Évolution et Transformation automatisée de modèles de Systèmes d'Information

Abdoulkader Osman Guédi

ACADÉMIE DE MONTPELLIER

UNIVERSITÉ **M**ONTPELLIER **II**

Sciences et **T**echniques du **L**anguedoc

THÈSE

présentée au Laboratoire d'Informatique de Robotique
et de Microélectronique de Montpellier pour
obtenir le diplôme de doctorat

Spécialité	:	**Informatique**
Formation Doctorale	:	**Informatique**
École Doctorale	:	**Information, Structures, Systèmes**

Évolution et Transformation automatisée de modèles de Systèmes d'Information

Une approche guidée par l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts

par

Abdoulkader OSMAN GUEDI

Version du 7 octobre 2013
Date prévue de la soutenance : Le 10 Juillet 2013

Directrice de thèse
Madame Marianne HUCHARD, Professeur ..... Université Montpellier II

Co-encadrant de thèse
Monsieur André MIRALLES, Ingénieur de recherche ..... UMR TETIS, IRSTEA
Madame Clémentine NEBUT, Maître de Conférences ..... LIRMM, Université Montpellier II

Président du jury
Madame THÉRÈSE LIBOUREL, Professeur ..... Université Montpellier II

Rapporteurs
Madame ISABELLE COMYN-WATTIAU, Professeur ..... Conservatoire National des Arts et Métiers(CNAM)
Monsieur PHILIPPE LAHIRE,Professeur ..... Université de Nice Sophia Antipolis

Examinateurs
Madame ISABELLE BORNE, Professeur ..... Université de Bretagne Sud
Monsieur DJAMA MOHAMED HASSAN, Maître de Conférences, ... Université de Djibouti

*À **mon papa** infiniment exemplaire*

*À **ma maman** fabuleusement exceptionnelle*
À **MA SŒUR** CADETTE BIEN AIMÉE ET
À **MON FRÈRE** AINÉ TRÈS RESPECTUEUX

Remerciements

Les travaux présentés dans cette thèse ont été effectués en alternance entre l'Université de Djibouti et l'Université de Montpellier, plus précisément au laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) et à l'Institut national de recherche en sciences et technologies pour l'environnement et l'agriculture (Irstea).

Ces travaux de recherche n'auraient pas pu aboutir sans les échanges précieux avec l'ensemble des acteurs de la recherche de l'UMR Tetis, du Lirmm, d'IRSTEA de Lyon, de Bordeaux et de Clermont-Ferrand et, à l'Université de Djibouti, le Service d'Étude et de Développement d'Applications (SEDA). Je souhaite remercier ceux qui ont contribué de près ou de loin à la réussite et à la bonne réalisation de ces travaux. Qu'ils trouvent trace dans ce mémoire de ma profonde reconnaissance.

Mes premiers remerciements et ma gratitude vont tout naturellement à l'ensemble de l'équipe qui a dirigé et encadré ma thèse, au professeur Marianne Huchard ma directrice de thèse, à André Miralles mon encadrant, à Clémentine Nebut mon encadrante. Je vous remercie chaleureusement pour les nombreux conseils prodigués et la confiance tout au long de ce travail.

J'adresse mes plus vifs remerciements au professeur Isabelle Comyn-Wattiau du Conservatoire National des Arts et Métiers de Paris et au professeur Philippe Lahire de l'Université de Nice Sophia Antipolis d'avoir accepté de juger et d'être les rapporteurs de mes travaux de thèse.

Mes amis, à Montpellier, à Djibouti, je vous remercie de vos encouragements et vos motivations.

J'adresse mes sincères remerciements à Jean-Pierre Chanet et à Thérèse Libourel pour avoir participé aux différents comités de suivi de thèse, et pour les conseils, les discussions et les échanges tout au long de la thèse.

Par ailleurs, je remercie l'ensemble des doctorants du LIRMM et de l'UMR Tetis pour les nombreuses discussions enrichissantes.
Une reconnaissance toute particulière aux autorités Djiboutiennes qui m'ont accordé leur soutien et ont financé la totalité de ma thèse.
Enfin, je voudrais tout particulièrement témoigner ici de toute mon affection à :
Mes très chers parents, ma sœur et mon frère sans oublier mes neveux et nièces et toute ma famille qui m'ont apporté leur soutien bien avant la thèse et qui continuent encore aujourd'hui. Grâce au grand Dieu, je vous dois TOUT. Pour croire en chacun de mes projets, pour votre soutien, vos encouragements et vos motivations.
À vous, je dédie donc cette thèse ainsi que l'ensemble de l'effort.

Résumé

De nos jours, l'évolution rapide des besoins dus à l'innovation technique, la concurrence, la réglementation, etc. conduit de plus en plus à décrire le cadre d'étude par des modèles conceptuels, métiers, etc. pour faciliter l'évolution du fonctionnement des systèmes informatiques. Actuellement, l'Ingénierie Dirigée par les Modèles (IDM) propose des outils permettant de transformer ces modèles en applications ou en systèmes d'information qui, bien évidemment, doivent évoluer comme les systèmes réels qu'ils sont sensés représenter. Généralement, le développement d'une application s'effectue en plusieurs phases qui constituent le cycle de développement du logiciel. Plusieurs équipes de nature différente contribuent aux différentes phases. Des intervenants, experts des domaines étudiés, produisent des modèles traduisant leur perception propre du système. Ainsi, les différentes perceptions des intervenants au cours de la phase d'analyse d'un système d'information donneront lieu à des modèles représentant des sous-systèmes spécifiques qu'il faudra ensuite réunir pour obtenir l'intégralité du modèle du système d'information étudié.

L'objectif essentiel de la thèse est de concevoir et d'implémenter dans un atelier de génie logiciel les mécanismes et les transformations consistant d'une part à extraire le *Plus Grand Modèle Commun*, modèle factorisant les concepts communs à plusieurs modèles sources et, d'autre part, à proposer aux concepteurs une méthodologie de suivi de l'évolution de la factorisation des modèles sources.

Pour réaliser la factorisation, nous avons mis en œuvre l'Analyse Formelle de Concepts (AFC) et l'Analyse Relationnelle de Concepts (ARC), méthodes d'analyse des données utilisées en Ingénierie Dirigée par les Modèles et basées sur la théorie des treillis. Dans un ensemble d'entités décrites par des caractéristiques, ces deux méthodes extraient des concepts formels qui associent un ensemble maximal d'entités à un ensemble maximal de caractéristiques partagées. Ces concepts formels sont structurés dans un ordre partiel de spécialisation qui les munit d'une structure de treillis. L'ARC permet de compléter la description des entités par des relations entre entités.

La première contribution de la thèse est une **méthode d'analyse de l'évolution de la factorisation d'un modèle basée sur l'AFC et l'ARC**. Cette méthode s'appuie la capacité de l'AFC et de l'ARC à faire émerger au sein d'un modèle des abstractions thématiques de niveau supérieur, améliorant ainsi la sémantique des modèles. Nous montrons que ces méthodes peuvent être aussi employées

pour suivre l'évolution du processus d'analyse avec des acteurs. Nous introduisons des métriques sur les éléments de modélisation et sur les treillis de concepts qui servent de base à l'élaboration de recommandations s'appuyant sur une expérimentation dans laquelle nous étudions l'évolution de la factorisation des 15 versions du modèle de classes du système d'information SIE-Pesticides. Ces versions ont été établies en phase d'analyse au cours de séances avec un groupe variable d'experts du domaine.

La seconde contribution de la thèse est une **étude approfondie du comportement de l'ARC sur des modèles UML**. Nous montrons l'influence de la structure des modèles sur les temps d'exécution, la mémoire occupée, le nombre d'étapes et la taille des résultats au travers de plusieurs expérimentations sur les 15 versions du modèle SIE-Pesticides. Pour cela, nous étudions plusieurs configurations (choix d'éléments de modélisation et de relations entre eux dans le méta-modèle) et plusieurs paramètres (choix d'utiliser les éléments non nommés, choix d'utiliser la navigabilité). Des métriques sont introduites pour guider le concepteur dans le pilotage du processus de factorisation et des recommandations sur les configurations et paramétrages à privilégier sont faites. Nous étudions également les modèles en tant que graphes au travers de plusieurs indicateurs tels que la densité ou le degré.

La dernière contribution est une nouvelle **approche pour assister la factorisation inter-modèles** dont l'objectif est de regrouper au sein d'un modèle unique l'ensemble des concepts communs à différents modèles sources conçus par des experts ayant des points de vue différents sur le système. Ce modèle unique que nous appelons *Plus Grand Modèle Commun* permet de capitaliser la connaissance commune à plusieurs experts. Cette factorisation est basée sur l'Analyse Formelle de Concepts et les concepts formels sont classés à l'aide d'un arbre de décision. Outre le regroupement des concepts communs, cette analyse produit de nouvelles abstractions généralisant des concepts thématiques existants. Nous appliquons notre approche sur les 15 versions du modèle SIE-Pesticides.

L'ensemble de ces travaux s'inscrit dans un cadre de recherche dont l'objectif est de factoriser des concepts thématiques au sein d'un même modèle et de contrôler par des métriques la profusion de concepts produits par l'AFC et surtout par l'ARC. Ces contributions et la robustesse des recommandations seront prochainement validées et consolidées en reproduisant l'expérimentation sur les différents modèles du projet de Système d'information pour la gestion du personnel et des étudiants de l'Université de Djibouti dont le but est de regrouper tous les modèles des systèmes d'information scolaires et universitaires des différentes institutions de la République de Djibouti.

Table des matières

CHAPITRE

1 Introduction

Préambule

Dans ce chapitre, nous décrirons le contexte de nos travaux, qui concernent la conception des systèmes d'information dans une approche orientée objets. Cette conception, qui est rendue délicate par la multiplicité des points de vue, par la taille des systèmes et leur constante évolution, nécessite des guides et des outils d'assistance. Nous présenterons notre problématique et nous introduirons notre approche. Puis nous détaillerons le plan de ce manuscrit.

Sommaire

1.1 Contexte

La multiplicité des points de vue des utilisateurs, l'évolution rapide des besoins dus entre autres à l'innovation technique, la concurrence ou la réglementation conduisent à privilégier des modèles conceptuels comme support de la construction et de l'évolution des systèmes d'information.

De fait, la conception des applications est de plus en plus rapide mais aussi de plus en plus complexe et se doit de prendre en compte des informations de plus en plus volumineuses. Cette complexité impose des méthodes de travail rigoureuses afin de répondre aux exigences des utilisateurs.

Généralement, le développement d'un projet est subdivisé en plusieurs phases qui constituent le cycle de développement de l'application, dont les plus importantes sont les phases d'analyse, de conception, d'implémentation et de déploiement.

Le point de départ est l'analyse des besoins de l'utilisateur lors de laquelle les besoins sont transcrits sous forme de modèles, lesquels modèles sont ensuite enrichis lors des phases de conception avant de déboucher sur l'implémentation. La mise au point de ces modèles s'effectue en plusieurs phases au cours desquelles collaborent plusieurs équipes de nature différente, chaque intervenant apportant sa perception du système à construire en se limitant à la partie de son domaine de spécialisation. Il faut alors concilier les différentes perceptions. Assez souvent, ces équipes composées de plusieurs intervenants vont produire plusieurs modèles qu'il faut ensuite mettre en commun.

Le travail en équipe, fait d'interactions et d'échange, rassemble les efforts et les connaissances expertes des membres du projet [Prégent, 1990]. Un réel potentiel naît de ce travail qui se vérifie dans le cycle de développement. Néanmoins, le travail en équipe a ses risques, les membres du projet peuvent être privés de leur part de satisfaction du travail par la structure hiérarchique [Perrenoud, 1994].

De nombreux auteurs [Beck, 2000; Bénard *et al.*, 2002; Boehm et Turner, 2004; Guimond, 2005; Jacobson *et al.*, 1999; Egyed et Kruchten, 1999; Martin, 1991; Miralles, 2006; Royce, 1970] ont souligné l'importance de la phase d'analyse qui constitue une étape clé du développement d'une application informatique ou d'un système d'information. En effet, c'est au cours de cette phase que les concepts du domaine et les besoins des acteurs doivent être transmis au concepteur de l'application qui doit se les approprier afin de les reformuler dans un ou plusieurs langages propres au domaine informatique. La qualité finale de l'application et son adéquation aux besoins des acteurs dépendent directement de cet exercice.

Les cycles de développement itératifs et les méthodes agiles [Beck, 2000; Highsmith, 2002; Jacobson *et al.*, 1999; Schwaber et Beedle, 2001] ont été introduits en réaction aux méthodes inspirées par la méthode Merise (cycle en V, cycle en cascade, etc. cf. chapitre 2) qui préconisent des cycles de développement long [Bénard *et al.*, 2002]. Ces méthodes inspirées par la méthode Merise s'appuient sur des cycles souvent qualifiés de linéaires car l'analyse est effectuée au début du projet et l'équipe de développement réalise ensuite l'application sans quasiment aucun contrôle de la part des acteurs. L'équipe présente l'application aux acteurs une fois celle-ci totalement finie. L'expérience montre que souvent les applications développées selon un processus linéaire ne répondent pas au strict besoin des acteurs.

Ce constat est confirmé par les statistiques du Standish Group [Group *et al.*, 2001]. En 2000 sur un échantillon de 150 000 projets de développement informatique, 49 % d'entre eux ont subi des

dépassements financiers ou des retards de livraison ou bien encore l'application finale comportait des fonctionnalités non prévues au départ. Dans 23 % des projets, la dérive par rapport aux objectifs initiaux est devenue trop importante et a conduit à abandonner le développement. Seulement 28 % des développements ont été réalisés dans le cadre budgétaire initialement prévu, dans le temps imparti et sans fonctionnalités superflues.

Fort de ce constat, de nombreux auteurs ont réfléchi et proposé de nouvelles méthodes qui impliquent de façon plus conséquente et plus formalisée les acteurs du système étudié. Les processus de développement itératifs et les méthodes agiles s'inscrivent dans cette logique.

Ces nouvelles méthodes font intervenir les acteurs tout au long du développement soit de façon périodique soit de façon continue.

Par exemple, la méthode itérative Unified Process fait intervenir les acteurs au début de chaque itération pour faire l'analyse d'une nouvelle brique du système à développer et à la fin de l'itération pour valider le produit de l'itération. Dans la méthode eXtreme Programming, l'acteur est plongé au sein de l'équipe tout le long du développement. Ces méthodes sont dites centrées sur les acteurs. Ces méthodes sont fondées sur l'hypothèse qu'un prototype est un objet technique qui fluidifie les échanges entre les acteurs et le concepteur et qu'il permet d'accroître la zone de connaissances communes [Guimond, 2005], zone appelée *Commonness* [Schramm et Roberts, 1971]. De plus, la mise en œuvre d'un prototype en phase d'analyse accélère l'apprentissage par les acteurs du langage de modélisation utilisé par le concepteur [Guimond, 2005].

1.2 Problématique et Objectif

1.2.1 Problématique à résoudre

Dans la pratique, l'analyse d'un système est effectuée lors de séances de travail qui doivent regrouper l'ensemble des acteurs intervenant sur un système. Il existe plusieurs types d'acteurs : des utilisateurs qui vont faire simplement la saisie des informations, d'autres qui vont remobiliser les informations pour les exploiter et prendre des décisions de gestion, le commanditaire soucieux de disposer rapidement d'informations, etc. Par exemple, les systèmes d'informations territoriaux et environnementaux ont comme caractéristique de faire intervenir un nombre important d'acteurs : citoyens, agriculteurs, syndicats agricoles, instituts professionnels, Service de la Protection des Végétaux, associations de protection de la nature, chercheurs, etc.

Regrouper l'ensemble de ces acteurs en un même lieu afin de faire l'analyse s'avère souvent être une vraie gageure car certains d'entre eux ont un emploi du temps très chargé. Lorsque cela est possible, les échanges entre certains acteurs peuvent devenir rapidement conflictuels, car ils ont des intérêts divergents. Dans un contexte potentiellement semi-conflictuel, certains types d'acteurs font de la rétention d'information ou donnent une vision déformée du système. La qualité sémantique du modèle final en est donc affectée. Ce processus d'analyse est illustré par la figure 1.1.

La mise au point de ces modèles s'effectue ainsi en plusieurs phases au cours desquelles collaborent plusieurs équipes de nature différente, chaque intervenant apportant sa perception du système à construire en se limitant à la partie de son domaine de spécialisation. Il faut alors concilier les différentes perceptions.

Face à ce type de situation, une solution consiste à faire travailler les acteurs par petits groupes ayant les mêmes domaines d'activités, de sensibilité et/ou d'affinité, puis à confronter les diffé-

rents points de vue au cours d'une séance plénière. Appliquée aux séances de travail, cette solution conduit à réaliser un modèle par groupe de travail. La figure 1.2 montre ce nouveau processus.

1.2.2 Objectif et Approches mises en œuvre

Dans un processus de modélisation, les premières étapes de l'analyse se nourrissent de l'inventaire de l'existant. Les données disséminées au sein d'organismes, d'institutions universitaires, etc. font partie de cet existant. Qu'elles soient structurées ou non, les données ont souvent un schéma connu ou sous-jacent. Comme précédemment, la difficulté technique réside dans le fait de disposer d'une vision claire de cet existant. La factorisation de différents modèles en un modèle va faciliter et accélérer l'appropriation des connaissances existantes sur le système.

Dans cette thèse, nous nous concentrerons sur l'évolution des aspects structurels des différentes versions de modèles à savoir les éléments de modélisation du domaine métier au moyen de classes, d'attributs, d'opérations ou d'associations. Le but est de comprendre comment guider le mieux possible l'équipe vers la version finale qui reflètera l'ensemble des points de vue des équipes et/ou les modèles antérieurs.

Cette opération de regroupement, ou d'assemblage, s'appuiera sur une factorisation de différents modèles afin de proposer un seul modèle commun. Cette opération est accompagnée d'une analyse de l'évolution des versions de modèles.

Comme évoqué précédemment, du fait des différentes perceptions des membres d'une équipe d'analyse, nous aurons en effet plusieurs modèles répondant soit à des parties spécifiques soit à la totalité du système d'information étudié.

L'objectif essentiel de la thèse est de concevoir les mécanismes permettant d'une part d'obtenir le modèle factorisant les concepts communs à plusieurs modèles et, d'autre part, de proposer aux

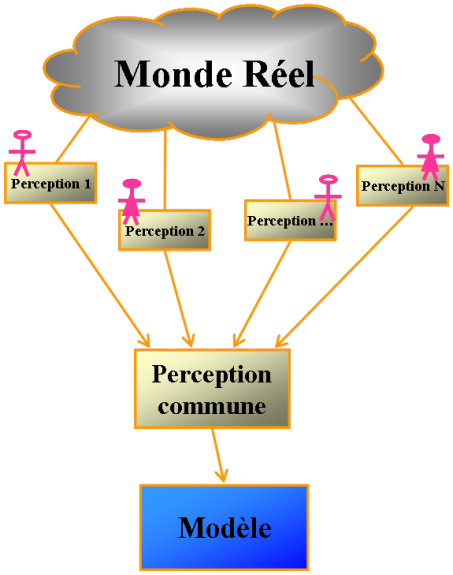


FIGURE 1.1: **Analyse en groupe**

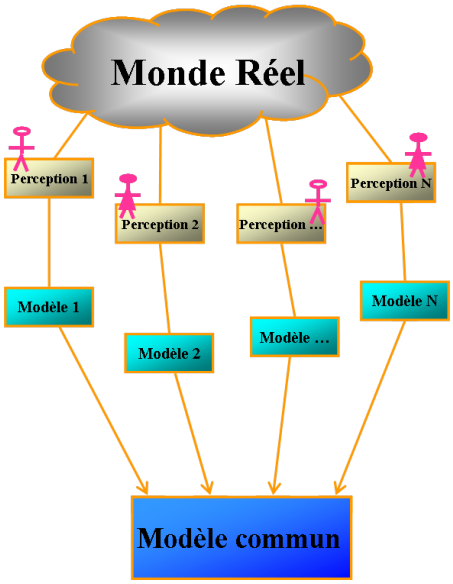


FIGURE 1.2: **Analyse individuelle**

concepteurs une méthodologie de suivi de l'évolution de la factorisation. Notre réflexion s'inscrit dans le cadre du suivi de l'évolution des modèles dans le cycle de vie d'un système d'information. À une étape du cycle de vie, chaque modèle peut gagner ou perdre des concepts liés aux aspects structuraux ou relationnels pour sa mise en œuvre. Nous proposerons des méthodes et des mécanismes permettant d'obtenir automatiquement ou quasi-automatiquement cette factorisation ainsi que les outils informatiques qui permettront de la réaliser. Ces derniers seront implémentés dans un Atelier de Génie Logiciel (AGL).

Pour réaliser la factorisation et suivre l'évolution, nous avons mis en œuvre l'Analyse Formelle de Concepts (AFC) [Ganter et Wille, 1999] et l'Analyse Relationnelle de Concepts (ARC) [Huchard, 2003] qui sont des méthodes d'analyse de données basées sur la théorie des treillis. Dans un ensemble d'entités décrites par des caractéristiques, les deux méthodes extraient des concepts formels qui associent un ensemble maximal d'entités à un ensemble maximal de caractéristiques partagées. Ces concepts formels sont structurés dans un ordre partiel de spécialisation qui les munit d'une structure de treillis. L'Analyse Relationnelle de Concepts permet de compléter la description des entités par des relations entre entités.

Ces approches s'inscrivent dans le cadre conceptuel de l'Ingénierie Dirigée par les Modèles qui permet de mieux pérenniser les modèles de haut niveau ayant une grande complexité (par exemple le modèle du projet *SIE-Pesticides* (cf. section 4.2.1)).

Le suivi de l'évolution (cf. chapitre 5) est traité par le biais de l'introduction de métriques sur les éléments de modélisation et sur les treillis de concepts qui servent de base à l'élaboration de recommandations. Les deux approches (AFC et ARC) sont également mises en œuvre pour factoriser les différentes perceptions représentées sous forme d'un modèle UML. Elles s'appuient sur la factorisation maximale des attributs et des méthodes pour la découverte de nouvelles abstractions. Les informations utilisées lors de cette approche peuvent être le nom ou la signature des attributs et des méthodes. On peut également utiliser des relations de spécialisation connues sur ces attributs, leur type, ou sur les méthodes [Roume, 2004]. Nous montrons que la capacité de ces approches à faire émerger au sein d'un unique modèle des abstractions thématiques de niveau supérieur (factorisation intra-modèle [Falleri, 2009]) s'étend également au cas où on est en présence de plusieurs modèles (factorisation inter-modèles, cf. figure 1.3), comme nous l'expliquerons au chapitre 7.

Dans notre contexte, nous allons utiliser ces méthodes sur des modèles de classes complexes issus d'un système d'information réel. Le but est de disposer d'une meilleure abstraction de ces modèles afin de faciliter la réutilisation et de simplifier la compréhension des concepts du domaine. Nous allons mettre en œuvre ces méthodes qui s'appuient sur le principe de la généralisation et de la spécialisation. Par généralisation, nous entendons l'action de réunir les caractéristiques communes à plusieurs concepts dans un nouveau concept de niveau d'abstraction supérieur. Inversement, la spécialisation répartit les caractéristiques d'un concept abstrait dans des concepts plus spécifiques.

1.3 Motivations personnelles et institutionnelles

L'origine et la motivation de cette thèse sont issues du cadre de mon travail au sein de l'université de Djibouti [1]. En effet, mes attributions principales en complément de l'enseignement sont

1. http://www.univ.edu.dj

l'étude et le développement de plusieurs applications permettant la gestion des examens des étudiants et du personnel de l'Université de Djibouti et des régions environnantes.

Les travaux présentés dans le document ont été effectués en alternance par période de six mois entre Djibouti et Montpellier. La thèse a commencé au 01 septembre 2009 et s'est terminée au 10 juillet 2013, soit une durée d'un peu plus de 46 mois. Dans les faits, seulement 24 mois ont été pleinement consacrés aux travaux de recherche lors des différents séjours à Montpellier, le reste du temps était consacré à mes obligations professionnelles au sein de l'Université de Djibouti du fait de mon statut de permanent.

À Djibouti, j'étais dans mon cadre professionnel à l'Université de Djibouti et, à Montpellier, j'étais en accueil au sein de l'Unité Mixte de Recherche Territoires, Environnement, Télédétection et Information Spatiale (UMR TETIS) d'Irstea [2] et au sein du laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) de l'UMII [3].

Cette thèse est donc intimement couplée au développement réalisé à Djibouti d'autant plus que les contributions de la thèse doivent apporter efficacité et robustesse à l'application. Par ailleurs au sein d'Irstea, nous travaillons sur des développements plus aboutis (le Système d'Information Environnemental pour les Pesticides) qui nous servent de terrain d'étude.

1.4 Organisation du rapport de thèse

Le mémoire de cette thèse est organisé en quatre parties. La *première partie* correspond à l'étude bibliographique du contexte et de différentes approches pour résoudre la problématique de la thèse. Elle est composée de deux chapitres.

2. Institut national de recherche en sciences et technologies pour l'environnement et l'agriculture
3. Université de Montpellier II

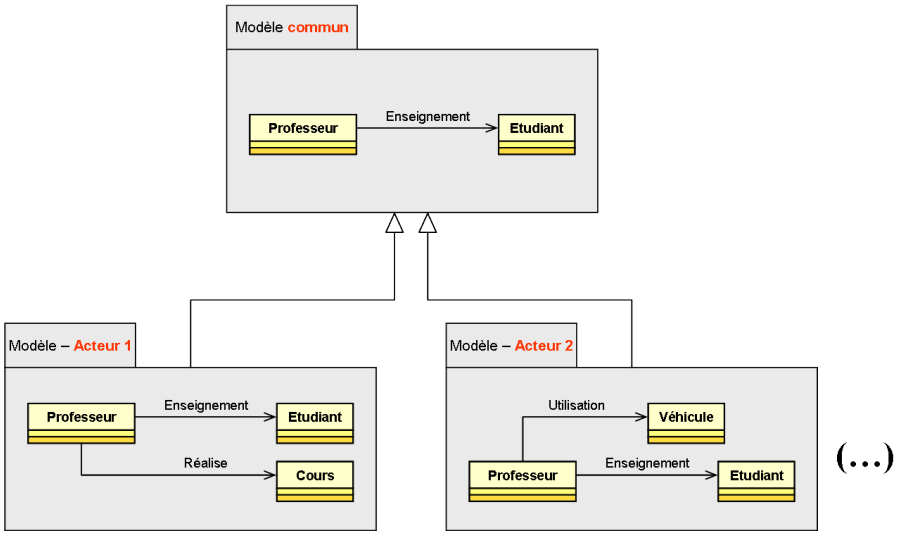


FIGURE 1.3: **Factorisation inter-modèle**

Tout d'abord, dans le chapitre 2, nous abordons l'évolution des systèmes d'information dans le cadre de l'Ingénierie Dirigée par les Modèles (IDM). Ainsi, nous aborderons les principaux outils et/ou méthodes basées sur l'évolution dans le contexte de l'IDM, domaine dans lequel nos contributions ont pris leur source. D'une part, nous présenterons la comparaison des modèles afin d'identifier de nouveaux concepts et plus généralement comprendre l'évolution et d'autre part, nous verrons les métriques existantes comme outils d'analyse de l'évolution.

Ensuite, dans le chapitre 3, nous abordons les différentes approches qui nous permettent de factoriser les modèles. D'une part, nous présentons l'Analyse Formelle de Concepts [Ganter et Wille, 1999] une technique de regroupement d'entités ayant des caractéristiques communes dans des concepts. Ensuite, nous verrons l'Analyse Relationnelle de Concepts [Huchard, 2003] qui est une extension de la précédente approche. Cette deuxième approche prend non seulement en compte les caractéristiques communes des entités, mais aussi les relations qu'elles entretiennent entre elles. Ces deux approches sont basées sur la théorie des treillis de concepts [Birkhoff, 1940; Barbut et Monjardet, 1970].

Dans une *seconde partie*, mise en œuvre dans le chapitre 4, nous décrivons le contexte d'expérimentation de notre approche. Nous nous plaçons dans un cadre de développement d'une application informatique en différentes phases dans lesquelles contribuent plusieurs équipes de natures diverses. Les spécificités propres aux domaines d'application avec leurs impacts sur les modèles sur les différents projets applicatifs font l'objet de ce chapitre. Le premier de ces cas d'études (*SIE-Pesticides*) sert de plus de socle expérimental pour valider les contributions. Nous expliquons quels sont les principaux éléments de modélisation qui nous serviront pour le suivi de l'évolution ainsi que pour la factorisation de modèles. Puis nous introduisons les divers outils utilisés et développés dans les expérimentations.

La *troisième partie* regroupe les contributions de la thèse et se compose de trois chapitres.

Le chapitre 5 correspond à la première contribution de la thèse. Nous y exposons une méthode d'analyse de l'évolution de la factorisation d'un modèle basée sur l'AFC et l'ARC. Cette méthode s'appuie sur la capacité de l'AFC et de l'ARC à faire émerger au sein d'un modèle des abstractions thématiques de niveau supérieur, améliorant ainsi la sémantique des modèles. Nous montrons que ces méthodes peuvent aussi être employées pour suivre l'évolution du processus d'analyse avec des acteurs. Nous introduisons des métriques sur les éléments de modélisation et sur les treillis de concepts qui servent de base à l'élaboration de recommandations. Nous effectuons une expérimentation dans laquelle nous étudions l'évolution des quinze versions du modèle de classes du système d'information *SIE-Pesticides*.

Le chapitre 6 présente la seconde contribution de la thèse et effectue une étude approfondie du comportement de l'ARC sur des modèles UML. Nous montrons l'influence des données prises en compte dans les modèles et de la structure des modèles sur différentes variables étudiées telles que les temps d'exécution et la mémoire occupée. Ceci est le fruit de plusieurs expérimentations sur les 15 versions du modèle *SIE-Pesticides*. Nous étudions plusieurs configurations (choix d'éléments et de relations dans le méta-modèle) et plusieurs paramètres (choix d'utiliser les éléments non nommés, choix d'utiliser la navigabilité). Des métriques sont introduites pour guider le concepteur dans le pilotage du processus de factorisation et des recommandations sur les configurations et paramétrages à privilégier sont faites.

Le chapitre 7 aborde la troisième contribution et propose une approche de factorisation inter-modèles afin de regrouper au sein d'un unique modèle les concepts communs à différents modèles sources conçus par différents experts. Outre le regroupement des concepts communs, cette analyse produit de nouvelles abstractions généralisant des concepts thématiques existants. Nous appliquons notre approche sur les quinze versions du modèle du *SIE-Pesticides*.

Dans la *quatrième et dernière partie* du manuscrit, nous présentons les conclusions et les perspectives. Le chapitre 8 rappelle les contributions de la thèse. Le chapitre 9 ouvre des perspectives à nos différentes contributions.

Pour finir, nous présentons en complément dans l'annexe A un récapitulatif complet des résultats des expérimentations issus des relations et des configurations sur le cas d'étude.

Par ailleurs, cette annexe contient un article en soumission à EDOC 2013 où nous étudions systématiquement une application pratique de l'Analyse Relationnelle de Concepts sur plusieurs versions d'un modèle de classes. Ce dernier est issu d'un cas réel et ces expérimentations permettent de donner des chiffres précis sur le comportement de L'Analyse Relationnelle de Concept.

Première partie

Étude Bibliographique

CHAPITRE

2 Factorisation et Évolution de modèles

Préambule

Dans ce chapitre, nous donnons des éléments de contexte de nos travaux, notamment l'ingénierie dirigée par les modèles et le cycle de vie des systèmes d'information. Nous présentons ensuite quelques approches de la littérature pour mesurer par des métriques la qualité et la complexité des modèles et quelle place elles peuvent occuper dans le suivi de l'évolution.

Sommaire

2.1 Introduction

Dans ce manuscrit, nous nous intéressons à divers aspects du cycle de vie d'un système d'information. Tout d'abord, lors de sa conception, nous nous intéressons à l'élaboration d'un modèle structurel permettant de concilier plusieurs modèles réalisés par plusieurs partenaires. Nous nous intéressons également à la présence et à la qualité des abstractions dans ce modèle, et enfin à l'évolution de ce modèle lors de son élaboration en phases successives.

Dans cette section, nous positionnons tout d'abord le contexte de ces travaux, notamment celui de l'ingénierie dirigée par les modèles (IDM). En effet, comme nous travaillons sur des modèles que nous raffinons et intégrons, notre approche relève de ce contexte. Nous donnons ici quelques éléments de l'IDM, et montrons sur quelles applications de l'IDM nous avons travaillé. Ensuite, nous positionnons nos travaux par rapport au cycle de vie d'un système d'information. Nous montrons rapidement les différents cycles de vie envisageables, et précisons dans quel cadre nos travaux peuvent s'articuler.

Dans le cadre défini, nous nous intéressons au suivi de l'évolution d'un modèle de système d'information, en nous basant sur des indicateurs de qualité du modèle, portant notamment sur les niveaux d'abstraction. Nous abordons dans la seconde section de ce chapitre différents indicateurs présents dans la littérature permettant de suivre la qualité et la complexité d'un modèle et qui motivent nos propositions ultérieures.

2.2 Contexte : les modèles dans le cycle de vie d'un système d'information

2.2.1 Ingénierie dirigée par les modèles

L'Ingénierie Dirigée par les Modèles est une approche contemporaine [Selic, 2003] très utilisée dans les développements des logiciels. Elle apporte une robustesse à la production des logiciels en termes d'efficacité et de gain de coût (le délai) tout en pérennisant le système d'information étudié.

D'après Barry Boehm [Boehm, 2006], cette approche n'est pas très récente en terme d'application, mais la formalisation de ces contributions est l'effort des recherches menées dans les années 1980. Notamment, le développement de CASE [1] a permis de faire émerger des outils et des techniques pour une large communauté.

L'Ingénierie Dirigée par les Modèles (IDM) ou Model Driven Engineering (MDE) en anglais [Kent, 2002] prône l'utilisation des modèles pour guider le développement en mettant à notre disposition un corpus de concepts, de méthodes et d'outils, mais aussi de langages pour créer, organiser, restructurer et transformer des modèles (cf. définition 2.1). Ces derniers sont décrits par des modèles de plus haut niveau appelés méta-modèles (cf. définition 2.2).

Un des principes de l'IDM est de découpler l'aspect métier des aspects structurels et organisationnels liés à l'implémentation. Il en résulte un gain de capitalisation. De ce fait, le modèle constitue l'artefact essentiel pour développer des applications informatiques. Ces modèles représentent une vision naturelle du contexte que l'utilisateur est amené à concevoir.

1. Computer Aided Software Engineering

Plusieurs langages de modélisation généralistes ont ainsi été créés, parmi lesquels les plus connus ont d'abord été Merise [Arnold Rochfeld, 1989] puis UML [Booch *et al.*, 2000]. Ce dernier propose une large gamme de diagrammes d'expression des besoins, de conception et de comportement dont l'utilité est intéressante dans les différentes phases de développement d'un logiciel.

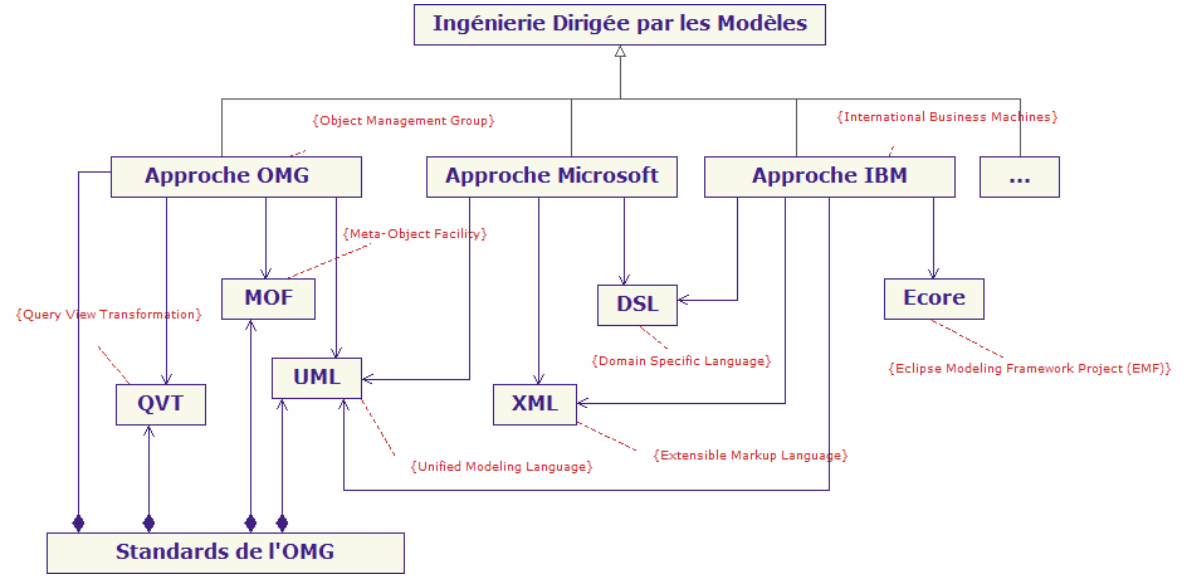


FIGURE 2.1: **Approches orientées modèles en ingénierie du logiciel** [FAVRE *et al.*, 2006]

L'Ingénierie Dirigée par les modèles concentre l'effort de développement des applications au niveau des modèles. Ces derniers peuvent décrire de façon relativement précise un système d'information. Ces modèles sont conformes à un méta-modèle dont ils sont une instance. Un métamodèle définit les concepts présents dont les modèles qui lui sont conformes. Par exemple, le métamodèle UML définit l'ensemble des concepts que l'on peut utiliser lorsque l'on développe un modèle UML (cf. figure 2.1). Les Ateliers de Génie Logiciel (AGL) tel qu'Objecteering (cf.section 4.4.1) implémentant ces méta-modèles permettent d'exploiter ces modèles pour générer automatiquement le code source d'un langage de programmation, de requête, etc. Ils permettent en effet de manipuler un modèle en tant qu'instance de son méta-modèle.

Définition 2.1 *Le* modèle *représente une description parfaite d'un système en sa globalité ou une partie correspondante à l'étude. Ce système est défini dans un langage bien précis qui effectue la transcription de ce système en respectant le méta-modèle [Kleppe* et al.*, 2003].*

Par exemple, dans ce manuscrit nous travaillons sur le projet *SIE-Pesticides* (cf. section 4.2.1), dont l'objectif est la production des connaissances pour évaluer l'impact des produits phytosanitaires dans les différents compartiments de l'environnement relevant des domaines d'intervention de IRSTEA. La structuration de la connaissance a été capitalisée sous forme d'un modèle de classes UML. Ce modèle peut ensuite être manipulé par différentes transformations, notamment on peut y rechercher de nouvelles abstractions, ou y détecter des duplications.

Définition 2.2 *Le* méta-modèle *est d'abord un modèle définissant un langage pour décrire, comprendre un modèle et il facilite le raisonnement sur sa structure, sa sémantique et son usage [Kleppe et al., 2003].*

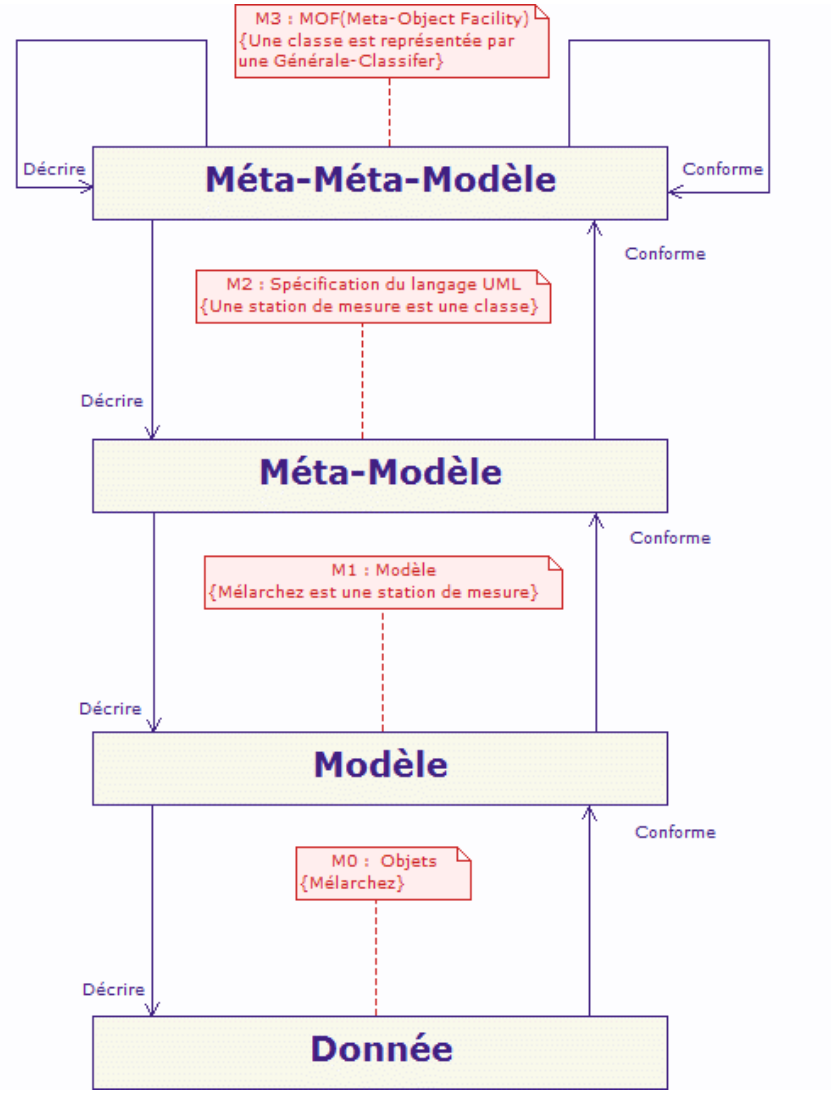


FIGURE 2.2: **Les différents niveaux de la modélisation** [FAVRE *et al.*, 2006]

Par exemple : les différentes versions de modèle issues du projet *SIE-Pesticides* sont conformes au méta-modèle d'UML. Le méta-modèle respecte à son tour un langage décrit par un méta-méta-modèle. Ce dernier définit bien un ensemble de concepts et de relations entre ces concepts pour modéliser le système étudié.

Dès lors que les modèles sont définis en respectant un métamodèle, lui-même défini grâce à un méta-méta-modèle connu, les modèles deviennent manipulables par programmes. Ces pro-

grammes sont appelés transformations de modèles. Les transformations peuvent être implémentées avec des langages de programmation généralistes comme JAVA, ou plus spécifiques comme Kermeta ou ATL. Les transformations de modèles permettent donc de manipuler des modèles, par exemple pour les transformer (application de refactorings), récolter des informations (calcul de métriques), ou encore pour générer d'autres modèles (modèle de classes vers modèle relationnel).

Ces transformations sont de deux types :

- **Transformation endogène**, les modèles sources et cibles sont conformes au même méta-modèle ; par exemple, transformation d'un modèle UML en un autre modèle UML.
- **Transformation exogène**, dans ce cas, les modèles n'ont pas le même méta-modèle ; par exemple, génération d'un modèle relationnel à partir d'un modèle UML.

2.2.2 IDM et MDA

L'Architecture Dirigée par les Modèles (Model Driven Architecture (MDA) en anglais) est une initiative du consortium industriel OMG ([2]) conçue pour réaliser la conception des applications, en proposant des outils, des langages, et des modèles pour faciliter le développement, la maintenance et l'évolution des modèles. Le MDA est souvent vu comme une restriction de l'ingénierie dirigée par les modèles aux outils et aux modèles développés par l'OMG.

La principale idée de MDA réside en la séparation du métier de son implémentation, via l'introduction de modèles dits indépendants de la plateforme, et de modèles dits dépendants de la plateforme.

L'approche MDA propose ainsi de structurer le développement d'une application suivant trois types de modèles :

1. Computation Independent Model (CIM) : ce modèle représente la vue des utilisateurs sur le système et en particulier les exigences pour une meilleure utilisation. Ici les aspects structurels ne sont pas retenus. Autrement dit, le modèle représente l'application dans son environnement.
2. Platform Independent Model (PIM) : ce modèle, indépendant des plateformes de programmation cibles, modélise l'analyse et la conception abstraite tout en représentant l'architecture de l'application.
3. Platform Specific Model (PSM) : il étend le PIM en ajoutant les détails d'implémentation propres au langage de programmation cible.

Le modèle Computation Independent Model (CIM) a été ajouté à l'architecture MDA en 2003 afin de combler la distance existant entre les experts du domaine thématique et les experts de la conception et de la construction de l'application [Miller et Mukerji, 2003]. Le modèle CIM est destiné à décrire les exigences des acteurs du domaine. Une caractéristique importante du modèle CIM est que le vocabulaire familier des acteurs du domaine est utilisé pour sa spécification [Miller et Mukerji, 2003]. En outre, il est explicitement mentionné dans ce guide que ce modèle est utile, non seulement comme aide à la compréhension du problème, mais aussi comme source de vocabulaire partagé à utiliser dans d'autres modèles. Dans la communauté des bases de données, le dictionnaire de données joue partiellement le rôle de source de vocabulaire du modèle CIM.

2. http://www.omg.org/

Utilisation du MDA dans le cadre de ce manuscrit. Dans ce manuscrit, nous présentons une approche destinée à assister le développement de modèles pour les systèmes d'information : refactorisation, extraction de concepts communs dans plusieurs modèles du même système, étude de l'évolution des modèles.

Dans l'optique de pouvoir confier rapidement ces outils à des thématiciens dans le cadre du projet *SIE-Pesticides*, nous avons opté pour l'intégration des outils dans un atelier de génie logiciel majoritairement dédié à UML : les outils sont mis à disposition au sein d'un profil dans l'Atelier de Génie Logiciel (AGL) Objecteering (cf. section 4.4.1). Ce profil permettra d'appliquer nos propres transformations pour factoriser les modèles et générer les données nécessaires à l'analyse de l'évolution. Ainsi, notre utilisation du domaine de l'ingénierie dirigée par les modèles se positionne plutôt dans la branche MDA, avec le développement d'approche et d'outils pour le méta-modèle UML, et implémentés dans un modeleur UML.

2.2.3 Cycles de vie d'un système d'information

Le cycle de vie d'un système d'information regroupe toutes les phases qui ont donné naissance à un état du système courant. Dans le domaine du génie logiciel, cet état correspond à la notion de *version* qui est finalement un instantané à un stade de l'évolution [Royce, 1970].

Un système d'information existant n'est souvent que l'évolution d'un système déjà existant qu'il a été nécessaire de faire évoluer pour prendre en compte de nouveaux besoins par exemple. Cette évolution qui a généralement plusieurs origines implique souvent une modification profonde des processus d'entreprise, une rupture d'un rythme dans les habitudes l'ordre établi, etc. En d'autres termes, la perte ou l'acquisition des nouveaux comportements et/ou structures conduit à l'évolution du système d'information.

En général, les phases d'un projet dépendent du cycle de vie adopté. Chaque phase est découpée en plusieurs étapes et activités propres. L'aboutissement des étapes d'une phase peut être entérinée par une décision prise par le chef de projet. Pour chacune de ces étapes, le concepteur définit en accord avec l'utilisateur le produit à réaliser.

2.2.3.1 Les phases d'un cycle de développement

Les phases dépendent essentiellement du processus ou de la méthode de développement. Néanmoins, certaines d'entre elles sont récurrentes pour l'ensemble des processus :

1. **Phase de définition et d'analyse** : c'est la phase de l'étude des besoins et de la capture de quelques concepts majeurs du domaine.
2. **Phase de conception** : elle regroupe les principales activités concernant les spécifications détaillées propres au domaine tout en prenant en compte les exigences identifiées précédemment.
3. **Phase d'implémentation ou de réalisation** : Cette phase correspond à la concrétisation du projet, c'est-à-dire le passage des besoins abstraits aux résultats concrets.
4. **Phase de maintenance** : Cette phase permet de suivre l'évolution du projet et de maintenir le système informatique en accord avec les besoins.

Dans le domaine du génie logiciel, les différentes phases retracent l'ensemble des changements, en d'autres termes l'évolution du logiciel. Les méthodes traditionnelles de développement dérivent le plus souvent des méthodes de gestion de projet éprouvées en ingénierie industrielle ou dans le secteur du Bâtiment et des Travaux Publics [Larman, 2002]. Il existe une multitude de méthodes traditionnelles pour développer une application informatique, mais la plupart reposent sur quelques méthodes de référence :

1. Cycle en cascade.
2. Cycle en V.
3. Cycle en W.
4. Cycle en spirale.

D'après [Dowson, 1987], ces cycles sont orientés activités car ils se concentrent sur les activités exécutées pour réaliser un produit et sur l'ordonnancement de ces activités. Ces cycles définissent aussi d'autres sortes de modèle, d'une part, les modèles orientés produits qui couplent l'état du produit par rapport à l'activité générée et, d'autre part, les modèles orientés décision qui s'intéressent aux transformations subies par le produit tout au long de son cycle de vie.

2.2.3.2 Cycle en cascade, cycle en V et en W

Ces deux cycles définissent diverses activités dépendantes des unes des autres pour un objectif bien précis qui est de réaliser un produit final. Ces activités sont le fil conducteur d'une méthode de conduite du projet.

Cycle en cascade. Le cycle en cascade [Royce, 1970] décrit uniquement la phase de développement et le fait de manière linéaire. Il est calqué sur les cycles de construction utilisés dans le secteur du Bâtiment et des Travaux Publics. Les activités (cf. figure 2.3) sont réalisées de manière séquentielle les unes après les autres.

L'analyse, la conception et l'implémentation constituent le cœur du processus de développement. Les autres phases deviennent indispensables lorsque la taille de l'application devient conséquente. Chacune de ces phases peut contenir des activités identiques telles que la documentation. Cette dernière s'enrichit au fur à mesure de l'avancement du projet. Chaque phase de l'approche descendante donne lieu à une validation technique. Dans ce cycle, le retour à phase précédente n'est théoriquement pas possible. Toutefois, pour les projets importants, il est parfois nécessaire de revenir à la phase précédente. L'inconvénient majeur de ce cycle est que l'analyse est réalisée en une seule fois de manière globale. Aussi, si des concepts majeurs n'ont pas été appréhendés lors de cette phase, ils seront absents dans le produit final lequel ne répondra pas aux besoins des utilisateurs.

Cycle en V. Le cycle en V [McDermid et Ripken, 1984] permet d'améliorer le cycle en cascade en préservant la linéarité des phases (branche de gauche) tout en ajoutant une nouvelle branche de validation (branche de droite). En effet, chacune des phases de la branche gauche est synchronisée à une batterie de tests associés dont un certain nombre sont définis en accord avec l'utilisateur.

Dans ce cycle (cf. figure 2.4), la branche de gauche est d'abord réalisée et ensuite la branche de droite, branche de tests permettant de valider la conformité de l'application. Cette approche

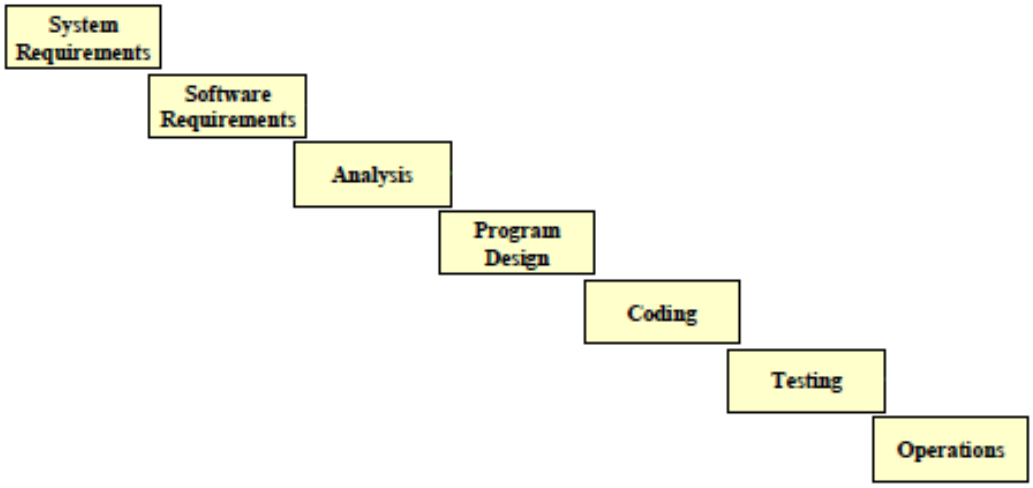


FIGURE 2.3: **Cycle de développement en cascade**

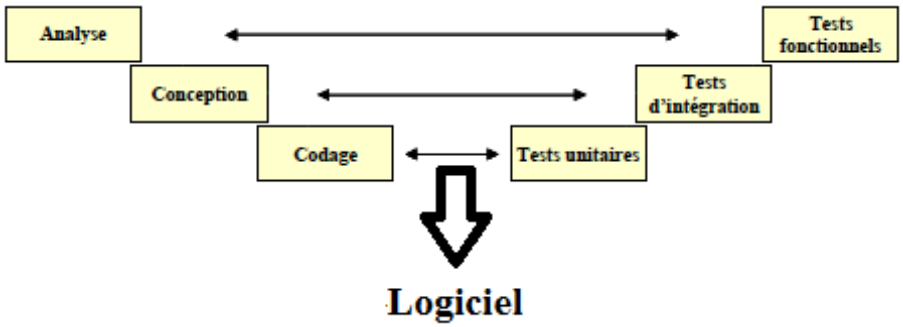


FIGURE 2.4: **Cycle de développement en V**

permet de vérifier la conformité à ce qui devrait être fait et non ce qui a été fait. Pour ce faire, les tests fonctionnels sont donc spécifiés lors de l'analyse, les tests d'intégration lors de la conception et les tests unitaires pendant la phase de codage.

Cycle en W. Ce modèle est une variante du modèle en "V". Le cycle de développement ne commence que quand les problèmes ou les besoins sont parfaitement connus. Un cycle en "W" (Waterfall en anglais) peut être couplé à un cycle en "V", quand il est nécessaire de décomposer le système en sous-systèmes. Un des avantages de cette approche est que deux types de tâches sont réalisées en parallèle : verticalement on prépare l'étape suivante et horizontalement on prépare la vérification de la tâche en cours. Par contre, les phases restent séquentielles [Kay, 2002].

2.2.3.3 Cycle en spirale

Le cycle de développement en spirale [Boehm, 1988] est basé sur les risques avec le déclenchement d'actions associées. L'objectif est de se concentrer sur le développement par les risques et non plus par la documentation ou par le code comme c'était le cas jusqu'alors.

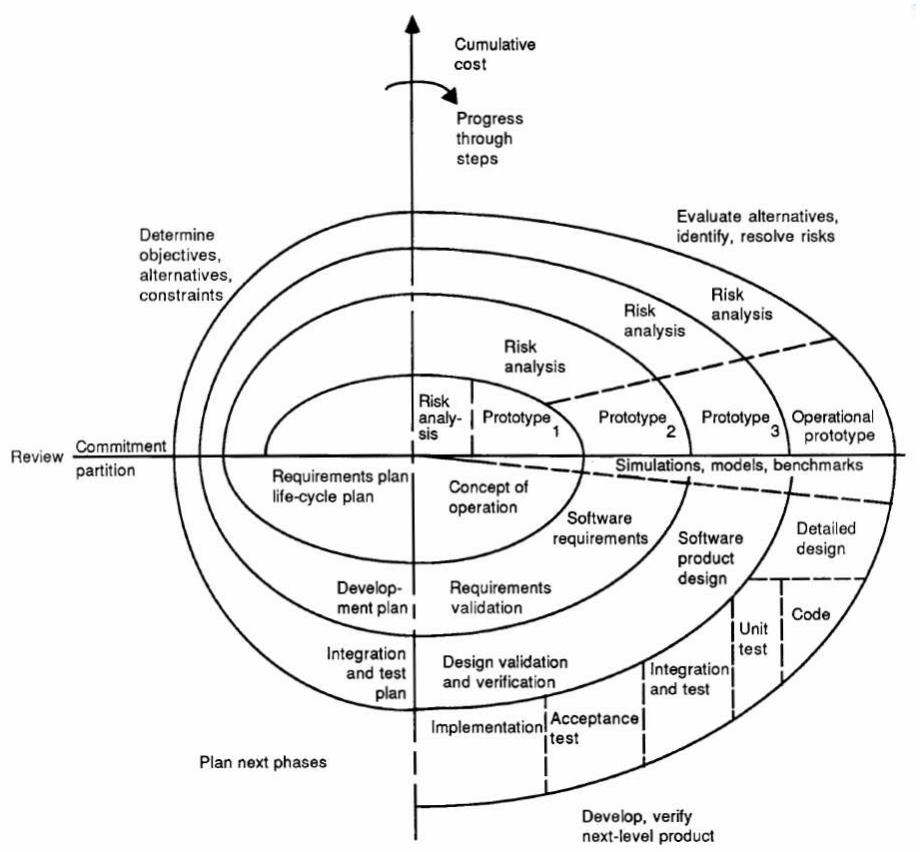


FIGURE 2.5: **Cycle de développement en spirale**

Chaque cycle de la spirale (cf. figure 2.5) est composé d'une phase d'analyse, d'une phase de développement ou d'une phase de test. Le dernier cycle donne lieu à la finalisation du projet. Cette méthode de conduite de projet commence par la définition des objectifs ainsi que l'identification des alternatives d'implémentation possibles et des contraintes. Ensuite, elle s'intéresse d'une part, à l'évaluation des risques induits par l'alternative d'implémentation et, d'autre part, à la conception et au développement incluant les tests unitaires, d'intégration et fonctionnels. Enfin, la dernière activité concerne la validation du produit de la phase par les acteurs du domaine et la planification de la phase suivante.

2.2.3.4 Méthode Rapid Application Development (RAD)

L'objectif de cette méthode de conduite de projet est de changer radicalement le processus de développement afin de réaliser très rapidement des applications de qualité. Une particularité de cette méthode est la rapidité du développement dans des délais réduits tout en restant sous le contrôle total des utilisateurs. Le cycle de développement [Vickoff, 2003] de cette méthode se compose de cinq phases : Initialisation, Cadrage, Design, Construction et Finalisation (cf. figure 2.6) tout en indiquant les tâches essentielles à accomplir au cours de chacune des phases. Le principal objectif de la phase de Design reste l'approfondissement des spécifications.

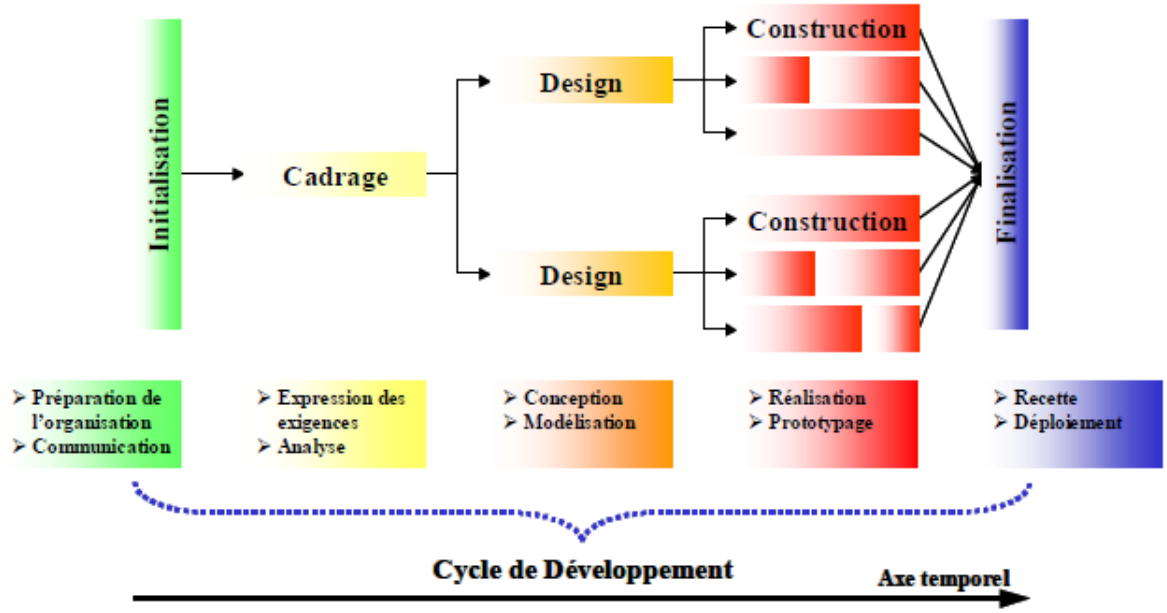


FIGURE 2.6: **Cycle de développement avec le RAD**

Bien évidemment, les acteurs du domaine sont au centre des préoccupations et ils participent à la phase d'analyse pour définir le cadre du projet. Ensuite, ils décrivent les grandes parties du système à modéliser et expriment leurs exigences. En phase de Construction, les acteurs du domaine participent à des réunions plénières, qui sont essentiellement l'occasion de présenter l'avancement du projet et de le valider dans toutes ses dimensions. La participation des acteurs tout au long du cycle de développement simplifie la validation finale de l'application.

2.2.3.5 Méthodes Agiles

Les méthodes agiles disposent d'un cycle de développement constitué d'itérations courtes permettant une grande réactivité dans la prise en compte des besoins des utilisateurs. Elles mettent en œuvre des moyens et des outils favorisant un échange constant avec les utilisateurs du domaine. La contrainte principale est la place qu'occupe l'utilisateur dans toutes les phases du processus. Ces méthodes agiles, dont les plus récentes sont eXtreme Programming, Dynamic Software Deve-

lopement Method, Adaptative Software Developement, SCRUM, etc., reposent sur une structure commune qui peut être itérative, incrémentale et adaptative [Bénard *et al.*, 2002].

Méthode eXtreme Programming (XP). La méthode eXtreme Programming a été conçue par Kent Beck[Beck et Andres, 2004], Ward Cunningham et Ron Jeffries, tous trois experts en développement logiciel. La phase du développement est au cœur du projet. Cette méthode est fondée d'une part, sur une implication forte des utilisateurs et, d'autre part, sur la mise en œuvre systématique de tests implémentés assurant la qualité du code remanié par les développeurs. De ce fait, elle facilite la maintenance du code mais aussi son évolution pour répondre à de nouveaux besoins. La factorisation des codes sources est présente tout au long du développement et l'utilisation de patrons (des méthodes recettes) mais aussi de frameworks est très importante. Elle prône la programmation en binôme avec une conception modulaire. Ainsi, la livraison se fait au fur et à mesure de l'état d'avancement afin d'avoir des retours par l'utilisateur final.

La communication constante avec l'utilisateur, la simplicité du développement et le retour d'information constituent ses principaux atouts par rapport aux méthodes traditionnelles.

2.2.3.6 Méthodes spécifiques

Ces méthodes sont souvent spécifiques aux systèmes pour lesquels elles ont été conçues. De ce fait, elles sont rarement généralisées. Dans le cadre général, elles sont constituées de trois grandes étapes : d'abord la spécification des besoins, la conception et la validation par des utilisateurs.

L'intérêt majeur de ces méthodes est que le passage d'une phase à l'autre est automatisé ou semi-automatisé à l'aide de transformation de modèles. De ce fait, l'effort principal peut être porté au niveau de l'identification des besoins et de l'analyse du domaine. Le code est alors considéré comme un produit.

Dans cette catégorie, il est possible de citer une nouvelle méthode de développement de prototypage rapide revisitée par MDA [Miralles et Libourel, 2006, 2009]. Cette méthode appelée *Continuous Integration Unified Process Method* est itérative et incrémentale et est conçue pour la production rapide d'exécutables (cf. figure 2.7). Son intérêt majeur est de recueillir les réactions instantanées des acteurs du domaine. Pour ce faire, les auteurs ont défini un artefact constitué de plusieurs modèles de modèles (cf. figure 2.8), désigné *Software Development Process Model (SDPM)*.

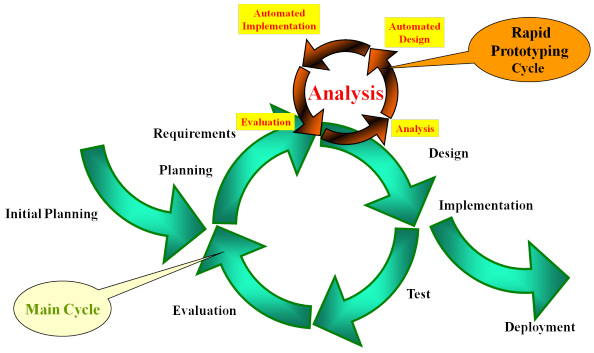


FIGURE 2.7: **Prinicpe de la méthode *Continuous Integration Unified Process***

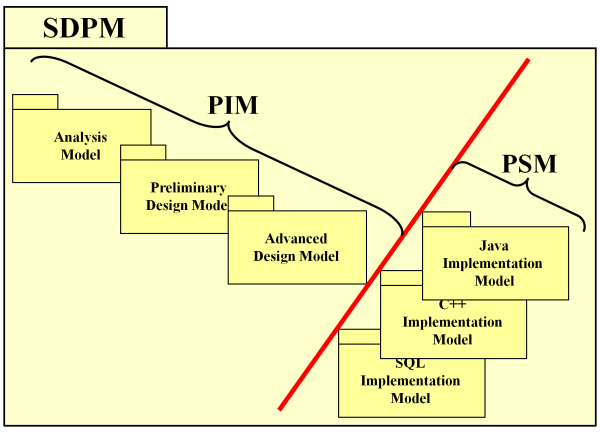


FIGURE 2.8: **L'artefact *Software Development Process Model***

Une transformation de clonage conditionnel (cf. figure 2.9) munie des propriétés de traçabilité réalise la diffusion des concepts entre phases de modèle. Une architecture de traçabilité "orthogonale" à la précédente, engendrée et gérée par la transformation de clonage, maintient les différentes phases du modèle en cohérence.

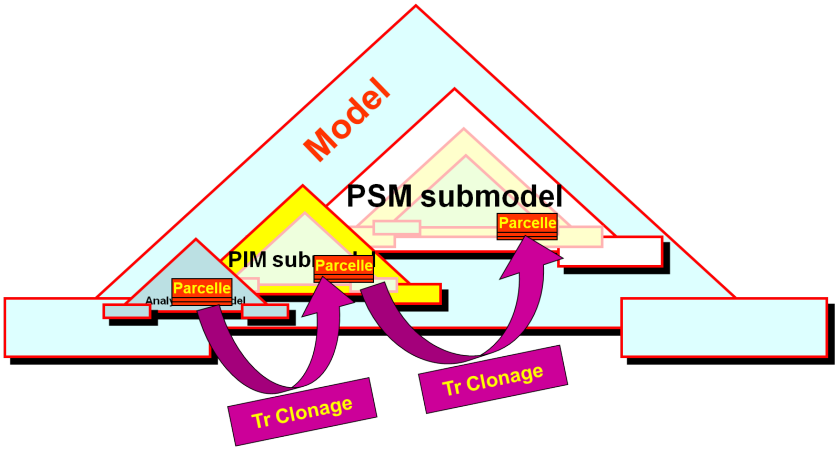


FIGURE 2.9: **Transformation de clonage**

Cette méthode est particulièrement attachée à l'amélioration de la qualité du modèle au cours de chaque étape du cycle de développement du système d'information. Elle repose sur l'artefact SDPM, en mettant l'importance sur la phase du développement. La transformation de clonage lui apporte une efficacité et les liens de traçabilité sous-jacents permettent d'avoir un lien constant sur les phases antérieures afin de suivre l'évolution de chaque concept thématique.

Dans cette thèse, nous ne définirons pas de nouvelle méthode, nous nous placerons dans le cadre de la méthode *Continuous Integration Unified Process Method*. Ce choix nous mettra en situation de suivre la qualité et l'évolution des modèles. Nous nous sommes également concentrés sur les qualités d'abstraction et de finesse de description du modèle car nous travaillerons en phase

d'analyse, moment crucial pour l'émergence des concepts de domaines, à tous niveaux, des plus concrets aux plus abstraits. Dans la section suivante nous nous intéresserons aux indicateurs de qualité et de suivi de l'évolution dans les modèles.

2.3 Indicateurs de qualité et de complexité d'un modèle

L'évolution d'un logiciel ou d'un système d'information peut se produire à des intervalles de temps aléatoire, générant de nouveaux cycles. Les travaux de recherche dans le domaine de l'évolution du logiciel sont principalement focalisés sur la minimisation des effets indésirables induits par un changement.

D'après [carpers Jones, 1991], l'obtention des mesures tôt dans le cycle de vie et la continuité de cette capture de mesures sont fondamentales. La mesure est essentielle pour l'analyse de l'évolution afin d'avoir une vision descriptive et globale du système d'information à tout instant et plus précisément dans le domaine de l'ingénierie [Zuse, 1998]. Une mesure (ou métrique) n'a de sens que si elle est accompagnée d'un attribut [Fenton et Pfleeger, 1991]. En général, ces métriques seront séparées en deux groupes : les métriques quantitatives et les métriques qualitatives. Les métriques quantitatives peuvent être mesurées automatiquement sur le modèle restructuré. Par contre, les métriques qualitatives nécessitent l'intervention d'un ou plusieurs experts.

Une classification des métriques, proposée par [Fenton et Pfleeger, 1991], est largement partagée dans le domaine de logiciel. Elle distingue trois types de mesure :

1. les métriques de processus sont liées aux activités.
2. les métriques de produits concernent les artefacts créés et modifiés.
3. les métriques de ressources mesurent les ressources nécessaires à chacune des activités du processus de développement.

Dans la littérature, nous observons que certaines métriques appartiennent à plusieurs de ces catégories. Néanmoins, elles participent à toutes les phases de développement de logiciel. En d'autres termes, elles mesurent l'évolution du logiciel en se concentrant sur sa fiabilité et sa capacité à répondre aux besoins définis par l'utilisateur [Fenton et Neil, 1999]. Actuellement, de plus en plus de résultats de recherche sur des mesures sur des diagrammes de classes sont rapportés dans la littérature. Ces mesures ont pour but d'étudier les caractéristiques des diagrammes de manière systématique. Avec [Yi *et al.*, 2004], l'analyse et la comparaison des mesures classiques sur les diagrammes de classes UML reflètent les différents types de relations existants, la complexité mais aussi la validation théorique et empirique de ces modèles. Ces auteurs définissent aussi les avantages et inconvénients de ces métriques ainsi que les problèmes existants.

2.3.1 Métriques sur la qualité des modèles

L'évaluation d'un modèle repose sur des métriques techniques et cognitives qui permettent d'évaluer l'effort réalisé dans la phase de développement mais aussi l'effort à produire pour la réutilisation et la maintenance.

Les outils commerciaux sont, au dire des auteurs, juste capables d'estimer une quantité limitée de métriques et donnent du système des vues spécifiques (propres). Un bon environnement de développement et de maintenance doit disposer d'outils pour définir et évaluer des métriques mais

aussi pour définir et présenter les diverses vues du système et de ses composants. Ces métriques peuvent être paramétrées pour contrôler l'évolution. Par exemple, plusieurs métriques permettent d'évaluer l'aspect quantitatif des relations entre les concepts thématiques d'un système d'information.

Nous rappelons ci-dessous quelques métriques parmi les plus connues.

1. La métrique *WCM* (Weighted Methods per Class) [Basili *et al.*, 1996] mesure le nombre de méthodes dans chaque classe (cf. pseudo-code en annexe A.2).
2. La métrique *NOC* (Number of Children) [Chidamber et Kemerer, 1994a] correspond au nombre de descendants directs pour chaque classe (cf. pseudo-code en annexe A.4).
3. La métrique *RFC* (Response For a Class) [Chidamber et Kemerer, 1994a] correspond au nombre d'échanges relatifs à la communication en comptabilisant les appels de méthodes (cf. pseudo-code en annexe A.6).
4. La métrique *LCOM* (Lack of Cohesion in Methods) [Chidamber et Kemerer, 1994a] mesure la relative "disparité" des méthodes d'une classe quant à l'utilisation des variables d'instance.
5. La métrique *CBO*(Coupling Between Object Classes) [Chidamber et Kemerer, 1994a] correspond à une mesure de couplage entre le nombre de classes couplées à une classe donnée (cf. pseudo-code en annexe A.5).
6. La métrique *DIT* (Depth in Inheritance Tree) [Chidamber et Kemerer, 1994b] mesure la longueur du chemin le plus long de la racine à une feuille de l'arbre d'héritage (cf. pseudo-code en annexe A.3).
7. La métrique *CNP* [Chidamber et Kemerer, 1994b] mesure le nombre de successeurs immédiats d'un nœud dans l'arbre d'héritage (les sous-classes directes d'un noeud).
8. La métrique *NIM* [Lorenz et Kidd, 1994] correspond au nombre de méthodes héritées.
9. La métrique *NOM* [Lorenz et Kidd, 1994] correspond au nombre de méthodes définies localement.
10. La métrique *NRM* [Lorenz et Kidd, 1994] correspond au nombre de méthodes de remplacement (non héritées, non spécialisées).
11. La métrique *SIX* [Lorenz et Kidd, 1994] mesure le rapport $(NRM\ timesDIT)/(NOM+NIM)$, globalement l'indice de spécialisation.

Un cadre global pour l'évaluation des schémas EER (Enhanced ENtity-Relationship) et schémas conceptuels UML est proposé dans [Cherfi *et al.*, 2003]. Les classes de mesures facilitent le processus d'évaluation et conduisent au choix de la représentation appropriée parmi plusieurs schémas décrivant la même réalité. Un sous-ensemble de critères pertinents pour l' évaluation conceptuelle de la qualité du schéma de EER sont définis. Pour chaque critère, un ou plusieurs paramètres permettent au concepteur de mesurer la qualité du schéma.

La qualité est un concept multidimensionnel englobant différentes sémantiques liées à des contextes différents. La qualité des systèmes d'information couvre plusieurs aspects et dimensions, y compris les dimensions humaines, technologiques et organisationnelles. Évaluer les dimensions d'un système d'information nécessite de prendre en compte respectivement la qualité des données ainsi que des modèles et la qualité des logiciels. Le processus d'évaluation est basé sur les objectifs,

les facteurs de qualité, les critères et les indicateurs associés, et enfin sur des modèles de qualité. Beaucoup de modèles et de cadres ont été proposés pour évaluer les données, les modèles et la qualité des logiciels. L'article [Comyn-Wattiau *et al.*, 2010] présente un état complet de l'art de ces approches, même s'il n'y a pas un cadre communément accepté.

On note que quelques métriques ont donc été proposées pour mesurer la qualité de l'héritage (qualité de la spécialisation et/ou de la généralisation des relations, de l'abstraction et la factorisation), sujet qui nous intéresse en priorité. PII(Pure Inheritance Index) [Miller *et al.*, 1999] est une variante qui indique si une classe préserve ce qui est hérité de ses super-classes. D'autres métriques s'intéressent à la succession des caractéristiques comme les méthodes et les attributs dans une hiérarchie de classes : MIF (Method Inheritance Factor) / AIF(Attribute Inheritance Factor) [e Abreu et Carapuça, 1994].

Lorsque ces mesures indiquent une utilisation intensive de l'héritage, intuitivement on peut dire que la complexité du modèle augmente. En contrepartie, l'utilisation intensive de l'héritage améliore le niveau d'abstraction du modèle et la possibilité de réutiliser des attributs et des méthodes dans les sous-classes.

2.3.2 Métriques sur la complexité et l'abstraction

Les évaluations empiriques [Briand *et al.*, 1996; Gyimóthy *et al.*, 2005] permettent d'estimer les défauts dans les logiciels et surtout la complexité. Les métriques essayent de prévoir le processus de développement du logiciel, de mesurer la productivité du développement et de quantifier la qualité du logiciel obtenu.

En partant des métriques précédemment présentées, de nouvelles métriques ont été introduites dans l'objectif de déterminer la complexité d'une classe dans une hiérarchie [Bansiya *et al.*, 1999] en phase de conception et d'implémentation :

1. **AMC** (Average Method Complexity) [McCabe, 1976] : cette métrique détermine la complexité d'une classe en faisant la moyenne des complexités sur toutes les méthodes de la classe (cf. equations 2.1 et 2.2).

$$\left(AMC_{Classe} = \frac{\sum_1^n CI}{n} \right) \tag{2.1}$$

$$\left(AMC_{Classe} = \frac{WMC_{Classe}}{n} \right) \tag{2.2}$$

2. **CDE** (Class Design Entropy) : cette métrique mesure la complexité d'une classe en mesurant l'entropie de l'information qu'elle comporte. Pour cela on regarde la quantité d'information

des chaînes de caractères contenues dans la classe et désignant des objets, des opérations, des attributs, et des noms d'associations. Elle est calculée à partir de l'équation 2.3) [3].

$$\left(CDE_{Classe} = \sum_{1}^{n1} CI((\frac{fi}{N1}) \times (log_2(\frac{fi}{N1})) \right) \tag{2.3}$$

Dans [Dao *et al.*, 2002], des métriques sont proposées pour mesurer précisément la qualité de la prise en compte des hiérarchies d'héritage par référence à une hiérarchie idéalement factorisée qui est obtenue par la construction d'une sous-hiérarchie de Galois associée à la hiérarchie de classes. Elle est calculée sur le code Java. Ici, nous préférerons observer les catégories de concepts formels dans des treillis (concepts fusionnés et de nouveaux concepts) car ils sont immédiatement compréhensibles par l'expert informatique qui peut en outre exploiter les concepts résultants pour construire de nouvelles classes ou associations dans son modèle.

2.3.3 Métriques sur l'analyse de l'évolution

Les métriques logicielles sont souvent incluses dans des frameworks destinés à analyser l'évolution des logiciels. Par exemple, l'apport original de Lanza et de Ducasse [Lanza et Ducasse, 2002] est de proposer une *matrice d'évolution* dans laquelle chaque classe correspond à une ligne, chaque version de logiciel à une colonne, mettant en valeur l'apparition et la disparition des classes. La taille des classes est représentée par une zone de taille variable représentant la classe dans une version spécifique. Plusieurs types d'évolution se dégagent (croissance, stabilisation, etc.) ainsi que le comportement spécifique des classes (*par exemple des grandes classes permanentes*).

[Pinzger *et al.*, 2005] utilise un diagramme Kiviat pour visualiser des métriques à travers plusieurs versions. Les techniques de visualisation peuvent être utilisées pour enrichir l'outillage de notre proposition.

Dans la littérature, d'autres approches utilisent l'information sur les versions [Gall *et al.*, 1999] ou sur des changements dans le code [Nagappan et Ball, 2005] plutôt que d'évaluer les paramètres directement sur les artefacts logiciels.

Dans [Kpodjedo *et al.*, 2011], les auteurs proposent une approche heuristique basée sur une méta-heuristique pour déterminer une distance d'édition de coût minimal entre deux versions successives d'un modèle de classes. Cette distance compte les modifications introduites entre les versions et elle est comparée avec des métriques traditionnelles de prédiction de défaillances.

Les métriques sur l'analyse de l'évolution sont principalement utilisées pour prédire les activités ou les initiatives telles que l'effort de maintenance, la prédisposition aux erreurs ou le taux d'erreur en génie logiciel.

[Alshayeb et Li, 2003] proposent une approche basée sur un double processus itératif, d'une part, un processus à court terme et, d'autre part, un processus à long terme. Le constat est que les mesures proposées dans cette approche sont efficaces pour prédire les efforts de conception et le suivi des lignes de code source ajoutées, modifiées et supprimées dans le processus à court terme.

3. N1 est le nombre de chaînes distinctes, N1 est le nombre de chaînes non distinctes et fi ($0 \leq fi \leq n$) est la fréquence des occurrences de la i ème chaîne.

Par contre elles sont inefficaces pour prédire les mêmes aspects dans le processus à long terme. Cela nous amène à penser que la capacité de prédiction de ces mesures est limitée à la conception et à la mise en œuvre des changements au cours des itérations de développement. Ces mesures ne sont pas adaptées à la capacité de prédiction à long terme de l'évolution d'un système au travers différentes versions.

2.3.4 Métriques sur les bases de données

Dans le domaine des bases de données, plusieurs approches ont été proposées pour gérer l'évolution du schéma des bases de données orientées objet [Monk et Sc, 1993] et [Skarra *et al.*, 1986]. Ces auteurs définissent principalement les transformations utilisées pour l'évolution du schéma conceptuel de la base de données. Une évolution combine un ensemble de transformations primitives ainsi que des règles globales utilisées pour valider ou invalider l'évolution.

2.3.5 Métriques quantitatives et qualificatives basées sur l'AFC et l'ARC

Les métriques utilisées sont essentiellement inspirées de celles basées sur l'Analyse Formelle de Concepts [Falleri, 2009]. Elles sont calculées sur les treillis générés par l'Analyse Formelle de Concepts et elles permettent d'une part, d'évaluer le niveau d'abstraction de modèle de manière quantitative et, d'autre part, de définir la qualité de la factorisation réalisée. Pour ce dernier point, le jugement d'un ou de plusieurs experts du domaine est souvent requis. Ces métriques sont définies comme suit :

Métriques quantitatives

1. La métrique NAC mesure le nombre de classes ajoutées.
2. La métrique NFC mesure le nombre de classes fusionnées.
3. La métrique NAA mesure le nombre d'attributs ajoutés.
4. La métrique NFA mesure le nombre d'attributs fusionnés.

Métriques qualitatives

5. La métrique NACR mesure le taux de classes correctes ajoutées.
6. La métrique NFCR mesure le taux de classes correctes fusionnées.
7. La métrique NAAR mesure le taux d'attributs corrects ajoutés.
8. La métrique NFAR mesure le taux d'attributs corrects fusionnés.

Ces métriques sont issues d'une étude récente [Falleri, 2009] qui compare trois stratégies d'application de l'Analyse Formelle de Concepts au logiciel open-source Java Salomé-TMF software [4] (gestion des tests, 37 classes, 142 attributs). Les trois stratégies sont désignées par AFC-NAME, AFCC-NAME et ARC-NAME. Nous les reverrons dans le chapitre suivant.

4. http://wiki.ow2.org/salome-tmf/

2.4 Conclusion

Nous avons présenté dans ce chapitre d'une part, un corpus de concepts, de méthodes et d'outils, mais aussi de langages pour créer, organiser, restructurer et transformer des modèles dans un cadre d'Ingénierie Dirigée par les Modèles (IDM) [Kent, 2002].

D'autre part, nous avons évoqué les principales métriques dans différents domaines liés au développement d'applications. Nous avons montré l'importance de la mesure pour l'analyse d'une application dans toutes les phases de son cycle de vie. Pour prendre en compte l'évolution lors de la phase d'analyse, le concepteur est amené à proposer de nouvelles abstractions en se basant sur des mesures afin de maintenir ou d'augmenter la qualité du logiciel. Les méthodes et mesures ci-dessus sont autant de possibilités d'analyser le suivi de l'évolution des systèmes d'information au cours de leur cycle de vie.

Dans nos travaux, nous voulons comprendre l'évolution des principales phases. Pour ce faire, nous utilisons les informations relatives aux éléments de modélisation dans une première partie des travaux, ce qui correspond à certaines des métriques existantes. De plus, comme l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts sont comme des modèles normalisés, nous avons exploré la manière dont elles peuvent aider le concepteur à analyser le manque de détails ou l'absence d'abstraction dans le modèle.

Dans nos recherches, une série de métriques sera proposée pour l'observation et la compréhension de l'évolution. Nous interprèterons les résultats par rapport à l'histoire connue et nous établirons les premières recommandations pour suivre une évolution du modèle. Nous présentons dans le chapitre 5, une méthode d'analyse de l'évolution à travers des métriques d'une part, sur les éléments de modélisation du modèle UML et, d'autre part, sur les treillis générés par l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts (cf. chapitre 3). Cette méthode permet l'étude de l'évolution des modèles. L'application systématique aux quinze versions du modèle *SIE-Pesticides* de ces deux approches nous a permis d'extraire des métriques et plusieurs recommandations qui constituent un début d'une méthode d'analyse.

Nous verrons comment mesurer les principaux éléments de modélisation (classes, attributs, méthodes, associations, rôles) donnera des indications sur le degré de réification du modèle (combien de concepts thématiques du domaine sont capturés ?), sur la description des concepts (méthodes et attributs) mais aussi sur le nombre de relations (associations) entre concepts thématiques.

Les indicateurs construits sur des treillis permettront d'évaluer le degré de réification du modèle mais aussi d'analyser plus en profondeur le niveau d'abstraction des concepts thématiques réifiés (mesure des nouveaux concepts) ainsi que leur structuration (via les treillis) et leur description (mesure des concepts fusionnés).

CHAPITRE

3 Analyse Formelle de Concepts et Analyse Relationnelle de Concepts

Préambule

L'approche de factorisation de modèle et de suivi et d'analyse de l'évolution de modèles que nous présentons dans ce manuscrit se base sur l'Analyse Formelle de Concepts et sur une de ses extensions, l'Analyse Relationnelle de Concepts. Ce chapitre se consacre à la présentation de ces deux théories.

Sommaire

3.1 Introduction

l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts permettent de construire des abstractions à partir d'une ou plusieurs relations binaires entre un ensemble d'objets et un ensemble de caractéristiques. Les abstractions construites sont des concepts formels formés d'un ensemble d'objets partageant un ensemble maximum de caractéristiques. L'Analyse Relationnelle de Concepts étend l'Analyse Formelle de Concepts en utilisant non seulement les caractéristiques mais aussi les relations entre les objets.

Les concepts sont groupés dans des treillis, appelés treillis de concepts ou treillis de Galois, par référence à la notion de correspondance de Galois mise en évidence en 1940 par le mathématicien Birkhoff. Ensuite, d'autres travaux français ont été effectués sur des analyses de questionnaires [Barbut et Monjardet, 1970] et enfin, l'Analyse Formelle de Concepts [Wille, 1982a] a été formalisée par le mathématicien allemand R. Wille [1]. Cette dernière est étendue de nombreuses manières, dont l'une est l'Analyse Relationnelle de Concepts[Hacene *et al.*, 2013].

Dans notre contexte, nous appliquerons ces approches sur des modèles de classes en nous en servant comme d'une méthode de transformation de modèles.

3.2 L'Analyse Formelle de Concepts

L'Analyse Formelle de Concepts [Ganter et Wille, 1999] est une méthode d'analyse de données basée sur la théorie des treillis [Birkhoff, 1940] et des treillis de Galois [Barbut et Monjardet, 1970], qui permet de faire émerger un ensemble de concepts par abstraction d'un ensemble d'objets (appelés parfois entités ou individus) décrits par des attributs (appelés parfois caractéristiques). Elle a été utilisée dans de nombreuses applications ayant trait ou se basant sur de la classification, telles que la structuration de connaissances, la recherche d'informations, l'extraction de règles, ou la recherche de motifs communs.

L'Analyse Formelle de concepts a comme données d'entrée des objets, décrits par des caractéristiques. Cette description se présente sous forme d'une table binaire nommée *contexte formel*.

Définition 3.1 (Contexte Formel) *Un contexte formel est un triplet $K = (O, A, R)$, où O et A sont respectivement l'ensemble des objets (ou entités) et l'ensemble des attributs (ou caractéristiques) qui décrivent les objets. $R \subseteq O \times A$ est une relation exprimant que $\forall (o, a) \in R$, a est un attribut de l'objet o.*

Un contexte peut être représenté graphiquement par un tableau de dimension $|O| \times |A|$. Un exemple de contexte formel est donné à la figure 3.1, dans un tableau qui a dû être scindé en deux parties pour rentrer dans la page. Ce contexte décrit des pizzas (les objets) par les ingrédients qu'elles contiennent ou les caractéritiques de la pizza (les attributs). Dans la table, les objets sont en ligne, et les attributs en colonne. Le fait qu'un objet possède un attribut (et donc qu'une pizza contienne un ingrédient ou possède une caractéristique) est dénoté par une croix dans la case correspondant à la ligne de l'objet et la colonne de l'attribut. On voit par exemple que la pizza lorraine est à pâte fine, et contient de la crème, des oignons, de l'emmental, et du bacon.

1. http://www.upriss.org.uk/fca/fca.html

has-topping	thin	thick	calzone	tomato-sauce	cream	tomato	basilic	olive	olive oil	soy	mushroom	eggplant	onion	pepper	ananas
okonomi			×	×					×	×	×				
alberginia		×		×					×	×		×	×		
margherita	×			×		×	×	×	×						
languedoc	×			×		×	×	×	×				×	×	
four-cheeses	×				×										
three-cheeses	×				×										
frutti-di-mare	×			×				×	×						
quebec		×		×											
regina	×			×							×				
hawai		×		×											×
lorraine	×				×								×		
kebab			×	×		×		×					×		

has-topping	mozza	goat-cheese	emmental	fourme-ambert	squid	shrimp	mussels	ham	bacon	chicken	maple-sirup	corn
okonomi												
alberginia												
margherita	×											
languedoc	×											
four-cheeses	×	×	×	×								
three-cheeses	×	×	×									
frutti-di-mare	×				×	×	×					
quebec	×							×			×	×
regina	×								×			
hawai	×							×				
lorraine			×						×			
kebab			×							×		

TABLE 3.1: **Contexte formel pour l'exemple de pizzas**

On note f et g deux applications caractéristiques de la relation R :

- $f : \mathscr{P}(O) \to \mathscr{P}(A)$
  $X \longmapsto \{y \in A \mid \forall x \in X, (x, y) \in R\}$
- $g : \mathscr{P}(A) \to \mathscr{P}(O)$
  $Y \longmapsto \{x \in O \mid \forall y \in Y, (x, y) \in R\}$

À partir d'un contexte formel, l'Analyse Formelle de Concepts va grouper les objets possédant des attributs communs dans des concepts formels. Un concept formel est défini comme suit.

Définition 3.2 (Concept) *Un concept est une paire $C = (X, Y)$ avec $X \subseteq O$, $Y \subseteq A$ et où $X = \{o \in O | \forall y \in Y, (o, y) \in R\}$ est l'extension (objets couverts), notée $Ext(C)$ et $Y = \{a \in A | \forall x \in X, (x, a) \in R\}$ est l'intension (attributs partagés), notée $Int(C)$.*

Définition 3.3 (Extension et intension) *L'extension (respectivement l'intension) d'un concept (X, Y) est l'ensemble X (resp. Y) des objets couverts (respectivement des attributs partagés) par ce concept.*

Graphiquement, dans le contexte formel, un concept peut être vu comme un pavé de croix de taille maximale, à l'ordre près des lignes et des colonnes. Si l'on reprend l'exemple des pizzas, on

peut par exemple trouver le concept des pizzas à pâte fine et contenant du bacon. Ce concept a pour extension $lorraine, regina$ et pour intension $thin, bacon$.

L'extension et l'intension d'un concept vérifient les propriétés suivantes : pour tout concept $C = (X, Y)$, $Y = f(X)$ et $X = g(Y)$.

L'Analyse Formelle de Concepts ordonne les concepts dans un treillis de concepts, aussi appelé treillis de Galois.

Définition 3.4 (Treillis de concepts) *Le treillis de concepts $\mathscr{L}$ se définit comme l'ensemble des concepts muni de l'ordre partiel suivant : c_1 spécialise c_2 ($c_1 \leq_{\mathscr{L}} c_2$) si l'extension de c_1 est incluse dans celle de c_2 et inversement, l'intension de c_2 est incluse dans celle de c_1). $\mathscr{L}$ possède au plus $2^{min(|O|,|A|)}$ concepts.*

On représente souvent un treillis de concepts par un diagramme de Hasse. Il est courant de représenter dans ce diagramme les concepts simplifiés sur les attributs et sur les objets. Cette simplification consiste à retirer des étiquettes des concepts les objets qui apparaissent dans les sous-concepts et les attributs qui apparaissent dans les super-concepts et se définit comme suit.

Définition 3.5 (Concept simplifié sur les attributs) *Soit $C = (X, Y)$ un concept du treillis de concepts , le concept associé par simplification des propriétés est noté (X, Y_s) avec $Y_s = Y \setminus herit(Y)$ où $herit(Y) = \{y \in Y \mid \exists C' \in \mathscr{L}, C \leq_{\mathscr{L}} C', y \in Int(C')\}$ La simplification selon les objets est définie de manière symétrique. On nommera treillis simplifié les représentations de treillis comportant l'intension et l'extension simplifiées des concepts.*

Pour illustrer ces notions, reprenons l'exemple des pizzas. Le treillis de concepts simplifiés est présenté à la figure 3.1. Dans ce treillis, les concepts sont représentés par des rectangles, dont la partie supérieure contient le nom, la partie centrale contient l'intension simplifiée, et la partie du bas contient l'extension simplifiée. Le concept des pizzas à pâte fine et contenant du bacon est nommé `Concept_36`. Son intension se compose de `bacon` (ici représenté dans l'intension simplifiée du Concept_36) et de `thin`, qui est un attribut hérité du concept `Concept_1`. Son extension simplifiée est vide, et son extension se compose de `lorraine`, objet hérité du concept `Concept_20` et de `regina`, hérité du concept `Concept_17`.

Dans un treillis de concepts, on trouve deux concepts particuliers : *Top*, qui est l'infimum du treillis, et *Bottom* qui est le supremum du treillis. Dans notre exemple de pizzas, le *Bottom* est le concept `Concept_2`. Son intension simplifiée est l'ensemble de toutes les caractéristiques des pizzas, et son extension est vide (aucune pizza ne possède toutes les caractéristiques décrites). Le *Top* est le concept `Concept_0`, son extension est l'ensemble des pizzas décrites, et son intension est vide (aucune caractéristique n'est partagée par toutes les pizzas).

3.3 Analyse Relationnelle de Concepts

L'Analyse Relationnelle de Concepts est une extension de l'Analyse Formelle de Concepts, qui prend en compte plusieurs types d'objets décrits par des attributs, et qui entretiennent des relations avec d'autres objets.

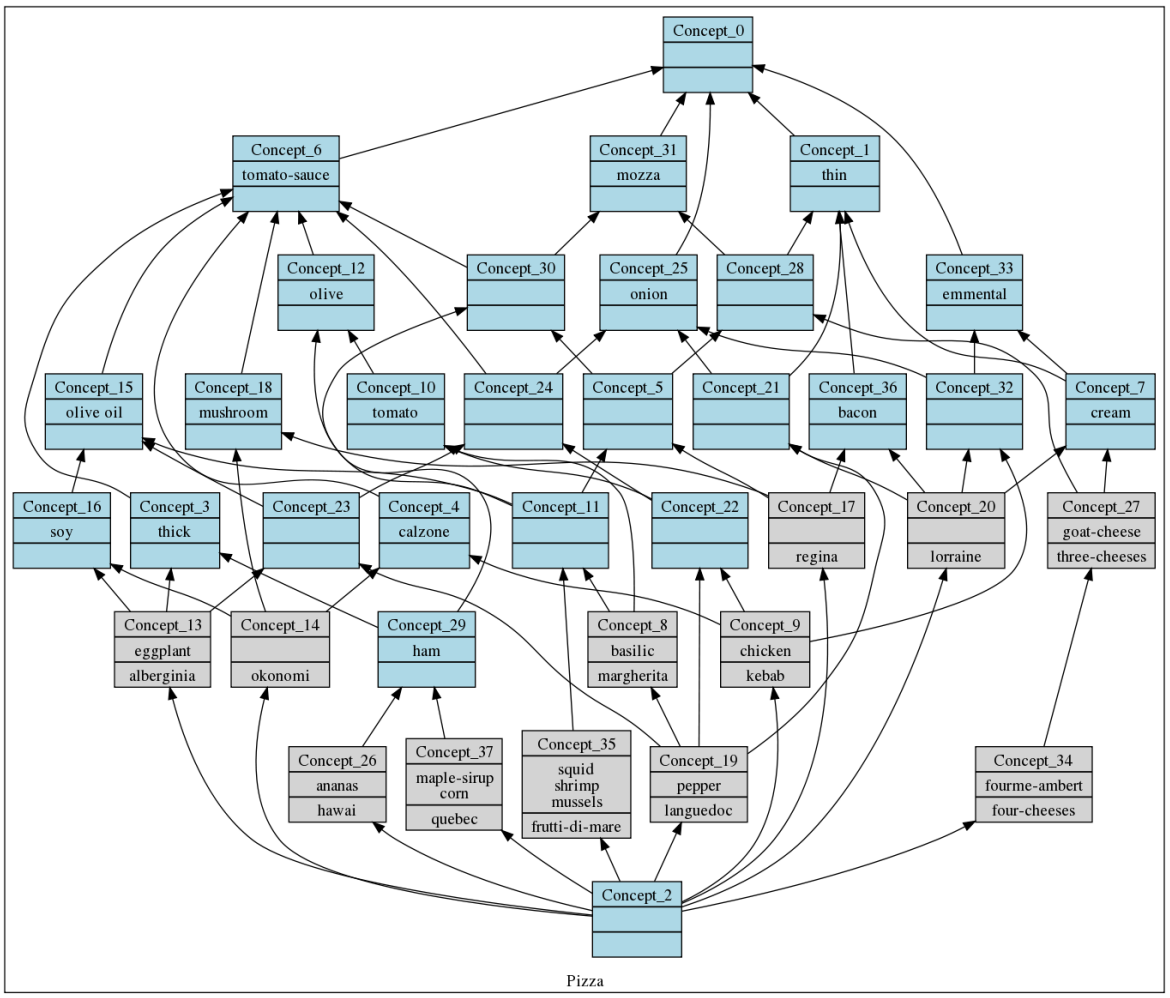


FIGURE 3.1: **Treillis des concepts simplifiés, issu du contexte de la figure 3.1**

L'Analyse Relationnelle de Concepts va donc opérer non pas sur un contexte formel, mais sur plusieurs contextes formels (un par type d'objets étudié) et sur des contextes dits relationnels, qui représentent les relations pouvant exister entre les objets de différents types. Cet ensemble de contextes sur lequel l'ARC travaille est nommé Famille de Contextes Relationnels, et est défini comme suit.

Définition 3.6 (Famille de Contextes Relationnels) *Une famille de Contextes Relationnels FCR est un couple* $(\mathcal{K}, \mathcal{R})$*.* $\mathcal{K}$ *est un ensemble de contextes formels* $K_i = (O_i, A_i, R_i)$*, et* $\mathcal{R}$ *est un ensemble de contextes relationnels* $R_j = (O_k, O_l, J_j)$ *(*O_k *et* O_l *sont les ensembles d'objets des contextes* K_k *et* K_l *de* $\mathcal{K}$*).*

L'ARC fonctionne par itération de FCA sur les contextes formels. A chaque itération, les concepts découverts pour chaque type d'objets sont propagés le long des relations aux autres types d'objets,

lors d'une phase dite de *scaling*. Plus formellement, à une itération donnée, chaque contexte formel $K_i = (O_i, A_i, R_i)$ est étendu par les contextes formels $K_{i,l}$ construits en utilisant tous les contextes relationnels $R_l = (O_i, O_j, J_l)$ et en considérant, pour chaque j, le treillis de concepts L_j calculé à l'itération précédente à partir du contexte formel $K_j = (O_j, A_j, I_j)$ (adéquatement étendu) et un opérateur de scaling S.

Un opérateur de scaling est une fonction booléenne $S(o, c, J)$, où o est un objet, c un concept et J une relation d'incidence. Pour chaque R_l, un contexte formel $K_{i,l} = (O_i, \{R_l\} \times L_j, M_l)$ sera créé avec $M_l = \{(o, (R_l, c)) | o \in O_i, c \in L_j, S(o, c, J_l)\}$. Cela signifie qu'une relation entre types d'objets est transformée en remplaçant les objets en colonne par les concepts du treillis construit à l'itération précédente sur ces objets, et l'opérateur de scaling définit comment les objets de O_i sont reliés aux concepts de L_j. Un exemple d'opérateur de scaling, qui est celui que nous avons utilisé dans la suite de nos travaux, est l'opérateur appelé existentiel, qui est vrai si et seulement si $((\{o\} \times extent(c)) \cap J) \neq \emptyset$, de telle manière qu'une relation entre un objet et un concept est établie dans le nouveau contexte formel (après scaling) quand l'objet est en relation avec au moins un objet de l'extension du concept.

Le processus itère jusqu'à l'obtention d'un point fixe, c'est-à-dire lorsque d'une étape à l'autre, aucun nouveau concept n'est découvert.

Nous allons illustrer RCA en utilisant l'exemple de pizzas, légèrement modifié. Nous prenons en compte deux types d'objets : les pizzas, et les aliments. Les pizzas ont pour attributs leurs caractéristiques (calzone, pâte fine, pâte épaisse). Les aliments ont pour attributs les caractéristiques viande, poisson, laitage, etc. On prend en compte la relation entre pizza et aliment : a pour garniture (has topping).

TABLE 3.2: **Contexte formel des pizzas**

Pizza	**thin**	**thick**	**calzone**
okonomi			×
alberginia		×	
margherita	×		
languedoc	×		
four-cheeses	×		
three-cheeses	×		
frutti-di-mare	×		
quebec		×	
regina	×		
hawai		×	
lorraine	×		
kebab			×

RCA va commencer par appliquer FCA sur chaque contexte formel, et va ainsi produire les treillis donnés à la figure 3.2.

Nous voyons apparaître dans le treillis des ingrédients une classification naturelle des aliments en concepts : aucun aliment ne possède plus d'une caractéristique, aussi le treillis est assez plat. Dans le treillis des pizzas, nous avons une classification des pizzas par caractéristique, par exemple, on voit apparaître le concept des pizzas à pâte fine, dont l'extension est assez grosse. En revanche, comme aucune information relationnelle n'a encore été utilisée, on ne trouve pas le concept des pizzas à pâte fine et avec un laitage. Ce concept sera découvert lors des itérations suivantes, en prenant en compte l'information relationnelle. Nous donnons à la figure 3.3 le résultat de RCA sur

TABLE 3.3: **Contexte formel des ingrédients**

Ingredient	fruit-vegetable	meat	fish	dairy	cereal-leguminous	veg-oil
tomato-sauce	×					
cream				×		
tomato	×					
basilic	×					
olive	×					
olive oil						×
soy	×					
mushroom	×					
eggplant	×					
onion	×					
pepper	×					
ananas	×					
mozza				×		
goat-cheese				×		
emmental				×		
fourme-ambert				×		
squid			×			
shrimp			×			
mussels			×			
ham		×				
bacon		×				
chicken		×				
maple-sirup	×					
corn					×	

cet exemple, avec l'opérateur de scaling existentiel. On y voit le concept des pizzas à pâte fine et avec un laitage : il s'agit du concept `Concept_1`. L'attribut `thin` est présent dans l'intension simplifiée, et l'attribut relationnel `has-topping:Concept_10` est hérité du concept `Concept_21`. L'attribut relationnel `has-topping:Concept_10` se réfère au concept `Concept_10` du treillis des aliments, qui correspond au concept des laitages. Notons ici que le scaling existentiel a été utilisé : le concept `Concept_1` correspond aux pizzas à pâte fine et ayant `au moins un` ingrédient qui est un laitage (il existe un ingrédient qui est un laitage). En utilisant d'autres opérateurs de scaling, comme le scaling universel, on verrait plutôt apparaître le concept de pizza dont tous les ingrédients sont des laitages (ici aucune pizza n'a cette caractéristique).

3.4 Description des formes normales

Les méthodologies proposées pour la conception des bases de données relationnelles incluent une étape de normalisation [Codd, 1970]. Cette étape utilise "un schéma universel" où les dépendances fonctionnelles prennent en compte tous les attributs et les contraintes sémantiques. Elle vise l'élimination de redondances d'attribut et de dépendance. Les schémas obtenus ont la même sémantique, dans une forme réduite et facile à manipuler. Cela permet d'éviter des échecs pendant le chargement des données dans la base de données. Nous transposons ce principe aux diagrammes de classes d'UML des systèmes d'information.

De même que l'Analyse Formelle de Concepts [Godin et Valtchev, 2005], l'Analyse Relationnelle de Concepts peut être employée pour produire des formes normales pour les modèles d'UML, en

TABLE 3.4: **Contexte relationnel entre les pizzas et les ingrédients pour la relation *a pour garniture* (*has topping*)**

has-topping	tomato-sauce	cream	tomato	basilic	olive	olive oil	soy	mushroom	eggplant	onion	pepper	ananas
okonomi	×					×	×	×				
alberginia	×					×	×		×	×		
margherita	×		×	×	×	×						
languedoc	×		×	×	×	×				×	×	
four-cheeses		×										
three-cheeses		×										
frutti-di-mare	×				×	×						
quebec	×											
regina	×							×				
hawai	×											×
lorraine		×								×		
kebab	×		×		×					×		

has-topping	mozza	goat-cheese	emmental	fourme-ambert	squid	shrimp	mussels	ham	bacon	chicken	maple-sirup	corn
okonomi												
alberginia												
margherita	×											
languedoc	×											
four-cheeses	×	×	×	×								
three-cheeses	×	×	×									
frutti-di-mare	×				×	×	×					
quebec	×							×			×	×
regina	×								×			
hawai	×							×				
lorraine			×						×			
kebab			×							×		

éliminant des redondances, en ajoutant tous les liens implicites de spécialisation et en mettant à jour des abstractions appropriées. Nous illustrons ces caractéristiques sur le modèle du schéma (cf. figure 4.3) qui contient des redondances d'attribut (par exemple **DeviceType**) et des associations qui méritent d'être généralisées (par exemple **Groundwater Monitoring** et **Rainfall Monitoring**).

3.5 Application de l'AFC et l'ARC dans le domaine du génie logiciel

L'AFC et dans une moindre mesure l'ARC ont eu de nombreuses applications dans le domaine du génie logiciel. Nous dressons ici un panorama de ces applications, en distinguant les applications ayant trait à la factorisation de hiérarchies de classes des autres applications. L'idée n'est pas ici de lister de manière exhaustive les applications, mais de donner une idée des applications possibles. Nous terminons cette partie par une analyse des différentes expérimentations ayant été menées sur des modèles de classes.

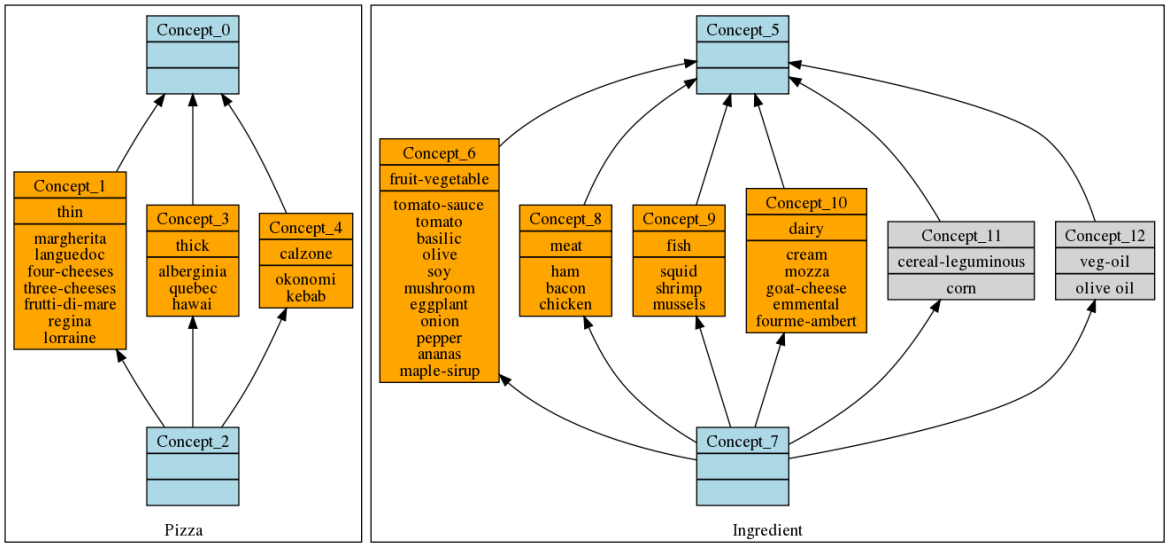


FIGURE 3.2: **Treillis obtenus lors de l'étape d'initialisation de RCA pour l'exemple des pizzas (contextes des tables 3.2 et 3.3)**

3.5.1 Différentes applications de l'AFC et l'ARC en Génie Logiciel

Les applications de l'AFC et de l'ARC ont généralement pour objectif l'amélioration de la qualité de différents artéfacts d'un système.

Snelting *et al* [Snelting, 1998] proposent une approche permettant d'émettre des recommandations pour améliorer la qualité d'un système orienté-objet, comme la suppression de certains attributs, ou la remodularisation. Les recommandations portent sur le modèle de classes du système. L'idée utilisée ici est de s'intéresser aux utilisations du modèle de classe par des programmes. L'AFC porte donc sur des programmes orientés objets, en prenant en compte les définitions et utilisation de variables des différents artéfacts du programme. L'étude des concepts du treillis complet se fait avec des moyens algébriques, et permet entre autres de donner des indications sur la création de composants logiciels, et de la modularisation du code existant.

Siff *et al* [Siff et Reps, 1999] utilisent l'AFC pour identifier des modules dans une application existante. L'Analyse Formelle de Concepts est utilisée pour construire un treillis de concepts, où chaque concept représente un module potentiel. Une nouvelle notion de partition du concept est introduite pour découvrir toutes les partitions du concept d'un treillis de concept donné, afin de déterminer une modularisation pertinente.

Caprile et al [Caprile et Tonella, 1999] utilisent l'AFC pour analyser les structures lexicales, syntaxiques et sémantiques d'un programme. Les identifiants choisis par les programmeurs comme les noms de fonctions contiennent des informations précieuses. Ils sont souvent le point de départ pour les activités de compréhension du programme. L'analyse des structures lexicales, syntaxiques et sémantiques des identifiants, des noms de fonctions peut permettre une meilleure compréhension du programme, et peut apporter une aide lors de son évolution. L'approche proposée par les

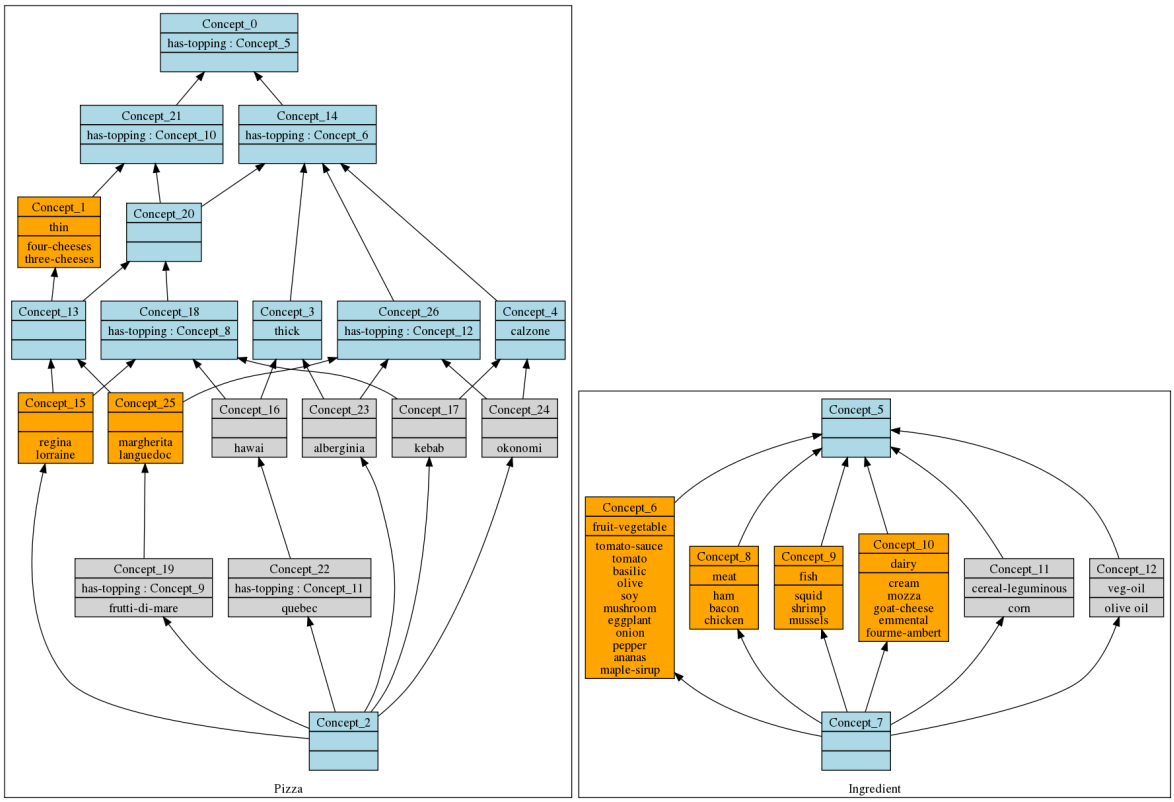


FIGURE 3.3: **Résultat de l'application de RCA avec l'opérateur de scaling exists**

auteurs se base sur une technique de segmentation, puis sur l'application de l'AFC pour classifier les identifiants en fonction des noms le composant.

Moha et al [Moha, 2008; Moha *et al.*, 2008a,b] utilisent l'AFC et l'ARC pour suggérer des refactorisations afin de corriger certains défauts dans les systèmes orientés objets. La méthode DECOR proposée par les auteurs est une approche de détection et de correction semi-automatiques pour faciliter les phases de maintenance et d'évolution. Cette méthode permet de spécifier des règles de détection à un haut niveau d'abstraction et de suggérer des restructurations de code afin d'automatiser la correction des défauts. Cette méthode utilise l'Analyse Relationnelle de Concepts. L'utilisation de RCA sur un modèle de code pertinent permet d'y détecter certains anti-patrons comme le blob, et d'y apporter des corrections (par exemple, dans le cas du blob, en identifiant des sous-parties cohésives pour scinder la classe Blob en plusieurs parties).

L'ARC a également été utilisée dans le domaine des applications orientées services, afin de classifier les services web d'un annuaire. L'approche proposée [Azmeh *et al.*, 2008] classifie les services web en fonction des différents éléments qu'ils contiennent (en termes d'opérations, de paramètres, etc) et propose une approche permettant également une recherche dans le treillis des services ainsi construit.

3.5.2 Utilisation de l'AFC et de l'ARC pour la restructuration de modèles de classes

L'utilisation de l'Analyse Formelle de Concepts dans le domaine de la factorisation de modèle de classes a une longue et riche histoire dans la littérature scientifique. Cette problématique a été introduite par Godin *et al.* [Godin et Mili, 1993] pour extraire des interfaces abstraites à partir d'une hiérarchie de classes Smalltalk. Des extensions de ces travaux ont été publiées dans [Godin *et al.*, 1998]. D'autres approches ont été proposées, qui prennent en compte plus d'informations extraites des codes sources comme les signatures de méthode dans [Dicky *et al.*, 1996]. Des approches avec des objectifs et des techniques similaires ont également été proposées pour refactoriser des systèmes de base de données Orientée Objet [Missikoff et Scholl, 1989; Rundensteiner, 1992; Yahia *et al.*, 1996]. Des approches plus récentes peuvent être trouvées dans la littérature [Dao *et al.*, 2004a; Cyril Roume, 2004; Hacène, 2005; Falleri *et al.*, 2008; Falleri, 2009], qui sont en quelque sorte les sources du travail présenté dans ce manuscrit. Ces approches ont toutes pour objectif la refactorisation de modèles de classes avec FCA et RCA. Ces approches sont abordées plus longuement dans la section suivante dans la mesure où chacune a été expérimentée sur différents modèles. Nous reportons dans la section suivante un bilan de ces expérimentations.

Nous retenons trois grandes caractéristiques de l'AFC et l'ARC qui ont mené à leur application dans le cadre de la factorisation de modèles de classes et qui sont illustrés Figure 3.4 :

1. La factorisation maximale des attributs dans les concepts, ce qui permet dans le domaine des modèles de classes la suppression des redondances.
2. Le respect de la relation de spécialisation : la relation d'ordre dans le treillis de concepts correspond bien à la relation d'héritage présente dans les modèles de classes.
3. La minimalité du nombre de concepts introduits : lors de l'application de FCA/RCA pour la refactorisation, il y a une garantie que le nombre de concepts (i.e. de classes) à introduire pour supprimer les redondances est minimal.

3.5.3 Expérimentations menées dans le cadre de l'application de FCA/RCA pour la factorisation de modèles de classes

3.5.3.1 Expérimentations sur un modèle de classes de *France Télécom (Orange)*

Le premier cas d'application pratique de l'Analyse Relationnelle de Concepts est rapporté dans [Dao *et al.*, 2004a]. Il a été mené sur plusieurs modèles de classes de taille moyenne dans le cadre d'un projet RNTL commun soutenu par le ministère français de la recherche et de l'industrie. La configuration choisie pour l'application de l'ARC était composée de classes, méthodes, attributs et d'associations.

Les classes n'ont aucune description (non relationnelle) ; les attributs sont décrits par leur nom, la multiplicité et la valeur initiale ; les méthodes sont décrites par le nom et le corps de la méthode, les associations sont décrites par le nom, le rôle et de l'information sur l'origine et la destination comme multiplicité et la navigabilité. Les associations sont décrites par les classes (origine et destination), les classes sont décrites par les attributs et les opérations qui leur appartiennent. De même les attributs des classes ont leur type, etc.

Les modèles contiennent quelques dizaines de classes et les nouveaux concepts qui seront examinés par des experts se complexifient jusqu'à atteindre plusieurs centaines concepts dans les

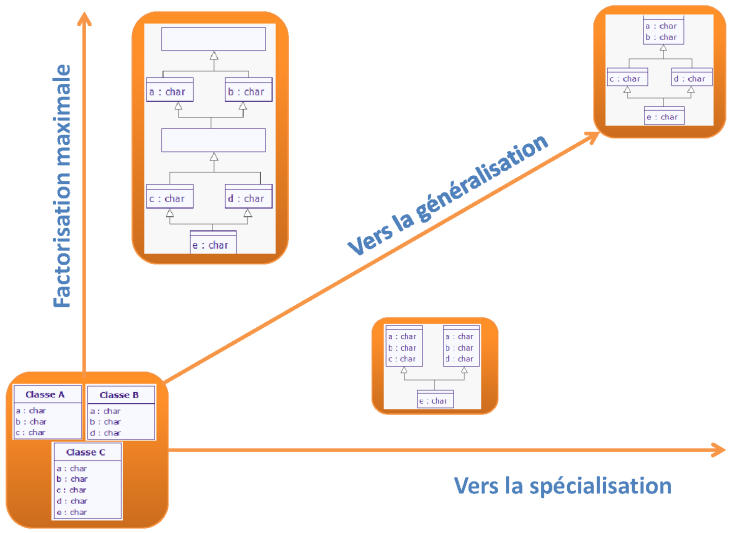


FIGURE 3.4: **Principe Général**

treillis. Les chiffres détaillant l'expérience sont disponibles dans la thèse de référence [Cyril Roume, 2004]. Dans le plus grand projet (*57 classes*), le nombre de *nouveaux* concepts était de *110 pour les classes, 9 pour les associations* et *59 pour les propriétés*. Parmi ces concepts, les experts ont été en mesure de sélectionner ceux qui sont pertinents.

3.5.3.2 Expérimentation sur un modèle de classe de la société *JETSGO*

Dans la thèse [Hacène, 2005], l'Analyse Relationnelle de Concepts a été expérimentée sur un extrait du modèle de classes issu du logiciel *Jetsmiles* de la société *JETSGO*[2]. Cet extrait de modèle de classes est composé de seulement *6 classes*, reliées par *9 associations*, et environ *30 attributs*. Les attributs et les rôles sont mélangés et les classes sont reliées par un contexte relationnel aux attributs et aux rôles. Ainsi, ces rôles sont reliés par un autre contexte relationnel définissant leur type (quand il s'agit bien d'une classe).

Les éléments de modélisation en UML sont décrits par de nombreuses caractéristiques techniques : la multiplicité, la visibilité, la valeur de la méta-popriété "isAbstract", etc. Cette description technique est importante pour l'introduction des nouveaux éléments de modélisation UML. Après application du processus RCA, émergent beaucoup de nouveaux concepts, parmi lesquels de nombreux concepts sont non pertinents et apparaissent en nombre très important, *par exemple : la super-classe de toutes les classes abstraites* . Un post-traitement d'analyse des concepts est mise en place, afin de construire et de maintenir les concepts les plus pertinents. Le treillis des concepts

2. http://www.jetsgo.net/

pour les classes contient environ *35 concepts* tandis que le treillis de concepts attribut + rôle a environ *25 concepts*.

3.5.3.3 Expérimentations pour des modèles UML2 et EMF

L'Analyse Relationnelle de Concepts a été aussi expérimentée sur deux modèles *Ecore*[3] issus de deux programmes Java et cinq modèles UML dans [Falleri *et al.*, 2008]. La configuration utilisée est composée des classes décrites par leur nom et les propriétés (c'est-à-dire les attributs et les rôles). Ces derniers sont décrits par leurs noms et par les classes qui contiennent ces propriétés. Ainsi, les propriétés sont décrites par leur type (quand il s'agit d'une classe).

Les premiers résultats représentatifs de cette expérience sur un modèle issu d'Apache Common Collections [4] composé de 250 classes montrent que l'Analyse Relationnelle de Concepts trouve *34 nouveaux concepts de classe* qui correspondent essentiellement à des concepts de propriété. Pour Le méta-modèle d'UML2 [5], nous avons *246 classes* et *615 propriétés*, l'Analyse Relationnelle de Concepts extrait *1534 nouveaux concepts de classe* et *998 nouveaux concepts de propriété*.

Dans cette expérience, les associations n'ont pas été encodées, contrairement à ce que nous faisons dans le modèle issu du projet *SIE-Pesticides*. Néanmoins, l'explosion du nombre de concepts apparaît déjà. Dans notre cas, nous introduisons les associations dans la configuration et nous montrerons qu'avec certaines précautions telles que l'annotation par la navigabilité et le fait de nommer les rôles, la refactorisation de données, y compris avec les associations, reste possible.

3.5.3.4 Expérimentations sur plusieurs configurations de FCA/RCA sur un modèle de classes de *l'open-source Java Salomé-TMF*

Parmi les plus récentes des études, une nouvelle approche propose trois stratégies d'application de l'Analyse Formelle de Concepts et de l'Analyse Relationnelle de Concepts [Falleri, 2009] sur le modèle issu d'une partie du logiciel open-source Java Salomé-TMF [6]. Le modèle concerne la gestion des tests, ce modèle est composé de 37 classes avec 142 attributs.

Les stratégies utilisées sont les suivantes : AFC-NAME, AFCC-NAME et ARC-NAME et sont décrites ci-après.

Dans AFC-NAME, les entités sont les classes et les caractéristiques sont les noms des attributs ; une analyse sémantique des identifiants permet d'assigner à une classe ses noms d'attributs et ainsi que leurs hyperonymes.

Dans AFCC-NAME, les classes sont décrites par des caractéristiques composites contenant le nom de l'attribut, son type, si elle est statique ou non ; l'hypéronymie est également encodée dans la relation. Les opérations n'ont pas été prises en compte, car elles étaient très techniques et l'auteur estime que les attributs à eux seul apportaient l'information principale.

Dans ARC-NAME, un premier contexte formel comprend les classes (avec aucune description), un deuxième contexte formel décrit les attributs par leur nom et les hyperonymes de leurs

3. Ecore est un modèle EMF définissant les concepts manipulables dans EMF
4. http ://commons.apache.org/collections/
5. http ://www.eclipse.org/uml2
6. http ://wiki.ow2.org/salome-tmf/

noms, un premier contexte relationnel associe une classe aux noms de ses attributs, un second contexte relationnel associe un attribut à son type quand ce dernier correspond à une classe.

L'AFC-NAME produit *26* nouveaux concepts de classe et *3* concepts de fusion. Ensuite, l'AFCC-NAME produit *17* nouveaux concepts de classe et *3* concepts de fusion. Enfin, l'ARC-NAME produit *33* nouveaux concepts de classe et *3* concepts de fusion.

Définition 3.7 (Concepts de fusion) *les* concepts formels de fusion (ou fusionnés) *ont strictement plus d'une entité dans leur extension simplifiée. Remarquons que toutes les entités de l'extension sont décrites par exactement le même ensemble de caractéristiques.*

Par ailleurs, la stratégie ARC-NAME produit 21 nouveaux concepts d'attributs et 13 concepts d'attributs fusionnés.

Tous ces résultats sont très acceptables à des fins d'analyse par des experts, mais les modèles traités n'ayant pas d'associations, il est difficile de généraliser ces résultats au cas général des modèles de classe.

Par rapport à ce travail, ici, nous n'utiliserons pas l'information linguistique (cela pourrait être fait dans un travail futur). Les termes du modèle ne seront pas des identificateurs techniques, mais plutôt les termes de domaines soigneusement sélectionnés par un groupe d'experts.

3.6 Conclusion

Dans ce chapitre, nous avons présenté l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts tout en exposant les principales approches qui ont été élaborées à ce jour pour leur application dans le cadre général du génie logiciel et plus particulier de la factorisation dans les modèles de classes. Nous avons ensuite rapporté les expérimentations récentes issues de plusieurs contextes d'application. Nous nous sommes intéressés à la complexité des modèles produits en termes de nombres de concepts introduits, qui doivent donc être vérifiés par un expert. Cette complexité dépend essentiellement de la taille des modèles de départ. Dans ce contexte, nous nous sommes intéressés à étudier le comportement de l'Analyse Relationnelle de Concepts sur des modèles UML (cf. chapitre 6. Cette étude s'interroge sur la part de l'influence de la structure des modèles sur différentes variables étudiées (comme les temps d'exécution et la mémoire occupée) au travers de plusieurs expérimentations sur les 15 versions du modèle SIE-Pesticides (cf. section 4.2.1).

Ces méthodes d'analyse proposent un cadre théorique pour la découverte exhaustive des nouvelles abstractions dans le modèle. Elles sont utilisées pour l'amélioration de la qualité d'abstraction et l'élimination des duplications dans les modèles de classes au travers de divers travaux. Ainsi, ces méthodes définissent bien une forme normale dans le cadre des modèles conceptuels. Nous nous appuyons sur la capacité de l'AFC et de l'ARC à faire émerger au sein d'un modèle des abstractions thématiques de niveau supérieur, améliorant ainsi la sémantique des modèles pour l'analyse de l'évolution de la factorisation d'un modèle (cf. chapitre 5). Mais surtout dans la présente thèse nous les exploiterons d'une manière originale pour construire le plus grand modèle commun de plusieurs modèles de classes d'entrée (cf. chapitre 6).

Deuxième partie

Contexte expérimental : Cas d'Étude

CHAPITRE

4

Cas d'Étude : Projets applicatifs et outils

Préambule

Dans ce chapitre, nous décrivons le contexte d'expérimentation de notre approche. Nous nous plaçons dans un cadre de développement d'une application informatique en différentes phases dans lesquelles contribuent plusieurs équipes de natures diverses. Ces équipes produisent des artefacts informatiques (besoins, modèles UML, codes, etc.) qu'il faut ensuite partager. Nous évoquons deux projets cibles : le projet SIE-Pesticides (cf. 4.2.1) pour lequel nous disposons de versions successives et le projet SI-GPE qui fera l'objet de travaux futurs. Nous expliquons quelles données nous extrayons des différentes versions de modèle du projet SIE-Pesticides (cf. 4.2.1) pour expérimenter nos méthodes. Nous indiquons enfin quels outils ont été choisis et développés pour mener notre travail.

Sommaire

4.1 Introduction

Dans les chapitres précédents, nous avons présenté les états de l'art sur l'évolution des modèles au cours du cycle de vie d'une application (cf. chapitre 2) et sur l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts (cf. chap 3). Le présent chapitre est dédié à la description des cas d'étude qui ont permis d'identifier les problématiques sur lesquelles se basent les contributions décrites dans les chapitres suivants. Nous exposons les spécificités propres aux domaines d'application avec leurs impacts sur les modèles de chacun des projets applicatifs. Le premier de ces cas d'études (SIE-Pesticides) sert de plus de socle expérimental pour valider les contributions. Nous expliquons quels sont les principaux éléments de modélisation (instances de la méta-classe `ModelElement` d'UML) qui nous serviront pour le suivi de l'évolution ainsi que la factorisation de modèles. Enfin, nous détaillons les divers outils utilisés et développés dans les expérimentations.

4.2 Projets applicatifs

Généralement, plusieurs modèles sont produits pour représenter le système d'information du domaine étudié. Ces différents modèles résultent soit des perceptions et des besoins exprimés individuellement ou en groupes par les utilisateurs, soit des différentes versions du modèle global qui retracent l'avancement du projet tout le long du cycle de développement. Ces versions correspondent à un archivage effectué après chaque séance d'analyse. Le modèle global est le « regroupement » des différentes perceptions et besoins des utilisateurs. Nous avons formulé nos problématiques à partir de deux projets. Le premier projet (SIE-Pesticides, projet IRSTEA) s'inscrit dans un processus d'analyse comme celui qui est précédemment décrit (séances d'analyse collectives successives). Le second projet (SI-GPE, projet Université de Djibouti) a pour but est de regrouper des connaissances existantes, mais dupliquées dans des systèmes d'information de plusieurs institutions de Djibouti. Dans ce projet, il a été nécessaire de prendre en compte les nouveaux besoins issus de la réforme des examens de l'Education Nationale (BAC 2014, etc.). Les expérimentations seront effectuées sur le projet SIE-Pesticides, car, pour le projet SI-GPE, les différents modèles des systèmes d'information existants sont encore en cours d'élaboration.

4.2.1 Projet SIE-Pesticides

Dans les domaines d'intervention de IRSTEA [1] (environnement, territoires, etc.), le développement d'un système d'information suppose de capturer des connaissances de plus en plus complexes. En outre, les recherches de cet institut mobilisent des objets dont la description et le comportement ne sont pas entièrement connus, car c'est sur eux que portent les recherches thématiques. Les équipes informatiques de IRSTEA développent des méthodes et des outils intégrant, dès la conception, l'évolution des systèmes. Les méthodes de développement de nature itérative sont privilégiées, telles que la méthode *Continuous Integration Unified Process* proposée par [Miralles, 2006]. Elle est actuellement mise en œuvre dans un projet dont le but est de mettre en place un système d'information pour les recherches autour des pesticides (SIE Pesticides).

1. Anciennement CEMAGREF.

L'objectif sociétal de ce projet est de réduire l'impact des produits phytosanitaires dans les différents compartiments de l'environnement en France. Pour ce faire, deux objectifs ont été définis. Le premier objectif concerne la production des connaissances par les équipes thématiques et le second la structuration de la connaissance par les équipes informatiques. Le premier objectif consiste à faire évoluer les méthodes, les indicateurs et les outils utilisés par les thématiciens pour évaluer l'impact des produits phytosanitaires dans les différents compartiments de l'environnement relevant des domaines d'intervention de IRSTEA. Le second consiste à concevoir un système d'information améliorant l'accès à la connaissance relative aux produits phytosanitaires afin de faciliter sa remobilisation pour des analyses par exemple.

Ces deux objectifs fixés sont complémentaires et leur synergie représente un challenge fort du projet. Par ailleurs, la synergie entre équipes thématiques et équipes informatiques permet de répondre à une volonté interne de l'établissement. L'objectif interne est de créer, autour du système d'information, une synergie entre sept équipes dont les recherches intéressent l'application, la gestion ou l'action des pesticides sur le milieu. Le système d'information devrait permettre de mieux capitaliser les connaissances thématiques et les données de ces équipes afin de les diffuser plus largement et de les partager lorsque cela est autorisé.

Dans le cadre de ce projet [Miralles *et al.*, 2009, 2010], les connaissances thématiques sont produites par sept équipes de IRSTEA impliquées dans les domaines de l'écotoxicologie, de l'hydrologie, de l'instrumentation, de la modélisation du ruissellement, de la conception des agro-équipements ou de la géomatique. Par souci de simplification, seules deux équipes ont participé au démarrage de l'analyse : une équipe de Lyon dont les recherches portent sur le transfert des pesticides de la parcelle vers les cours d'eau et une équipe de Bordeaux dont l'activité principale est l'étude des pratiques agricoles des agriculteurs.

Les modèles sont structurés en paquetages, chacun d'eux modélisant un « thème » : les éléments du paysage d'un bassin versant (parcelle, cours d'eau, barrage, etc.), l'instrumentation mise en œuvre pour étudier l'impact des pesticides, etc. Cette approche n'est pas sans poser de problèmes.

En effet, il n'est pas rare que des concepts thématiques soient modélisés deux fois dans deux paquetages du modèle global, que des concepts de niveau supérieur d'abstraction ne soient pas identifiés, etc.

Ainsi, lors de la consolidation et de l'appropriation du système d'information par l'équipe du projet, de nouveaux concepts pourront apparaître, d'autres seront détruits et/ou pourront changer de nom (c'est-à-dire que leur sens pourra évoluer en apportant plus de précision par exemple). De ce fait, le projet SIE-Pesticides a donné lieu à plusieurs versions de modèles pour des périodes données et pour des équipes données. À ce jour, le projet compte une quinzaine de versions de modèles, dont les dernières contiennent plus de 170 classes.

Le projet SIE-Pesticides s'inscrit à la fois dans les recherches thématiques et les recherches informatiques afin de les mettre en synergie. La thèse se situe essentiellement dans le domaine de l'informatique et son $r\hat{o}le$ est d'enrichir le projet SIE-Pesticides avec de nouvelles méthodologies d'abstraction de données et de nouvelles techniques d'analyse de l'évolution. Ces méthodologies intègreront toujours une formulation basée sur le langage UML. Au terme du projet, l'objectif sera d'assister les thématiciens pour factoriser les modèles. Concrètement, un environnement de travail sera implémenté dans un Atelier de Génie Logiciel (voir 4.4.1) pour que l'utilisateur puisse mettre en place le processus de factorisation spécifique à ses besoins (cf. figure 4.2).

Le schéma 4.1 illustre le processus itératif de factorisation des concepts des modèles UML. Lors d'une première transformation, une partie des informations contenues dans les modèles est transposée dans des contextes formels. Une deuxième transformation calcule des treillis de concepts sur ces contextes, puis un modèle appelé *modèle générique* est construit à partir de ces treillis.

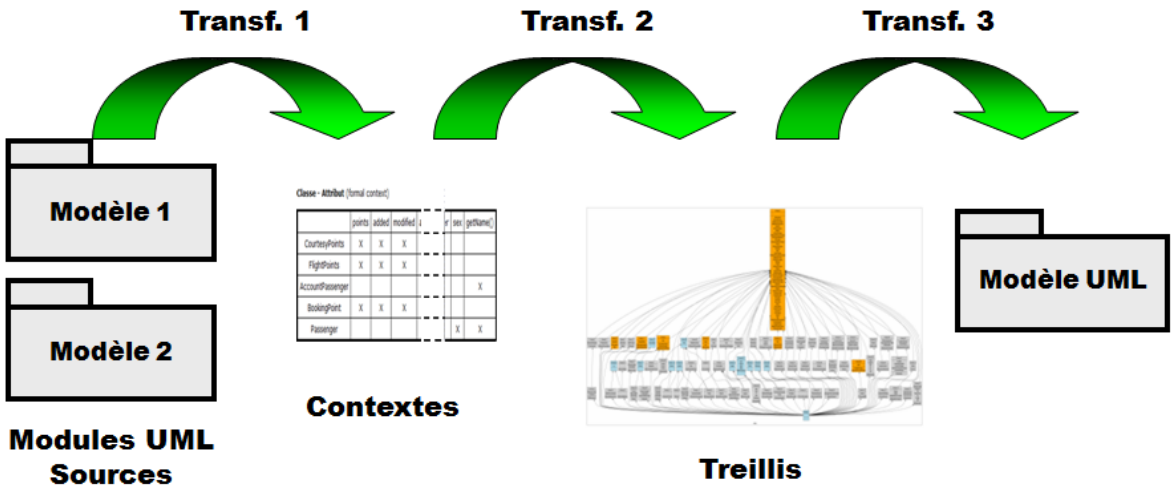


FIGURE 4.1: **Processus global de factorisation**

L'ensemble des différentes versions du modèle de classes du projet SIE-Pesticides nous servira comme cadre d'étude pour toutes les expérimentations durant la thèse. La démarche mise en place a consisté pour chacune des expérimentations à tester les pistes de réflexion en cours sur un paquetage du modèle du projet SIE-Pesticides et dans un second temps à valider cette piste avec l'ensemble des modèles.

4.2.2 Projet SI-GPE

L'origine de cette recherche est un projet de refonte du système d'information permettant la gestion des étudiants et du personnel. Cette démarche s'inscrit essentiellement dans une suite logique d'acquisition et de maîtrise de l'ensemble de la chaîne éducative afin de mettre en place d'une part à l'université de Djibouti et, par la suite d'agrandir au niveau national, tous les outils nécessaires pour la gestion efficace des systèmes d'informations pour l'enseignement de base, intermédiaire, secondaire et universitaire. En particulier, la massification de la scolarisation à Djibouti, conséquence de la réforme du système éducatif et de l'allongement du droit à la scolarité jusqu'à 16 ans, va générer dans les années à venir une forte augmentation des effectifs dans tous les cycles d'enseignement.

L'évolution attendue des effectifs entre 2009 et 2014 est la suivante :

- 35% d'écoliers supplémentaires en sixième année (1ère année de collège),
- 72% de collégiens supplémentaires en neuvième année (1ère année de lycée),
- 190% de lycéens supplémentaires en terminale de lycée, soit un triplement des effectifs à ce niveau.

L'examen qui est organisé à chacun de ces niveaux, garantissant l'égalité des chances pour la poursuite des études, ira donc en se complexifiant, notamment jusqu'à celui du baccalauréat. Il convient donc de s'y préparer. En novembre 2007, un audit de la situation a été effectué par une équipe du rectorat de Bordeaux pilotée par la directrice de la Division des Examens et Concours. Le rapport de mission [ROIDOR et MAUVILAIN, 2007] propose, d'une part, un nouvel organigramme pour le service des examens et des concours qui serait érigé en direction et, d'autre part, la création d'une nouvelle plateforme informatique pour gérer les examens et concours. Cela prend forme dans un projet regroupant l'ensemble des applications existantes tout en intégrant les nouveaux besoins. Ce projet repose exclusivement sur le système d'information scolaire et universitaire.

L'objectif principal est de faire une analyse profonde de l'existant ainsi que la collecte des nouvelles informations. Le but serait de regrouper l'ensemble des différents outils (applications/logiciels) existants afin de fusionner dans une même plate-forme informatique l'existant, mais aussi d'améliorer le système en prenant en compte les nouveaux besoins et en introduisant plus de performance et de flexibilité.

Le projet doit prendre en compte les similitudes et les convergences des outils existants après avoir extrait les modèles UML sur lesquels reposent ces applications (la collecte des données). Pour ce faire, il est nécessaire d'identifier l'ensemble des besoins des différentes institutions appartenant à des niveaux différents dans les domaines de la gestion administrative du personnel et des étudiants (Gestion des examens et concours). La réalisation d'un projet informatique débute toujours par une analyse des besoins qui comprend dans certains cas une analyse de l'existant, en particulier l'analyse des systèmes d'informations existants comme dans le projet SI-GPE. Au cours du développement, il n'est pas rare que plusieurs versions du modèle soient effectuées comme dans le projet SIE-Pesticides.

Lors de l'analyse de ces besoins, plusieurs équipes sont souvent amenées à travailler ensemble de même que dans le projet SIE-Pesticides. Il en résulte plusieurs perceptions du même système d'information. Nous sommes exactement dans cette configuration pour le projet lié à la gestion du personnel et des étudiants de l'ensemble des institutions (SI-GPE).

Comme nous possédons plusieurs modèles à un moment donné, une solution consiste à "regrouper" ces différents modèles en un modèle unique, qui regroupe l'ensemble des connaissances du système d'information global. Ce dernier modèle peut notamment être discuté en séance plénière avec les différents acteurs du projet.

Finalement, cette opération de "regroupement" est la problématique essentielle de ma thèse. Elle se traduit par une factorisation inter-modèles complétée ou précédée par une factorisation intra-modèle. Elle a nécessité la mise en place de plusieurs métriques d'analyse de l'évolution et de choix de factorisation afin d'obtenir le plus rapidement possible un résultat satisfaisant. Cette thèse est donc intimement couplée à ce projet d'autant plus que les résultats de la thèse devraient accélérer l'avancement du projet en phase d'analyse et de conception du nouveau système d'information.

4.3 Eléments de modélisation intervenant dans l'AFC et l'ARC

Les éléments du métamodèle intervenant dans l'AFC et l'ARC résultent d'un choix des concepteurs des projets SIE-Pesticides et SI-GPE qui ont adopté les diagrammes de classes du langage

UML pour modéliser les systèmes d'information. Les principaux éléments de modélisation utilisés dans ces diagrammes de classe sont les classes, les attributs, les opérations, les associations et les *rôles*. Comme les diagrammes de classes du langage UML sont imposés, les transformations T1 et T3 sont spécifiques à UML. Mais en toute généralité, il est très facile de les adapter à un autre langage de modélisation basé sur les mêmes principes, comme le méta-modèle KM3, ou MOF. Les éléments de modélisation que nous utilisons et leurs relations sont présentés à la Figure 4.2. Ils sont inspirés par le métamodèle d'Objecteering, proche du métamodèle UML 1 [2].

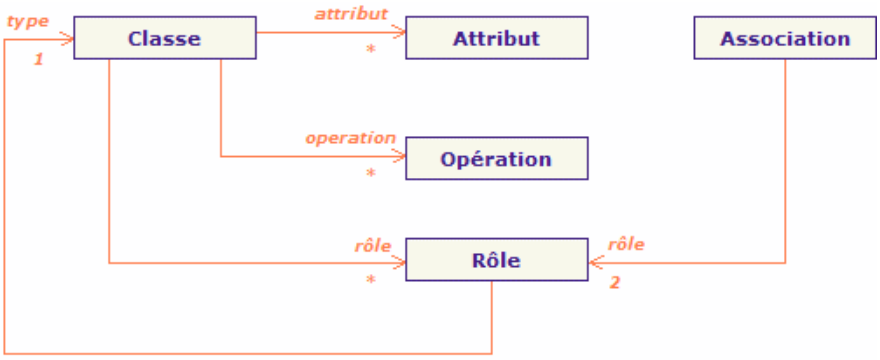


FIGURE 4.2: **Principaux éléments de modélisation**

Les deux projets décrits ci-dessus ont des similitudes en terme de processus de développement qui ont conduit à la problématique posée dans la thèse. Dans les deux cas de figure, les projets ont pour objectif le développement d'un système d'information (SIE-Pesticides et SI-GPE) dont il faut prendre en compte l'évolution dans le temps. Pour le projet SIE-Pesticides, le processus d'évolution est le versionnement des modèles et se situe principalement lors de l'analyse. Après chaque séance d'analyse, les modèles UML sont archivés afin de tracer l'avancement du projet mais aussi l'évolution des concepts thématiques. Pour le projet SI-GPE, il existe plusieurs applications qui répondent aux mêmes besoins et chacune d'elles repose sur un modèle UML qui lui est propre. Dans ce cas, le processus d'évolution réside dans l'intégration de toutes les applications ainsi que la prise en compte des concepts introduits pour modéliser les nouveaux besoins. De ce fait, le but essentiel de la thèse est de factoriser dans un modèle unique les concepts communs contenus dans plusieurs modèles « sources ».

Nous allons expérimenter l'utilisation de l'AFC et de l'ARC comme outil de factorisation des concepts thématiques identiques des systèmes d'information des deux projets d'application. Puisque les deux projets sont modélisés en UML , les sections suivantes vont présenter les choix d'éléments de modélisation retenus pour la factorisation ainsi que les relations et configurations qui vont en découler. Ces mêmes relations et configurations entre éléments de modélisation UML seront utilisées tout au long de la thèse. Nous les illustrons sur le paquetage `Station de Mesure` du modèle du projet SIE-Pesticides (voir figure 4.3). Ces relations et configurations définissent des contextes formels et relationnels qui sont représentés sous forme de tables.

2. Un méta-modèle décrit de manière formelle les éléments de modélisation ainsi que la syntaxe et la sémantique de la notation qui permet de le manipuler.

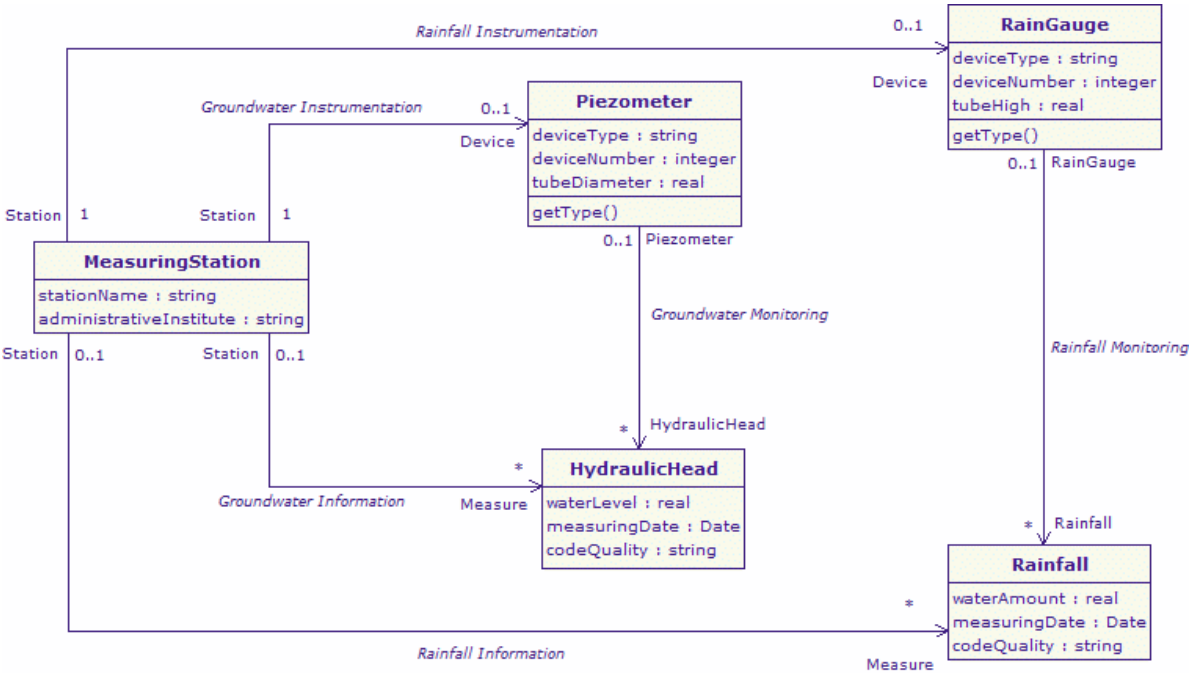


FIGURE 4.3: **Paquetage station de mesure extrait du modèle SIE-Pesticides**

4.3.1 Relations retenues pour l'Analyse Formelle de Concepts (AFC)

L'AFC est décrite dans le chapitre 3. Nous rappelons que son objectif est d'atteindre la factorisation maximale de caractéristiques entre entités. Son résultat est donc que chaque caractéristique apparaît dans un et un seul concept du treillis. Lorsque les entités sont les classes et les caractéristiques les attributs, le treillis peut être interprété comme une nouvelle hiérarchie de classes (chaque concept est une classe) dans laquelle les attributs sont factorisés de manière maximale. Une factorisation au sein d'un même modèle sera qualifiée de factorisation intra-modèle [Falleri, 2009; Huchard, 2003]. L'AFC utilise une relation binaire entre les éléments de modélisation[Falleri *et al.*, 2007; Falleri, 2009; Hacene *et al.*, 2007a]. Cette méthode a été utilisée pour l'amélioration de la qualité d'abstraction et l'élimination des doublons dans les modèles de classes au travers de divers travaux [Falleri, 2009; Arévalo *et al.*, 2006; Godin et Mili, 1993; Hacene *et al.*, 2007a; Dao *et al.*, 2004b]. De nombreuses variantes ont été étudiées dans les recherches citées. Elles prennent en compte plus ou moins de caractéristiques pour les entités (souvent des classes) : nom des attributs, types des attributs, noms des opérations, signature des opérations, spécialisation des types, etc. Son intérêt est lié aux fortes propriétés que possède le modèle de classes après refactorisation : les doublons sont tous supprimés, les relations de spécialisation/généralisation respectent l'inclusion des ensembles des caractéristiques des classes. La description complète est exposée dans le chapitre 3.

Dans nos expérimentations, nous avons choisi d'une part trois principales relations simples décrites par une paire d'éléments de modélisation et, d'autre part, deux relations supplémentaires impliquant plusieurs éléments de modélisation. Ces relations ont été appliquées aux différentes versions du modèle du projet SIE-Pesticides. Elles s'appuient sur les éléments de modélisation les

plus fréquemment utilisés en UML pour modéliser un système d'information à savoir les classes, les attributs, les opérations, etc. Les modèles correspondant aux deux projets d'application sont réalisés en vue d'obtenir un schéma de base de données relationnelle [Hacene *et al.*, 2007a]. De ce fait nous observerons que certains éléments de modélisation, comme les opérations, sont en nombre limité.

4.3.1.1 Description des éléments de modélisation [3]

Les relations se baseront sur le nom de l'élément de modélisation c'est-à-dire sur la chaîne de caractères qui permet de le désigner. Le choix des noms pour identifier et/ou décrire un élément de modélisation est donc fondamental. Ce travail de capture par le concepteur des concepts thématiques manipulés par les acteurs est une activité essentielle de la phase d'analyse d'un projet. Dans le cadre de cette thèse, nous avons débuté un travail de réflexion sur l'usage d'un réseau lexical pour assister les concepteurs et/ou guider la méthode de factorisation. Ce travail sera une prochaine étape de nos recherches et nous le décrivons en conclusion de ce manuscrit.

Pour le projet SIE-Pesticides, le travail en plusieurs équipes thématiques fait qu'un même concept peut être modélisé par un ou plusieurs de ces équipes dans des paquetages différents, mais un même concepteur a présidé toutes les réunions. Dans le projet SI-GPE, les modèles émanent d'applications physiquement différentes, réalisées par des équipes disjointes, et répondant au même besoin, qui ont de ce fait une forte probabilité d'avoir modélisé les mêmes concepts.

Autant dans SIE-Pesticides, nous pensons que travailler sur les noms tels qu'ils sont dans le modèle, sans l'appui de ressources lexicales extérieures n'est pas une hypothèse trop forte, car un unique acteur (le concepteur chef de projet) a travaillé sur l'uniformisation du vocabulaire donné lors des réunions par les différents experts. Autant pour le projet SI GPE, ce sera par contre une problématique incontournable.

Cette démarche nous permettra de traiter le cas de la répétition d'un même concept dans un modèle issu d'un cas concret, par exemple dans le cas de la dernière version du modèle SIE-Pesticides qui compte plus de 170 classes.

Par ailleurs, pour respecter le modèle, les relations créées doivent préserver les liens d'héritage définis par le concepteur du modèle ; il faut donc "remettre à plat le modèle". Pour cela, lors de la création des relations (qui portent sur les classes), une classe qui hérite d'une autre classe va se voir affecter tous les noms des éléments (par exemple les noms des attributs) qui décrivent sa super-classe.

Les lignes des relations binaires correspondent aux différentes classes du modèle. Les colonnes correspondent aux différents noms des éléments de modélisation mis en relation avec les classes (attributs, $rôles$, opérations). Un couple dans la relation binaire indique qu'une classe (en ligne) possède l'élément de modélisation du nom donné en colonne, directement ou par héritage.

4.3.1.2 Relation R1

La première relation binaire, notée `R1`, implique d'une part, les classes qui décrivent le domaine de définition des objets qui regroupent des éléments d'un monde (abstrait ou réel) interagissant entre eux. D'autre part, les attributs d'une classe correspondent aux propriétés de la classe et en

3. Par leur nom et mise à plat de l'héritage.

Classe × Attribut	deviceType	deviceNumber	tubeDiameter	waterLevel	measuringDate	codeQuality	tubeHigh	waterAmount	stationName	administrativeInstitute
Piezometer	×	×	×							
HydraulicHead				×	×	×				
RainGauge	×	×					×			
Rainfall					×	×		×		
MeasuringStation									×	×

TABLE 4.1: **R1 =** $owns_1 \subseteq$ **classe** × **Attribut (cf. section 4.3.1.2)**

UML ils sont définis au minimum par un nom, un type et éventuellement une valeur initiale. Cette relation permettra de rassembler l'ensemble des classes partageant les mêmes attributs. On notera que, lors de la reconstruction de la hiérarchie, les attributs seront redistribués entre les classes hiérarchiquement. Les hiérarchies de classes ou les classifications permettent de gérer la complexité en ordonnant les objets au sein d'arborescences de classes par niveau d'abstraction croissant. Ces hiérarchies seront elles-mêmes organisées par le lien d'héritage qui exprime soit la généralisation soit la spécialisation. En d'autres termes, `R1` permettra de décrire les classes par les noms des attributs et dans un second temps permettra de regrouper les classes partageant le maximum d'attribut similaires. Ainsi, cette relation ordonnera les classes selon une hiérarchie où chacun des niveaux maximise le partage d'attributs entre classes qui s'y trouvent.

$$\textbf{R1 :}\ owns_1 \subseteq classes \times attributs \tag{4.1}$$

La factorisation se fera grâce au nom des attributs qui sont partagés par plusieurs classes, le type de l'attribut est ignoré, car nos modèles ne possèdent que des types primitifs. Ces types primitifs ne sont pas définis par une classe modélisée par l'utilisateur, mais sont des types fournis par l'atelier de génie logiciel (AGL) et/ou le compilateur du langage ; par exemple en Java les entiers : int, long, byte, short ; les réels : float, double ; les booléens : boolean et les caractères : char..

4.3.1.3 Relation R2

La deuxième relation binaire, notée `R2` permet la factorisation des classes selon leurs comportements, ce mécanisme est matérialisé en UML par les opérations. La spécification du comportement d'un objet est donnée par les opérations décrites par sa classe et sa réalisation est exprimée dans les méthodes. En UML, on distingue deux termes : une opération est un service offert par les instances de la classe alors que la méthode est une implémentation de l'opération.

$$\textbf{R2 =}\ owns_2 \subseteq classes \times opérations \tag{4.2}$$

Classe × Opération	getType
Piezometer	×
HydraulicHead	
RainGauge	×
Rainfall	
MeasuringStation	

TABLE 4.2: **R2 =** $owns_2 \subseteq$ **Classe × Opérations (cf. section 4.3.1.3)**

Cette relation décrit les classes par le nom des opérations sans prendre en compte ni leurs signatures ni le type de la valeur de retour. La signature d'une opération regroupe en particulier le nom et le type de ses arguments.

Les signatures portent une sémantique importante, cependant nos modèles d'expérimentation issus du projet *SIE-Pesticides* ne possède pas beaucoup d'opérations, nous sommes donc restés sur une description simplifiée. Ainsi, leur apport sémantique dans nos factorisations sera limité, néanmoins il pourrait être intéressant dans d'autres types de modèles tels que les modèles d'implémentation de prendre mieux en compte les signatures. Nous faisons l'hypothèse que dès lors que nos modèles UML se situent en phase d'analyse ou de conception des projets d'applications respectifs, l'apport de cette relation est moindre. Toutefois (voir chapitre 5, 7, 6), elle peut s'avérer importante pour le suivi de l'évolution des modèles (voir chapitre 5).

4.3.1.4 Relation R3

Dans la liste des principales relations, nous introduisons un autre élément de modélisation auquel nous attachons une grande importance car il participe à la relation entre les classes dans le langage UML. Il s'agit des $rôles$. L'extrémité des associations est représentée par cet élément de modélisation, le $rôle$, et décrit quelle est la fonction d'un objet d'une des classes reliées dans l'association.

Le $rôle$ peut être assimilé à un pseudo-attribut d'une des classes reliées. A ce titre, son nom doit être unique. Cet élément de modélisation s'est avéré très intéressant pour factoriser les classes. Cet aspect relationnel sera encore mieux pris en compte avec l'ARC (voir le chapitre 3), mais même avec l'AFC, l'utilisation des $rôles$ va compléter les factorisations amenées par les attributs (cf. la figure 4.4).

$$\textbf{R3} = owns_3 \subseteq classes \times rôles \tag{4.3}$$

Comme le montre la figure 4.4, dans la relation $owns_3$ les $rôles$ seront des caractéristiques, au même titre que les attributs. Le suivi de l'évolution des factorisations avec cette relation sera détaillé dans le chapitre 5 et comparé avec les factorisations en provenance des autres relations ci-dessus mentionnées.

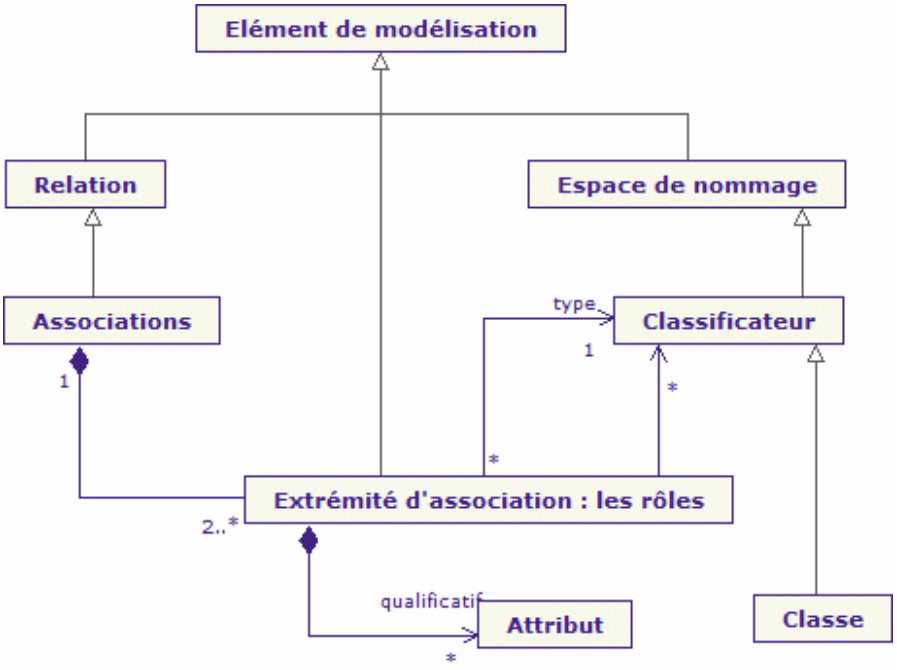


FIGURE 4.4: **Description des classes par les** *rôles*

Classe × *Rôles*	HydraulicHead	Station	Piezometer	Rainfall	RainGauge	Device	Measure
Piezometer	×	×					
HydraulicHead		×	×				
RainGauge		×		×			
Rainfall		×			×		
MeasuringStation						×	×

TABLE 4.3: **R3 =** $owns_3 \subseteq Classe \times r\hat{o}les$ **(cf. section 4.3.1.4)**

4.3.1.5 Relation R4

Alors que les relations précédentes décrivent les classes par un seul élément de modélisation, nous nous intéressons ici à évaluer l'apport de factorisations plus complexes qui prennent en compte plusieurs éléments de modélisation. Cette quatrième relation s'intéressera à décrire les classes à la fois par leurs attributs et leurs *rôles*.

$$\textbf{R4} = owns_4 \subseteq classes \times attributs \cup r\hat{o}les \tag{4.4}$$

Cette relation fera émerger les concepts correspondant à des groupes de classes partageant des attributs et des liens (cf. en annexe A.1 le contexte formel correspondant à cette relation).

4.3.1.6 Relation R5

Enfin, nous avons réuni l'ensemble des éléments de modélisation qui nous semble les plus importants pour décrire les classes, à savoir les attributs (R1), les $rôles$ (R3) et les opérations (R2) dont elles disposent. En d'autres termes, cette relation va compléter la précédente en y ajoutant les opérations (voir en annexe A.2 cette relation). Les modèles qui vont être utilisés dans les expérimentations relèvent de projets de bases des données, donc contiennent peu d'opérations, mais cela nous permettra malgré tout d'évaluer l'apport de cet élément de modélisation dans la factorisation.

$$\textbf{R5} = owns_5 \subseteq classes \times attributs \cup opérations \cup rôles \tag{4.5}$$

Dans la section suivante, nous présentons les données retenues pour appliquer l'ARC.

4.3.2 Configurations retenues pour l'Analyse Relationnelle de Concepts (ARC)

En prélude, il est important de rappeler que l'ARC est une extension de l'AFC qui permet d'utiliser non seulement des entités décrites par des caractéristiques, mais également des relations entre entités. A titre d'exemple, les associations entre classes font partie de cette seconde catégorie [Arévalo *et al.*, 2006; Hacene *et al.*, 2007a; Dolques *et al.*, 2008, 2009; Falleri, 2009].

L'ensemble de données considéré est regroupé dans une famille relationnelle de contextes. L'objectif principal de factorisation reste le même. Pour différencier cette méthode de la précédente, le terme de configuration a été adopté pour qualifier l'ensemble des relations binaires mises en jeu lors d'une factorisation (la famille relationnelle de concepts).

Pour appliquer l'ARC, il est nécessaire d'identifier les éléments de modélisation qui caractérisent les relations pour produire le treillis. Nous considérons deux configurations, notées C1 et C2, composées de relations entre les principaux éléments de modélisation comprenant les classes, les attributs, les opérations, les associations et les $rôles$. Dans les configurations (dans les familles relationnelles de contextes), on trouve des contextes formels (relations entité-attribut) et des contextes relationnels (relation entité-entité). Les relations entité-attribut qui nous serviront de base décrivent les entités par leur nom. Elles sont présentées dans les tables 4.4, 4.5, 4.6, 4.7 et 4.8 pour le modèle de la figure 4.3. Dans les deux sections suivantes nous indiquons quels contextes relationnels (relations entité-entité) comprend chacune des configurations.

4.3.2.1 Configuration C1

La première configuration retenue comprend les éléments suivants : les classes, les attributs, les opérations, les associations et les $rôles$. Toutes ces entités sont désignées par leur nom.

Classe × nom	Piezometer	HydraulicHead	RainGauge	Rainfall	MeasuringStation
Piezometer	×				
HydraulicHead		×			
RainGauge			×		
Rainfall				×	
MeasuringStation					×

TABLE 4.4: **Contexte formel : Classe × Nom**

Opération × nom	getType
Piezometer.getType	×
RainGauge.getType	×

TABLE 4.5: **Contexte formel : Opération × Nom**

Configuration C1

$$Les\ contextes\ formels\ associés\ aux\ tables\ 4.4,\ 4.5,\ 4.6,\ 4.7 et\ 4.8 \tag{4.6a}$$
$$owns_1 \subseteq classes \times attributs \tag{4.6b}$$
$$owns_2 \subseteq classes \times opérations \tag{4.6c}$$
$$owns_3 \subseteq classes \times rôles \tag{4.6d}$$
$$owns_4 \subseteq Associations \times rôles \tag{4.6e}$$

Les contextes relationnels sont présentés dans les tables 4.9, 4.10, 4.11 et 4.12.

4.3.2.2 Configuration C2

La configuration C2 étend la configuration C1 par une relation supplémentaire très importante qui consiste à décrire les *rôles* par leurs types (qui sont des classes). Comme nous le montre la figure 4.2, il s'agit de la relation entre *Rôles* et Classe qui détermine le type du *rôles*. Cette configuration permettra de factoriser les *rôles* ayant le même type. Le contexte relationnel est présenté par la table 4.13.

Attribut × nom	deviceType	deviceNumber	tubeDiameter	waterLevel	measuringDate	codeQuality	tubeHigh	waterAmount	stationName	administrativeInstitute
Piezometer.deviceType	×									
Piezometer.deviceNumber		×								
Piezometer.tubeDiameter			×							
HydraulicHead.waterLevel				×						
HydraulicHead.measuringDate					×					
HydraulicHead.codeQuality						×				
RainGauge.deviceType	×									
RainGauge.deviceNumber		×								
ainGauge.tubeHigh							×			
Rainfall.waterAmount								×		
Rainfall.measuringDate					×					
Rainfall.codeQuality						×				
MeasuringStation.stationName									×	
MeasuringStation.administrativeInstitute										×

TABLE 4.6: **Contexte formel : Attribut × Nom**

Configuration C2

$$Les\ contextes\ formels\ associés\ aux\ tables 4.4,\ 4.5,\ 4.6,\ 4.7 et\ 4.8 \tag{4.7a}$$
$$owns_1 \subseteq classes \times attributs \tag{4.7b}$$
$$owns_2 \subseteq classes \times opérations \tag{4.7c}$$
$$owns_3 \subseteq classes \times rôles \tag{4.7d}$$
$$owns_4 \subseteq Associations \times rôles \tag{4.7e}$$
$$hasType_5 \subseteq rôles \times classes \tag{4.7f}$$

4.3.2.3 La notion de la navigabilité

En UML, la notion de navigabilité est essentielle car elle oriente les associations entre classes qui sont par défaut bidirectionnelles. Le fait qu'une association soit navigable de manière bidirectionnelle signifie qu'un objet d'une des classes extrémité connaît les objets de l'autre classe extrémité auxquels il est relié. Certains AGL orientent par défaut dans l'interface les associations en introduisant automatiquement la navigabilité de la source de la souris vers sa destination lors du tracé

Association × nom	Groundwater Monitoring	Groundwater Instrumentation	Groundwater Information	Rainfall Monitoring	Rainfall Instrumentation	Rainfall Instrumentation
Groundwater Monitoring	×					
Groundwater Instrumentation		×				
Groundwater Information			×			
Rainfall Monitoring				×		
Rainfall Instrumentation					×	
Rainfall Information						×

TABLE 4.7: **Contexte formel : Association × Nom**

$Rôles \times nom$	HydraulicHead	Rainfall	Device	Measure
HydraulicHead :HydraulicHead	×			
Rainfall :Rainfall		×		
Device :Piezometer			×	
Device :RainGauge			×	
Measure :HydraulicHead				×
Measure :Rainfall				×

TABLE 4.8: **Contexte formel :** $r\hat{o}les \times Nom$

$owns_1$ Classe × Attribut	Piezometer.deviceType	Piezometer.deviceNumber	Piezometer.tubeDiameter	HydraulicHead.waterLevel	HydraulicHead.measuringDate	HydraulicHead.codeQuality	RainGauge.deviceType	RainGauge.deviceNumber	RainGauge.tubeHigh	Rainfall.waterAmount	Rainfall.measuringDate	Rainfall.codeQuality	MeasuringStation.stationName	MeasuringStation.administrativeInstitute
Piezometer	×	×	×											
HydraulicHead				×	×	×								
RainGauge							×	×	×					
Rainfall										×	×	×		
MeasuringStation													×	×

TABLE 4.9: **Contexte relationnel $owns_1$ (cf. section 4.3.2.1)**

$owns_2$ Classe × Opération	Piezometer.getType	RainGauge.getType
Piezometer	×	
HydraulicHead		
RainGauge		×
Rainfall		
MeasuringStation		

TABLE 4.10: **Contexte relationnel $owns_2$ (cf. section 4.3.2.1)**

$owns_3 Classe \times rôles$	HydraulicHead :HydraulicHead	Rainfall :Rainfall	Device :Piezometer	Device :RainGauge	Measure :HydraulicHead	Measure :Rainfall
Piezometer	×					
HydraulicHead						
RainGauge		×				
Rainfall						
MeasuringStation			×	×	×	×

TABLE 4.11: **Contexte relationnel** $owns_3$ **(cf. section 4.3.2.1)**

$owns_4 Association \times rôles$	HydraulicHead :HydraulicHead	Rainfall :Rainfall	Device :Piezometer	Device :RainGauge	Measure :HydraulicHead	Measure :Rainfall
Groundwater Monitoring	×					
Groundwater Instrumentation			×			
Groundwater Information					×	
Rainfall Monitoring		×				
Rainfall Instrumentation				×		
Rainfall Information						×

TABLE 4.12: **Contexte relationnel** $owns_4$ **(cf. section 4.3.2.1)**

$hasType_5$ Role × Type (Classe)	Piezometer	HydraulicHead	RainGauge	Rainfall	MeasuringStation
HydraulicHead :HydraulicHead		×			
Rainfall :Rainfall				×	
Device :Piezometer	×				
Device :RainGauge			×		
Measure :HydraulicHead		×			
Measure :Rainfall				×	

TABLE 4.13: **Contexte relationnel $hasType_5$ (cf. section 4.3.2.2)**

de l'association. C'est à dire que lors de la création de l'association dans l'interface de l'AGL, le mouvement de la souris est utilisé et l'association est par défaut navigable depuis la classe source du mouvement vers la classe cible. Toutefois, lorsque l'on crée un diagramme de classes, il est possible d'indiquer la navigabilité entre les deux classes. Ceci est illustré par l'ajout d'une flèche à une extrémité de la relation. La navigabilité permet de spécifier dans quel(s) sens il est possible de traverser l'association à l'exécution. C'est une notion qui s'applique à une association $R \subseteq C_{source} \times C_{cible}$. Ainsi, si R est navigable à partir de la classe C_{source} à la classe C_{cible}, cela signifie que depuis un objet de la classe C_{source}, on peut atteindre (soit par une requête, soit par envoi d'un message à) un objet de la classe C_{cible}. Cependant, dans notre analyse actuelle, nous distinguerons les $rôles$ qui sont nommés, parce qu'ils ont une sémantique plus forte, de ceux qui ne le sont pas. Nous verrons l'impact des $rôles$ non nommés pour lesquels certains AGL proposent une dénomination par défaut mais unique à tous (par exemple undefined).

Toutes nos relations et nos configurations ont été extraites du modèle UML en prenant en compte ou non la navigabilité des associations. Nous verrons dans les chapitres suivants l'importance de cette notion sur notre processus de factorisation via des métriques qui permettent de mesurer l'apport de cette dernière. La prise en compte de la navigabilité influencera toutes les relations et les configurations qui font référence aux $rôles$ (R3, R4, R5, C1, C2). Dans les contextes formels et relationnels précédents, la navigabilité a été prise en compte par défaut puisqu'elle existe par défaut en UML. Sa suppression modifie les relations $owns_3$(cf. le tableau 4.14), $owns_4$(cf. le tableau 4.15) et $hasType_5$(cf. le tableau 4.16) mais aussi la configuration (C2).

4.4 Aspects techniques et outils utilisés

Tout au long de la thèse, nos expérimentations ont été effectuées avec des outils informatiques dont une partie avaient déjà été conçue et implémentée soit au LIRMM [4] soit à l'UMR TETIS. Une

4. Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)

$Sansnavigabilitéowns_3 Classe \times Rôles$	HydraulicHead :HydraulicHead	Rainfall :Rainfall	Device :Piezometer	Device :RainGauge	Measure :HydraulicHead	Measure :Rainfall	Station :MeasuringStation(1)	Station :MeasuringStation(2)	Station :MeasuringStation(3)	Station :MeasuringStation(4)
Piezometer	×						×			
HydraulicHead								×		
RainGauge		×							×	
Rainfall										×
MeasuringStation			×	×	×	×				

TABLE 4.14: **Contexte relationnel sans la navigabilité :** $owns_3$ **(cf. section 4.3.2.1)**

$Sansnavigabilitéowns_4 Association \times Rôles$	HydraulicHead :HydraulicHead	Rainfall :Rainfall	Device :Piezometer	Device :RainGauge	Measure :HydraulicHead	Measure :Rainfall	Station :MeasuringStation(1)	Station :MeasuringStation(2)	Station :MeasuringStation(3)	Station :MeasuringStation(4)
Groundwater Monitoring	×									
Groundwater Instrumentation			×				×			
Groundwater Information					×			×		
Rainfall Monitoring		×								
Rainfall Instrumentation				×					×	
Rainfall Information						×				×

TABLE 4.15: **Contexte relationnel sans la navigabilité :** $owns_4$ **(cf. section 4.3.2.1)**

Sans navigabilité $hasType_5$ Role × Type (Classe)	Piezometer	HydraulicHead	RainGauge	Rainfall	MeasuringStation
HydraulicHead :HydraulicHead		×			
Rainfall :Rainfall				×	
Device :Piezometer	×				
Device :RainGauge			×		
Measure :HydraulicHead		×			
Measure :Rainfall				×	
Station :MeasuringStation (1)					×
Station :MeasuringStation (2)					×
Station :MeasuringStation (3)					×
Station :MeasuringStation (4)					×

TABLE 4.16: **Contexte relationnel sans la navigabilité :** $hasType_5$

description de ces outils sera présentée ci-après. Nous limiterons cette description aux développements spécifiques pour réaliser les diverses expérimentations durant la thèse.

4.4.1 L'atelier de génie logiciel Objecteering

On peut définir un Atelier de Génie Logiciel (AGL) comme un logiciel d'aide à la conception et/ou au développement de logiciels assisté par ordinateur. C'est une famille de logiciels offrant un environnement complet de développement de logiciels en équipe. Ces logiciels permettent de couvrir le cycle de vie du logiciel, à savoir l'Analyse, la Conception, la Réalisation et enfin la Maintenance. Nous utiliserons plus particulièrement Objecteering [5].

Objecteering est un outil de modélisation UML (Unified Modeling Language) avancé. Il offre un ensemble d'outils permettant de structurer via la même interface la conception, la modélisation, le développement, la génération des codes et le débogage des applications. Nous utiliserons le Modeler d'Objecteering qui permet notamment de manipuler les modèles conçus par l'acteur du système d'information et de générer les code associé via une générateur SQL.

L'AGL peut en outre être couplé à l'environnement de développement Eclipse [6]. L'intérêt principal de cet AGL est de pouvoir développer des *profils*, qui forment un mécanisme d'extension du métamodèle UML prévu par l'OMG en ajoutant des concepts métiers et des fonctionnalités. Ces fonctionnalités permettent de parcourir les diagrammes UML, de supprimer, de modifier mais aussi d'ajouter des éléments de modélisation dans les diagrammes. De façon générale, elles per-

5. http ://www.objecteering.com/

6. «IDE pour le développement », un environnement de développent qui prend en compte plusieurs langages de développement.http://www.eclipse.org

mettent des transformations sur les diagrammes plus ou moins complexes. C'est grâce à cet outil de développement des profils UML que nous avons pu développer plusieurs modules Objecteering et adapter l'application de l'AFC et l'ARC sur les modèles des projets d'applications.

Notre intention est de proposer un service interactif et par ailleurs que l'utilisateur puisse aussi définir de nouvelles extensions sous forme de profil (une sorte de plugin ou module), des commandes qui seront des entrées de menus dans « UML Modeler », des traitements, des produits de développement afin de pouvoir les tester sur un modèle UML immédiatement. Dans un usage classique, le module « UML Profile » présente le méta-modèle Objecteering (conforme au métamodèle UML 1.3) à l'utilisateur, 6.10 qui va l'exploiter pour parcourir les éléments de modélisation afin de générer les contextes. L'utilisateur se servira également du plugin de ERCA (Voir la section 4.4.2) pour générer les treillis de concepts.

4.4.2 Le Plugin ERCA : Eclipse's Relational Concept Analysis

ERCA [Falleri, 2010] « Eclipse's Relational Concept Analysis » est un framework permettant la construction et la manipulation de treillis de concepts dans un cadre d'Ingénierie Dirigée par les Modèles. ERCA est intégré dans l'environnement de développement Eclipse. Ce framework offre une syntaxe spécifique pour définir des contextes formels et une API Java qui permet d'instancier, de manipuler et de sauvegarder des contextes et des familles relationnelles de contextes à partir des modèles conçus par l'acteur du système d'information. Ce framework a été intégré dans un profil UML d'Objecteering. Cet environnement est extrêmement modulable et facile d'utilisation. Une fois intégré, nous avons défini la transformation T1 qui permet de transformer les modèles en une famille de contextes formels et relationnels (entrée de ERCA).

Tous ces outils ont permis d'implémenter un profil UML « MDAC sous Objecteering » qui peut être déployé dans tous les différentes versions de modèle du projet SIE-Pesticides et à terme, dans le projet SI-GPE. Ce profil permettra d'adapter, d'automatiser et de contrôler le développement des applications en adaptant l'atelier à une problématique métier.

4.4.3 Caractéristique du cluster utilisé pour les expérimentations

L'ensemble des expérimentations a été réalisé sur un cluster de calcul disposant de Linux [7], 64 bits, comme système d'exploitation. Un cluster est constitué d'une batterie de ressources (dans notre cas de processeurs) dans le but de les partager. Cette machine possède neuf nœuds dont chacun dispose de huit processeurs [8] de huit gigaoctets de mémoire. Un des nœuds, nommé Master ; ne sert qu'à la gestion des autres ressources du cluster.

7. Linux debian http://www.debian.org/index.fr.html
8. Intel (R) Xeon (R) CPU E5335 @ 2.00GHz

Troisième partie

Contributions de la thèse :

Analyse de l'évolution de modèle UML

Étude comportementale de l'ARC

Factorisation Inter-modèle

CHAPITRE

5 Analyse de l'évolution de modèles UML

Préambule

Un modèle de système d'information est généralement développé en plusieurs étapes, surtout dans les contextes multi-acteurs, lors de sessions de travail, donnant lieu à plusieurs versions. Il subit ensuite un certain nombre d'évolutions dues à l'innovation technique, à la concurrence et à la réglementation, ou à de nouveaux besoins utilisateurs. Beaucoup de paramètres peuvent être observés lors de cette évolution. Dans ce chapitre, nous étudions l'évolution des nombres relatifs d'éléments de modèles. Puis de manière plus originale, nous montrons comment des métriques issues de l'Analyse Formelle de Concepts viennent compléter cette analyse et la renforcent, notamment pour l'observation de la factorisation et de l'émergence de nouveaux concepts de domaine. Le but est de procurer au concepteur un outil de suivi de cette évolution pour l'aider à mener à bien sa tâche de modélisation.

Sommaire

5.1 Introduction

Dans les domaines thématiques, comme l'environnement et les territoires qui sont le domaine d'intérêt de Irstea, la conception et le développement des systèmes d'information environnementaux impliquent souvent de nombreux acteurs incluant des scientifiques, chacun ayant son propre point de vue sur un système complexe et hétérogène. Il n'est pas rare que ces points de vue soient divergents voire opposés.

De ce fait, le développement d'une application ou la conception d'un système d'information implique une complexité accrue en raison de sa nature hétérogène et de ses interactions. En dépit de cette complexité, la capture des connaissances sur le système est une tâche obligatoire qui ne peut être évitée.

En premier lieu, nous rappelons l'origine du modèle *SIE-Pesticides* et présentons l'historique des versions successives disponibles ce qui permettra d'expliquer les analyses effectuées et les recommandations.

5.1.1 Rappel sur l'origine du modèle *SIE-Pesticides*

À Irstea, comme dans plusieurs instituts de recherche, plusieurs équipes effectuent des travaux autour du même domaine ou dans ces domaines connexes. Elles mobilisent souvent plusieurs domaines scientifiques comme l'hydrologie, l'agronomie, la chimie, etc. Ces différentes équipes produisent un grand nombre des connaissances, de données méthodologiques, etc. Chacune de ces équipes dispose de ses propres pratiques pour stocker et documenter sa production. Ainsi, l'archivage de cette production n'est souvent pas structuré de manière identique et, quand il l'est, il est effectué souvent sur des supports et avec des formats différents (fichiers textes, fichiers Excel, bases de données, etc.). Cette absence de structuration et cette absence de cohérence dans l'archivage sont à l'origine de pertes importantes de connaissance. Il n'est pas rare que cette production soit réutilisable avec beaucoup de difficultés pour de nouvelles études, de nouvelles analyses statistiques ou l'étude de l'évolution dans le temps des systèmes. Ces institutions sont souvent confrontées à un problème de capitalisation de leur propre production.

Afin de pallier ce problème, au moins partiellement, un projet de capitalisation des connaissances a été mis en place dans le domaine des pesticides, domaine comprenant sept équipes de recherche Irstea. Ce projet, intitulé Système d'information environnementale pour les pesticides (*SIE-Pesticides* cf. 4.2.1), a pour objectif de mettre en place un système d'information permettant de regrouper les connaissances et de structurer les informations produites par les équipes. Par souci de simplification, seules deux équipes ont participé au démarrage de l'analyse : une équipe de Lyon dont les recherches portent sur le transfert des pesticides de la parcelle vers les cours d'eau (équipe Transfert) et une équipe de Bordeaux dont l'activité principale est l'étude des pratiques agricoles des agriculteurs (équipe Pratiques).

Pour atteindre cet objectif général, les équipes informatiques ont collaboré avec les deux équipes thématiques impliquées dans le projet. Cet objectif général a été scindé en deux objectifs opérationnels :

1 le premier concerne les équipes thématiques qui ont à produire des connaissances, des méthodologies et les données dans le domaine des pesticides.

2 le second est relatif aux équipes informatiques et consiste à structurer et à stocker la production thématique afin de la remobiliser.

La synergie entre équipes informatiques et équipes thématiques a été un défi très fort dans ce projet. Bien que les deux équipes fortement impliquées dans le projet mobilisent chacune la production de l'autre, l'une d'elles est spécialisée dans l'étude du transfert des pesticides vers les cours d'eau tandis que l'autre travaille sur les pratiques agricoles des agriculteurs et sur les effets de ces pratiques sur le territoire.

Afin de capitaliser la connaissance des thématiciens, le langage de modélisation UML a été utilisé pour capturer et stocker, dans des modèles métiers (principalement des diagrammes de classes), les concepts métiers décrivant les "objets" du domaine. Grâce à l'approche Model Driven Architecture (MDA), le modèle métier a été transformé en un schéma de base de données [Miralles, 2006], où seront capitalisées les données produites par les thématiciens.

Ce travail de modélisation des concepts métiers a été effectué au cours d'une série de séances de travail impliquant les thématiciens. Au cours de cette phase d'analyse, la méthodologie adoptée consistait à archiver les modèles après chaque séance afin de pouvoir revenir à l'étape précédente en cas de besoin. Ce processus d'archivage explique les 15 versions du modèle *SIE-Pesticides* (cf. figure 5.1).

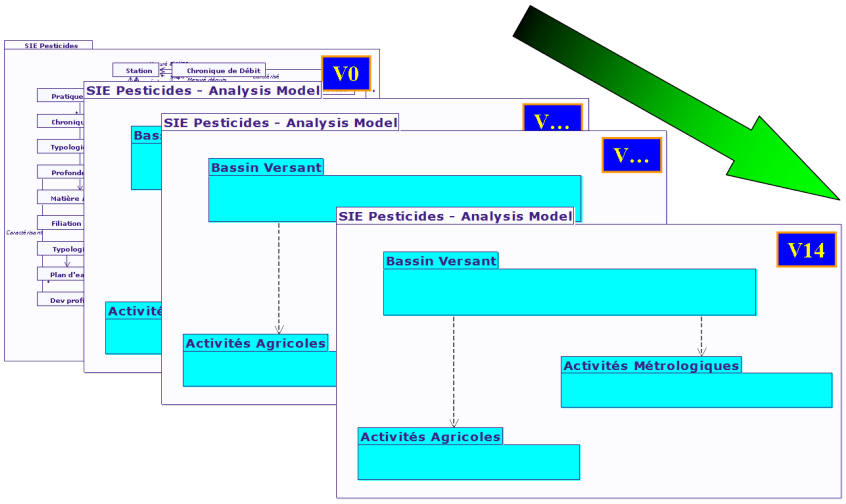


FIGURE 5.1: **Versionnement du modèle *SIE-Pesticides***

Ces archives constituent un capital de connaissances qui évolue avec l'avancement du projet. En effet, le nombre d'éléments de modélisation augmente avec l'avancement de projet, donc des versions du modèle. Cette augmentation est due soit à la prise en compte d'un nouveau besoin soit une restructuration (généralisation/spécialisation) des concepts existants. En d'autres termes, la restructuration permet de mieux répondre aux besoins existants sans en introduire de nouveaux.

5.1.2 Historique des versions du modèle *SIE-Pesticides*

Dans cette section, nous présentons l'historique des différentes versions du modèle *SIE-Pesticides*. À travers cet historique, nous allons suivre les travaux réalisés par les équipes intervenant sur les modèles. Cela permet de suivre l'évolution des besoins au fur et à mesure de la modélisation.

Le problème de capitalisation des données avait été identifié par les équipes impliquées. Aussi, afin de trouver une solution, un étudiant en deuxième année de Master informatique avait été pris en stage pendant six mois par l'équipe travaillant sur le transfert des pesticides. L'analyse effectuée par l'étudiant avait permis d'établir un modèle entités/relations du bassin versant expérimental suivi par cette équipe. La version *V0* du modèle *SIE-Pesticides* est la transposition en langage UML de ce premier modèle. Le bassin versant est instrumenté et suivi depuis plusieurs années tant au niveau de l'évolution des entités du paysage que des mesures de débit ou de pluviométrie ou bien encore des pratiques agricoles. Cette version *V0* ne possède pas de relations de généralisation et donc pas d'abstractions (super-classes) portant sur les concepts thématiques du domaine.

La version *V1* est le résultat de la première session d'analyse avec l'équipe Transfert. Les entités hydrographiques du modèle *V0* ont été mieux affinées en prenant en compte une partie des spécifications de la base de données *BD CARTHAGE* ® [1].

L'apport de l'équipe de Bordeaux est très important au niveau des nouvelles abstractions introduites dans la troisième version du modèle *V2*. Ces abstractions permettent de généraliser plusieurs concepts métiers par un concept d'un niveau plus élevé en abstraction. Bien entendu, beaucoup d'associations ont été déplacées vers ces nouvelles super-classes. La première partie de cette rencontre a été essentiellement focalisée sur la factorisation des concepts métiers du modèle et, en particulier, sur une meilleure représentation des éléments du paysage. Les bassins versants étant imbriqués les uns dans les autres de manière récursive et contenant chacun une partie des entités hydrographiques, ces concepts thématiques ont été modélisés autour d'un patron composite dont le composite est le bassin versant et les feuilles les entités hydrographiques.

Une autre partie de la même réunion a été consacrée à la restructuration du modèle *SIE-Pesticides* en trois "sous-modèles" essentiels qui font l'objet de la quatrième version du modèle *V3* : le paquetage décrivant les objets du paysage des bassins versants, le paquetage représentatif de l'activité agricole sur le territoire et le dernier paquetage consacré à l'activité de métrologie pour suivre le transfert des pesticides vers les cours d'eau.

Au cours de cette restructuration, certains concepts thématiques autour de la métrologie de la version *V0* ne représentant pas bien le métier de l'équipe Transfert ont été abandonnés et remplacés par des concepts plus appropriés. À cette occasion, des concepts plus abstraits ont été introduits, factorisant ainsi manuellement des concepts du domaine. Dans ce travail de restructuration, plusieurs associations ont été déplacées et/ou supprimées.

Au cours d'une autre réunion, le paquetage qui regroupe les concepts métiers de l'activité agricole a été détaillé. Ce travail a impliqué une forte augmentation de nombre de classes (introduction de nouveaux concepts métiers) et, naturellement, du nombre d'éléments du modèle. La version *V4* du modèle est le résultat de cette analyse.

Comme le modèle sur l'activité de métrologie n'était pas satisfaisant, il a été entièrement réanalysé donnant la sixième version (*V5*) du modèle. Pour ce faire, une copie du paquetage a été ef-

1. http ://professionnels.ign.fr/ficheProduitCMS.do ?idDoc = 5323714

fectuée afin d'éviter la perte de concepts. Cela explique la forte progression (cependant artificielle) du nombre d'éléments de modélisation dans le modèle.

La septième version du modèle (*V6*) est tout simplement le résultat d'un travail classique d'analyse et de restructuration des concepts métiers en les spécialisant.

La huitième version du modèle (*V7*) a été réalisée par le chef de projet. Entre deux séances d'analyse, il procède à une opération de nettoyage des concepts redondants suite à la duplication des paquetages décrivant l'activité de métrologie (cf. figure 5.2). Entre autres, l'évolution principale dans ce modèle correspond au déplacement des concepts métiers utiles restant dans la copie. Il en résulte une forte diminution des éléments de modélisation. Une autre évolution importante dans ce modèle réside dans l'ajout de hiérarchies consécutives à une factorisation manuelle. Par exemple, les attributs de même nom des classes appartenant à la même hiérarchie sont déplacés dans les super-classes, réduisant par là le nombre d'attributs dans le modèle.

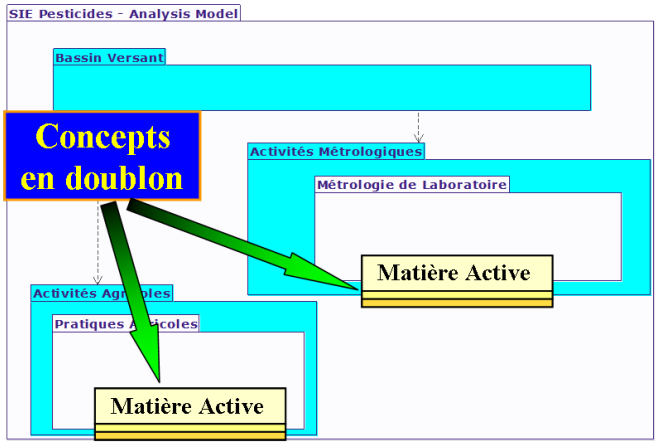


FIGURE 5.2: **Duplication de la classe Matière Active**

Depuis le début de la phase d'analyse, le premier objectif du chef de projet était de faire progresser rapidement la modélisation pour stabiliser globalement le modèle. Pour cette raison, certaines caractéristiques très importantes comme le type des attributs ou comme le nom des rôles mais aussi les multiplicités des associations n'ont pas été définies. Mais, comme par défaut, l'Atelier de Génie Logiciel (cf. section 4.4.1) utilisé lors de la modélisation nomme certains éléments de modélisation, un grand nombre d'attributs sont typés *undefined* tout comme la plupart des noms de rôles des associations. L'essentiel du travail de la neuvième version du modèle (*V8*) a consisté à définir le type des attributs et à préciser toutes les caractéristiques des associations.

Dans la dixième version du modèle (*V9*), un langage pictogrammique [A.Miralles et Libourel, 2009; Miralles et Libourel, 2009] a été mis en œuvre pour indiquer la spatialité (point, ligne et polygone) et la temporalité (instant et période) des concepts du domaine. Ce langage facilite la communication avec les thématiciens ce qui permet de mieux capturer les concepts du domaine et de stabiliser le modèle plus rapidement. Lors de cette opération, certains attributs introduits pour représenter la géométrie ou le temps dans les concepts sont supprimés et remplacés par des sté-

réotypes, éléments de modélisation permettant de porter un pictogramme. Il en découle une forte baisse du nombre d'attributs dans le modèle *V9*.

Dans la onzième version du modèle (*V10*), aucun changement important n'a été effectué à part la poursuite du processus classique d'analyse tout en intégrant le langage pictogrammique.

La douzième version du modèle (*V11*) est le résultat d'une refonte importante du paquetage des activités agricoles. L'ajout essentiel est un modèle de gestion prévisionnelle de l'activité agricole à deux échelles : la gestion prévisionnelle au niveau des exploitations et au niveau sectoriel. En outre, la description du sol et du sous-sol qui sont dans le paquetage paysage, a également été fortement retouchée et beaucoup d'attributs, d'associations et des nouvelles abstractions de généralisation ont été introduits. Dans cette partie du modèle, des groupes de classes présentes uniquement pour une raison de spécification, n'ayant pas d'attribut et appartenant toutes à la même hiérarchie ont été remplacés par des énumérations. Par conséquent, plusieurs classes et plusieurs associations ont été supprimées.

L'évolution vers la treizième version du modèle (*V12*) concerne l'ajout d'une classe et d'un attribut qui résulte de l'impossibilité de spécialiser une énumération (type de récoltes) en UML dans deux énumérations (type de récoltes annuelles et récoltes pérennes). Ceci a conduit à introduire dans la quatorzième version du modèle (*V13*) une hiérarchie de classes représentant le type de récolte, le type de récolte annuelle et le type de récolte éternelle. Naturellement, des attributs, des associations et des généralisations ont été ajoutés pour reconnecter ces classes au reste du modèle.

La quinzième et dernière version du modèle (*V14*) est le résultat d'un travail de réorganisation dû à l'ajout d'une classe plus abstraite.

5.1.3 Présentation de la problématique

Pour le modèle *SIE-Pesticides*, la capture de la multitude des concepts thématiques manipulés par les équipes scientifiques impliquées a posé un réel problème. Étant donné qu'il est difficile de réunir l'ensemble des chercheurs simultanément pour faire l'analyse, elle est effectuée au cours de séances avec les équipes thématiques, entrecoupées de réunions de consolidation permettant de croiser et d'unifier les points de vue sur les concepts thématiques ou sur des sous-ensembles du modèle UML. Nous disposons donc à ce jour de quinze versions archivées du modèle UML. Les modèles sont structurés en paquetages, chacun d'eux modélisant un "thème", comme par exemple les éléments du paysage d'un bassin versant (parcelle, cours d'eau, barrage, etc.) ou l'instrumentation mise en œuvre pour étudier l'impact des pesticides.

Ce processus d'analyse pose un certain nombre de problèmes. En effet, il n'est pas rare que des concepts thématiques soient modélisés deux fois dans deux sous-ensembles (paquetages) différents du modèle global ou que des concepts de niveau supérieur d'abstraction ne soient pas identifiés, etc. Confrontés à ces problématiques, nous avons mis en œuvre d'une part, l'Analyse Formelle de Concepts et d'autre part, l'Analyse Relationnelle de Concepts afin d'évaluer le potentiel de regroupement de concepts thématiques similaires, mais aussi la possibilité de faire émerger de nouveaux concepts thématiques plus abstraits.

Dans ce chapitre , nous rapportons une expérimentation portant sur l'analyse de l'évolution du modèle de classes du système d'information *SIE-Pesticides* au travers de ses quinze versions successives. Nous appliquons un ensemble de métriques sur les modèles successifs pour analyser cette évolution. Ensuite, nous établissons des recommandations qui devraient permettre de suivre

l'évolution des modèles dans d'autres projets de développement. Nous utilisons des métriques sur le modèle de classes mais aussi des métriques issues des treillis successifs générés par l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts.

Nous avons conçu des méthodes et des outils permettant d'obtenir automatiquement ou quasi-automatiquement ces différentes métriques. Les outils ont été implémentés dans un Atelier de Génie Logiciel (cf. section 4.4.1).

5.2 Analyse de l'évolution basée sur des métriques portant sur le modèle

Notre objectif est de suivre l'évolution des différentes versions du modèle d'un système d'information au cours de son cycle de vie. Pour ce faire, nous mettons en œuvre des mécanismes pour que les acteurs puissent suivre cette évolution tout en mesurant le gain (la valeur ajoutée) en abstraction d'un modèle tout le long de son cycle de vie. Notre approche consiste à extraire et à analyser les métriques d'une part, sur les quinze versions du modèle et, d'autre part, sur les treillis résultant de l'application de l'Analyse Formelle de Concepts et de l'Analyse Relationnelle de concepts.

5.2.1 Métriques basées sur le nombre d'éléments de modélisation

Notre première tâche consiste à étudier cette évolution via des mesures quantitatives sur les aspects structurels des quinze versions du modèle *SIE-Pesticides*, à savoir le nombre d'éléments de modélisation tels que les classes, les attributs, les opérations, les associations et les rôles. La seconde tâche consiste à considérer l'évolution des ratios (les rapports) entre éléments de modélisation.

Dans un premier temps les métriques étudiées seront : le nombre de classes ($\sum_{Classes}$), le nombre d'attributs ($\sum_{Attributs}$), le nombre d'opérations ($\sum_{Opérations}$), le nombre d'associations ($\sum_{Associations}$) et le nombre de rôles nommés ($\sum_{Rôles}$). La figure 5.3 montre l'évolution de ces métriques.

Dans la suite de cette section, l'analyse des métriques s'appuie sur l'historique du modèle développé dans le cadre du projet *SIE-Pesticides* (cf. 5.1.2).

5.2.1.1 Évolution du nombre de classes (#Classes)

Une tendance croissante du nombre de classes est observée pour les quinze versions du projet *SIE-Pesticides*. En effet, ce nombre passe de 36 classes pour la version *V0* à 171 pour la quinzième version, malgré certaines baisses qui s'expliquent par le processus d'analyse. Dans un premier temps, nous remarquons une augmentation du nombre de classes de *V0* à *V2* qui correspond au travail d'analyse approfondie effectué au cours de cette période. Ce travail se poursuit par l'ajout de nouveaux concepts entre *V3* et *V4*. L'augmentation entre *V4* et *V5* est principalement due à la copie du modèle sur les activités de métrologie.

La baisse du nombre de classes entre *V6* et *V7* correspond à la suppression de la copie du modèle sur les activités de métrologie.

Une quasi-stabilité du nombre de classes est observée entre *V8* et *V10* puis entre *V11* et *V14*. Au cours de ces deux séries de modèles, le travail d'analyse a essentiellement porté sur d'autres

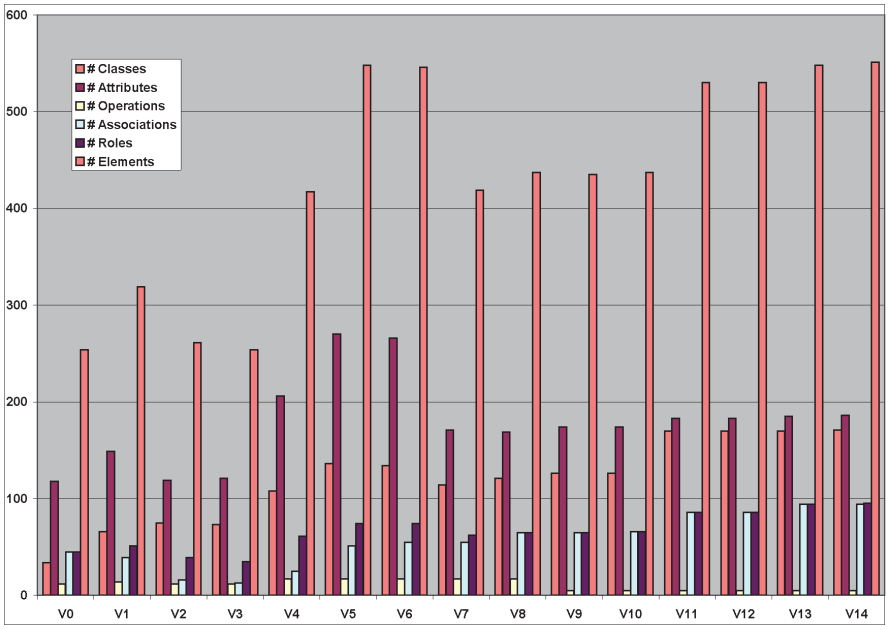


FIGURE 5.3: **Évolution du nombre d'éléments de modélisation**

éléments de modélisation qui seront évoqués dans les paragraphes suivants (opérations en *V8* et associations/rôles en *V13*).

Cependant, la forte progression du nombre de classes et des caractéristiques au cours du processus d'analyse débute depuis la version *V0* jusqu'à la version *V14*. Au début, le nombre de caractéristiques est supérieur au nombre de classes (*V0* à *V2*), il reste du même ordre de jusqu'à la version *V9* et une chute est observée dans la version *V10*. La version *V10* du modèle *SIE Pesticides* est une étape importante dans l'analyse où la description du sous-modèle Sol et du Sous-sol (*V9*→*V10* : 6 à 36 classes) a été fortement affinée ainsi que celle du sous-modèle Gestion de l'Exploitation (*V9*→*V10* : 16 à 23 classes). Cette métrique permet de suivre l'évolution du nombre de classes et elle indique des informations importantes pour les acteurs sur le niveau d'encapsulation des données de leur système d'information. En fait, cette métrique n'est pas en mesure à elle seule de discerner le degré de finalisation d'un modèle par rapport au domaine étudié. Toutefois, elle est un indicateur de la complexité du modèle (cf. figure 5.3).

5.2.1.2 Évolution du nombre d'attributs (#Attributs)

Les attributs sont l'une des principales caractéristiques des classes car ils stockent des informations importantes des concepts du domaine qu'elles représentent.

La figure 5.3 montre une tendance croissante de l'évolution des attributs avec quelques décroissances qui s'expliquent par l'historique des modèles.

Pour ces différentes versions, l'augmentation du nombre indique que les concepts métiers sont décrits de façon plus précise. C'est le cas entre les versions *V0* et *V1*. L'introduction de super-classes (concepts plus abstraits) en *V2* pourrait expliquer la diminution du nombre d'attributs, car des re-

dondances ont été supprimées. La poursuite de l'analyse avec les thématiciens explique la forte augmentation observée en *V4*. Comme pour les classes, la duplication du modèle des activités de métrologie et sa suppression ensuite expliquent d'une part, l'augmentation en *V5* et, d'autre part, la diminution entre les versions *V6* et *V7*. De *V8* à *V14*, nous observons une stabilité générale de cette métrique avec une très légère augmentation entre *V10* et *V11*. Entre ces deux versions, les acteurs ont ajouté quelques précisions sur les concepts métiers.

La contribution de cette métrique pour l'acteur est double. D'une part, elle mesure l'ampleur de la description des classes dans un modèle (faible, moyenne, etc.). D'autre part, l'analyse de son évolution à travers les quinze versions permet d'avoir une vision de l'évolution de la description du modèle (cf. figure 5.3).

5.2.1.3 Évolution du nombre d'opérations (#Opérations)

Les opérations caractérisent également les classes en décrivant leur comportement aussi l'analyse de leur évolution est intéressante.

Dans notre contexte, les opérations ne sont pratiquement pas décrites car le modèle est conçu pour être implémenté par une base de données. Les quelques opérations qui ont été introduites l'ont souvent été pour des raisons pédagogiques afin d'expliquer aux thématiciens que certains concepts de *V0* pouvaient être obtenus par calcul.

Le nombre d'opérations est relativement constant (autour de 14) jusqu'à *V8* et ce nombre chute à une paire d'opérations à partir de *V9* et jusqu'à *V14*.

Ce faible nombre d'opérations est très intéressant car il caractérise cette typologie de modèles destinés aux systèmes d'information. En effet, le modèle SIE Pesticides est dédié à la modélisation d'un système d'information dont la finalité est l'implémentation d'une base de données. De ce fait, l'analyse privilégie la description statique des concepts thématiques et ne s'intéresse que marginalement au comportement thématique de ces mêmes concepts.

Par conséquent, dans notre cas, l'évaluation quantitative de cet élément de modélisation ne fournit pas d'information significative sur l'analyse de l'évolution (cf. figure 5.3). La seule information significative est le typage du modèle : modèle dédié aux systèmes d'information versus "modèle d'interaction".

5.2.1.4 Évolution du nombre d'associations (#Associations)

Cette métrique est un indicateur de la description des relations existant entre les concepts métiers. Ces relations portent aussi des informations telles la multiplicité, la navigabilité, etc.

Nous observons dans la figure 5.3 une légère diminution des associations pour les premières versions (*V0* à *V3*). Elle s'explique par une restructuration profonde du modèle *V0*, modèle réalisé dans le cadre d'un stage de Master.

Ensuite, nous remarquons une légère augmentation en *V4*, suivi d'une augmentation en *V5*. Cette dernière augmentation s'explique sûrement par la duplication du modèle des activités métrologiques. L'augmentation se poursuit régulièrement entre *V6* et *V14* avec un saut en *V11*, version correspond à une refonte importante du modèle des activités agricoles.

Cette faible progression entre *V6* et *V14* est due à l'introduction de super-classes tout au long de l'avancement du projet répondant ainsi aux nouvelles exigences exprimées par les nouveaux experts de différents domaines d'application.

L'analyse de cette mesure exprime d'une part, la dimension relationnelle entre classes d'un modèle et, d'autre part, son évolution (cf. figure 5.3).

5.2.1.5 Évolution du nombre de rôles (#Rôles)

En UML, un rôle représente une extrémité d'association. Il peut être navigable, ce qui signifie qu'un objet possède un accès à l'objet opposé à travers ce rôle. Dans notre contexte, nous prenons en considération le sens de la navigabilité et uniquement les rôles nommés sémantiquement. Les rôles navigables nommés par les experts devraient évoluer de la même manière que les associations dans les différentes versions. Dans le cas contraire, nous en déduirions un manque d'information et/ou de précision de la description des associations. Pour les quinze versions du modèle (cf. figure 5.3), nous observons les mêmes tendances de croissance et de décroissance que pour les associations. Le rôle est une des informations les plus importantes sur les associations dans les modèles UML.

5.2.2 Métriques basées sur le rapport entre éléments de modélisation

Dans cette section, nous avons sélectionné des ratios pour suivre l'évolution des quinze versions du modèle. Ces ratios permettent d'évaluer l'influence d'un élément de modèle par rapport à d'autres.

1 Rapport entre les classes et l'ensemble des éléments de modélisation du modèle : (#Classes / #Éléments du Modèle)

2 Rapport entre les attributs et les classes : (#Attributs / #Classes)

3 Rapport entre les associations et les classes : (#Associations / #Classes)

5.2.2.1 Évolution du rapport classes / éléments de modélisation (#Classes / #Éléments du Modèle)

Cette métrique est définie par le rapport entre le nombre de classes et le nombre d'éléments du modèle ($\#Classes/\#ÉlémentsduModèle$) :

$$\left(\frac{\sum_{Classes}}{\sum_{ÉlémentsduModèle}}\right) \times 100 \tag{5.1}$$

Dans la version *V0*, les classes représentent seulement 15% de l'ensemble des éléments de modélisation. L'analyse avançant, elles atteignent environ 45% dans les dernières versions avec un maximum de 47% pour les versions *V11* et *V12*. Nous allons maintenant examiner les différentes raisons d'augmentation et/ou de baisse de ce ratio.

Ce rapport illustre la proportion de classes dans le modèle. La tendance croissante montre d'une part l'importance des classes dans les modèles, car elles caractérisent les informations décrites par le concepteur du système. D'autre part, ce rapport nous montre le degré de réification à travers les différentes versions du modèle (cf. figure 5.4).

5.2.2.2 Évolution du rapport attributs / classes (#Attributs / #Classes)

Cette métrique est définie par le rapport entre le nombre d'attributs et le nombre de classes ($\#Attributs$ / $\#Classes$) :

$$\left(\frac{\sum_{Attributs}}{\sum_{Classes}} \right) \times 100 \tag{5.2}$$

Dans la version *V0*, le nombre d'attributs par classe est le plus élevé car l'étudiant en stage de Master n'avait pas introduit de généralisation. En moyenne, il y a 3,5 attributs par classe.

Pour les versions suivantes, la factorisation manuelle effectuée lors de l'analyse fait décroître ce ratio (cf. figure 5.3). Nous confirmerons cette observation avec les métriques sur les treillis dans la section suivante.

La forte décroissance observée de *V0* à *V2* est essentiellement due à l'ajout de concepts (classes) sans forcément finaliser leur description. L'augmentation en *V4* est due à une certaine stabilisation des concepts et une description plus approfondie des concepts existants. L'augmentation en *V5* et en *V6* est consécutive à la duplication du modèle des activités métrologiques. La chute entre textitV10 et *V11* résulte de la refonte de modèles des activités agricoles. L'effort d'analyse se traduit sur les dernières versions par un faible nombre d'attributs par classes, qui est à ce moment-là d'un attribut en moyenne.

Ce rapport du nombre d'attributs par classes est intéressant pour le suivi de l'évolution d'un modèle. Il constitue une information sur la description moyenne de ces classes (cf. figure 5.5).

5.2.2.3 Évolution du rapport associations / classes (#Associations / #Classes)

Cette métrique est définie par le rapport entre le nombre d'associations et le nombre de classes ($\#Associations/\#Classes$) :

$$\left(\frac{\sum_{Associations}}{\sum_{Classes}} \right) \times 100 \tag{5.3}$$

Ce ratio mesure l'aspect relationnel entre les différentes classes. Dans la version *V0*, nous avons en moyenne un peu plus d'une association par classe mais cette valeur diminue rapidement dans les versions suivantes.

Comme pour les attributs, sa forte baisse jusqu'en *V3* est due à l'ajout de classes sans finaliser leur description.

Ensuite, ce rapport augmente de *V4* à *V8* lorsque d'autres relations sont ajoutées par les acteurs et par l'effort de factorisation manuelle effectué par le chef de projet en transformant certaines associations en des relations de spécialisation et/ou de généralisation. Finalement, ce rapport se stabilise jusqu'à la version *V14*.

L'intérêt de ce rapport pour l'acteur est de mesurer pour un modèle donné le niveau moyen de la description relationnelle entre ces classes (cf. figure 5.5).

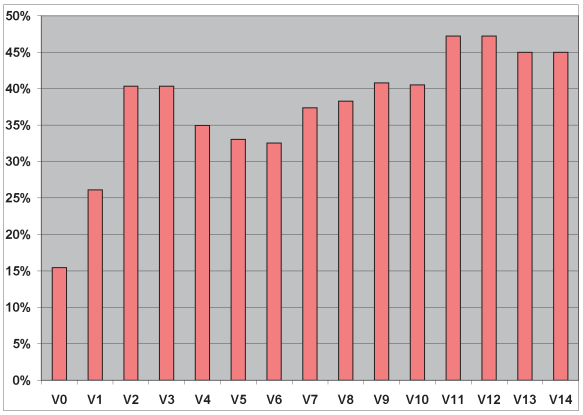


FIGURE 5.4: **Évolution des classes par rapport aux éléments du modèle**

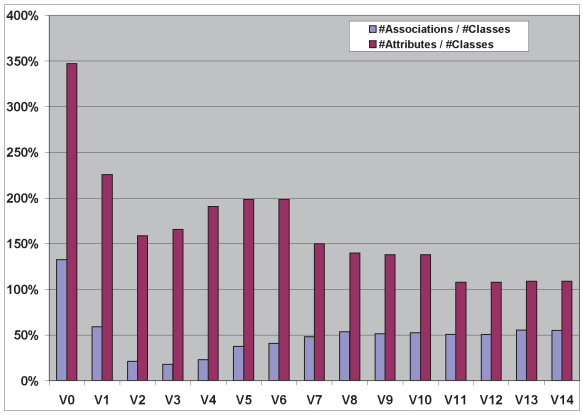


FIGURE 5.5: **Évolution des attributs et des associations par rapport aux classes**

5.3 Analyse de l'évolution basée sur des métriques issues de l'Analyse Formelle de Concepts

L'Analyse Formelle de Concepts [Ganter et Wille, 1999] regroupe des concepts métiers de même nom et fait apparaître de nouveaux concepts métiers plus abstraits, auxquels les scientifiques n'auraient pas pensé, étant donné qu'ils ne sont pas nécessairement focalisés sur le modèle global. De fait, l'expérience montre que les scientifiques concentrent leurs efforts d'analyse sur les parties du modèle correspondant à leurs recherches.

Par ailleurs, l'Analyse Formelle de Concepts fusionne les concepts métiers dupliqués d'une manière systématique, tout en ajoutant de nouvelles abstractions au sein d'un processus robuste qui conduit à une solution unique. Pour le modèle de classes du système d'information, elle génère des treillis, artefacts équivalent à une forme normale (solution unique et sans redondance).

L'Analyse Formelle de Concepts a été utilisée dans plusieurs domaines au cours des dernières années et notamment pour l a refactorisation de modèles de classes [Godin et Mili, 1993]. Elle intervient sous forme itérative dans l'Analyse Relationnelles de concepts (ARC [Huchard *et al.*, 2007]) pour la prise en compte de la richesse des relations entre les classes, attributs, méthodes, etc.

Dans cette utilisation maintenant bien connue, l'Analyse Formelle de Concepts (AFC) a pour rôle de faire émerger au sein d'un modèle des concepts thématiques ayant un niveau d'abstraction supérieur en vue d'améliorer le modèle. Ici, nous montrons que l'Analyse Formelle de Concepts

peut aussi être employée pour suivre l'évolution de la factorisation dans le processus d'analyse avec des acteurs. Nous en avons conçu une méthode, fondée sur des recommandations et des métriques, permettant de suivre et de vérifier la bonne évolution du processus d'analyse sur ce plan.

Puisque que l'Analyse Formelle de Concepts produit une forme normale, elle offre un cadre d'étude pour comparer les versions successives du modèle, en particulier les nouvelles abstractions issues du potentiel de la factorisation du modèle. Ainsi, les métriques sur ces formes normales complètent l'information fournie par les métriques précédentes sur le modèle et précisent les observations.

5.3.1 Description des expérimentations

La figure 5.6 est un exemple d'un treillis montrant des concepts formels avec leurs intensions et leurs extensions. Elle montre aussi l'héritage descendant des caractéristiques et l'héritage ascendant des individus. L'intension propre d'un concept formel est l'intension de ce concept privée des caractéristiques des intensions des super-concepts. L'extension propre est définie de manière inverse.

Le Concept formel Top, à la racine du treillis, regroupe dans son intension les caractéristiques communes à toutes les classes du modèle. Si l'intension du top est vide, les classes qui y figurent sont exemptes de caractéristiques.

Les *concepts formels fusion* sont des concepts qui possèdent plus d'une classe dans leur extension propre. Cela signifie que les classes de l'extension propre sont décrites exactement par les mêmes caractéristiques.

Les *concepts formels créés* sont des concepts dont l'extension propre est vide. Ce sont de nouveaux concepts formels factorisant des caractéristiques communes à plusieurs concepts thématiques.

Dans nos expérimentations, nous avons choisi cinq relations différentes (cf. section 4.3.1) pour appliquer l'Analyse Formelle de Concepts aux quinze versions du modèle du projet *SIE-Pesticides*. Ces relations sont constituées par les principaux éléments de modélisation utilisés pour modéliser un système d'information dans un cadre général, à savoir les classes, les attributs, les opérations et les rôles.

Afin de faciliter la génération des contextes formels et des treillis pour l'ensemble des versions et pour toutes ces relations, un profil UML a été développé sous Objecteering (cf. section 4.4.1, pour transformer les modèles UML en contextes formels, eux-mêmes convertis ensuite en treillis.

Cette technique de profil nous a aussi permis d'intégrer les fonctionnalités implémentées dans le framework eRCA (Eclipse Relational Concept Analysis)(cf. section 4.4.2) développé par [Falleri, 2009], permettant la manipulation des treillis sur la base de transformations de modèles.

Avec ce profil, il a été possible de multiplier les expérimentations, de produire de nombreux treillis et l'ensemble des métriques d'analyse associées. Ces métriques permettent de suivre l'évolution des factorisations tout au long des versions. La définition de ces métriques et l'analyse des principales métriques sont décrites dans la section suivante.

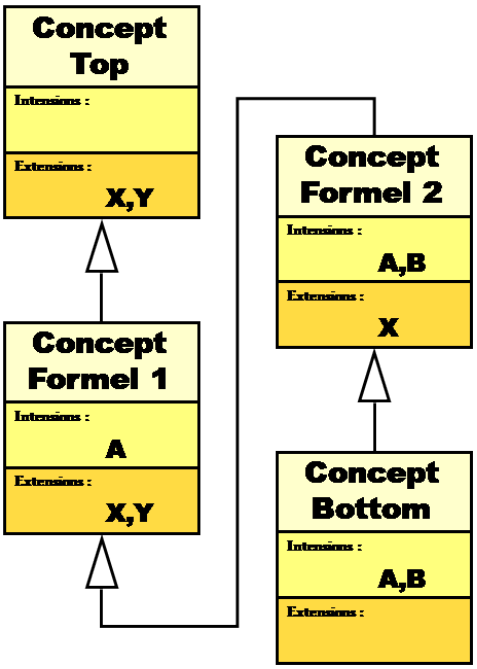


FIGURE 5.6: **Exemple de treillis**

5.3.2 Quantification de l'évolution des éléments de modélisation

L'analyse visuelle des treillis (cf. les deux colonnes de gauche Table 5.1) montre que le modèle se complexifie au fur à mesure des versions et cela passe par une augmentation du nombre de concepts formels dans ces treillis. Dans cette section, nous allons nous consacrer à l'analyse des concepts formels fusion (aussi appelés "fusionnés") qui sont de couleur orange. Les concepts formels de couleur bleu sont les concepts créés.

D'après l'historique (cf. section 5.1.2), les dernières versions du modèle du projet *SIE Pesticides* sont dans un stade de finalisation bien avancé. Il en est de même pour les concepts thématiques qui sont introduits dans le modèle mais leur description par les attributs et les opérations n'est pas totalement finalisée. On retrouve cette analyse avec l'Analyse Formelle de Concepts puisque certains concepts thématiques, ceux qui sont sans attributs ni opérations sont factorisés dans l'extension au sein du Concept formel **Top**. Le nombre de concepts thématiques dans l'extension du Concept formel **Top** est une métrique caractérisant le *"degré de finalisation d'un modèle pour la relation R1"* qui associe les attributs aux classes auxquelles elles sont attachées.

Il est important de noter que toute métrique doit être interprétée et ne sert qu'à mettre le focus sur des parties de modèle à examiner plus attentivement et que chaque relation choisie pour l'analyse a ses propres intérêts et limites. Par exemple, le concept **Top** de plusieurs treillis contient une classe abstraite **ComposantDeBassin** qui joue le rôle de la classe Composant d'un patron composite, patron de concept introduit dès la version *V2* pour organiser l'imbrication des bassins versants les uns dans les autres. Cette classe n'a pas d'attribut car les concepts thématiques qu'elle abstrait sont de nature très différente et, de ce fait, ils n'ont pas de caractéristiques communes. Pour la re-

lation R1, la classe **ComposantDeBassin** est toujours dans le concept formel **Top** depuis *V2* jusqu'à *V14* (cf. figure 5.11). Pourtant, cette partie du modèle est stabilisée dès la version *V2*. Si on intéresse à la relation *R5*, la classe **ComposantDeBassin** appartient au concept formel **Top** jusqu'en *V6* et disparaît ensuite (cf. figure 5.12). Elle n'aurait pas dû apparaître dans le concept formel **Top** dès la *V2*,. Cela provient simplement du fait que le rôle de l'association père-fils du patron composite n'était pas nommé (il était *undefined*). Dès que ce dernier a été nommé en *V7* (cf. figure 5.13), la classe **ComposantDeBassin** a disparu du concept formel **Top**. Cet exemple montre bien que le degré de finalisation est une métrique à manipuler avec précaution car elle est très sensible à la relation étudiée.

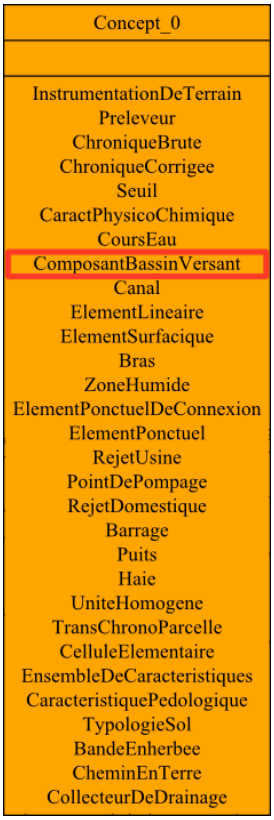


FIGURE 5.7: **Concept Top V2**

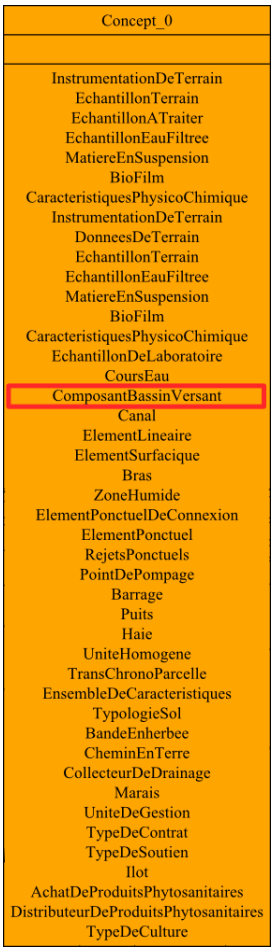


FIGURE 5.8: **Concept Top V6**

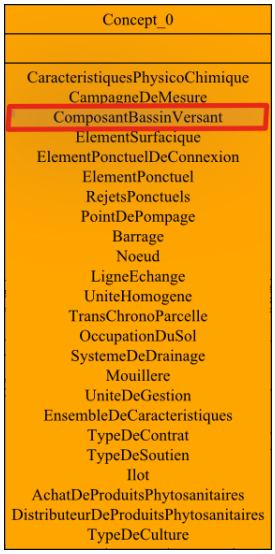


FIGURE 5.9: **Concept Top V7**

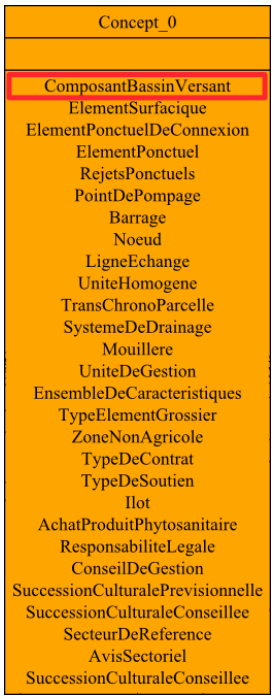


FIGURE 5.10: **Concept Top V14**

FIGURE 5.11: **Concept Top V2, V6, V7 et V14 pour la relation R1=$owns_1 \subseteq classes \times attributes$ (cf. section 4.3.1.2)**

Les treillis des tables 5.1 et 5.2 (colonnes de gauche) montrent l'évolution des treillis pour les

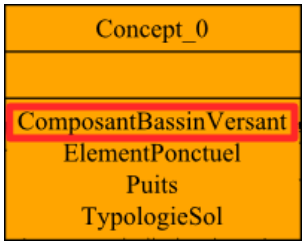


FIGURE 5.12: **Concept Top V6**

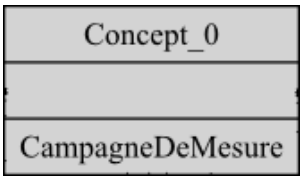


FIGURE 5.13: **Concept Top V7**

FIGURE 5.14: **Concept Top V6 et V7 pour la relation R5=**$owns_5 \subseteq classes \times attributs \cup opérations \cup rôles$ **(cf. section 4.3.1.6)**

relations *R1* (Classe/Attribut) et *R3* (Classe/Rôle). Les analyses ci-après sont relatives à ces deux relations et, dans certains cas, il est nécessaire de conforter l'analyse avec les treillis produits par d'autres relations.

En *V0*, les extensions des concepts formels **Top** pour les relations *R1* et *R3* sont vides ou peu fournies. Cela montre que, dans le périmètre des besoins fixés à l'étudiant de Master, l'analyse est aboutie. Tous les concepts thématiques ont au moins une caractéristique mais celle-ci n'est pas partagée.

Globalement, il est possible de voir une progression des concepts thématiques pour les versions *V1*, *V2* et *V3* suivi d'une brusque augmentation pour la *V4* correspondant à l'analyse approfondie du sous-système relatif aux activités agricoles avec l'équipe de Bordeaux. Ce nombre tend à baisser légèrement pour *V6* et *V7* résultant du travail d'analyse. Une augmentation est observée en *V8* suivie d'une tendance stable pour les versions de *V9* à *V14*. Les versions *V4* et *V8* correspondent à des restructurations profondes du modèle.

Outre le concept formel **Top** très fourni, le treillis *V14* contient un concept formel fusionné (orange) regroupant un nombre important de concepts thématiques. Ce regroupement vient de l'introduction assez systématique d'un attribut "*Commentaire*" dans la description des concepts thématiques.

La colonne de droite (cf. tables 5.1 et 5.2) montre quant à elle le nombre de classes (barres en mauve) dans les extensions des treillis qui représentent dans notre cas des concepts thématiques. Les nombres de caractéristiques (barres bordeaux) sont issues des relations *R1* à *R5* qui prennent en compte les éléments de modélisation tels que les attributs, les opérations ou bien la combinaison de ces caractéristiques.

On peut remarquer le faible nombre de concepts fusionnés ou créés par l'application de la relation *R2* qui s'intéresse à la factorisation des opérations des classes. Ce résultat n'est pas surprenant au vu faible du nombre d'opérations du modèle.

Il est également possible d'observer que le nombre de concepts fusionnés est faible au début de la modélisation et croît avec le nombre de concepts thématiques introduits lors des séances d'analyse, ce qui est un comportement normal.

Le ratio du nombre de concepts formels fusionnés ou créés sur le nombre de classes d'un modèle est aussi une métrique adimensionnelle intéressante comme nous allons le voir.

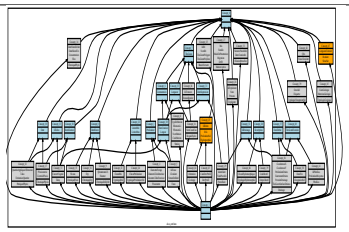
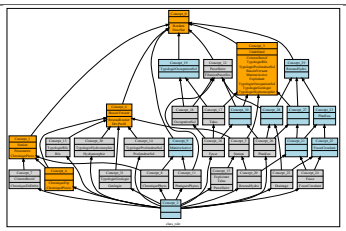
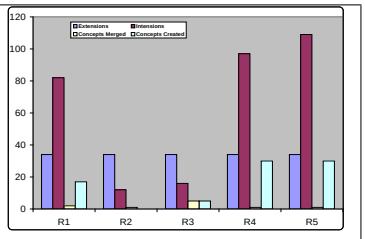
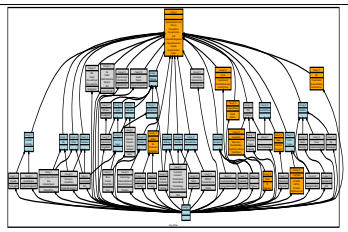
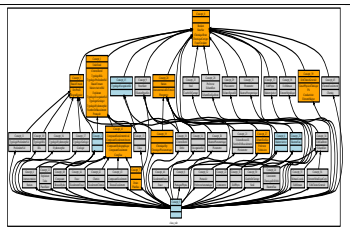
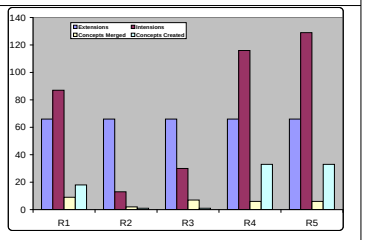
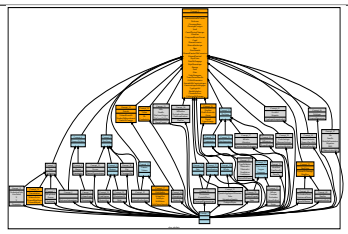
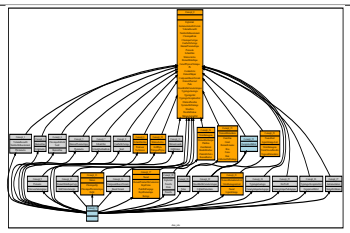
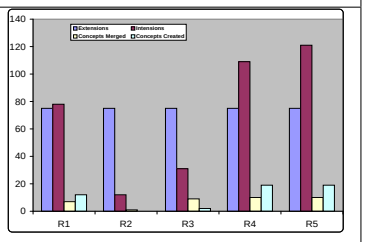
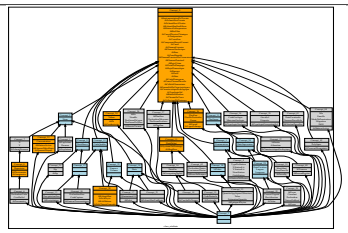
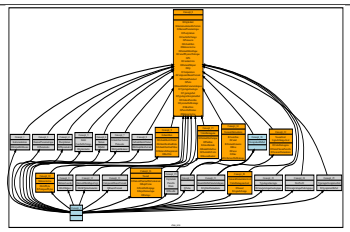
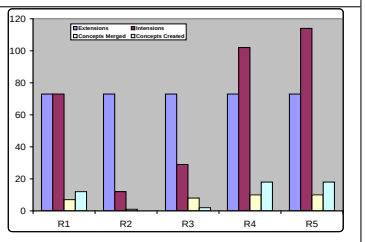
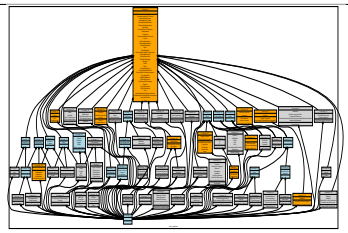
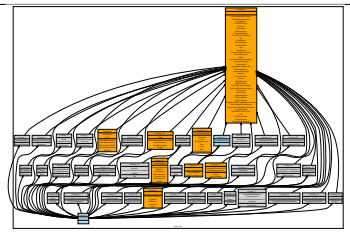
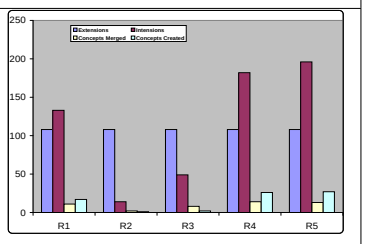
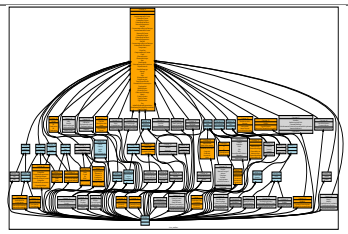
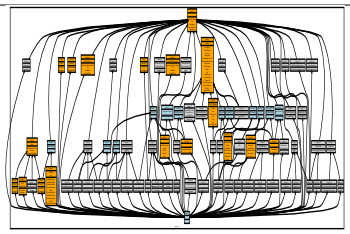
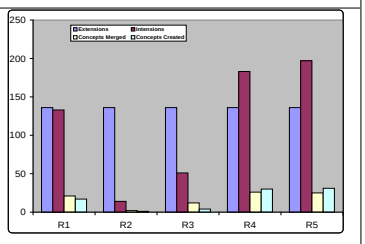
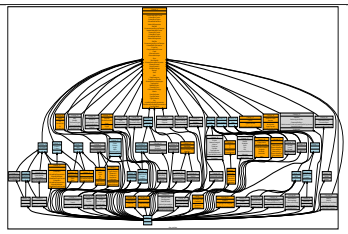
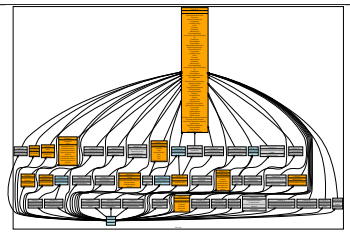
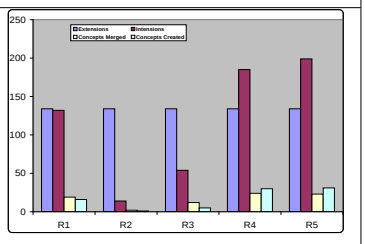
	Relation Classe/Attribut (R1)	Relation Classe/Rôle (R3)	Concepts Créés et Fusionnés
V0			
V1			
V2			
V3			
V4			
V5			
V6			

TABLE 5.1: **Tableau montrant l'évolution des treillis et du nombre de concepts fusionnés ou créés par l'Analyse Formelle des Concepts**

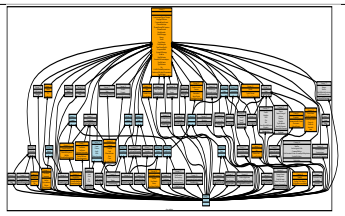
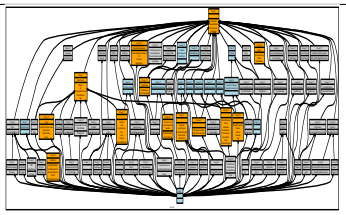
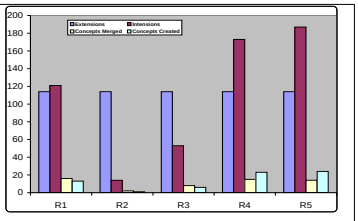
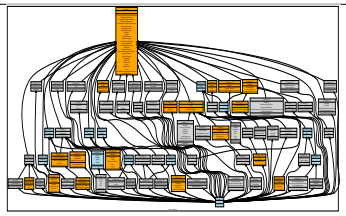
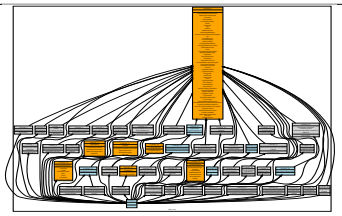
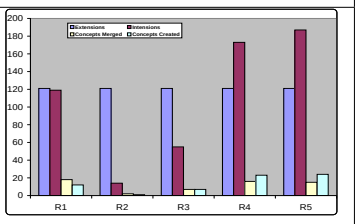
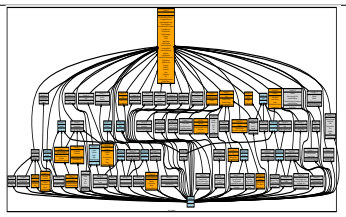
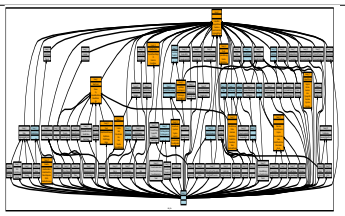
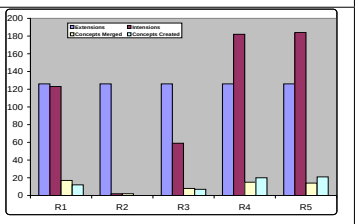
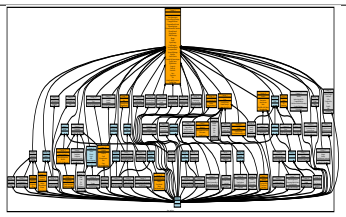
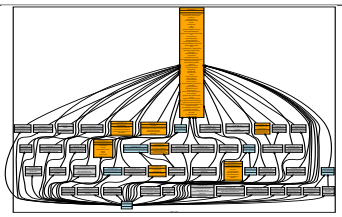
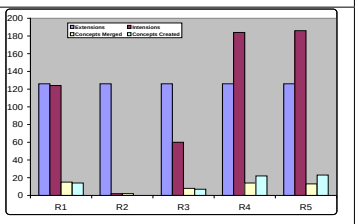
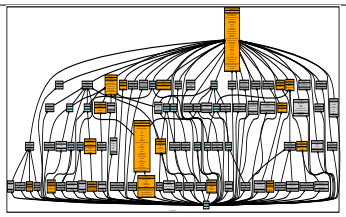
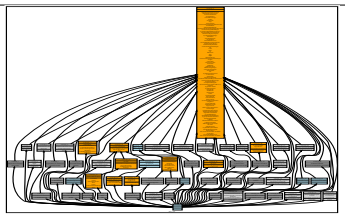
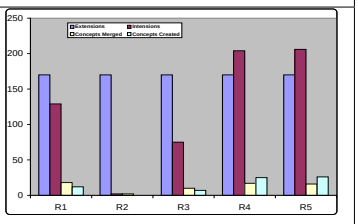
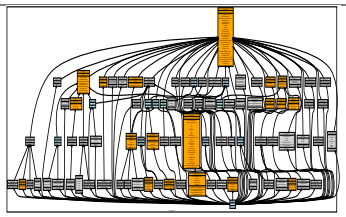
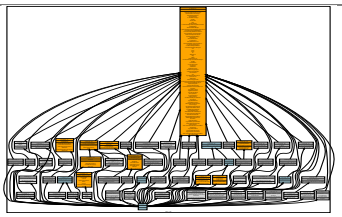
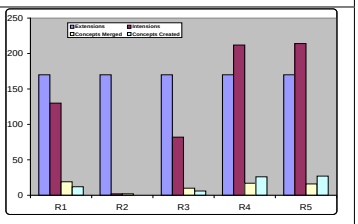
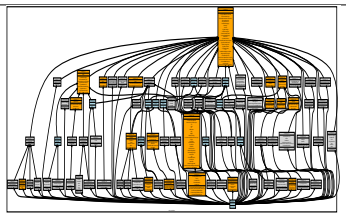
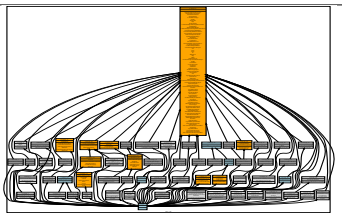
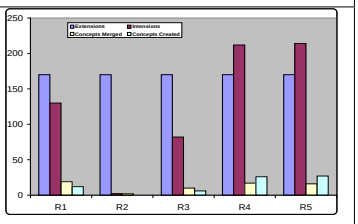
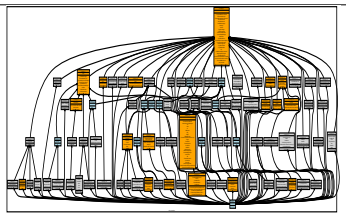
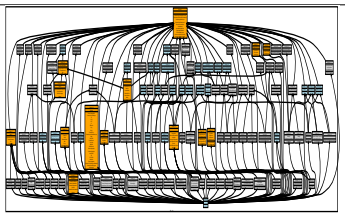
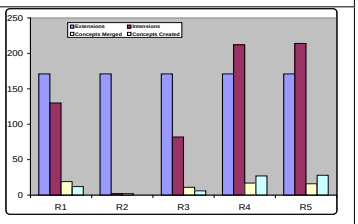
	Relation Classe/Attribut (R1)	Relation Classe/Rôle (R3)	Concepts Créés et Fusionnés
V7			
V8			
V9			
V10			
V11			
V12			
V13			
V14			

TABLE 5.2: **Tableau montrant l'évolution des treillis et du nombre de concepts fusionnés ou créés par l'Analyse Formelle des Concepts**

Le graphique de la figure 5.15 relatif à la relation *R3* implique aussi bien la factorisation des attributs que celle des opérations qui, comme on l'a vu précédemment, intervient marginalement. Ce graphique montre l'évolution de ce ratio pour tous les modèles *SIE Pesticides*. Dans ce graphique, le nombre total de concepts formels fusionnés ou créés (barres couleur paille) décroît malgré la multiplication par cinq du nombre de concepts thématiques. L'origine de cette décroissance vient surtout d'une forte diminution des concepts créés (barres bordeaux). Cela signifie que les concepts thématiques introduits au fur et à mesure des séances d'analyse sont des concepts de niveau d'abstraction supérieur qui factorisent les premiers concepts thématiques et que le modèle a bien évolué. La diminution du nombre de concepts fusion indique, de son côté, que la description a été approfondie. Le graphique est révélateur d'une modélisation qui suit un cours favorable.

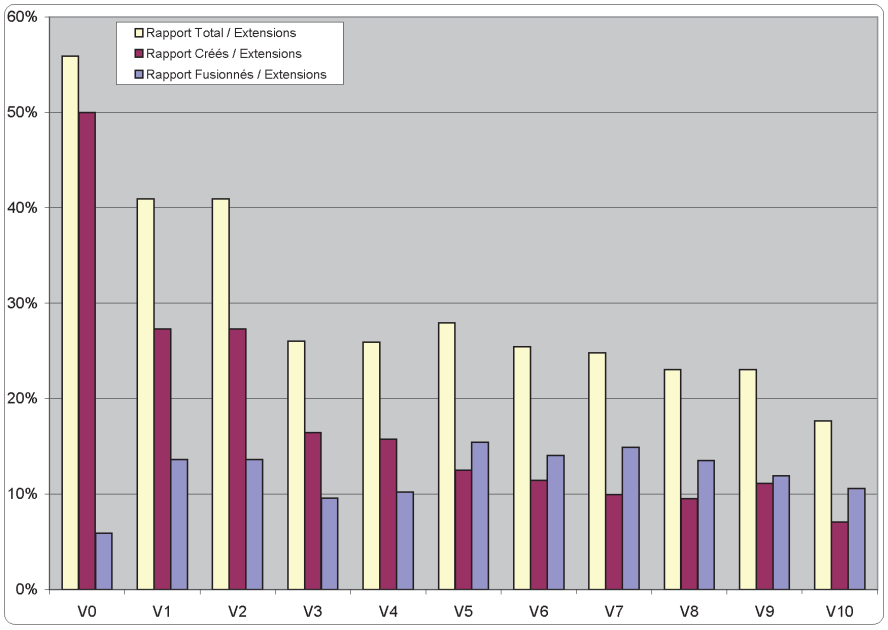


FIGURE 5.15: **Évolution de concepts total, fusionnés et créés (Relation *R3*)**

5.3.3 Vers une méthode d'analyse de l'évolution d'un modèle

Sans faire appel à l'Analyse Formelle de Concepts, l'évolution, en nombre et pourcentage (cf. section 5.2.1) des éléments de modélisation dans les différentes versions d'un modèle constitue une source d'information sur le processus d'analyse. En effet, les ruptures de ces métriques entre deux versions successives indiquent des temps forts dans ce processus. Ces ruptures se traduisent soit par une augmentation de ces métriques suite à la participation d'une nouvelle catégorie d'acteurs ou à l'analyse d'un sous-système du fait de la prise en compte de nouveaux besoins, soit par une réduction résultant par exemple de l'abandon d'un sous-système mais plus vraisemblablement d'une analyse approfondie.

L'Analyse Formelle de Concepts produit des treillis qui factorisent les concepts thématiques au sein de concepts formels éliminant ainsi les redondances de caractéristiques (attributs+opérations)

décrivant les concepts thématiques. De ce fait, un treillis peut être vu comme une représentation normalisée du modèle initial [Godin et Valtchev, 2005]. Dans l'espace des treillis, la normalisation offre un cadre de référence "canonique" facilitant la comparaison des modèles. L'application de cette approche à différentes versions de modèle permet de mettre en évidence un certain nombre de recommandations pour guider l'avancement du processus d'analyse d'un système d'information.

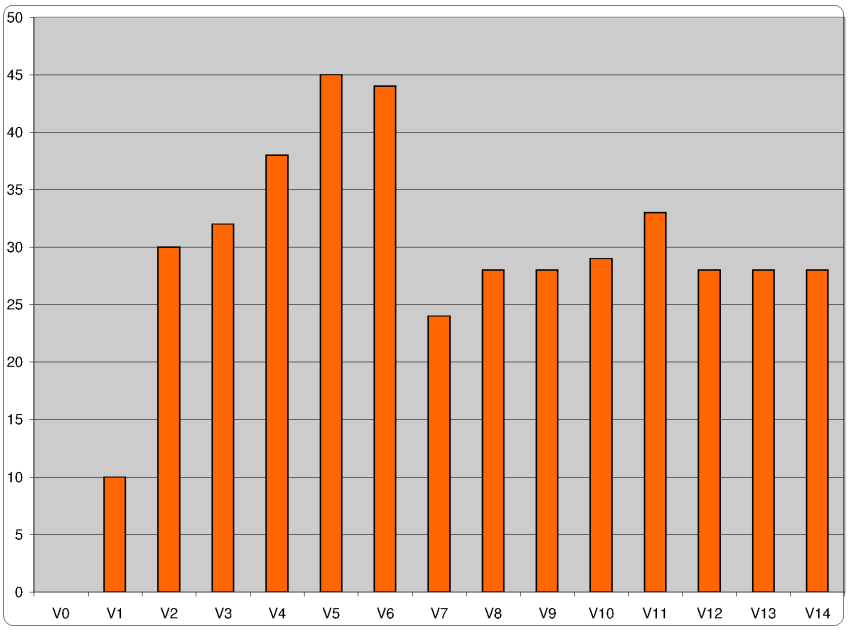


FIGURE 5.16: **Évolution du concept Top**

Les concepts thématiques, dont la description des caractéristiques (attributs et/ou opérations) n'est pas finalisée, sont factorisés dans l'extension du concept formel **Top** du treillis. Si la présence de ces concepts thématiques au sein du concept formel **Top** (cf. figure 5.17) se prolonge au cours des versions (cf. figure 5.16), il faut s'interroger sur la pertinence de ces concepts sachant qu'il faut confronter ou affiner l'analyse avec d'autres treillis produits avec d'autres relations.

Les concepts formels fusionnés (orange) regroupent les concepts thématiques (classes) ayant des descriptions communes. Lorsque le nombre de concepts thématiques dans l'extension est important et qu'il n'évolue pas à la baisse avec les versions successives, il faut vérifier si certains d'entre eux ne représentent pas le même concept thématique sous des noms différents. Lors de cette analyse on pourra souvent laisser de côté le Top surtout si son intension est vide.

Au cours du processus d'analyse, si le pourcentage de concepts formels créés tend à augmenter au fur et à mesure des versions, il faut reconsidérer la qualité de l'analyse en cours. En effet, lorsqu'un modèle est retravaillé avec des acteurs pour approfondir la description et le comportement des concepts thématiques, le concepteur fait émerger tout naturellement des concepts thématiques plus abstraits qui factorisent bien évidemment les concepts thématiques déjà dans le modèle. Il en résulte forcément une diminution nombre de concepts créés (relativement au nombre de classes

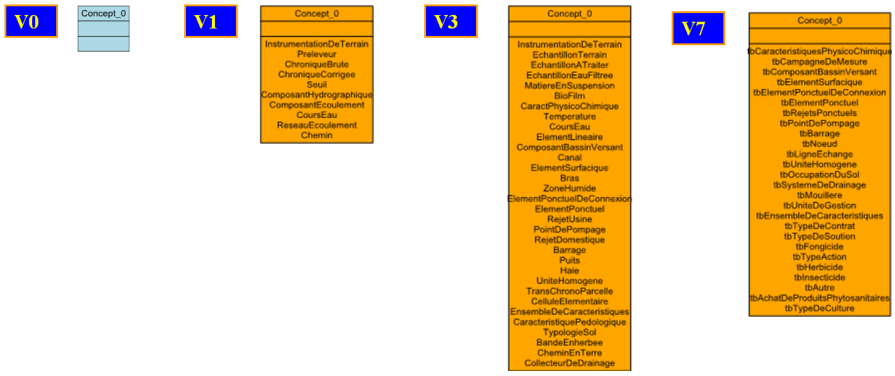


FIGURE 5.17: **Évolution du concept Top**(Relation *R1*)

introduites). Cette évolution est particulièrement perceptible dans le rapport entre le nombre de concepts créés et le nombre de concepts du modèle (de classes) (cf. figure 5.15).

5.4 Analyse de l'évolution basée sur des métriques issues de l'Analyse Relationnelle de Concepts

Dans cette section, nous élargissons la méthode décrite dans la section précédente en ajoutant l'aspect relationnel entre les éléments de modélisation par application de l'Analyse Relationnelle de Concepts. L'Analyse Relationnelle de Concepts est utilisée de manière itérative dans le processus de suivi de l'évolution. Son intérêt majeur est qu'elle prend en compte toute la richesse des relations entre les éléments de modélisation (Classes, Attributs, Méthodes, Associations, Rôles, etc.) et produit une forme normale intégrant cette dimension. Comme précédemment, nous appliquons donc cette méthode aux quinze versions du modèle *SIE-Pesticides* afin d'en étudier l'évolution. En outre, comme nous nous intéressons à des propriétés spécifiques au modèle de classes, comprenant le niveau de précision et le niveau d'abstraction, nous introduisons des métriques sur les treillis résultant. Ces treillis sont générés à partir du framework eRCA (cf. 4.4.2) qui implémente l'Analyse Relationnelle de Concepts (ARC), une variante itérative de l'Analyse Formelle de Concepts (AFC). Ces métriques mesureront le nombre de concepts fusionnés, créés et pérennes (ou immuables). Ces derniers sont les concepts qui contiennent un seul élément dans leur extension propre et ne participent pas à la factorisation.

L'interprétation et l'appréciation de ces métriques nous permettent également d'évaluer le niveau de finalisation et d'avancement de chacun des modèles pris séparément des autres, ainsi que le niveau d'optimisation des uns par rapport aux autres. L'analyse des métriques permet de suivre le taux de concepts fusionnés, créés et immuables dans le processus de factorisation. Enfin, l'analyse visuelle des treillis des quinze versions du modèle montre que ce dernier devient progressivement plus complexe. Le nombre de concepts formels augmente avec la même tendance au cours du cycle de vie de l'application.

5.4.1 Description des paramètres de factorisation

Dans notre contexte, l'application de l'Analyse Relationnelle de Concepts aux modèles d'UML implique les noms des éléments de modélisation : classe, attribut, association, rôle des associations, etc. (cf. chapitre 4). Il convient de noter que l'Analyse Relationnelle de Concepts peut être appliquée à une de ces familles (par exemple les classes) ou combinaisons plus ou moins complexes de familles : classe/attribut, classe/attributs/rôles, etc. Pour un modèle donné, le nombre de concepts nouveaux produits est directement lié à la combinaison entre les familles adoptées.

L'exploration du modèle peut être effectuée sous de nombreuses formes : notre approche retient deux configurations nommées C1 (cf. section 4.3.2.1) et C2 (cf. section 4.3.2.2) qui sont composées d'une combinaison entre relations des principaux éléments de modélisation tels que des classes, attributs, opérations, associations et rôles. En outre, le sens de la navigabilité est pris en compte (cf. section 4.3.2).

Pour générer les concepts formels et les treillis pour toutes les versions nous utilisons le profil d'UML développé dans Objecteering pour effectuer la transformation des modèles d'UML vers les contextes formels et celle des contextes formels vers les treillis (cf. section 4.4.2).

Dans toutes les expérimentations, nous avons choisi de prendre en compte la navigabilité, qui est une information importante sur les associations ; elle détermine la direction d'envoi des messages et elle ajoute une contrainte relationnelle entre les deux classes reliées par l'association. En pratique, nous choisissons seulement les rôles navigables. En conséquence, la factorisation extraite à partir du treillis correspond à la structure relationnelle définie par les modèles. La concordance du modèle vis-à-vis du treillis produit est préservée, ce qui serait essentiel pour une reconstruction du modèle à partir du treillis car la factorisation déduite sera compatible avec les aspects structurels et relationnels du modèle initial.

Nous avons donc appliqué l'Analyse Relationnelle de Concepts sous ses deux configurations à chacune des quinze versions du modèle *SIE-Pesticides*. En outre, comme l'Analyse Relationnelle de Concepts est de nature itérative, nous avons prévu dans le profil de générer les treillis et les métriques associées aux différents pas de l'itération. Le nombre d'itérations est spécifique à chaque modèle et dépend de la complexité du modèle mais aussi de la configuration choisie afin de réaliser une factorisation maximale.

5.4.2 Analyse de l'évolution basée sur des métriques issues du treillis

Dans cette section, nous analyserons le suivi de l'évolution des quinze versions du modèle *SIE-Pesticides* à partir des métriques extraites du treillis. Deux métriques seront principalement utilisées pour chacun des éléments de modélisation :

1. Le ratio des concepts fusionnés (#Fusionnés / #Eléments de modélisation) : Les concepts fusionnés sont des concepts dont l'extension contient plus d'un élément. Les concepts fusionnés regroupent plusieurs éléments qui ont la même description. Nous notons que les extensions propres contiennent des éléments non présents dans les extensions des sous-concepts.
2. Le ratio des nouveaux concepts (ou concepts créés) #New / #Eléments de modélisation : Les nouveaux concepts sont des concepts dont l'extension propre est vide. Ils correspondent à la factorisation d'un ensemble de caractéristiques. De plus cet ensemble de caractéristiques ne

correspond pas exactement à l'ensemble de caractéristiques d'un éléments (qui sinon serait dans l'extension propre).

Ici, nous nous concentrons sur l'analyse des classes, des attributs et des associations parce que le nombre d'opérations est faible et parce que les rôles se comportent plus ou moins comme les associations dans notre cas d'études. Plusieurs métriques sont proposées pour analyser le nombre de concepts fusionnés, créés et immuables. L'interprétation de ces métriques nous permettra de définir le niveau de la finalisation (plus ou moins avancée) pour chaque modèle ainsi que le niveau d'optimisation d'un élément de modélisation par rapport aux autres éléments du modèle, en d'autres termes, l'élément de modélisation qui a été plus factorisé que d'autres.

5.4.2.1 Métriques sur les classes

Dans le contexte formel des classes (cf. 4.4), nous décrivons les classes par leurs noms tout en distribuant le nom d'une classe mère à ses sous-classes (fille), et nous aplatissons la relation d'héritage au sens où les sous-classes sont décrites par les caractéristiques (attributs et opérations) de leurs classes mères (super-classes).

Évolution des concepts fusionnés dans le treillis des classes.

Cette métrique est définie par le rapport entre le nombre de concepts fusionnés et le nombre de classes du modèle (#Merge / #Classes) :

$$\left(\frac{\sum_{Merge}}{\sum_{Classes}} \right) \times 100 \tag{5.4}$$

Les concepts fusionnés correspondent à des classes ayant des noms identiques et une description identique suivant la configuration choisie. Celles-ci sont regroupées en un seul concept en orange dans le treillis.

Le rapport de concepts fusionnés sur le nombre de classes d'un modèle est une métrique qui donne une idée du niveau de redondance des classes dans les différentes versions du modèle. Toutefois, elle ne donne pas d'information sur l'ajout de classes dans les différentes versions au cours de notre processus d'abstraction de concepts.

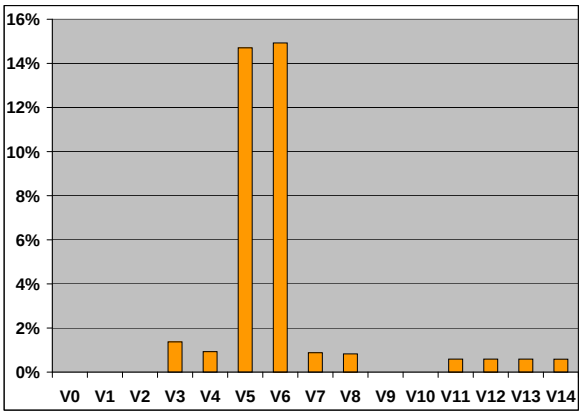


FIGURE 5.18: **C1/C2 : Concepts fusionnés/ Classes**

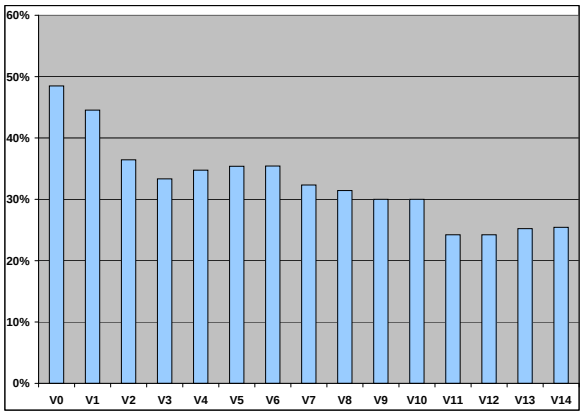


FIGURE 5.19: **C1 : Nouveaux Concepts/Classes**

Analyse des résultats sur les concepts fusionnés avec *C1* et *C2*. (cf. figure 5.18)

Cette métrique nous permet d'observer l'évolution des classes fusionnées dans les quinze versions du modèle. Le premier constat est que le nombre de concepts fusionnés reste identique pour les deux configurations *C1* et *C2* lorsque l'on prend en compte la navigabilité. Le deuxième constat est que le taux de concepts fusionnés est faible, excepté pour les versions *V5* et *V6* qui sont les versions de travail où le modèle des activités métrologiques a été dupliqué pour ne pas perdre l'existant et améliorer le modèle. Pour ces modèles, de nombreuses classes sont dupliquées provoquant de nombreux concepts fusionnés. Par exemple, la version *V5* a trois classes "matière active (cf. figure 5.2)" ayant les mêmes attributs.

Pour les versions *V0*, *V1*, *V2*, *V9* et *V10*, nous observons qu'il n'y a pas de fusion de classe, cela s'explique par le fait que toutes les classes sont uniques et distinctes pour *C1* et *C2*.

Dans le processus itératif, les concepts fusionnés apparaissent à partir de la première itération de l'Analyse Relationnelle de Concepts et ils restent inchangés sur toutes les autres itérations.

La règle que nous pouvons apprendre à ce stade est que plus ce ratio est important plus il y a de classes identiques en fonction de la relation choisie dans notre configuration. Ce ratio devrait attirer l'attention du concepteur quand il est élevé. Si le concepteur duplique des classes pour se faciliter le travail, cela lui rappelera qu'il faut supprimer les concepts dupliqués. Par ailleurs, son attention sera attirée lors de l'ajout de nouveaux concepts qui pourraient être en doublon avec les classes déjà présentes.

Évolution des concepts créés dans le treillis des classes.

Cette métrique est définie par le rapport entre le nombre de concepts créés et le nombre de classes du modèle (#New / #Classes) :

$$\left(\frac{\sum_{New}}{\sum_{Classes}} \right) \times 100 \tag{5.5}$$

Les concepts créés sont les nouvelles abstractions de classes regroupant des caractéristiques similaires existant dans plusieurs autres classes. Ce ratio diffère pour les deux configurations C1 et C2 utilisées pour les factorisations.

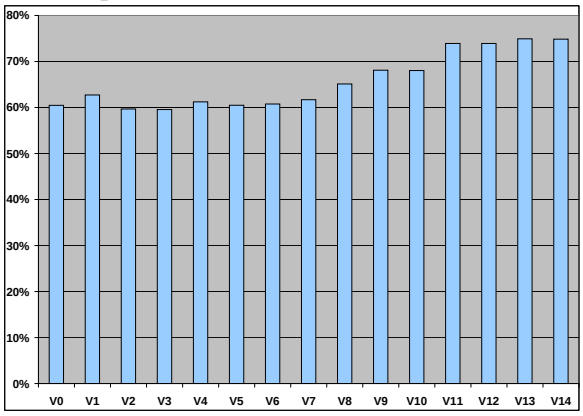


FIGURE 5.20: **C2 : New Concepts/classes à l'itération finale**

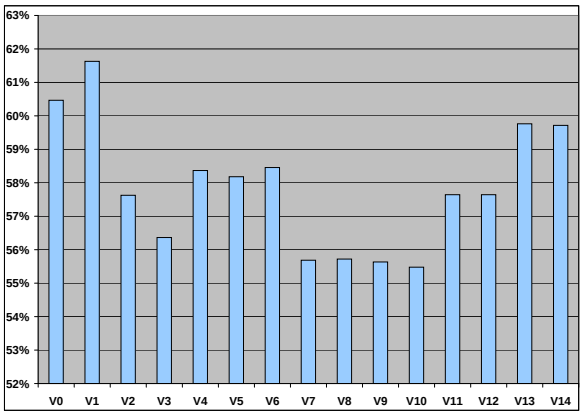


FIGURE 5.21: **C2 : NewConcepts/Classes à l'itération 5**

Analyse des résultats sur les concepts créés avec C1 et prise en compte de la navigabilité. (cf figure 5.19)

Pour la configuration C1, nous observons une décroissance progressive du nombre de nouveaux concepts. Cette décroissance s'accentue à partir de la *V1* malgré une augmentation forte du nombre de classes. Cela reflète le fait qu'il y a de moins en moins de factorisations automatiques par l'ARC dans les différentes versions du modèle au fur et à mesure de l'avancement du projet.

Plus l'analyse avance, plus les équipes informatiques et thématiques font soit émerger des concepts de niveau supérieur d'abstraction (effectuant manuellement la factorisation) soit spécialisent des concepts existants. Au cours de ce travail d'analyse, les associations sont souvent remontées au niveau des classes mères.

Cependant, la petite hausse est enregistrée pour la *V5* et la *V6* s'explique par la duplication de modèle des activités métrologiques.

Analyse des résultats sur les concepts créés avec C2 pour toutes les étapes et pour l'étape 5. (cf. figures 5.20 et 5.21)

La configuration C2 nécessite beaucoup d'itérations pour atteindre la factorisation maximale à cause de la relation Rôle-Type [2] des associations (cf. table de contexte 4.13) qui introduit un circuit dans les données étudiées.

En configuration C2 pour toutes les étapes, nous observons seulement une faible diminution dans la transition *V1* à *V2*, suivi d'une quasi-stabilité et d'une tendance croissante dans les dernières versions à cause d'un nombre important d'associations ajoutées dans les modèles. En fait, l'ajout des nouvelles associations dans le modèle fait émerger de nouvelles classes comme, par exemple, dans le paquetage de "station de mesure" dont certaines sont pertinentes, et d'autres sont trop abstraites.

2. Le type correspond à la classe opposée de l'extrémité du rôle.

La refonte des activités agricoles en *V11* s'est effectuée en ajoutant un nombre d'associations en concepts thématiques. Cela se traduit dans le graphique (cf. figure 5.20) par l'augmentation brusque de cette métrique entre la *V10* et la *V11*.

À la première itération, les résultats sont en fait assez similaires à ceux obtenus pour la configuration *C1*. Dans les premières versions, nous observons une tendance à la hausse jusqu'à l'itération cinq. À partir de cette itération cinq, la tendance s'inverse par rapport au résultat de la configuration *C1*. Pour l'itération finale, nous enregistrons une augmentation continue jusqu'à la saturation des exécutions.

Premier constat : nous observons que les factorisations dues aux associations sont très importantes et qu'elles créent beaucoup de nouvelles classes dans toutes les versions du modèle. De ce fait, nous arrivons très vite à une saturation qui rend l'interprétation difficile et complexe. Aussi, nous nous concentrons à l'étape cinq (*step 5*), qui correspond à la dernière étape qui reflète des résultats à l'échelle d'analyse proche de la configuration C1 et plus riche que la configuration *C1*. Nous observons à l'étape 5, les pics liés à la duplication du modèle des activités métrologiques pour les versions V5 et V6, ces pics ont été observés avec l'analyse de l'évolution des éléments du modèle. Cette duplication entraîne plus de factorisation. Nous notons également que l'évolution du nombre de concepts est influencée par le nombre de concepts immuables et que le nombre d'itérations de factorisation dépend de la taille d'un chemin (la distance entre éléments de modélisation dans le méta-modèle pour faire un circuit sur le même élément de modélisation cf. chap 6). En d'autres termes, le nombre d'itérations dépend d'une part, des éléments de modélisation contenus dans le modèle et d'autre part, des éléments de modélisation définissant la configuration. Par conséquent, le nombre d'éléments de modélisation du modèle et les relations définies dans la configuration ont un impact sur la complexité des résultats obtenus (par exemple le nombre de concepts dans le treillis) et, de fait, les factorisations, c'est-à-dire les nouvelles abstractions des éléments de modélisation, ont tendance à diminuer lors d'un premier cycle de factorisation, ce qui correspond à la cinquième itération de l'analyse relationnelle de concepts [Osman Guédi *et al.*, 2012] (classe-rôle-association-rôle-classe).

Deuxième constat (sur l'itération 5) : Une baisse significative de ce ratio est observée entre *V1* et *V3*, qui est dû à la factorisation des associations de la première version du modèle. Ensuite, nous observons une augmentation du niveau d'abstraction qui crée de nouveaux concepts dans la version *V4*. Ceci est expliqué par l'introduction de nouvelles classes qui ne sont pas factorisées (l'analyse est similaire à celle de *C1*. Cette remarque est accentuée avec la configuration *C2* à *l'itération 5*). L'ajout des rôles par l'introduction des nouvelles associations augmente également les nouveaux concepts de classe. De même, ce rapport diminue entre *V9* et *V10*. Cela s'explique par le fait que les associations ajoutées n'impliquent pas des factorisations.

Conclusion pour les configurations C1 et C2 .

La règle que nous pouvons tirer est qu'avec la configuration *C1* le ratio diminue, cela signifie que nous sommes sur le point d'améliorer l'abstraction (et la factorisation) des classes en fonction des attributs, des rôles et des opérations. Par contre, lorsque ce ratio augmente, cela exprime qu'il y a des redondances dans le modèle d'où l'intérêt de la question de les factoriser ou non. Pour l'itération *l'étape 5* de la configuration *C2*, les conclusions sont les mêmes mais plus accentuées. En ce qui concerne l'itération finale de *C2*, elle est très difficile à interpréter et de tirer des conclusions si-

gnificatives en raison de la saturation (cf. figure 5.21). Elle a motivé l'étude menée dans un prochain chapitre.

Évolution des concepts immuables dans le treillis des classes.

Cette métrique est définie par le rapport entre le nombre de concepts immuables et le nombre de classes du modèle (#Immuables / #Classes) :

$$\left(\frac{\sum_{Immuables}}{\sum_{Classes}} \right) \times 100 \tag{5.6}$$

Ce ratio caractérise les classes qui n'ont pas été regroupées avec d'autres. Nous notons que leur évolution diminue vers les dernières versions, car il y a de plus en plus de nouveaux concepts créés et fusionnés impliquant une diminution des concepts immuables. Par exemple, dans les versions *V5-V6* nous observons une diminution significative due à l'augmentation des concepts fusionnés (cf. figure 5.22).

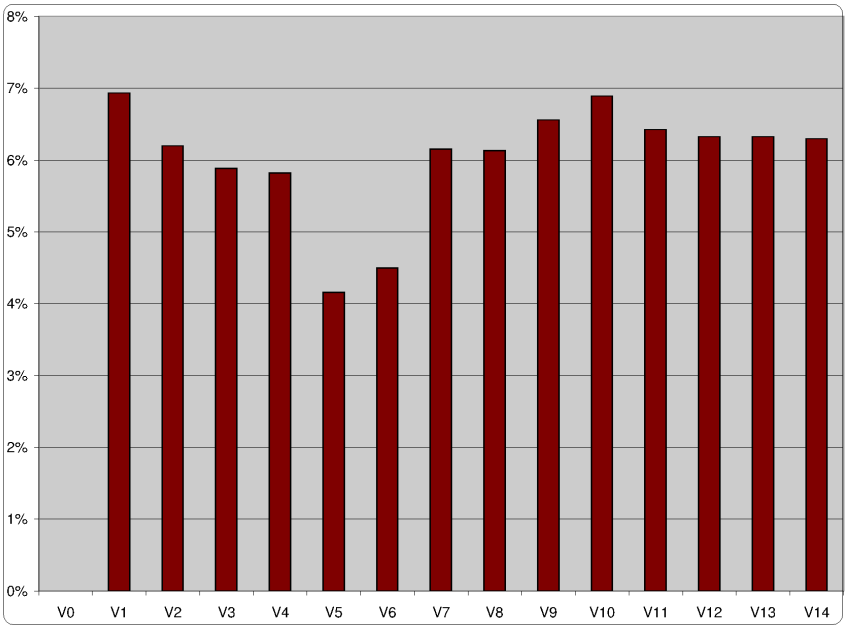


FIGURE 5.22: **Évolution du concept Immuable**(Relation *R1*)

5.4.2.2 Métriques sur les attributs

De la même manière, nous allons étudier dans cette section les métriques similaires impliquant les attributs afin d'analyser l'évolution des quinze versions du modèle *SIE-Pesticides*.

Évolution des concepts fusionnés dans le treillis des attributs. (cf. figure 5.23)

Cette métrique est définie par le rapport entre le nombre de concepts fusionnés et le nombre d'attributs du modèle (#Merge / #Attributs) :

$$\left(\frac{\sum Merge}{\sum Attributs} \right) \times 100$$

Globalement, nous observons une tendance générale à la baisse au fur et à mesure des versions, malgré une forte augmentation du nombre des attributs, ce ratio est en baisse continue parce que l'ajout de nouveaux attributs ne crée pas de nouvelles fusions ou du moins il en crée proportionnellement moins (cf. figure 5.23).

La conséquence directe est qu'il y a de moins en moins d'attributs dupliqués au sein des versions c'est-à-dire que les nouveaux attributs introduits ont de moins en moins de similarité avec ceux qui sont déjà présents dans le modèle. Le pic observé aux versions *V5* et *V6* est essentiellement dû à la duplication des modèles contenant les mêmes classes donc les mêmes attributs. D'autres pics importants sont observés pour les versions *V1* à *V3* qui correspondent à un ajout important d'attributs ayant le même nom, "Commentaire" par exemple.

En règle générale, l'ajout de nouveaux attributs entraîne de moins en moins de fusion. Ceci est dû à l'opération réalisée par le concepteur qui est de lever l'ambiguïté sur les attributs mais aussi à la factorisation manuelle effectuée par une analyse poussée.

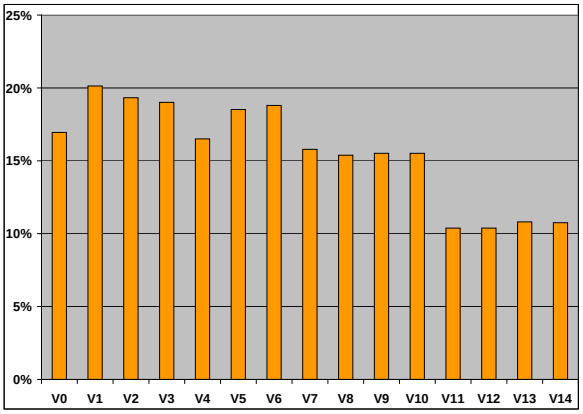


FIGURE 5.23: **C1/C2 : Merge / Attributes**

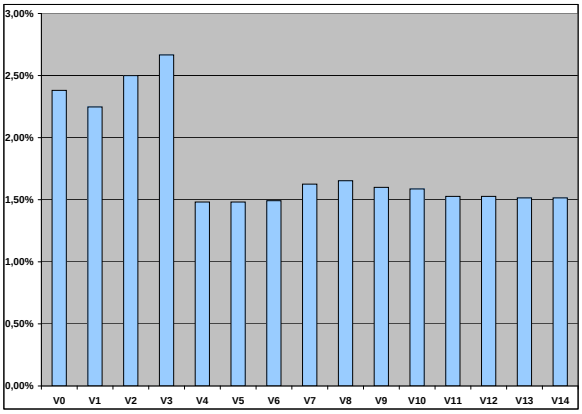


FIGURE 5.24: **C1 : New / Attributes**

Évolution des concepts créés dans le treillis des attributs. (cf. figure 5.24)

Cette métrique est définie par le rapport entre le nombre de concepts créés et le nombre d'attributs du modèle (#New / #Attributs) :

$$\left(\frac{\sum_{New}}{\sum_{Attributs}} \right) \times 100 \tag{5.7}$$

Ce ratio exprime la proportion de nouveaux concepts créés donc plus abstraits du fait d'attributs portant le même nom (cf. figure 5.24). Le niveau de cette métrique est faible de l'ordre de 1,50 % ce qui signifie que les attributs participent peu à la création de nouveaux concepts d'attributs. Même s'il y a une baisse importante entre la *V3* et la *V4*, elle reste faible puisque cette métrique passe de 2,50 à 1,50 %. En effet, l'analyse visuelle des treillis montre que, pour la configuration *C1*, il n'y a pas de nouveaux concepts dans le treillis des attributs à l'exception des concepts formels Top et Bottom qui représentent les extrémités supérieure et inférieure du treillis. Par contre, les attributs participent massivement à la création de nouvelles abstractions dans les treillis liés, par exemple, pour la relation $Classe \times Attribut$ (cf. table 4.9).

Évolution des concepts immuables dans le treillis des attributs.

Cette métrique est définie par le rapport entre le nombre de concepts immuables et le nombre d'attributs du modèle (#Immuables / #Attributs) :

$$\left(\frac{\sum_{Immuables}}{\sum_{Attributs}} \right) \times 100$$

Le graphique correspondant à ce ratio serait presque à l'opposé du précédent, avec une légère déformation en raison des concepts fusionnés. Nous observons une tendance croissante qui s'interprète par l'ajout des attributs qui ne participent pas à la factorisation d'attributs, mais peut-être qu'ils participeront à la création de nouvelles abstractions pour d'autres éléments de modélisation qu'ils décrivent dans les relations de la configuration c'est-à-dire par exemple pour les classes avec la relation $owns_1 \subseteq Classes \times Attributes$.

5.4.2.3 Métriques sur les associations

L'analyse des métriques sur les associations permet d'extraire des informations sur les factorisations introduites par les associations, factorisations qui sont la base de notre méthode de normalisation basée sur l'analyse du suivi de l'évolution du modèle.

La première information produite correspond à la part des associations qui ont été fusionnées. La seconde information est la proportion de nouvelles abstractions créées par la relation association-rôle des deux configurations *C1* et *C2* (cf. les tables de contextes formels en 4.12 et 4.15).

Enfin, la troisième information, importante pour l'évaluation de notre méthode de normalisation, concerne l'interprétation de l'augmentation du nombre de concepts d'associations immuables. Plus nous introduisons des associations, plus nous retirons des similitudes au niveau des rôles qui décrivent les associations dans les configurations *C1* et *C2*.

Évolution des concepts fusionnés dans le treillis des associations. (cf. figure 5.25)

Cette métrique est définie par le rapport entre le nombre de concepts association fusionnés et le nombre d'associations du modèle (#Merge / #Associations) :

$$\left(\frac{\sum Merge}{\sum Associations} \right) \times 100$$

Les résultats sont donnés à la figure 5.25.

Une tendance à la baisse est globalement observée dans la figure 5.25 avec deux creux très marqués. Les concepts fusionnés diminuent dans le treillis des associations et cela s'explique par la relation suivante : plus nous introduisons des associations, moins nous avons des concepts fusionnés. L'ajout de ces associations factorise notre modèle en ôtant les ambiguïtés.

En effet, nous travaillons avec les associations "bien formées" c'est-à-dire qui ont été complètement renseignées par les acteurs. Pour ce faire, nous avons adopté une règle qui exclut les associations dont le nom n'a pas été défini. Cette règle diminue énormément les incohérences dues aux associations qui n'ont pas été bien décrites dans les premières versions où nous observons de nombreuses associations fusionnées qui ont été améliorés par la suite en les remplaçant par d'autres associations mieux définies.

La tendance globale à la baisse observée pour cette métrique exprime que le concepteur améliore le modèle par la factorisation des associations et des rôles. Dans les versions *V5* et *V6*, les paquetages dupliqués produisent des pics de cette métrique exprimant qu'il y a moins de factorisation manuelle sur les associations (taux élevé). Les pics des versions *V7* et *V8* montrent de nombreuses associations n'ont pas été remontées au niveau des classes mères.

Par conséquent, cette métrique confirme notre idée initiale qui est de montrer la proximité du modèle et de sa forme normale en utilisant notre méthode d'analyse de l'évolution avec l'Analyse Relationnelle de Concepts sur des modèles UML.

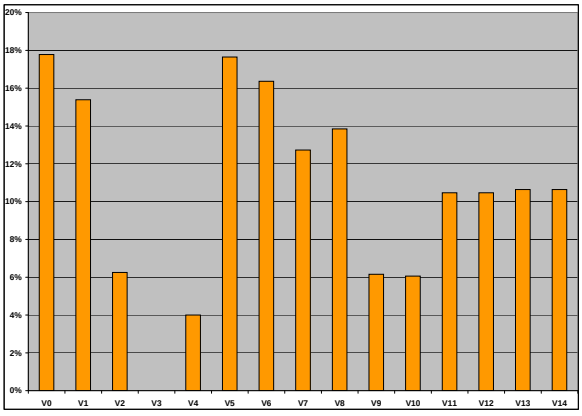


FIGURE 5.25: **C1/C2 : Merge/Associations**

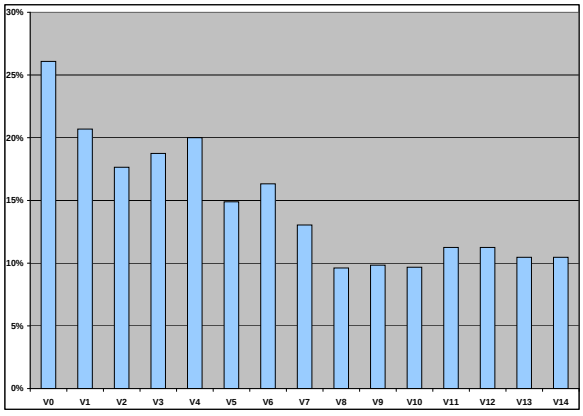


FIGURE 5.26: **C1 : New/Associations**

Évolution des concepts créés dans le treillis des associations.

Cette métrique est définie par le rapport entre le nombre de concepts créés et le nombre d'associations du modèle (#New / #Associations) :

$$\left(\frac{\sum_{New}}{\sum_{Associations}} \right) \times 100 \tag{5.8}$$

Cette métrique quantifie le nombre de nouveaux concepts d'association par rapport au nombre d'associations du modèle.

Cette métrique permet de suivre l'évolution des nouvelles abstractions créées dans le treillis des associations par l'apport de la relation "$owns_4 \subseteq Associations \times rôles$" (cf. table de contexte 4.12), relation indiquant que les associations sont décrites par les rôles. Cette relation apparaît dans les deux configurations *C1* et *C2*.

La même règle sur le nommage est appliquée pour les rôles, c'est-à-dire que nous sélectionnons uniquement les rôles dont le nom est sémantiquement bien défini. De ce fait, les rôles dont le nom n'est pas défini (undefined) sont ignorés dans les tables de contextes relationnels.

Dans la figure 5.26, nous observons une tendance globale à la baisse pour la configuration *C1* avec une légère augmentation pour les versions *V3*, *V4* et *V6* due à l'ajout d'un nombre important d'associations. Ensuite le processus d'ajout des associations se poursuit mais il fait l'objet de factorisations manuelles de la part du concepteur du système d'information.

D'une part, certains factorisations sont liées à la duplication d'associations qui a donné naissance à des concepts fusionnés et non à de nouvelles abstractions (cf. figure 5.26). Le pic de la version *V6* est typique de ce mécanisme. Une légère augmentation est remarquée en V11 et V12. Elle s'explique par la refonte du modèle des activités agricoles, refonte qui, bien évidemment, nécessite l'ajout d'associations pouvant faire l'objet de factorisations.

D'autre part, ces ajouts sont aussi très importants pour la factorisation des classes grâce aux deux relations $owns_3 \subseteq classes \times roles$, $hasType_5 \subseteq roles \times classes$ des configurations de l'Analyse Relationnelle de Concepts. En effet, l'une permet de décrire les classes par les rôles qu'elles possèdent et l'autre les rôles par les classes (pour définir leur type).

Pour la configuration *C2*, nous constatons le même phénomène de saturation lorsque toutes les itérations de l'analyse relationnelle de concepts sont calculées (cf. figure 5.27). Par contre, l'analyse de la cinquième itération permet d'énoncer des conclusions semblables à celles vues pour les classes pour la même itération (cf. figure 5.28).

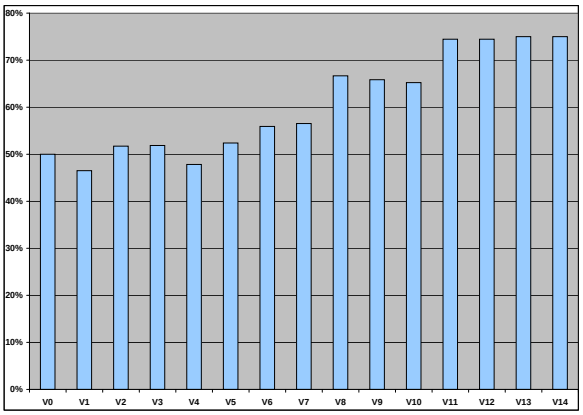


FIGURE 5.27: **C2 : New/Associations à l'étape finale**

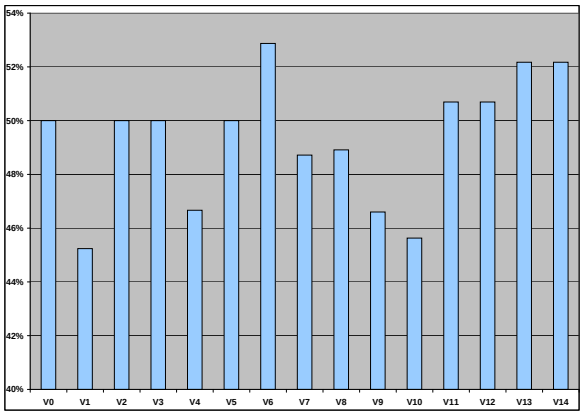


FIGURE 5.28: **C2 : New/Associations à l'étape 5**

Évolution des concepts immuables dans le treillis des associations.

Cette métrique est définie par le rapport entre le nombre de concepts immuables et le nombre d'associations du modèle (#Immuables / #Associations) :

$$\left(\frac{\sum Immuables}{\sum Associations} \right) \times 100 \tag{5.9}$$

De même que dans les autres concepts immuables du treillis, ce ratio nous permet de suivre les associations qui n'ont pas participé à des fusions. Ce ratio se comporte de manière opposée à celui des nouvelles abstractions créées, ce qui montre le degré de finalisation du modèle, qui progresse.

5.4.2.4 Métriques sur les opérations et les rôles

Dans notre contexte, les métriques calculées à partir du treillis des opérations n'apportent pas de conclusions significatives car, comme nous l'avons vu dans la section 5.2.1.3, cet élément de modélisation n'est pratiquement pas présent dans les modèles.

En ce qui concerne les métriques sur le treillis des rôles, nous obtenons les mêmes conclusions que pour les associations, et nous ne les développerons pas ici. En effet, les rôles participent massivement à la factorisation des associations auxquelles ils appartiennent et des classes qui définissent leur type.

5.5 Conclusion

Dans ce chapitre, nous avons présenté une analyse de l'évolution des modèles à travers des métriques portant d'une part, sur les éléments de modélisation du modèle de classe UML et, d'autre

part, sur les treillis générés à partir de l'Analyse Formelle de Concepts et de l'Analyse Relationnelle de Concepts. L'application systématique aux quinze versions du modèle *SIE-Pesticides*, nous a permis de proposer des métriques et plusieurs recommandations qui constituent le début d'une méthode d'analyse.

5.5.1 Analyse de l'évolution sur les métriques entre éléments de modélisation

Dans le contexte des modèles UML, nous proposons plusieurs métriques facilitant la comparaison des modèles en se basant sur le nombre d'éléments de modélisation. Ces métriques ont permis d'établir plusieurs recommandations pour le concepteur, et elles permettent de suivre l'évolution de ces modèles par la simple analyse des éléments de modélisation. En premier lieu, la métrique sur l'évolution du nombre de classes (cf. figure 5.4) donne une idée du niveau d'encapsulation des données qui composent le système d'information. En deuxième lieu, la description structurelle et comportementale des classes du modèle est abordée par la métrique sur l'évolution des attributs et des opérations (cf. figure 5.5). En troisième lieu, l'aspect relationnel entre concepts thématiques est introduit avec la métrique sur les associations qui permet d'en suivre l'évolution. Ces deux derniers points reflètent une vision globale des descriptions structurelles et relationnelles des éléments de modélisation qui composent le modèle du système d'information (cf. figure 5.5).

5.5.2 Analyse de l'évolution basée sur l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts

L'application de l'Analyse Formelle de Concepts et de l'Analyse Relationnelle de Concepts à l'étude de l'évolution du processus d'analyse d'un modèle UML est une approche nouvelle et originale. Dans nos travaux, nous avons défini des métriques pour supporter ce processus et nous l'avons évalué sur les quinze versions du modèle *SIE-Pesticides*.

D'une part, ces métriques permettent de suivre le nombre de concepts fusionnés pour chacun des éléments de modélisation. Cela permet de détecter les doublons (cf. figure 5.18, 5.23, 5.25). D'autre part, ces métriques sont des outils pour suivre l'évolution des nouvelles abstractions. Ces résultats indiquent au concepteur le niveau d'abstraction de son modèle (cf. figure 5.19, 5.24, 5.26). Par examen ultérieur des treillis, ces métriques permettent au concepteur de se focaliser sur les éléments impliqués dans des fusions ou dans des abstractions pour atteindre une factorisation maximale et une description enrichie.

L'analyse des résultats à chaque itération de l'Analyse Relationnelle de Concepts permet aux acteurs d'étudier les abstractions à une itération donnée et faire des choix. Avec l'Analyse Relationnelle de Concepts, nous introduisons une méthode de normalisation qui permet à l'acteur d'analyser le suivi de l'évolution du modèle au cours de son cycle de vie. L'étude de l'évolution du processus d'analyse d'un modèle par application systématique des métriques aux quinze versions du modèle *SIE-Pesticides* a permis de proposer les premières recommandations qui constituent également le début d'une méthode, utilisant cette fois l'ARC.

Les travaux décrits dans ce chapitre s'inscrivent dans un cadre de recherche dont l'objectif est de factoriser des concepts thématiques au sein d'un même modèle. Toutefois, comme l'AFC et l'ARC produisent dans certains cas une profusion de concepts formels, nous pensons que les métriques peuvent aussi être utilisées pour suivre et contrôler cette profusion.

En effet, le contrôle de l'émergence de nouveaux concepts de l'Analyse Relationnelle de Concepts, l'automatisation de certaines phases des transformations des modèles UML en treillis, mais aussi la phase de reconstruction du modèle UML seront au programme de travaux à venir.

CHAPITRE

6 Étude comportementale de l'Analyse Relationnelle de Concepts sur des modèles UML

Préambule

Une analyse détaillée sur les aspects structurels et comportementaux de l'analyse relationnelle de concepts[Godin et Valtchev, 2005] sur des modèles UML [Booch et al., 1999] fera l'objet de ce chapitre. Ainsi, nous allons en déduire des recommandations pour une utilisation optimale à l'échelle de l'utilisateur. A travers ces recommandations, l'acteur du système d'information disposera d'une panoplie de métriques pour utiliser au mieux l'analyse relationnelle de concepts (ARC).

Sommaire

6.1 Introduction

Les expérimentations réalisées dans le cadre de l'analyse de l'évolution ont apporté, d'une part, des recommandations sur l'analyse de l'évolution des différentes versions du modèle SIE-Pesticides [Osman Guédi *et al.*, 2011] et d'autre part, sur le suivi de l'évolution des factorisations avec l'analyse relationnelle de concepts [Osman Guédi *et al.*, 2012].

Nous avons maintenant, d'un côté, des métriques descriptives du modèle qui sont en amont de la factorisation et de l'autre côté, des métriques basées sur les treillis c'est à dire en aval sur les résultats de la factorisation. Cette dernière découle d'une méthode basée sur l'analyse formelle de concepts[Wille, 1982b] (AFC et l'analyse relationnelle de concepts[Huchard, 2003].

Cependant, certaines expérimentations se sont avérées très conséquentes en termes de temps d'exécution et en espace mémoire occupé pendant leurs exécutions. Une utilisation interactive du processus de la factorisation par l'acteur du système d'information peut donc être complexe dans certains cas de figure, notamment avec la configuration C2 (cf. section 4.3.2.2). Par conséquent, nous avons exploré le comportement de l'analyse relationnelle de concepts afin d'en déduire une méthodologie d'utilisation (cf. section 1).

Si on prend le cas de la configuration C2, cette dernière comporte les principaux éléments de modélisation (classe, attribut, opération, association et le rôle) et elle ajoute des relations pour une factorisation maximale. Elle fournit des résultats certes intéressants et justes théoriquement mais très complexes à utiliser ou à explorer. Comme toutes les expérimentations avec l'analyse formelle de concepts [Ganter *et al.*, 2005] arrivent bien à terme tout en produisant des résultats exploitables à l'échelle utilisateur, notre étude du comportement se limitera dans ce chapitre à l'analyse relationnelle de concepts. L'objectif de ce chapitre est d'émettre des recommandations à partir des métriques sur les modèles UML [Hacene *et al.*, 2007b], à destination de l'acteur du système d'information.

Dans un premier temps, nous allons présenter les paramètres choisis ainsi que la démarche suivie pour étudier le comportement de l'analyse relationnelle de concepts et des algorithmes l'implémentant.

Ensuite, nous allons évaluer d'une part l'impact de ces paramètres sur la taille des résultats, et d'autre part, la complexité en terme de temps et d'espace mémoire. Puis des recommandations seront déduites de ces métriques afin de proposer un meilleur cadre d'utilisation de l'analyse relationnelle de concepts.

Dans la partie suivante, nous analysons la forme et/ou la structure des modèles UML à partir des relations et des configurations choisies et nous étudions leur impact sur le comportement de l'analyse relationnelle de concepts. Là aussi des métriques sont déduites afin d'aider l'acteur du système d'information dans son processus de factorisation de modèles.

Enfin nous proposerons une synthèse des recommandations basées sur l'ensemble des métriques pour une utilisation interactive de l'analyse relationnelle de concepts avec des modèles UML.

6.2 Démarche suivie et paramètres d'analyse

La démarche d'expérimentations et d'analyse présentée dans le chapitre précédent (cf. chapitre 5) a été consolidée avec d'autres mesures qui nous permettront de comprendre le comportement de l'ARC sur des modèles UML réels. Dans un premier temps, nous décrivons les grandes lignes de ces expérimentations.

6.2.1 Description de la démarche suivie

Les expérimentations s'appuient ici encore sur les quinze versions du modèle UML du projet SIE-Pesticides. Ces versions retracent les nombreuses réunions de travail des experts et rendent compte des évolutions dans la modélisation du domaine. Le plugin développé sous Objecteeering est utilisé pour le parcours des principaux éléments de modélisation choisis (cf. figure 4.2) en fonction des liens définis dans le méta-modèle (cf. figure 6.10). Il génère les contextes formels et relationnels correspondant aux configurations définies dans le chapitre 4.

De plus, ce plugin utilise la librairie ERCA (cf. section 4.4.2) qui met en œuvre l'AFC et surtout l'ARC. ERCA a été étendu en ajoutant plusieurs métriques. Certaines métriques évaluent les aspects structurels des modèles, notamment le nombre d'éléments de modélisation de chaque type. D'autres métriques permettent de suivre les temps d'exécution et l'espace mémoire occupé. Enfin, une dernière catégorie de métriques, portant sur le graphe d'entités et de liens que manipule l'AFC et l'ARC est définie pour tenter de comprendre certains aspects du comportement de l'AFC et de l'ARC. Pour finir, les résultats correspondant aux métriques sont exploités d'une part pour évaluer les outils proposés et d'autre part pour en déduire des recommandations relatives à leur utilisation.

6.2.2 Paramètres choisis

Pour la réalisation de ces expérimentations, un processus d'exécution a été mis en place sur une machine cluster (cf. section 4.4.3). Ce processus consiste dans un premier temps à générer pour chacune des versions du modèle les contextes formels de chaque relation (R1 (cf. section 4.3.1.2), R2 (cf. section 4.3.1.3), R3 (cf. section 4.3.1.4), R4 (cf. section 4.3.1.5) et R5 (cf. section 4.3.1.6)) et relationnels relatifs à chaque configuration (C1 (cf. section 4.3.2.1), C2 (cf. section 4.3.2.2). Lors de la génération, nous prenons également en compte deux paramètres qui influenceront les quantités de concepts produits, le temps d'exécution et l'espace de mémoire occupé. Ces paramètres correspondent, d'une part à la prise en compte de la *navigabilité* sur les associations, et d'autre part, à la prise en compte des rôles non nommés, que nous appellerons ici les "undefined". De ce fait, chaque configuration fera l'objet de quatre exécutions qui correspondent aux différentes combinaisons des paramètres précédemment mentionnés. Pour préciser, voici les quatre combinaisons possibles :

1. Avec la navigabilité et avec les undefineds
2. Avec la navigabilité et sans les undefineds
3. Sans la navigabilité et avec les undefineds
4. Sans la navigabilité et sans les undefineds

L'ensemble des exécutions a été effectué sur la même machine cluster (plus précisément sur les différents nœuds du cluster). Les expérimentations se font en parallèle sur des nœuds distincts. Elles ne partagent aucune information tout au long de leurs exécutions car chacune d'elle se trouve dans un espace bien spécifique avec un processeur et une mémoire allouée.

Enfin, nous utilisons le caractère itératif de l'ARC (cf. 3). Les exécutions correspondant aux configurations de l'ARC se font de deux façons : d'un côté, une exécution globale qui comprend l'ensemble des étapes de factorisation et d'un autre côté, une exécution étape par étape. La condition d'arrêt pour ce type d'exécution est la même que pour l'exécution globale (le point fixe est atteint).

6.2.3 Récapitulatif des exécutions

Cette démarche génère un peu plus de 1070 exécutions [1] avec des résultats très intéressants en termes de fusion et d'émergence de nouvelles abstractions.

Néanmoins, un peu moins de 50 exécutions se sont avérées trop longues pour produire des résultats. Ces dernières correspondent aux versions de modèles les plus complexes avec la configuration C2 qui contient les relations principales. L'analyse des métriques choisies pour comprendre le comportement de l'ARC sur des modèles UML fait l'objet des sections suivantes.

6.3 L'impact des paramètres

L'ARC effectue plusieurs étapes pour arriver à une factorisation maximale (cf. chapitre 3), au cours desquelles elle crée des concepts par fusion ou abstraction d'entités. Le nombre d'étapes dépend principalement de la quantité d'éléments de modélisation pris en compte dans la configuration. Par conséquent, cette quantité a un impact sur la complexité des résultats obtenus, par exemple sur le nombre de concepts dans le treillis [Osman Guédi *et al.*, 2012] et sur le comportement général de l'ARC.

Durant toutes nos expérimentations, nous observons que le choix des paramètres présentés précédemment a des conséquences sur la complexité en termes de temps d'exécution et de résultats produits. Nous étudions l'effet de ces paramètres à l'aide de métriques et nous en déduisons des recommandations qui permettront d'améliorer la complexité et de favoriser une utilisation interactive de l'approche.

6.3.1 Le nommage des éléments de modélisation

En général, Les outils de modélisation s'utilisent d'une manière plus ou moins similaire dans les divers ateliers de génie logiciel. Créer un modèle pour un système d'information consiste à instancier des éléments du méta-modèle. Ainsi, ces outils de modélisation offrent une manipulation flexible, transparente et interactive à l'utilisateur, par exemple avec l'aide d'un système de "Glisser-déplacer" (connu sous le terme de Drag-and-drop en anglais). En premier lieu, lors de l'ajout d'un élément de modélisation, un nom par défaut lui est attribué, qui dépend dans certains cas du nom

1. http://www.lirmm.fr/~osmanguedi/

de l'élément du méta-modèle instancié. Dans notre cas de figure, Objecteering propose systématiquement un nom par défaut pour l'ensemble des élément de modélisation. Par exemple pour les classes, Objecteering propose Classe1, Classe2 etc. et pour les attributs, il propose attribut1, attribut2, etc.

Par contre, nous remarquons que les rôles portent par défaut un nom unique, le nom "undefined", signifiant que le nom du rôle en question n'est pas défini. L'utilisation de ce nom par défaut aura un impact sur les exécutions et les résultats de nos expérimentations. Dès lors que les modèles possèdent des rôles dont le nom n'est pas défini (donc portant le nom "undefined") nous constatons une hausse du nombre de rôles sélectionnés, c'est-à-dire que nous avons plus de rôles dans le contexte relationnels de rôles pour les configurations (C1 et C2). Des métriques sont introduites pour évaluer les conséquences de ce paramètre, d'une part sur les factorisations obtenues (cf. chapitre 5) et d'autre part sur le comportement de l'ARC.

Remarquons que pour notre cas de figure, les seuls rôles dont le nom n'est pas défini dans les modèles, sont les rôles qui sont à l'opposé du sens de la navigabilité. Donc leur étude se limite à la configuration C2 et au cas où la navigabilité est ignorée. Dans ce cas (sans navigabilité et en considérant les undefineds), la fin de l'exécution n'a été atteinte que pour les quatre premières versions. Nous n'étudierons donc ce paramètre que pour ces quatre premières versions.

Le rôle est un élément très important dans la configuration C2, car il décrit les associations (qui possèdent des rôles), ainsi que les classes (qui possèdent aussi des rôles) et les rôles sont décrits à son tour par les classes pour définir leur type. Une première observation montre que le nombre d'étapes augmente sensiblement lors de la prise en compte de ces éléments de modélisation non nommés (cf. figure 6.1). Ce nombre d'étapes s'accroît de 44% en moyenne. Enfin, plus les modèles deviennent complexes plus cette hausse se confirme.

Version	Avec undefineds	Sans undefineds	Rendement
V0	17	6	64,71%
V1	22	11	50%
V2	16	12	25%
V3	22	14	36,36%

TABLE 6.1: **Gain de nombre d'étapes**

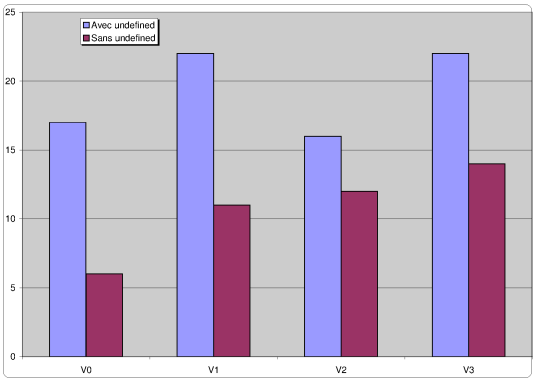


FIGURE 6.1: **Gain de nombre d'étapes**

Par voie de conséquence, les rôles non nommés ont, d'une part, un impact sur la cohérence des factorisations (cf. chapitre 5) et d'autre part, ils augmentent la complexité des exécutions. Donc, les premières recommandations à l'utilisateur consisteront à ne prendre en compte dans les configurations que les rôles dont le nom est défini. Ces conséquences se généralisent pour tout autre élément de modélisation présentant les mêmes caractéristiques. En d'autres termes, il vaut mieux ne prendre en compte que les classes, associations, méthodes, attributs, etc. dont le nom a été défini par l'utilisateur et non pas généré automatiquement.

6.3.2 L'impact du sens de la navigabilité des associations

Le second paramètre des expérimentations s'avère essentiel sur le comportement de l'ARC. Il concerne la prise en compte du sens de la navigabilité (cf. section 4.3.2.3, qui est aussi déterminant sur l'aboutissement des exécutions.

En effet, l'ARC procède en plusieurs étapes pour arriver à une factorisation maximale. De ce fait, la prise en compte du paramètre correspondant à la navigabilité influence d'une part le nombre d'étapes et d'autre part la complexité liée à l'exécution pour arriver à terme, autrement dit, son temps de calcul et l'espace mémoire utilisé.

La prise en compte du sens de la navigabilité diminue le nombre d'éléments de modélisation dans les contextes et nous remarquons que l'ensemble de ces exécutions arrivent bien à leur terme. A l'inverse, lorsque le sens de la navigabilité est ignoré et que seuls les éléments nommés sont pris en compte (resp. non pris en compte), seules les exécutions des cinq (resp. quatre) premières versions du modèle arrivent à terme.

Les contextes relationnels sont moins volumineux lorsqu'ils prennent en compte le sens de la navigabilité (cf. les tables de contextes du chapitre 4), notamment le cas de la configuration C2, car cette dernière contient plusieurs relations liées aux rôles. Dans le cadre de nos modèles, l'avantage de ce paramètre est de sélectionner uniquement les rôles qui sont du côté navigable des associations et d'éviter les rôles dont le nom n'est pas défini.

Un gain de 22% en moyenne est observé sur le nombre d'étapes lors de la prise en compte de la navigabilité (cf. le tableau 6.1).

Version	Avec navigabilité	Sans navigabilité	Rendement
V0	6	6	0%
V1	8	11	27,27%
V2	10	12	16,67%
V3	8	14	42,86%

TABLE 6.2: **Gain sur le nombre d'étapes**

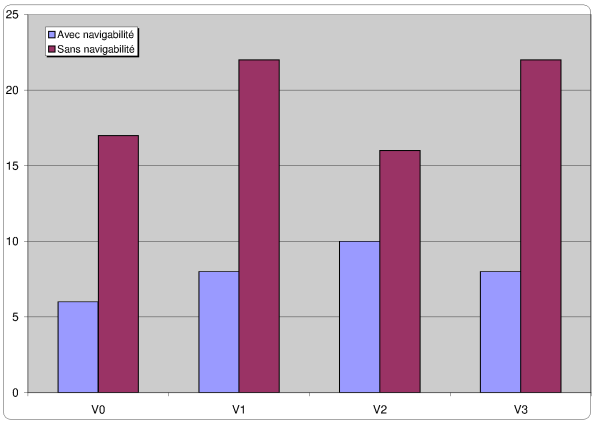


FIGURE 6.2: **Gain sur le nombre d'étapes**

En effet, la figure 6.2 montre que la prise en compte de la navigabilité permet d'atteindre en peu d'étapes la factorisation maximale, tout en produisant moins de concepts dans le treillis final. De même que la précédente, les figures 6.3 et 6.4 indiquent le nombre de concepts produits par l'ARC pour la première étape et la sixième étape. Cette dernière est l'étape la plus lointaine atteinte par les expérimentations pour l'ensemble de versions du modèle. Ainsi, nous constatons d'une part, un écart sur le nombre de concepts et d'autre part, que cet écart augmente au cours des étapes comme le montre la figure 6.4 qui correspond à l'étape 6 de la factorisation pour l'ensemble de versions.

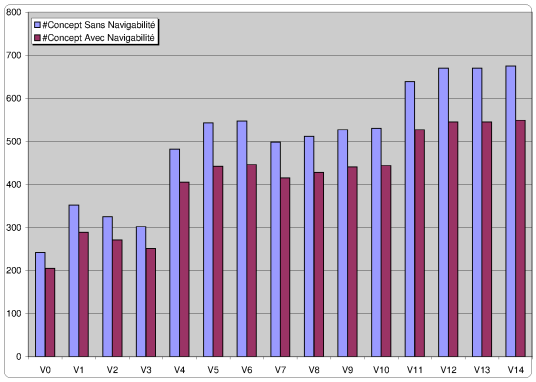


FIGURE 6.3: **Le métrique sur le rendement à l'étape 1**

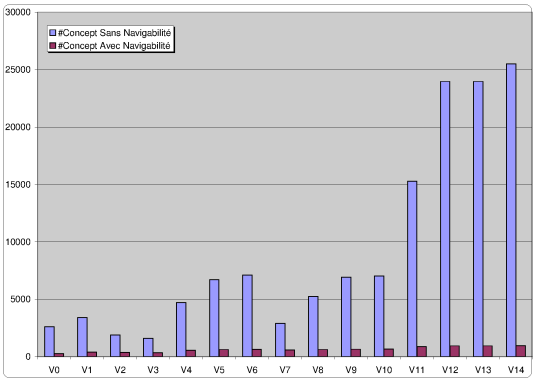


FIGURE 6.4: **Le métrique sur le rendement à l'étape 6**

Nous définissons ci-après une métrique de rendement qui exprime le gain en nombre de concepts créés en prenant en compte la navigabilité par rapport au cas où la navigabilité n'est pas prise en compte.

$$\textbf{Rendement} = \left(\frac{Nombre\ de\ concepts\ sans\ la\ navigabilité - Nombre\ de\ concepts\ avec\ la\ navigabilité}{Nombre\ de\ concepts\ sans\ la\ navigabilité} \right) \times 10 \tag{6.1}$$

Le gain en nombre de concepts est à 17% en moyenne à la première étape et il passe à 89% en moyenne à l'étape 6 de la factorisation, cette étape 6 correspond à la plus lointaine que nous ayons pu atteindre dans nos exécutions. En d'autres termes, l'ARC produit de plus en plus de concepts tout au long des étapes et ce nombre de concepts augmente particulièrement dans le cas où le sens de la navigabilité est ignoré.

6.3.3 Discussion et recommandation

Nous avons vu dans cette section, d'une part, l'importance de nommer chaque élément de modélisation, surtout dans notre cadre d'étude où la sémantique a une forte importance pour la factorisation des éléments de modélisation. D'autre part, nous avons constaté l'influence de la prise en compte de la navigabilité des associations. Ces deux paramètre améliorent considérablement les résultats, tant sur la cohérence des factorisations obtenues que sur le nombre d'étapes de factorisation et sur le nombre de concepts obtenus dans les treillis.

6.4 Mesure de la complexité pratique

Après avoir évalué les conséquences des paramètres sur certains aspects du comportement de l'ARC (nombre d'étapes et nombre de concepts), nous allons mesurer dans cette section la complexité de l'exécution. La notion de complexité est très importante dans le domaine de l'algorithmique, elle permet d'évaluer, de comparer une méthode à une autre, et elle est fortement liée aux

données entrant et sortant de la méthode en question. Un algorithme est évalué par des complexités théoriques ; par exemple, une mesure du cas trivial, du cas moyen et/ou du pire de cas, en d'autres termes le cas le plus complexe pour l'algorithme.

La complexité dépend et s'exprime souvent en fonction de la taille, de la forme, et de la structure de la donnée. C'est pourquoi nous étudierons l'influence de la forme et/ou la structure des modèles UML sur l'ARC afin de déterminer des recommandations pour une utilisation, notamment sous une forme interactive. Les mesures en terme de temps d'exécution et d'espace mémoire allouée sont très importantes pour l'utilisateur afin d'avoir une meilleur interactivité. Dans cette section, nous les étudierons pour l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts. Nous prendrons en compte comme préalables les recommandations décrites dans la section précédente (cf. 6.3), à savoir le sens de la navigabilité sur l'ensemble des associations et nous allons nous limiter aux éléments de modélisation dont le nom a été défini sémantiquement.

6.4.1 Temps d'exécution

Nous mesurons le temps d'exécution sur toutes les données choisies, et nous appliquons d'une part l'AFC sur les données issues des cinq configurations ou relations (R1, R2, R3, R4 et R5) et d'autre part l'Analyse Relationnelle de Concepts sur les données issues des deux configurations (C1, C2).

Une première comparaison a été effectuée entre la relation R4 (cf. section 4.3.1.5) et la configuration C1 (cf. section 4.3.2.1). Ces dernières sont en effet proches, possédant les mêmes éléments principaux de modélisation et surtout elles s'exécutent en une seule étape pour l'AFC et deux étapes pour l'ARC. La première observation concerne l'écart important en temps d'exécution entre l'AFC et l'ARC pour toutes les versions du modèle (cf. figure 6.5. Cet écart s'explique par le nombre de concepts que contiennent le(s) treillis résultant de la factorisation.

La seconde comparaison concerne la configuration ou relation R5 (cf. section 4.3.1.6) et la configuration C2 (cf. section 4.3.2.2) qui correspondent aux cas les plus complexes dans nos expérimentations. Notre observation précédente se confirme avec un plus grand écart sur les temps d'exécution et les nombres de concepts (cf. figure 6.6).

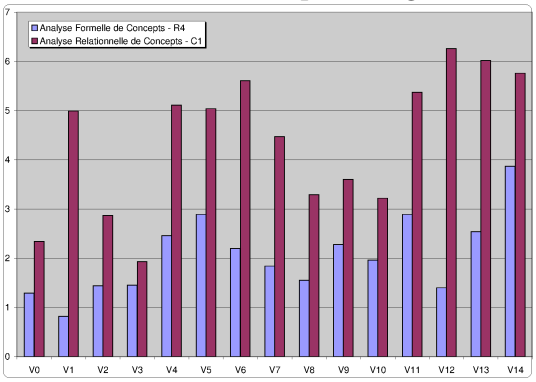


FIGURE 6.5: **Temps d'exécution (en seconde) de R4 et C1**

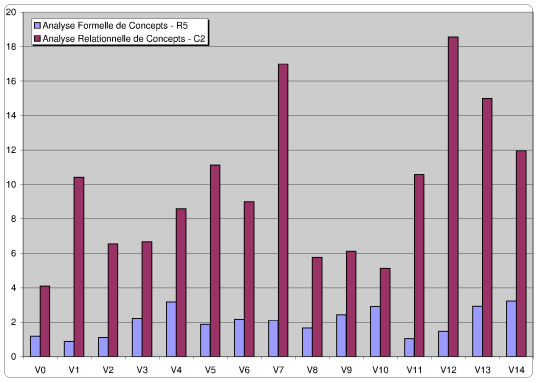


FIGURE 6.6: **Temps d'exécution (en seconde) de R5 et C2**

6.4.2 Espace mémoire utilisé

Dans cette partie, nous allons considérer une autre métrique qui est très intéressante pour l'utilisateur. Cette métrique se concentre sur l'espace mémoire occupé pendant une exécution. Nous allons suivre le même procédé que celui suivi dans la section précédente pour le temps d'exécution, et en particulier nous comparons les mêmes relations et configurations : R4 et C1, puis R5 et C2.

Comme les figures (cf. 6.7) et (cf. 6.8) le montrent, un écart existe entre les deux analyses aussi en terme d'espace occupé pendant les différentes exécutions. Cet écart vient du fonctionnement de l'approche RCA, qui construit des concepts à une étape et les utilise à l'étape ultérieure pour produire de nouveaux concepts.

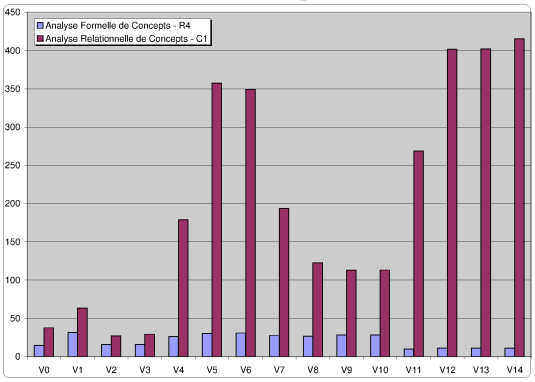


FIGURE 6.7: **Espace mémoire (en méga-bit) utilisé par R4 et C1**

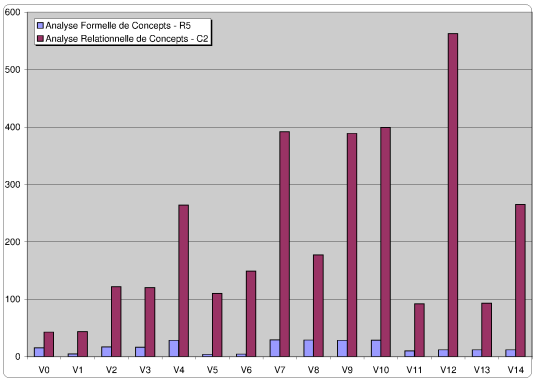


FIGURE 6.8: **Espace mémoire (en méga-bit) utilisé par R5 et C2**

Néanmoins, nous avons défini dans la section précédente des paramètres déterminants pour minimiser le temps d'exécution et l'espace mémoire occupé. La prise en compte de la navigabilité et le fait d'écarter les rôles *undefineds*, nous ont permis d'amener à leur terme toutes les exécutions avec l'ARC.

6.5 Métriques sur la complexité du modèle

La section précédente montre à travers plusieurs métriques l'importance des paramètres choisis par l'utilisateur et leur influence d'une part sur la complexité des résultats produits (taille des treillis) et d'autre part sur le temps d'exécution et l'espace mémoire occupé. Plus précisément, la complexité du modèle peut influencer :

1. Le nombre d'étapes, c'est-à-dire le nombre d'itérations pour atteindre une factorisation maximale,
2. Le nombre de concepts dans le(s) treillis final(aux),
3. Le temps d'exécution,
4. L'espace mémoire occupé.

Ces mesures sont liées les unes aux autres, par exemple plus il y a d'étapes, plus on peut s'attendre à avoir de concepts dans le résultat et plus le temps d'exécution et l'espace mémoire risquent d'augmenter.

Remarquons que lors du suivi de l'évolution de modèle basé sur l'analyse relationnelle de concepts, nous avons vu la particularité de la cinquième étape de la factorisation qui correspond au chemin allant d'une classe à une autre classe via les rôles des associations[Osman Guédi *et al.*, 2012] tout en parcourant le méta-modèle. Ce chemin correspond à un circuit du méta-modèle et nous conjecturons qu'en pratique, le nombre d'étapes est lié à la forme du méta-modèle et ne dépasse pas la longueur d'un plus long chemin simple dans le méta-modèle. C'est l'un des points que nous étudierons dans cette section.

De plus, le temps d'exécution et l'espace mémoire occupé ne dépendent pas seulement de la taille de la donnée mais aussi de sa structure. Nous allons étudier dans cette section plusieurs métriques qui permettront de décrire la structure du modèle afin d'apporter au concepteur des recommandations d'utilisation de l'ARC. Pour cette étude, nous nous intéressons aux graphes sous-jacents aux modèles.

6.5.1 Description de la structure des modèles par des graphes

La théorie des graphes, partagée par plusieurs disciplines (les réseaux, la génétique etc.), permet de modéliser une grande variété de problèmes. Dans cette partie, nous allons identifier les graphes sous-jacents aux configurations choisies pour analyser nos modèles. Nous allons donc définir quels sont leurs sommets et quels sont leurs arcs.

Nous avons abordé dans le chapitre 4, un cas d'étude qui présente les relations ainsi que les configurations retenues pour effectuer la factorisation de modèles UML du projet SIE-Pesticides. De ce fait, nous avons défini plusieurs contextes, formels et/ou relationnels qui représentent notamment les relations binaires entre les éléments de modélisation. Les contextes formels correspondent aux méta-classes et les contextes relationnels correspondent aux associations du méta-modèle (cf. Figure 6.9 pour la configuration C2). Dans notre cas de figure, le méta-modèle est inspiré de l'Atelier de Génie Logiciel (AGL) dont nous avons traduit les noms des méta-classes et ajouté les relations $owns_i$. Ce méta-modèle est conforme à celui d'UML 1.3 (cf. figure 6.9). Nous notons la présence des deux méta-classes `Attribut` et `Rôle` qui ont été fusionnées dans `EndAssociation` dans UML 2.

Dans une première phase, nous allons générer une matrice d'adjacence à partir des contexte relationnels. Les éléments de modélisation constituent les sommets du graphe tandis que les liens entre ces éléments de modélisation constituent les arcs entre les sommets du graphe. Comme les relations entre les éléments de modélisation sont orientées (Par exemple : les classes sont décrites par les attributs par une relation incluse dans Classe × Attribut), nous utilisons des graphes orientés.

Par exemple, le tableau (cf. 6.3) correspond à la matrice d'adjacence associée aux contextes relationnels définis dans les tables 4.9, 4.10, 4.14, 4.15 et 4.16 de la configuration C2 (cf. section 4.3.2.2), avec prise en compte de la navigabilité des associations. La figure 6.11 représente le graphe correspondant à la matrice d'adjacence de la table 6.3.

Dans ce graphe, nous allons calculer plusieurs métriques, d'une part pour étudier le compor-

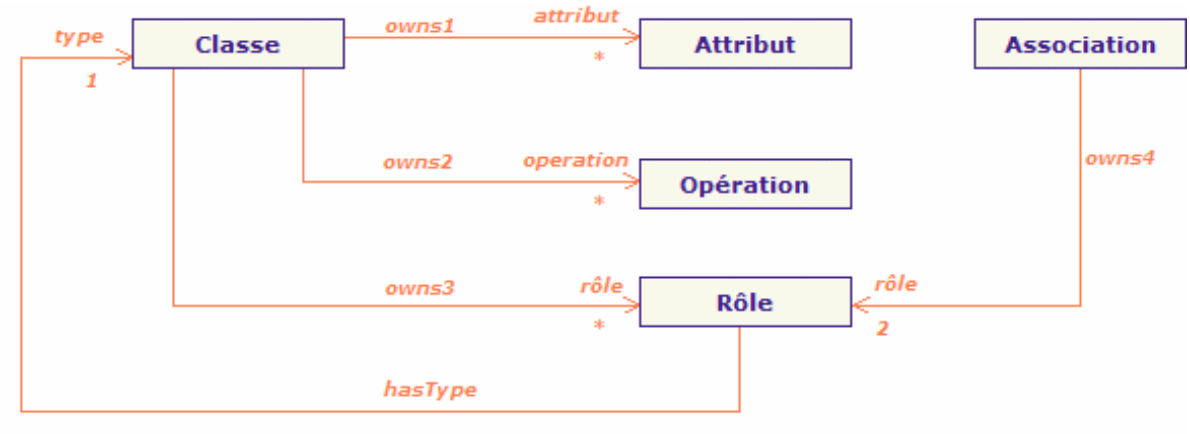


FIGURE 6.9: **Méta-modèle correspondant à la configuration C2**

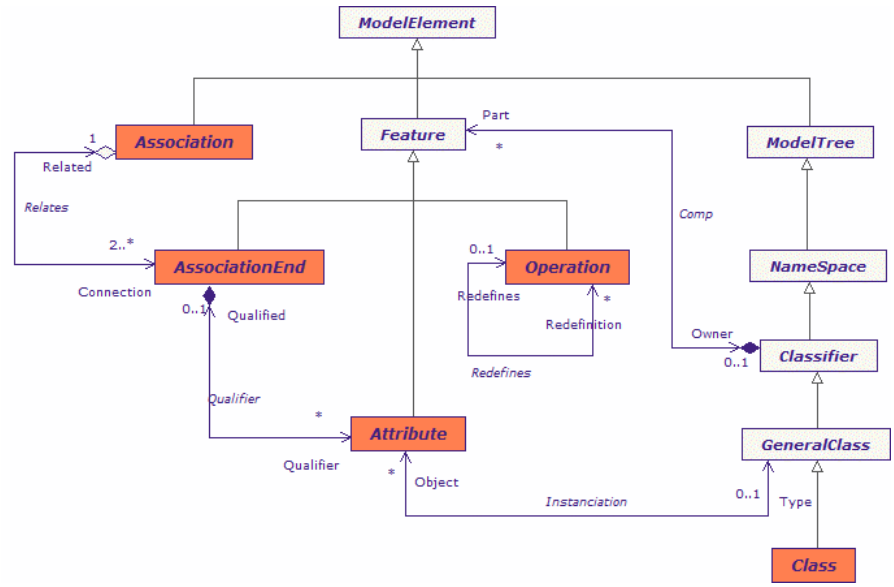


FIGURE 6.10: **Méta-Modèle Objecteering**

tement de l'ARC et si possible en déduire des recommandations pour le concepteur du système d'information.

6.5.2 Métriques basées sur les graphes

La théorie des graphes s'appliquant à plusieurs problématiques, elle propose une panoplie de métriques dont l'interprétation est spécifique à la problématique étudiée. Nous avons identifié quelques métriques qui nous ont semblé pertinentes pour notre étude.

	Piezometer.deviceType	Piezometer.deviceNumber	Piezometer.tubeDiameter	HydraulicHead.waterLevel	HydraulicHead.measuringDate	HydraulicHead.codeQuality	RainGauge.deviceType	RainGauge.deviceNumber	RainGauge.tubeHigh	Rainfall.waterAmount	Rainfall.measuringDate	Rainfall.codeQuality	MeasuringStation.stationName	MeasuringStation.administrativeInstitute	Piezometer	HydraulicHead	RainGauge	Rainfall	MeasuringStation	Piezometer.getType	RainGauge.getType	HydraulicHead :HydraulicHead	Rainfall :Rainfall	Device :Piezometer	Device :RainGauge	Measure :HydraulicHead	Measure :Rainfall	Groundwater Monitoring	Groundwater Instrumentation	Groundwater Information	Rainfall Monitoring	Rainfall Instrumentation	Rainfall Information
Piezometer.deviceType	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Piezometer.deviceNumber	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Piezometer.tubeDiameter	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HydraulicHead.waterLevel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HydraulicHead.measuringDate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HydraulicHead.codeQuality	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RainGauge.deviceType	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RainGauge.deviceNumber	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RainGauge.tubeHigh	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rainfall.waterAmount	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rainfall.measuringDate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rainfall.codeQuality	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MeasuringStation.stationName	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MeasuringStation.administrativeInstitute	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Piezometer	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
HydraulicHead	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RainGauge	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
Rainfall	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MeasuringStation	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Piezometer.getType	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RainGauge.getType	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HydraulicHead :HydraulicHead	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rainfall :Rainfall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Device :Piezometer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Device :RainGauge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Measure :HydraulicHead	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Measure :Rainfall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Groundwater Monitoring	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Groundwater Instrumentation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Groundwater Information	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Rainfall Monitoring	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Rainfall Instrumentation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Rainfall Information	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

TABLE 6.3: **Méta-Modèle Objecteering**

6.5.2.1 Métrique sur l'évolution de nombre d'arcs et de sommets du graphe

La première métrique retenue consiste à évaluer les nombres de sommets et d'arcs dans les différents modèles et à suivre leur évolution. La figure 6.12 montre l'évolution des sommets et des arcs pour la configuration C2 (avec navigabilité). Bien évidemment, nous observons la même tendance d'évolution pour les sommets et les arcs que les éléments du modèle par construction de nos graphes.

Ces entités voient leur nombre augmenter au fur et à mesure que l'on avance dans les versions avec une particularité au niveau des arcs. En effet, nous constatons que les nombres d'arcs qui définissent les liens entre les éléments de modélisation sont en hausse vers les dernières versions. En effet, l'ajout d'un nouvel élément de modélisation implique souvent l'ajout de plusieurs liens avec des éléments du modèle. Par exemple, l'ajout d'un nouveau rôle implique plusieurs liens, tels que classe-rôle (owns3), association-rôle (owns4) et rôle-classe (hastype).

FIGURE 6.11: **Graphe correspondant à la configuration C2**

6.5.2.2 La densité du graphe

Dans un second temps, nous allons aborder une métrique classique sur les graphes, qui permet d'évaluer le taux de connexité des éléments de modélisation au sein des différents modèles d'expérimentation.

La densité d'un graphe est donnée par le rapport entre le nombre de relations entre les sommets et le nombre de relations possibles c'est-à-direle nombre de sommets au carré. Plus le graphe est dense, plus le nombre d'arcs est proche du nombre maximal. De manière informelle, la densité représente à quel point tous les sommets sont liés. Dans le cas contraire, un graphe creux est un graphe avec un faible nombre d'arcs.

L'interprétation dépend fortement du domaine d'application. Dans notre cas de figure, la densité détermine comment les éléments de modélisation sont composés ou connectés entre eux dans le modèle. Par exemple, la densité d'un graphe représentant la relation Classe × Attribut c'est-à-dire

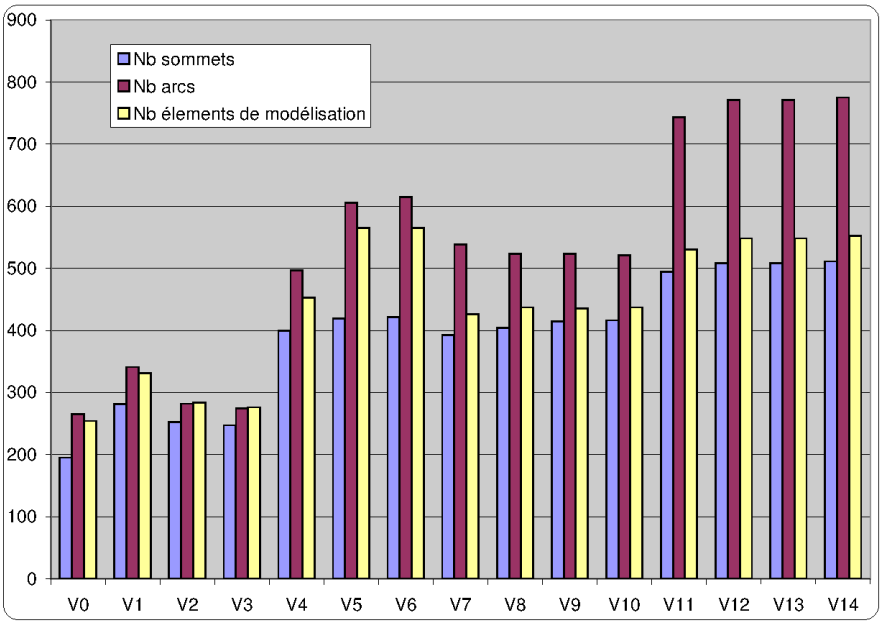


FIGURE 6.12: **Évolution des nombres de sommets, d'arcs et d'éléments du modèle**

les classes décrites par les attributs apporte une information sur le nombre d'attributs par classe dans le modèle.

$$\textbf{Densité(G) = (S,A)} = \frac{A}{S^2} \tag{6.2}$$

Nous observons une tendance générale décroissante de la densité qui se stabilise vers les dernières versions de modèle, à part deux pics issus des versions qui contiennent des paquetages en doubles. Ainsi, cette métrique exprime que les graphes deviennent de moins en moins denses. Dans les dernières versions, la densité est de 30%. A travers cette métrique, nous apprenons que les relations définies par nos configurations C1 et C2 entre les éléments de modélisation diminuent alors que le nombre de ces éléments du modèle augmentent.

Une densité importante est remarquée dans la première version de modèle (V1), qui s'explique par la forme spécifique de cette version du modèle qui dispose de beaucoup d'associations mais d'aucune relation d'héritage, autrement dit, très peu de factorisation.

Nous avons vu dans le chapitre 5 que les nouvelles abstractions augmentent significativement à partir de la cinquième itération de factorisations jusqu'à obtenir une saturation de nombre de concepts produits.

En revanche, les factorisations c'est-à-dire les fusions et les nouvelles abstractions des éléments de modélisation ont tendance à diminuer lors d'un premier circuit (cycle) de factorisation, ce qui correspond à la cinquième itération de l'Analyse Relationnelle de Concepts [Osman Guédi *et al.*, 2012].

A travers cette métrique, nous constatons que la densité évolue dans un sens opposé à l'évolution du nombre d'éléments de modélisation ainsi que de relations. (cf. 6.12. Ceci montre que le

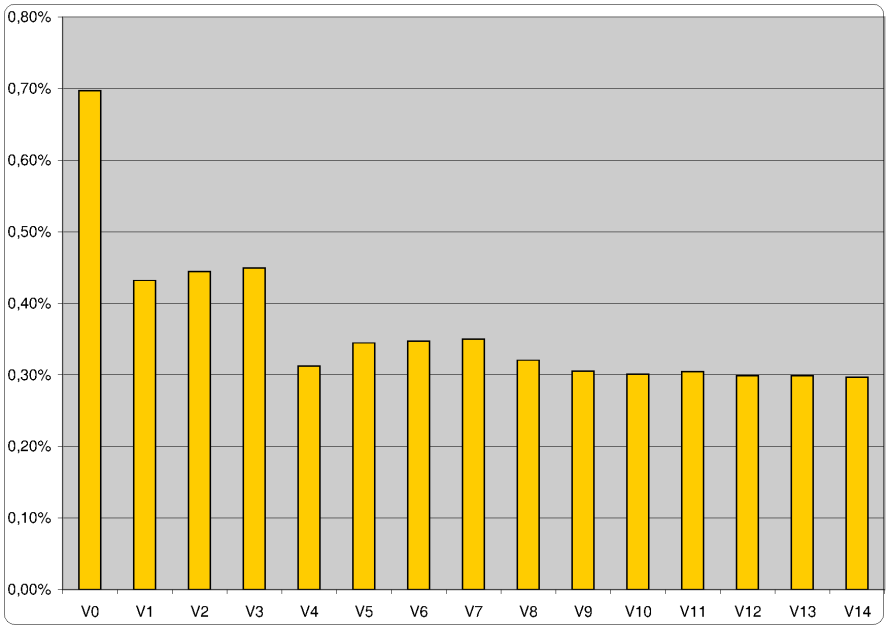


FIGURE 6.13: **Évolution de la densité du graphe correspondant à la configuration C2 (avec navigabilité)**

concepteur factorise les modèles (naturellement) d'une version à une autre, et que les différentes réunions avec les experts de domaine permettent de factoriser les modèles en ajoutant plus d'abstraction (cf. historique des modèles 5.1.2 du chapitre 5. En d'autres termes, plus de nouveaux éléments de modélisation (par exemple des classes mères) et plus de relations (par exemple, les relations d'héritage).

Cette métrique confirme nos interprétations sur les observations concernant l'analyse de l'évolution des nouvelles abstractions(cf. figure 5.19) du chapitre 5.

6.5.2.3 Le nombre d'étapes dans le pire des cas

Rappelons que l'ARC est une méthode itérative. De nouvelles abstractions sont tout d'abord créées sur la base des éléments de modélisation retenus dans la configuration (C1 ou C2). Dans l'étape suivante, les nouvelles abstractions créées participent à leur tour à la création d'autres abstractions. Enfin, ce processus se répète jusqu'à ce qu'il n'y ait plus de nouvelle abstraction créée.

La configuration C1 (cf. section 4.3.2.1) atteint en deux étapes la factorisation maximale, car les plus longs chemins dans le graphe sous-jacent sont de longueur 1. Par contre avec la configuration C2 (cf. section 4.3.2.2), le nombre d'étapes varie tout en présentant une tendance croissante au fur et à mesure des versions. La spécificité de la configuration C2 provient de la relation `hastype` que ne contient pas la configuration C1. Cette relation décrit les rôles par leurs types et ces derniers correspondent aux classes qui sont décrites par ces mêmes rôles. Cela cause un circuit dans le métamodèle et potentiellement des circuits dans certains modèles. De plus les associations sont aussi décrites par les rôles. Ces circuits augmentent le nombre d'étapes de factorisation.

La prise en compte de la navigabilité a également un impact sur le nombre d'étapes car elle influence le nombre de rôles considérés dans la factorisation. À leur tour, les rôles décrivent les classes et les associations, et sont décrits par les classes (leurs types), ce qui augmente la complexité du graphe.

La configuration C2 contient les principales relations entre les éléments de modélisation d'un modèle UML, il est donc intéressant d'analyser la quantité de nouvelles abstractions. Nous avons constaté une tendance croissante du nombre des nouveaux concepts lorsque l'on avance dans les étapes (cf. chap 5). Ceci nous a incité à évaluer la longueur d'un plus long chemin simple entre les éléments de modélisation dans cette configuration C2 et à la comparer au nombre d'étapes.

Démarche suivie .
Nous nous intéressons à calculer les plus longs chemins simples. Par "simple", nous entendons que ces chemins ne passent pas deux fois par le même arc c'est-à-dire tous les arcs du chemin sont distincts. Ils ne sont pas forcément élémentaires car ils peuvent traverser plusieurs fois le même sommet dans le cas de circuit. Par exemple, un rôle a un type qui est une classe, et cette classe a ce rôle. Nous procédons en plusieurs phases pour déterminer le taille du plus long chemin simple. Nous avons également calculé le nombre de circuits de chacun des graphes. Nous espérons obtenir avec le calcul de nombre de circuits des réponses sur le nombre de cycles de factorisation pour un modèle donné.

Première phase : génération de la matrice des arcs
La première phase consiste à transformer notre matrice d'adjacence en une relation (matrice) entre les arcs. Ces arcs sont nommés par concaténation des deux noms des sommets respectivement source et destination de l'arc. Par exemple, lorsqu'un arc relie un sommet A (source) à un sommet B (destination), le nom de l'arc est AB. Un arc est en relation avec un autre si la destination du premier est la source du second.

	A	B	C	D
A	0	1	1	1
B	0	0	0	1
C	0	1	0	0
D	0	0	1	0

TABLE 6.4: **Matrice d'adjacence**

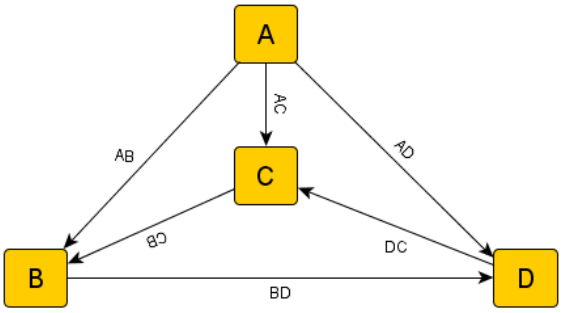


FIGURE 6.14: **Graphe de la matrice d'adjacence**

A titre d'exemple d'application, le tableau 6.4 correspond à une matrice d'adjacence dont le graphe est représenté à la figure 6.14. La relation entre les arcs correspondant à la matrice d'adjacence précédente est représentée dans le tableau 6.5 et le graphe associé est représenté par la figure 6.15.

	AB	AC	AD	BD	CB	DC
AB	0	0	0	1	0	0
AC	0	0	0	0	1	0
AD	0	0	0	0	0	1
BD	0	0	0	0	0	1
CB	0	0	0	1	0	0
DC	0	0	0	0	1	0

TABLE 6.5: **Relation entre les arcs**

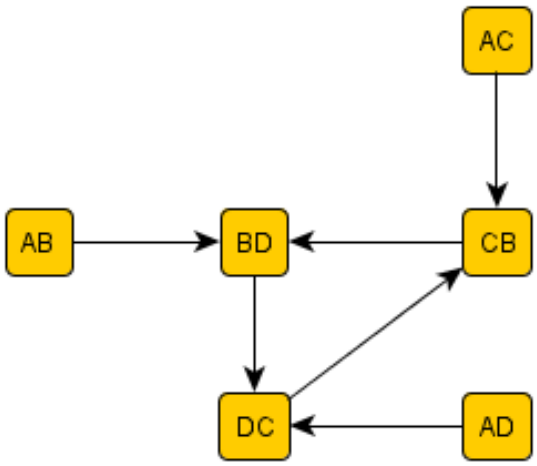


FIGURE 6.15: **Graphe associé à la relation entre les arcs**

Seconde phase : calcul de la longueur d'un plus long chemin simple

La seconde phase consiste à parcourir l'ensemble des chemins du graphe afin de déterminer la taille d'un plus long chemin simple. La recherche d'un plus court chemin est une problématique qui a été largement traitée et pour laquelle des algorithmes efficaces ont été proposés, et il est possible que ces algorithmes s'adaptent à la recherche d'un plus long chemin simple. Cependant nous n'avons pas cherché ici à proposer un algorithme particulièrement efficace et nous nous sommes inspirés d'une adaptation de l'algorithme de Bellman [Bellman, 1958] dont l'objectif reste le calcul du plus court chemin (cf. l'algorithme de calcul du plus long chemin A.1). Dans notre exemple, voici les chemins maximaux au sens où nous ne pouvons plus les allonger.

1 $AB \rightarrow BD \rightarrow DC \rightarrow CB \Rightarrow$ longueur = 4
2 $AC \rightarrow CB \rightarrow BD \rightarrow DC \Rightarrow$ longueur = 4
3 $AD \rightarrow DC \rightarrow CB \rightarrow BD \Rightarrow$ longueur = 4
4 $BD \rightarrow DC \rightarrow CB \Rightarrow$ longueur = 3
5 $CB \rightarrow BD \rightarrow DC \Rightarrow$ longueur = 3
6 $DC \rightarrow CB \rightarrow BD \Rightarrow$ longueur = 3

Nous observons six chemins maximaux possibles donc la longueur d'un plus long chemin simple est de quatre arcs. Dans notre algorithme, le parcours de l'ensemble des chemins est indispensable pour déterminer les plus longs chemins.

Nous avons appliqué notre algorithme au paquetage de station de mesure pour la configuration C2 en prenant en compte la navigabilité. Nous obtenons le graphe correspondant à la figure 6.16 avec la représentation d'un des plus longs chemins. Ce dernier parcourt cinq arcs du graphe. Cette longueur (5) est supérieure au nombre d'étapes effectuées par RCA dans ce cas.

Cependant, nous avons aussi expérimenté le cas où la navigabilité est totalement ignorée (cf. figure A.1), et nous observons là encore que la longueur du plus long chemin simple (15) est supérieure au nombre d'étapes de RCA (5).

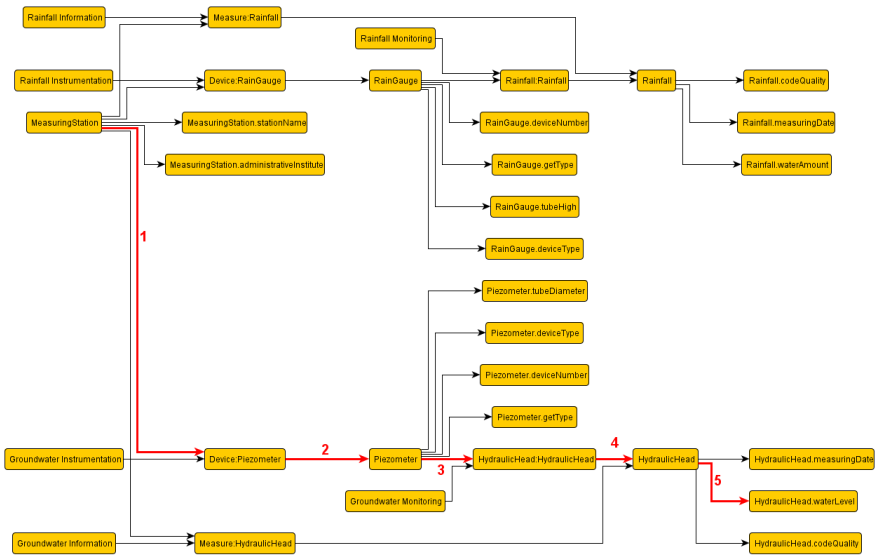


FIGURE 6.16: **Un plus long chemin dans le graphe de la configuration C2 avec la navigabilité**

Notre conjecture, qui dit que la longueur d'un plus long chemin simple est supérieure au nombre d'étapes de l'Analyse Relationnelle de Concepts, est confirmée en pratique par les expérimentations sur les quinze versions de modèle (cf. figure 6.17. Cela donne une borne pratique utilisable par un concepteur pour se limiter à un nombre d'étapes qu'il estimera proche ou éloigné de cette borne.

Troisième phase : calcul du nombre de circuits dans le graphe

Pour finir, nous exploitons les derniers résultats pour déterminer le nombre de circuits ainsi que leur longueur afin d'estimer s'il existe une relation possible entre ces valeurs et le nombre d'étapes. Un circuit correspond à un chemin dont les extrémités sont identiques. Dans notre précédent exemple, nous disposons de trois circuits.

1 $BD \rightarrow DC \rightarrow CB \Rightarrow$ longueur = 3

2 $CB \rightarrow BD \rightarrow DC \Rightarrow$ longueur = 3

3 $DC \rightarrow CB \rightarrow BD \Rightarrow$ longueur = 3

À travers la figure 6.18, nous n'observons pas de relation en pratique entre les deux quantités, à savoir le nombre de circuits et le nombre d'étapes.

Notons qu'un même algorithmique calcule ces deux métriques (cf. algo A.1), car le circuit est un chemin parmi ceux qui sont parcourus pour le calcul d'un plus long chemin, il faut juste détecter qu'un sommet atteint fait déjà partie du chemin parcouru.

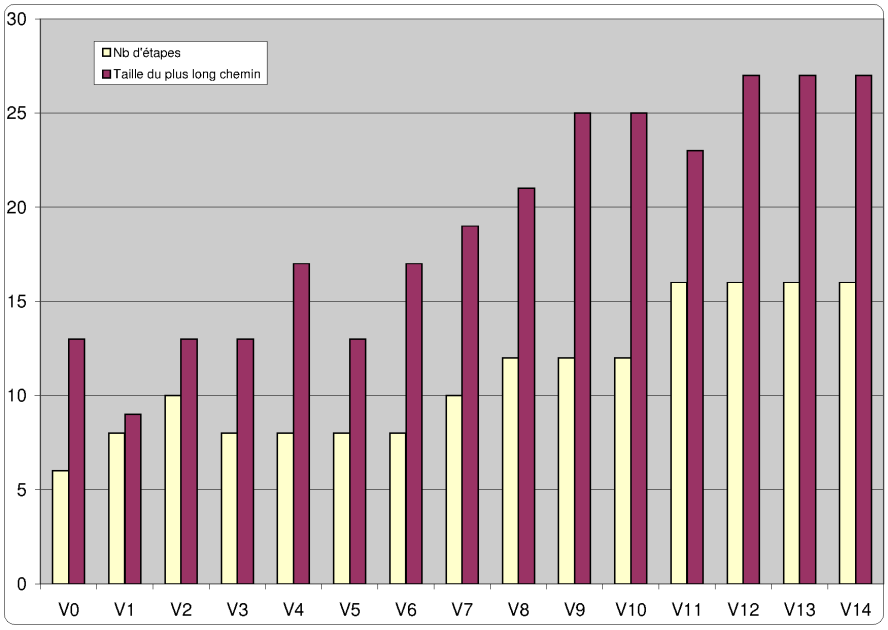


FIGURE 6.17: **Évolution du nombre d'étapes et la taille du plus long chemin dans le graphe de configuration C2 avec la navigabilité**

6.5.2.4 Degré des sommets du graphe

Dans un graphe orienté, le degré d'un sommet est le nombre d'arcs entrants et sortants de sommet. Étant donné que l'ensemble des sommets des graphes représente les éléments des modélisations, le degré permet de distinguer les éléments de modélisation ayant le plus de relations avec les autres, et on peut penser que ce sont des sommets susceptibles de générer le plus de factorisation.

Par exemple, le graphe correspondant au paquetage station de mesure présenté à la figure 6.20 distingue les sommets en fonction de leur degré en les présentant dans des boîtes dont la taille est d'autant plus grande que le degré l'est. Nous observons que les classes sont les éléments les plus reliés à d'autres éléments de modélisation, ce qui découle essentiellement du choix des relations retenues dans nos configurations. De même, dans le treillis de classes nous observons beaucoup d'attributs relationnels. (cf. figure 6.19).

Toutefois, la métrique sur le degré ne distingue pas que les classes, les rôles ont également un degré plus important que d'autres éléments. Nous le constatons avec le graphe correspondant à la configuration C2 où la navigabilité est ignorée (cf. figure A.2) et les mêmes conclusions se lisent dans le treillis des rôles (cf. figure A.3).

Par conséquent, cette métrique apporte une information sur les principaux éléments qui seront susceptibles de générer le plus de nouvelles abstractions. De ce fait, le concepteur peut d'une part, mieux choisir ces relations de factorisation et d'autre part, cette métriques renseigne le concepteur sur les éléments de modélisation qui influencent ces factorisations.

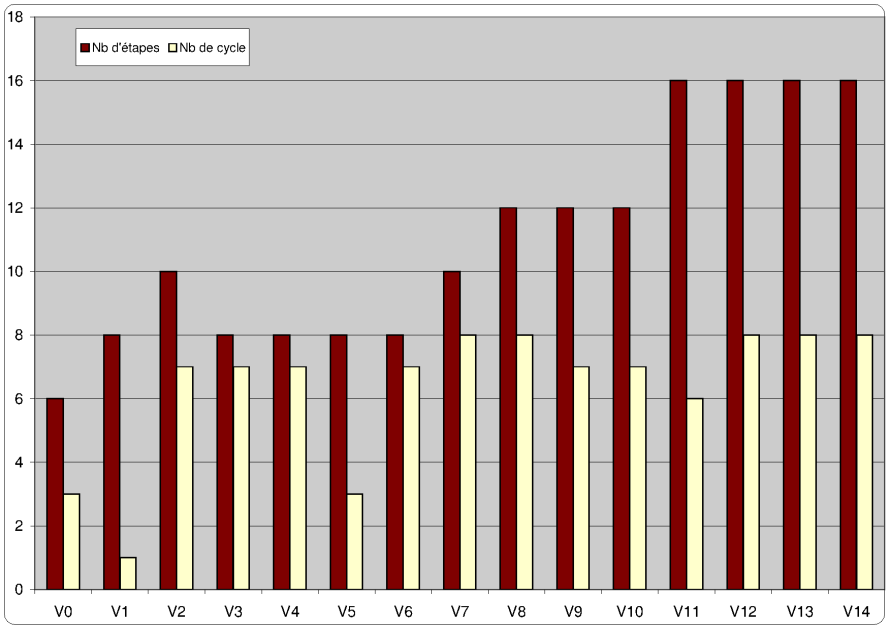


FIGURE 6.18: **Évolution du nombre d'étapes et du nombre de circuits dans le graphe de configuration C2**

6.6 Conclusion

L'ARC étend l'AFC aux relations entre les éléments de modélisation, ce qui permet d'obtenir des factorisations inédites mais au prix d'une augmentation de la complexité des résultats obtenus.

Pour assister le concepteur dans le contrôle de cette complexité, nous étudions le comportement de l'ARC sur les quinze versions de modèles du projet SIE-Pesticides. Dans le comportement, nous étudions le gain en nombre d'étapes, en nombre de concepts créés, en temps de calcul et en espace mémoire occupé. Les premières recommandations que nous en tirons concernent l'usage de la navigabilité, qui est recommandé, et l'usage des rôles *undefineds*, qui n'est pas recommandé.

Nous avons également étudié la structure des graphes sous-jacents aux modèles, et des indicateurs sur ces graphes, tels que les nombres de sommets et d'arcs, la densité, la longueur d'un plus long chemin simple, le degré ou encore le nombre de circuits. Lorsque l'on avance dans les versions, on observe que les nombres de sommets et d'arcs augmentent, tandis que la densité a tendance à diminuer au début, puis à se stabiliser. La longueur d'un plus long chemin simple augmente, mais on peut constater qu'elle reste toujours inférieure au nombre d'étapes effectuées par l'Analyse Relationnelle de Concepts.

Le degré permet d'identifier les éléments les plus connectés et qui seront potentiellement impliqués dans plus de factorisations que les autres, ou à l'inverse, les éléments isolés qui joueront un faible rôle dans la factorisation.

L'analyse approfondie de certaines métriques de cette étude peut être utilisée pour affiner l'analyse de l'évolution. Par exemple, le degré permettra d'apporter le choix des éléments de modélisation pour effectuer la factorisation (les paramètres de factorisation). Et d'autre part, l'analyse de

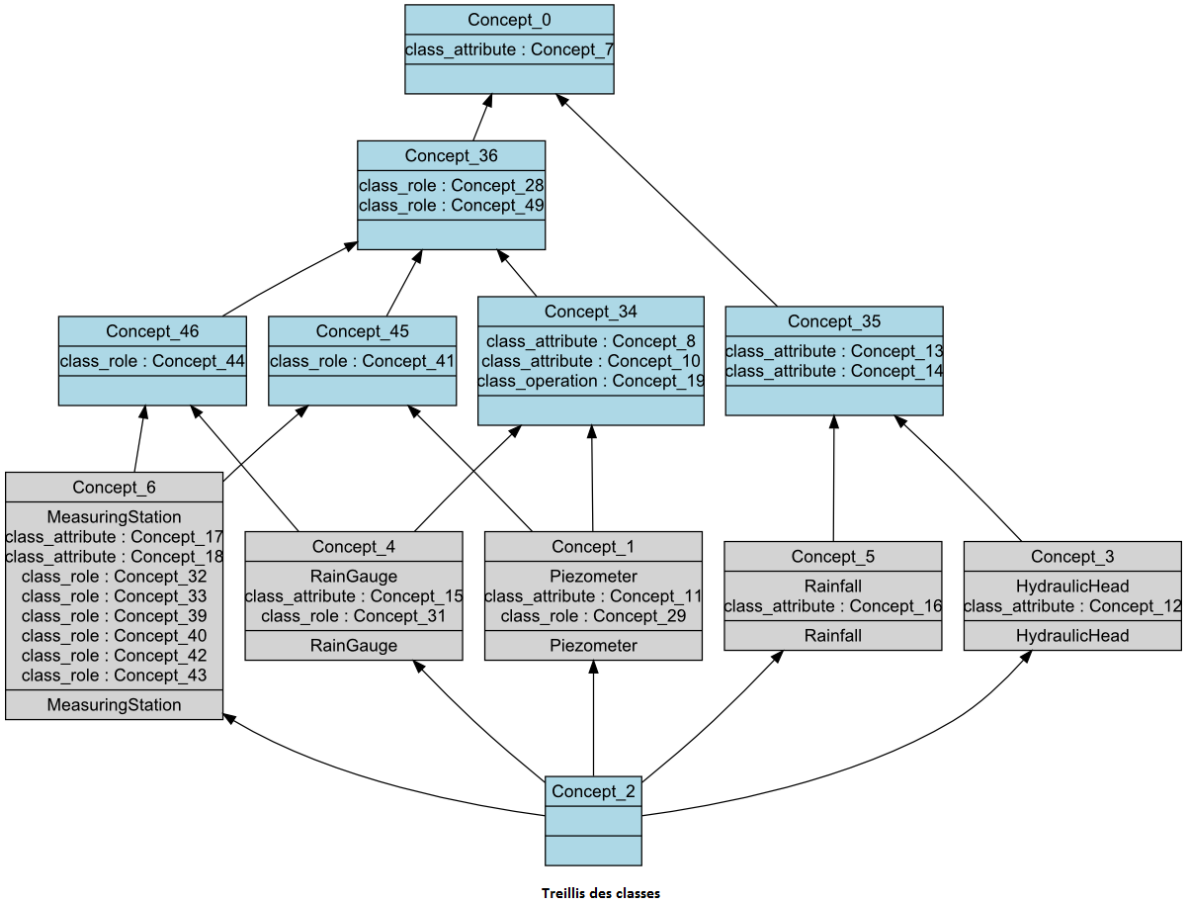


FIGURE 6.19: **Le treillis des classes (C2 avec la navigabilité)**

l'évolution de ces paramètres de factorisation permettra de déterminer les éléments de modélisation bien factorisés à travers les différentes versions du modèle et par rapport aux factorisations déduites.

Pour conclure, ce chapitre a proposé un ensemble de métriques et de recommandations utiles pour comprendre et définir un cadre d'utilisation de l'Analyse Relationnelle de Concepts et maîtriser ses résultats. Un article, joint en annexe, vient compléter ces conclusions.

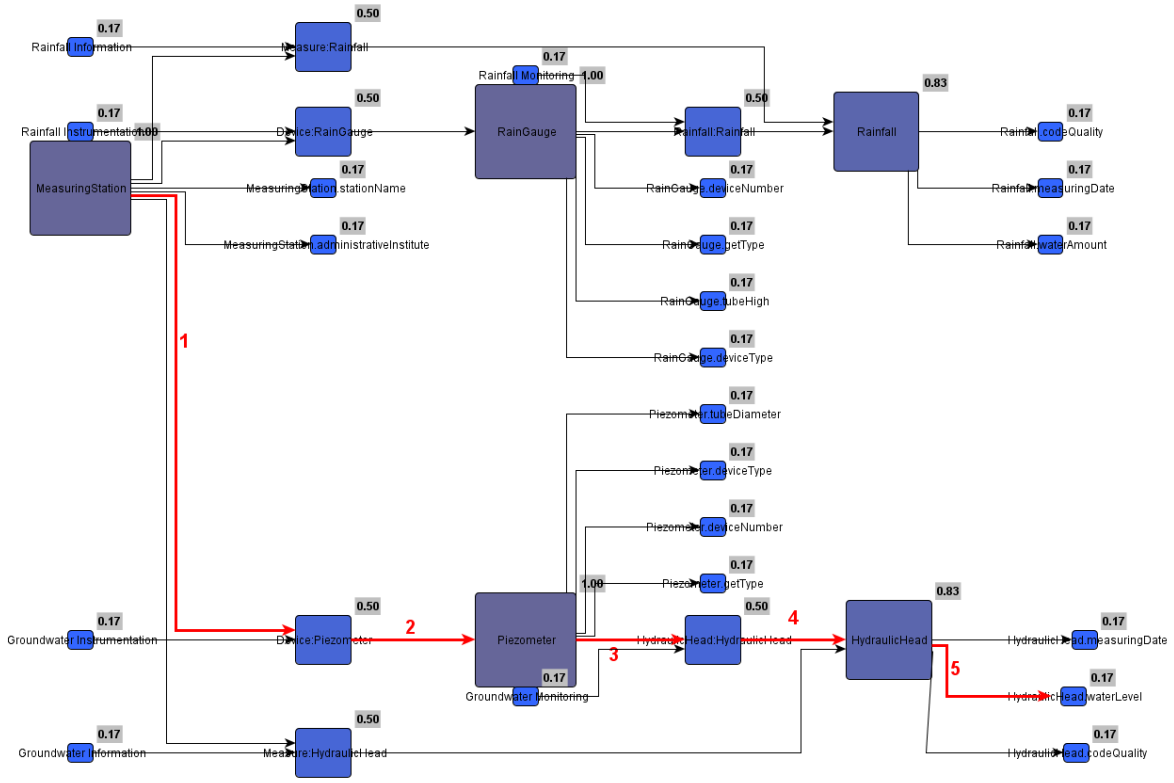


FIGURE 6.20: **Le degré de chaque sommet du graphe (C2 avec la navigabilité)**

CHAPITRE

7

Factorisation inter-modèles Introduction de la notion et construction du Plus Grand Modèle Commun

Avec l'Analyse Formelle de concepts

Préambule

L'intégration des données et la capitalisation des connaissances conduisent à manipuler des données et des informations produites par des utilisateurs ou des experts différents ayant chaque un point de vue distinct voire divergent sur un même système. Ces points de vue sont souvent exprimés sous forme de modèles. Lors du regroupement des différents modèles, il est nécessaire d'identifier l'ensemble des concepts communs au domaine afin de promouvoir la connaissance partagée et d'éviter les doublons. Cet ensemble est un modèle constitué de l'intersection maximale de tous les modèles sources. Nous appellerons ce nouveau modèle le Plus Grand Modèle Commun. Le présent chapitre présente la méthode permettant sa construction qui repose sur l'Analyse Formelle de Concepts. Nous appliquons cette approche au projet SIEPesticides.

Sommaire

7.1 Introduction

L'élaboration de modèles métiers est une activité de fond dans de nombreux projets de domaines différents et surtout pour des objectifs spécifiques tels que la construction de dictionnaires, la conception de bases de données ou le développement de logiciels dédiés à un domaine particulier. Habituellement, de tels modèles du domaine sont réalisés par plusieurs équipes d'experts. Par exemple, à Irstea, l'étude de l'impact des pesticides sur l'environnement fait intervenir des spécialistes de différents domaines scientifiques incluant l'hydrologie, l'agronomie, la chimie, etc.

Chaque spécialiste étant capable de modéliser la partie du modèle relative à son domaine d'expertise, il faut ensuite consolider l'ensemble au sein d'un modèle unique regroupant tous les modèles. Cette activité de regroupement est complexe et généralement effectuée manuellement. En effet, les modèles produits par les différents experts étant proches, il est nécessaire de détecter les concepts similaires dans les différents modèles spécialisés, afin de les intégrer sans redondance dans le modèle consolidé appelé le Plus Grand Modèle Commun. Ce modèle commun est particulièrement utile pour effectuer de l'intégration de données. De plus, il capitalise la connaissance experte commune aux différents participants. Nous avons partiellement automatisé la construction du Plus Grand Modèle Commun, qui est composé des concepts communs aux différents modèles.

Pour ce faire, notre méthodologie est basée sur l'Analyse Formelle de Concepts qui est une méthode d'analyse exacte et robuste de données basée sur la théorie des treillis. Ensuite, les concepts formels des treillis sont classés selon un arbre de décision pour assister le concepteur dans la création du Plus Grand Modèle Commun. Notre méthodologie permet en outre d'identifier les concepts spécifiques au sein de plusieurs modèles et propose de nouveaux concepts plus abstraits, à la fois dans le Plus Grand Modèle Commun donc entre plusieurs deux modèles distincts de données (factorisation inter-modèles) et dans les modèles d'origine c'est-à-dire à l'intérieur des modèles pris individuellement (factorisation intra-modèle).

Le terme "Plus Grand Modèle Commun", en anglais Greatest Common Model (GCM), est choisi par analogie avec la notion arithmétique de "Plus Grand Commun Diviseur" (PGCD). Le GCM contient tous les concepts du domaine qui sont présents dans tous les modèles étudiés. Le GCM peut être assimilé à une forme "normale" de tous les modèles sources.

Dans ce chapitre, nous allons appliquer l'Analyse Formelle de Concepts sur les modèles de classes du projet *SIE-Pesticides* (cf. section 4.2.1) pour détecter automatiquement les concepts communs au domaine et proposer éventuellement de nouvelles abstractions généralisant les concepts initiaux. La méthode est illustrée sur le même sous-modèle extrait de deux versions différentes du modèle *SIE-Pesticides*. Ces deux sous-modèles correspondent au paquetage *Station de mesure* (cf. figure 4.3). Cette approche est aussi en cours d'évaluation sur le modèle *SIE-Pesticides* dans son entier, dans lequel deux équipes thématiques coopèrent pour élaborer le modèle de données du système. L'équipe "transfert" est spécialisée dans l'étude du transfert des pesticides vers les cours d'eau et l'équipe de "pratique" effectue des recherches sur les pratiques agricoles des agriculteurs.

Dans un premier temps, nous allons présenter les sous-modèles *Station de mesure* avec leurs spécificités, les définitions des notions manipulées dans le reste du document et la ligne principale de notre approche.

Ensuite, nous allons décrire en détail l'approche permettant d'extraire automatiquement (ou quasi-automatiquement) le Plus Grand Modèle Commun, tout en expliquant la démarche utilisée

avec l'Analyse Formelle de Concepts sur les modèles sources et la façon dont les treillis qui en résultent sont analysés afin de fournir des recommandations au concepteur et/ou à l'utilisateur final.

Puis, nous allons présenter le processus que nous avons mis en place ainsi que l'outil permettant de générer semi-automatiquement le Plus Grand Modèle Commun. Enfin, nous appliquons l'approche sur différentes versions du modèle *SIE-Pesticides* global.

7.2 Description des deux modèles *Station de mesure* utilisés pour l'explication

Le Système d'information sur l'environnement des pesticides [Pinet *et al.*, 2010; Miralles *et al.*, 2011] a pour objectif de centraliser les connaissances et les informations produites par les équipes "Transfert" et "Pratique" (cf. section 7.1). Nous illustrons notre approche sur le modèle (*MeasuringStation*) représentant les stations de mesure situées sur le bassin versant et gérées par l'équipe "Transfert". Ce modèle comprend l'instrumentation nécessaire à l'étude du transfert des pesticides vers les cours d'eau.

La figure 7.1 montre les deux (sous-)modèles *Station de mesure* utilisés dans cette section. Ces modèles sont produits par les deux équipes impliquées dans le projet. Étant donné qu'ils sont très proches, nous avons regroupé, à droite du concept station de mesure (*cl_MeasuringStation*), les concepts identiques ayant aussi des relations (associations) identiques. Dans cette partie du modèle, les données mesurées sont associées à l'instrument de mesure associé : les précipitations (*cl_Rainfall*) et la pression hydrostatique (*cl_HydraulicHead*) de la nappe phréatique sont enregistrées en continu respectivement par le pluviomètre (*cl_RainGauge*) et par le piézomètre (*cl_Piezometer*). Chacune de ces mesures est datée (cf. propriété *att_MeasuringDate* de la figure 7.1).

À gauche du concept station de mesure *cl_MeasuringStation*, nous avons, pour le modèle M1 (*M1_MeasuringStation*), regroupé les concepts qui permettent d'enregistrer les données mesurées par une station météo, à savoir : la température (*cl_Temperature*), l'hygrométrie (*cl_Hygrometry*) et l'évapo-transpiration potentielle (*cl_PET*) des cultures vertes.

Ces derniers concepts du domaine sont absents du modèle M2 (*M2_MeasuringStation*) qui possède d'autres instruments de mesure tels que le limnimètre (*cl_Limnimeter*) pour mesurer en continu le débit (*cl_FlowRate*) du cours d'eau. Régulièrement, un technicien est chargé de prélever des échantillons d'eau afin de déterminer en laboratoire la quantité des pesticides dans le cours d'eau (*cl_PesticideMeasurement*). Enfin, la vitesse du vent (*cl_WindMeasurement*) est un paramètre provenant d'une station météorologique gérée soit par l'équipe elle-même, soit par Météo-France au niveau d'un bassin versant englobant. Nous présentons dans la section suivante, les mécanismes de la factorisation sur ces deux modèles.

7.3 L'approche basée sur la factorisation inter-modèles

Tout d'abord, nous allons définir les expressions utilisées dans la démarche qui nous amène à la factorisation inter-modèles.

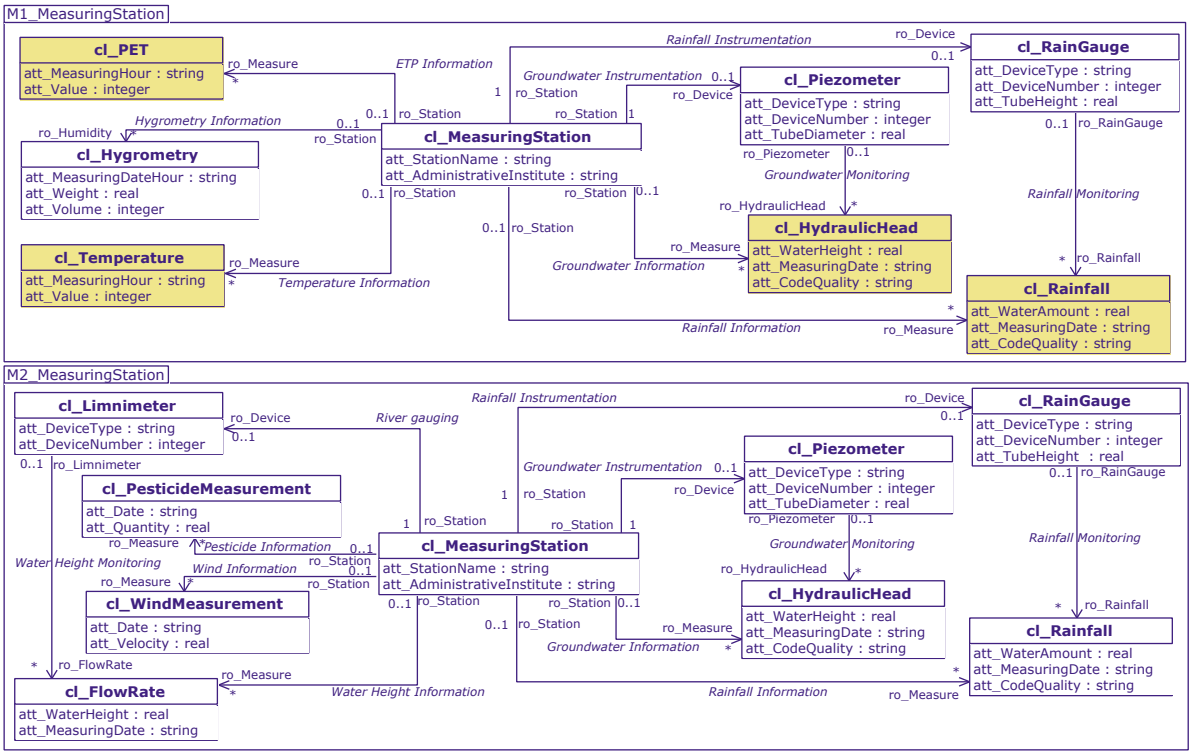


FIGURE 7.1: **Deux modèles de données de la station de mesure produits par deux équipes**

7.3.1 Définitions dans le contexte de la factorisation inter-modèles

Définition 7.1 *Un* concept spécifique *est un concept du domaine appartenant à un modèle unique.*

Par exemple, le concept de domaine *cl _PET* mais aussi *cl_Hygrometry*, *cl_Temperature*, etc. sont des concepts spécifiques au modèle de *M1_MeasuringStation.*

Définition 7.2 *Un* concept commun *est un concept de domaine existant au moins au sein de deux modèles sources et ayant un ensemble minimal de caractéristiques communes. Dans note cas, le nom du concept est déterminant pour identifier le concept commun de deux ou plusieurs modèles.*

Dans la figure 7.1, les concepts de domaine situés à droite de *cl_MeasuringStation* sont des concepts communs. Ces deux modèles expérimentaux *cl_MeasuringStation de M1 et M2* ont assez de caractéristiques communes pour trouver des concepts communs.

Définition 7.3 *Une* factorisation intra-modèle *est un mécanisme automatique ou semi-automatique au sein d'un modèle dont l'objectif est de faire émerger dans le modèle de nouveaux concepts spécifiques ayant un niveau supérieur d'abstraction.*

Définition 7.4 *Une* factorisation inter-modèles *est un mécanisme automatique ou semi-automatique entre au moins deux modèles dont l'objectif est de faire émerger des concepts du domaine communs aux modèles sources.*

Dans notre approche, l'Analyse Formelle de Concepts est utilisée pour les factorisations intra et inter-modèles. Cette analyse a été appliquée simultanément aux deux modèles de la figure 7.1. En factorisation inter-modèles, nous verrons que les deux mécanismes de factorisations existent simultanément et qu'ils sont indissociables (cf. figure 7.6).

Définition 7.5 *Le* Plus Grand Modèle Commun "Greatest Commun Model"(GCM) *est l'intersection maximale de l'ensemble des concepts communs dans tous les modèles sources.*

Les différents mécanismes produisant l'ensemble des concepts communs constituant le Plus Grand Modèle Commun sont détaillés dans la section 7.4.1.

7.3.2 L'objectif de la factorisation inter-modèles

L'objectif principal de notre approche est de regrouper les concepts communs à deux ou plusieurs modèles réalisés indépendamment au sein du Greatest Common Model. Le GCM contient tous les concepts communs du domaine qui sont introduits dans tous les modèles étudiés et s'apparentes à une forme normale de factorisation [1].

L'approche proposée est illustrée dans la figure 7.2. En entrée, le processus nécessite deux (ou plusieurs) modèles d'un domaine que nous avons, dans notre cas d'expérimentation, appelé M_1 et M_2. Dans un premier temps, les classes des modèles d'entrée sont décrites par leurs caractéristiques, ensuite, avec l'Analyse Formelle de Concepts (cf. chapitre 3) qui permet de regrouper les caractéristiques des concepts du domaine en concepts formels tout en générant des résultats dans des treillis offrant une vue hiérarchique de ces concepts formels. L'Analyse Formelle de Concepts peut traiter plusieurs caractéristiques de classe : les attributs, les opérations, les rôles... Nous l'appliquons pour décrire les classes contenues dans les modèles, ce qui entraîne plusieurs treillis. Ces treillis permettent l'identification de concepts communs, des concepts spécifiques et peut-être de nouvelles abstractions extraites de factorisation intra-modèle ou inter-modèles.

L'approche proposée est illustrée dans la figure 7.2. En entrée, le processus prend deux (ou plusieurs) modèles de classe d'un domaine que nous avons appelés ici M_1 et M_2. Dans un premier temps, les classes sont décrites par leurs caractéristiques dans des contextes formels. Puis l'Analyse Formelle de Concepts (cf. chapitre 3) permet de regrouper les caractéristiques des concepts du domaine en concepts formels tout en générant des résultats dans des treillis offrant une vue hiérarchique de ces concepts formels. L'Analyse Formelle de Concepts peut traiter conjointement plusieurs caractéristiques des classes telles que les attributs, les opérations ou les rôles.

Aussi bien en factorisation intra-modèle qu'inter-modèles, l'Analyse Formelle de Concepts produit plusieurs treillis qui permettent d'identifier les concepts communs (figures en trait plein), les concepts spécifiques (figures en trait plein) et, éventuellement, de nouvelles abstractions (figures en pointillé). En factorisation inter-modèles, il y a plusieurs résultats produits simultanément :

1. Ici, nous nous référons à une sorte de forme normale relationnelle telle que celles qui sont utilisées dans la normalisation des modèles de bases de données et qui ont ce même objectif d'éliminer les redondances.

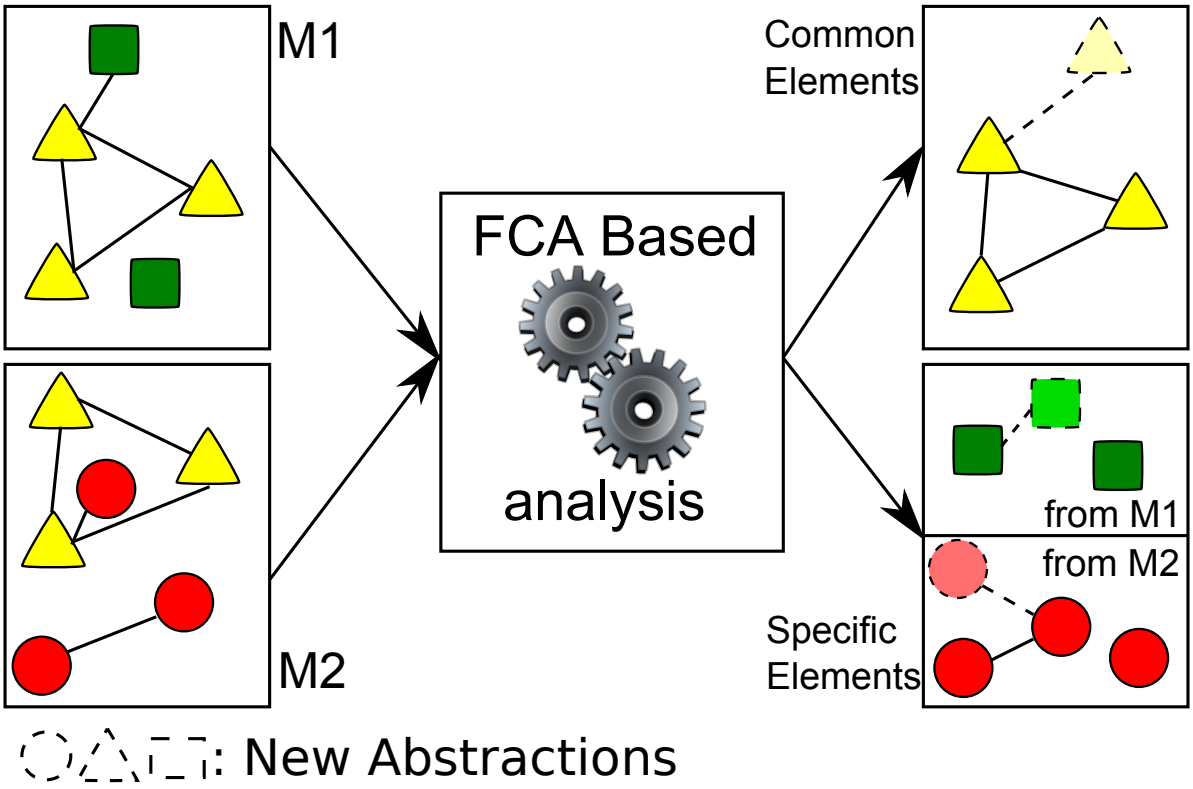


FIGURE 7.2: **Un aperçu schématique de notre approche**

- les concepts propres à un modèle : ce sont les concepts spécifiques (cercles et carrés en trait plein),
- les concepts communs, c'est-à-dire présents dans tous les modèles : ils sont dans le GCM (triangles en trait plein),
- les concepts correspondant aux nouvelles abstractions communes, en traits pointillés.

La factorisation intra-modèle, indissociable de la factorisation inter-modèles, fait émerger des abstractions au sein des modèles (cercle et carré en pointillé) qui peuvent se retrouver dans le GCM (triangle en pointillé).

Par exemple, si nous décrivons les classes par leurs attributs, le treillis qui en résulte (cf. figure 7.5) extrait les concepts du domaine commun de deux modèles tels que ceux de la figure 7.1. Il extrait également de nouvelles abstractions. Quelques nouvelles abstractions sont présentes à la fois dans M_1 et M_2 (par exemple, le concept *att_DeviceType* factorise les concepts communs de pluviomètre et de piézomètre : il s'agit d'une factorisation inter-modèles). Certaines autres abstractions extraites ne sont présentes que dans un même modèle (par exemple, le concept *att_Date* factorise la date de la mesure des pesticides et la mesure de vent dans M_2 : il s'agit d'une factorisation intra-modèle). Pour chaque treillis, nous avons deux degrés d'analyse pour ces concepts du domaine, autrement dit, certains de ces concepts sont susceptibles d'être dans le GCM tandis que d'autres doivent être plus précisément analysés, validés et nommés par l'expert final.

Cependant, comme nous produisons plusieurs treillis, l'expert en charge de l'intégration doit suivre une stratégie pour les analyser. Nous proposons d'ordonner les treillis obtenus en utilisant la force des différents critères de factorisation. Dans notre approche, nous classons en effet les classes du modèle en utilisant différentes caractéristiques (par exemple, leur nom et leurs propriétés). Les treillis sont ensuite analysés afin de catégoriser les concepts formels et de les interpréter, le cas échéant, pour déterminer les concepts du domaine.

Tout d'abord, nous produisons les concepts du domaine candidats pour le plus grand modèle commun (GCM). Ces concepts sont des concepts observés à la fois dans M1 et dans M2. Ces concepts du domaine peuvent également être des abstractions présentes à la fois dans les deux modèles M1 et M2, par exemple à partir des concepts de domaine de M1 et M2, on peut extraire le *measuring instrument* comme une abstraction du pluviomètre, du limnimètre et du piézomètre. Ils proviennent à la fois d'une factorisation intra-modèle et d'une factorisation inter-modèles entre M1 et M2.

Deuxièmement, nous identifions les *concepts spécifiques du domaine* de M1 (resp. M2), donc qui n'appartiennent qu'à M1 (resp. M2). Par exemple, dans la figure 7.1, *cl_PET* ou *cl_Hygrometry*, sont spécifiques au modèle de *M1_MeasuringStation* sans contrepartie en M2. Nous proposons également une factorisation intra-modèle des concepts spécifiques du domaine. Dans notre exemple, nous remarquons qu'un concept *dated measurements* pour dater les mesures du vent et des pesticides peut être déduit de la factorisation de l'attribut commun *att_Date*.

Les concepts de domaine reconnus par les experts comme étant dans le GCM sont appelés les *concepts de domaine commun*. Dans la figure 7.1, les concepts à la droite de *cl_MeasuringStation* sont certainement essentiels à tous les experts des deux équipes.

7.3.3 Treillis utilisés

Nous rappelons dans cette section quelques éléments principaux de l'Analyse Formelle de Concepts, en l'appliquant sur la problématique de ce chapitre. L'Analyse Formelle de Concepts [Ganter et Wille, 1999], décrite dans le chapitre 3, correspond à un cadre théorique pour la classification conceptuelle, l'analyse des données ou le regroupement conceptuel [Birkhoff, 1940]. Elle est appliquée dans de nombreux domaines, y compris la structuration des connaissances, la recherche d'information, la fouille de données mais aussi en factorisation de modèles de classe ou à partir du code source de programmes. Elle décrit les entités étudiées par leurs caractéristiques formelles et permet de découvrir les concepts qui sont des groupes maximaux d'entités partageant les mêmes caractéristiques. Un ordre partiel sur la base de l'inclusion ensembliste d'entités est fourni sous forme d'une structure en treillis (le treillis de concepts) et organise des concepts par la spécialisation.

Dans un même treillis, les entités sont organisées suivant un ordre ascendant alors les caractéristiques sont ordonnées dans l'ordre inverse. L'intension simplifiée d'un concept formel est son intension propre, privée des caractéristiques héritées des intensions de ses super-concepts. L'extension simplifiée est définie d'une manière symétrique. Pour des raisons de lisibilité, tous les treillis présentés par la suite contiendront des intensions et des extensions simplifiées.

La figure 7.3 montre le treillis de concepts construit à partir du contexte formel présenté par la table 7.1. Chaque concept formel est représenté par un rectangle constitué de trois parties : la pre-

	att_MeasuringHour	att_Value	att_WaterAmount	att_MeasuringDate	att_CodeQuality	att_WaterHeight
cl_PET	×	×				
cl_Temperature	×	×				
cl_Rainfall			×	×	×	
cl_HydraulicHead				×	×	×

TABLE 7.1: **Le contexte formel du modèle réduit**

mière partie contient le nom généré du concept formel, la deuxième partie contient son intension simplifiée et le dernier contient son extension simplifiée.

Par exemple, prenons `Concept_17` : il représente les entités (classes) décrites par la caractéristique de *att_WaterHeight* et par les caractéristiques héritées de ses super-concepts : *att_MeasuringDate* et *att_CodeQuality* (à partir de `Concept_16`). L'extension de `Concept_17` est l'ensemble contenant la classe *cl_HydraulicHead*. En ce qui concerne le `Concept_16`, son extension est composée de l'ensemble contenant *cl_HydraulicHead* et *cl_Rainfall* (hérités de `Concept_15` et `Concept_17` respectivement) (cf. chapitre 3).

Dans notre démarche, nous nous intéressons aux trois catégories de concepts formels :

Définition 7.6 (Concepts formels fusion (Merged formal concepts)) *Les* concepts formels fusionnés *ont strictement plus d'une entité dans leur extension simplifiée. Remarquons que toutes les entités de l'extension sont décrites par exactement le même ensemble de caractéristiques.*

Dans la figure 7.3, le `Concept_13` est un concept issu d'une fusion : les deux classes de son extension simplifiée, à savoir *cl_PET* et *cl_Temperature* sont exactement décrites par deux caractéristiques *att_MeasuringHour* et *att_value*.

Définition 7.7 (Nouveaux concepts formels (New formal concepts)) *Les* nouveaux concepts formels *ont une extension simplifiée vide. Ceux-ci sont des nouvelles abstractions issues de la factorisation sur des caractéristiques communes à plusieurs concepts formels.*

Dans la figure 7.3, le `Concept_16` est un nouveau concept formel issu de la factorisation des caractéristiques *att_MeasuringDate* et *att_CodeQuality*.

Définition 7.8 (Concepts formels immuables ou pérennes (Perennial formal concepts)) *Les* concepts formels immuables ou pérennes *ont une et une seule entité dans leur extension simplifiée et ils correspondent aux concepts du domaine initiaux.*

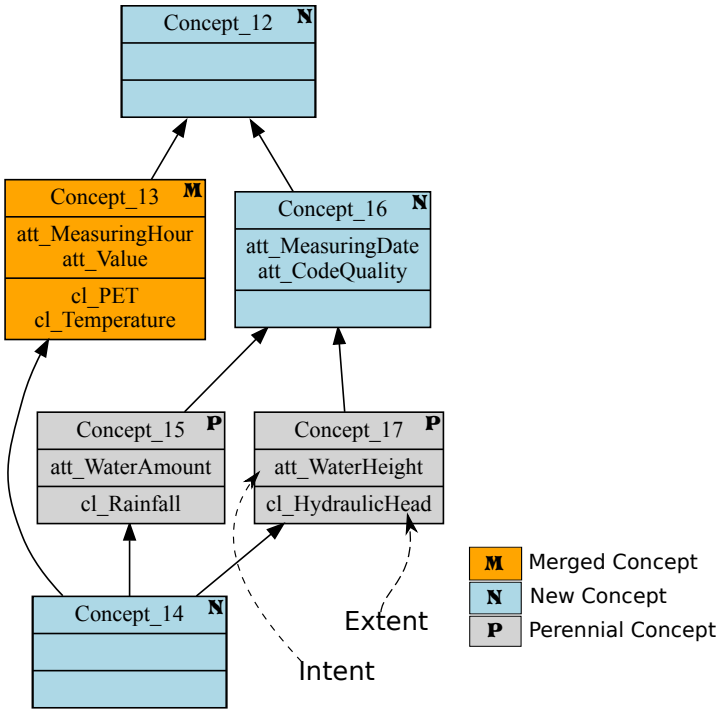


FIGURE 7.3: **Treillis nom de classe/attribut construit avec le contexte formel 7.1**

Dans la figure 7.3, `Concept_15` et `Concept_17` sont immuables (ou pérennes).

Dans la suite de ce chapitre, les concepts formels fusionnés, nouveaux et immuables sont respectivement annotés, sur les figures, **M**, **N** et **P** dans le coin en haut à droite de la boîte les représentant.

7.4 Processus d'extraction du Plus Grand Modèle Commun

Dans cette section, nous proposons une méthodologie basée sur deux étapes semi-automatisées qui utilisent l'Analyse Formelle de Concepts et un processus interactif pour extraire le Plus Grand Modèle Commun des deux modèles d'entrée M_1 et M_2 (cf. figure 7.1).

- La première étape consiste à établir les différents contextes formels en décrivant les classes par les différentes caractéristiques choisies pour la factorisation (noms de classes, attributs, rôles, etc.). Ces contextes formels sont établis sur l'union disjointe des deux modèles d'entrée, à savoir $M = M_1 \oplus M_2$. Ensuite, l'Analyse Formelle de Concepts appliquée à ces contextes formels donne un ensemble de treillis qui est fonction des caractéristiques choisies.
- Les concepts formels de ces treillis sont ensuite analysés grâce à un arbre de décision. Cette analyse s'intéresse principalement à l'extension des concepts.

7.4.1 Application de l'Analyse Formelle de concepts sur les modèles (M_1 et M_2)

Dans notre cas, les contextes formels décrivent les entités selon leurs caractéristiques à travers les relations citées dans les sections 4.3.1.2, 4.3.1.4 et 4.3.1.5 du chapitre 4. Plusieurs contextes formels peuvent être extraits à partir d'un modèle de classes. Ils définissent les éléments de modélisation qui seront les entités étudiées et ceux qui seront leurs caractéristiques. Ici, nous nous concentrons sur les trois contextes formels extraits de l'union disjointe des modèles en entrée $M = M_1 \oplus M_2$:

1. le contexte formel qui décrit les classes par leurs noms,
2. le contexte formel qui décrit les classes par leurs attributs,
3. le contexte formel qui décrit les classes par leurs attributs et leurs rôles.

La figure 7.4 présente le treillis obtenu avec le contexte formel des classes décrites par leurs noms (*classe/nom de classe*). Ce treillis regroupe dans un concept formel l'ensemble des classes qui partagent le même nom. Par exemple, le concept fusionné `Concept_1` a dans son extension l'ensemble des classes partageant le nom *cl_Piezometer* (le nom lui-même étant dans l'intension). Ici, l'Analyse Formelle de concepts fusionne en un seul concept les classes qui ont le même nom. C'est l'un des mécanismes qui nous intéressent pour construire le Plus Grand Modèle Commun. Par contre, les classes qui n'existent que dans un seul des modèles M_1 et M_2 restent dans un concept formel immuable ou pérenne. C'est le cas de la classe *cl_PET* du concept pérenne `Concept_7`. Dans le cadre de la factorisation inter-modèles, les trois catégories de concepts décrits précédemment existent : le concept fusionné `Concept_1` a plus d'une entité dans son extension simplifiée. De la même manière, le concept immuable ou pérenne `Concept_7` (*cl_PET*) a exactement un élément dans son extension simplifiée. Le concept `Concept_0` a une extension vide et correspond à une abstraction de toutes les classes existantes. Nous verrons plus loin des cas où de nouveaux concepts formels plus intéressants apparaissent.

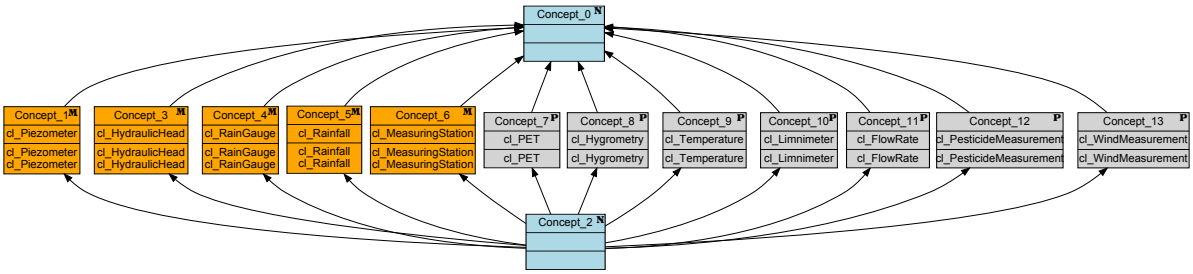


FIGURE 7.4: **Treillis correspondant aux classes décrites par leurs noms**

La figure 7.5 présente le treillis obtenu avec le contexte formel des classes décrites par le nom des attributs qu'elles contiennent. Dans ce treillis, un concept formel factorise dans son extension des classes partageant un groupe d'attributs de noms identiques. Ce treillis contient de nouveaux concepts formels. Ils sont facilement identifiables car leur extension simplifiée est vide. Par exemple, le `Concept_47` est une nouvelle abstraction factorisant l'attribut *att_Date*.

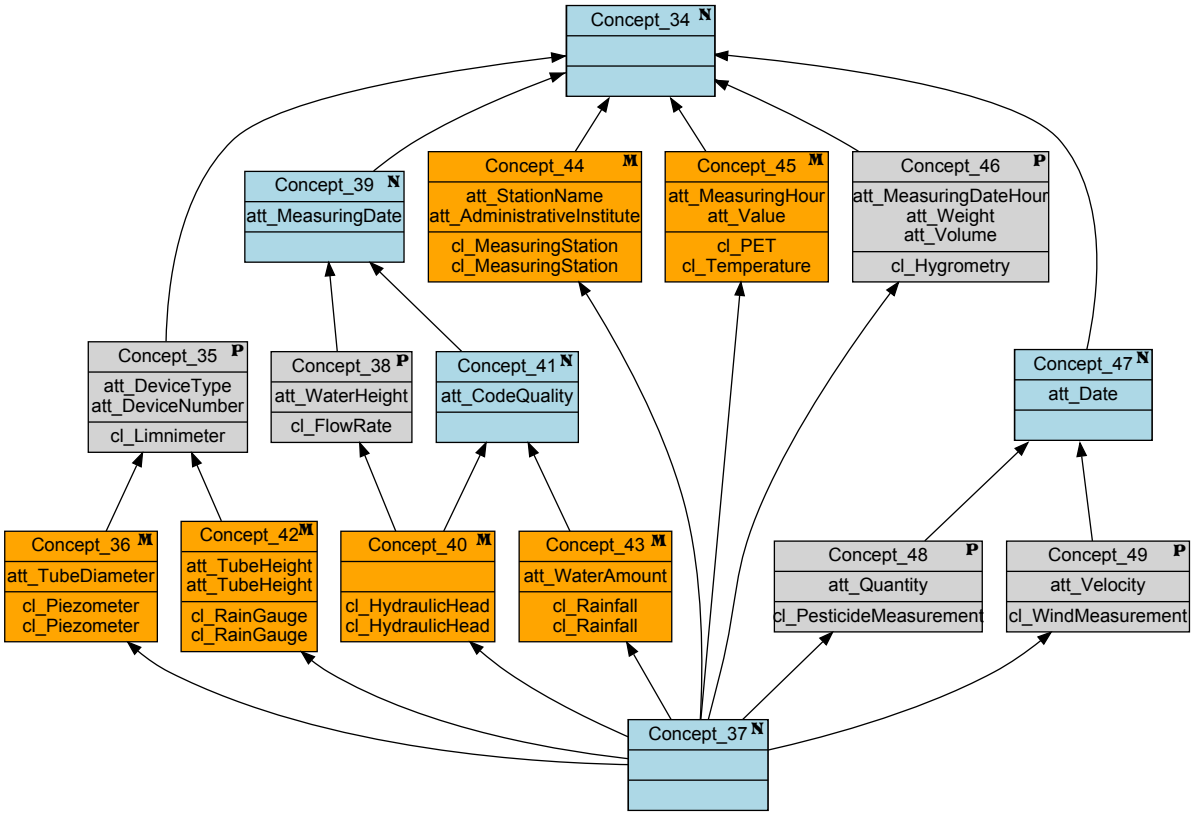


FIGURE 7.5: **Treillis correspondant aux classes décrites par les noms de leurs attributs**

La figure 7.6 présente le treillis obtenu avec le contexte formel des classes décrites par les noms des attributs et des rôles. Dans ce contexte, les rôles permettent de prendre en compte partiellement les associations. Par exemple, la classe *cl_FlowRate* possède un attribut *att_WaterHeight* et le rôle *ro_Station* de l'association *Water Height Information* (cf. figure 7.1) qui permet de gérer l'information sur la hauteur d'eau du cours d'eau. Dans ce treillis, nous constatons l'émergence d'un nouveau concept formel `Concept_30` qui factorise les classes ayant une association avec la classe *cl_MeasuringStation* via le rôle *ro_Station*.

7.4.2 Analyse des treillis

Dans cette section, nous présentons la manière dont l'analyse des treillis a permis de construire l'arbre de décision qui permet de classer chaque concept formel dans des listes à usage du concepteur. Tout d'abord, nous allons analyser le treillis qui résulte de la description des classes par leurs noms. Ce treillis permet au concepteur de modèle de détecter les classes ayant des noms similaires. Ensuite, nous analysons le treillis obtenu à partir du contexte formel décrivant les classes par le nom des attributs. Ce dernier permet de factoriser les classes en fonction des attributs identiques. Et enfin, nous renforcerons les conclusions de ce dernier treillis avec l'ajout de la description des

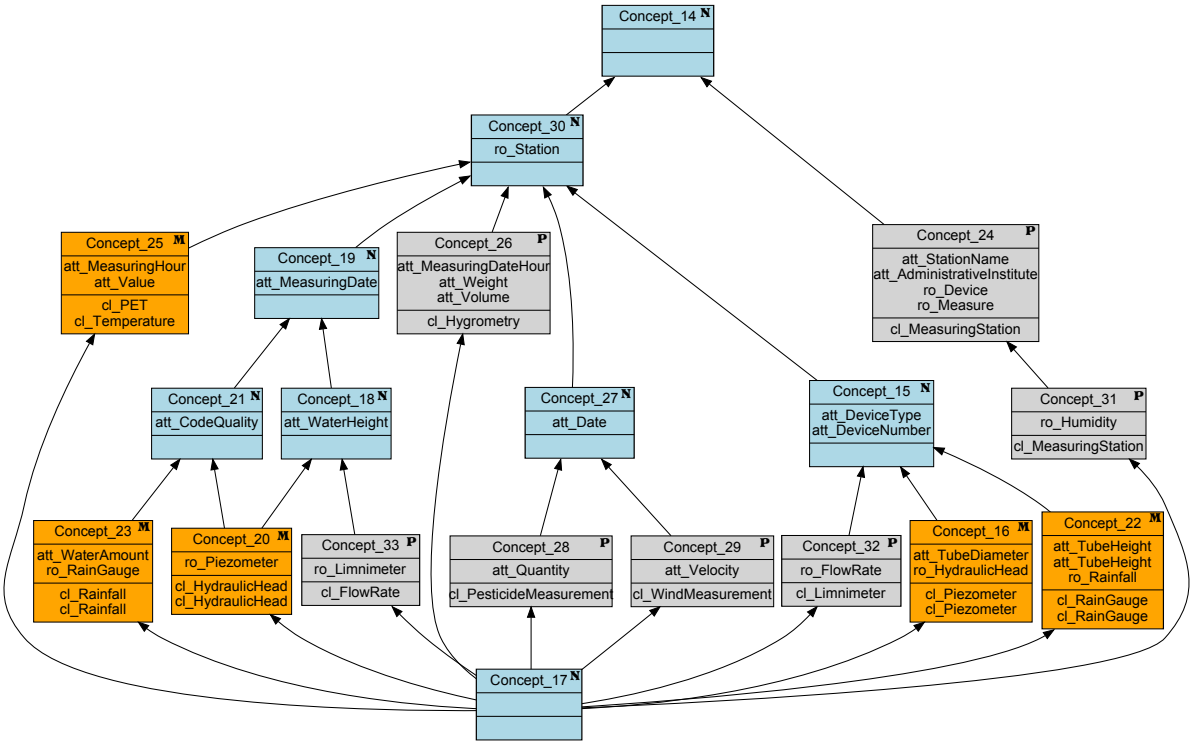


FIGURE 7.6: **Treillis correspondant aux classes décrites par les noms de leurs attributs et de leurs rôles**

classes à la fois par les attributs et les rôles apportant une aide considérable pour affiner les décisions concernant la factorisation.

Chaque concept formel $Concept_k = (Extension_k, intension_k)$, a été classé suivant son type (fusion, nouvelle abstraction ou immuable) et suivant le contenu de son extension *complète* $Extension_k$. Pour ce faire, nous avons listé les différents cas possibles qui constituent un arbre de décision (cf. figure 7.7). Chacun de ces concepts formels s'inscrit dans une des listes ci-dessous :

- L_{GCM} est la liste des concepts communs qui seront inclus dans le Plus Grand Modèle Commun.
- L_{pGCM} est une liste de concepts du domaine candidats au Plus Grand Modèle Commun mais qui doivent être validés par un expert.
- L_{M_1} (resp. L_{M_2}) est la liste des concepts spécifiques au modèle M_1 (resp. M_2).
- L_{nM_1} (resp. L_{nM_2}) est la liste des nouveaux concepts du domaine spécifiques à M_1 (resp. M_2). Ils factorisent les concepts spécifiques existants dans le modèle M_1 (resp. M_2). Comme précédemment, ces concepts du domaine ne sont pas destinés à être dans le Plus Grand Modèle Commun, mais ils peuvent être présentés à des experts pour améliorer la factorisation interne à M_1 (resp. M_2).

$\mathscr{C}_{M_i}$ (resp. $\mathscr{C}_{M_j}$) est l'ensemble des classes du modèle M_i (respectivement. M_j), et dans notre

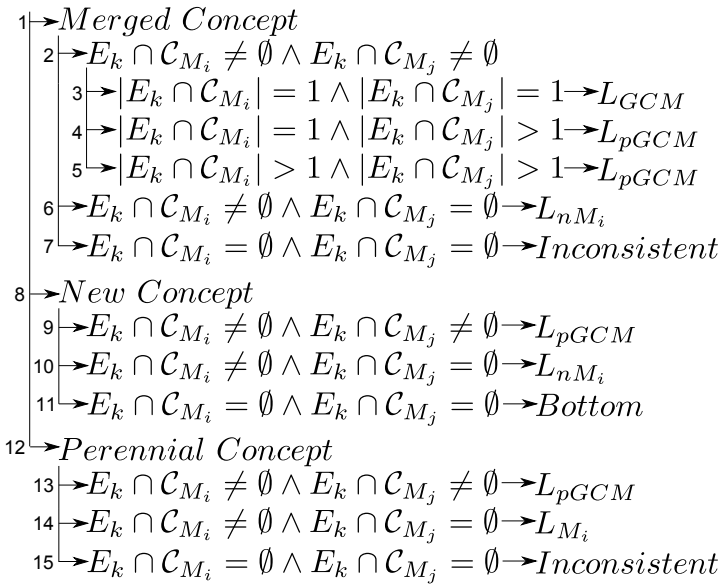


FIGURE 7.7: **Arbre de décision**

exemple l'arbre de décision classe les concepts issus de la factorisation de deux modèles différents, M_i et M_j avec $i \neq j$.

La figure 7.7 présente l'arbre de décision. Nous appliquons l'Analyse Formelle de Concepts avec des classes en tant qu'entités qui sont décrites par plusieurs caractéristiques comme le nom de la classe, les attributs et/ou les rôles. Ainsi, l'extension d'un concept formel contient uniquement des classes. Pour chaque concept, nous vérifions d'abord si le concept est une *fusion*, une *nouvelle abstraction (nouveau concept)* ou un *immuable* (nœuds 1, 8 et 12 respectivement de l'arbre de décision de la figure 7.7).

7.4.2.1 Analyse des concepts fusion

Dans le cas où le concept est issu d'une fusion, trois cas de figure peuvent se présenter : (i) son extension contient des éléments des deux modèles M_i et M_j (cf. nœud 2 figure 7.7), (ii) son extension contient uniquement les éléments de M_i (par exemple le nœud 6) et (iii) son extension est vide (par exemple le nœud 7).

Le premier cas de figure correspond au cas où l'extension du concept contient des éléments des deux modèles. Le cardinal de l'intersection entre l'extension et l'ensemble des classes du modèle doit alors être étudié.

Dans le premier sous-cas, l'extension ne contient qu'une seule classe de M_i et qu'une seule classe de M_j (nœud 3). Par exemple, le `Concept_1` est issu de la description des classes par leur nom *classe/noms de classe* (cf. figure 7.4). Le concept de domaine correspondant doit être ajouté dans L_{GCM} car il est commun aux deux modèles.

Ensuite, le deuxième sous-cas de figure correspond à une extension qui ne contient qu'une seule classe de M_i et plusieurs classes de M_j (nœud 4), ou plusieurs éléments des deux modèles

(nœud 5). Alors, le concept de domaine peut être considéré comme un concept potentiel nécessitant une expertise ultérieure. Il doit être mis dans la liste L_{pGCM}. L'expert peut alors choisir de fusionner ou de factoriser les classes en doublon si elles se correspondent sémantiquement, au sein d'un même modèle (factorisation intra-modèle), et relancer le processus pour extraire le Plus Grand Modèle Commun. L'expert peut aussi considérer que ces classes sont des concepts spécifiques à un domaine ce qui le conduit à les conserver dans les modèles spécifiques correspondants.

Enfin, le troisième sous-cas concerne les concepts fusion contenant uniquement des classes de M_i (nœud 6), comme le `concept_45` (cf. figure 7.5). Ils sont ajoutés à liste L_{nM_i}. La figure 7.8 présente un extrait du treillis obtenu à partir de contextes formels qui décrivent les classes par le nom des attributs. Pour illustrer nos propos, nous avons ajouté les classes correspondant au concept formel `concept_45` et mis en évidence les attributs factorisés.

L'extension du concept `concept_45` contient un groupe de classes provenant d'un même modèle. Elles sont décrites exactement par les mêmes caractéristiques. Par exemple, pour le `Concept_45`, l'AFC suggère de fusionner les classes *cl_PET* (représentant l'évapo-transpiration potentiel) et *cl_Temperature*. Le concept `concept_45` peut être présenté à un expert pour améliorer le modèle M_i, mais ce n'est pas un concept commun car il factorise des attributs de classes d'un même modèle. Comme dans ce cas particulier, ces deux classes sont sémantiquement différentes, l'expert ne veut pas les factoriser. Toutefois, dans d'autres situations, il pourrait envisager cette factorisation qui peut être intéressante.

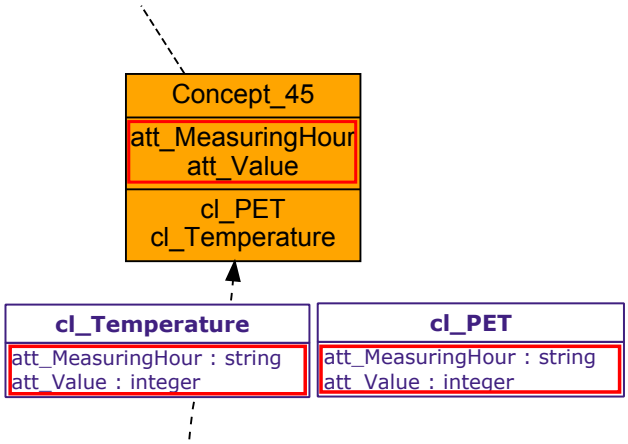


FIGURE 7.8: **Extrait du treillis de concept classe/attribut et les classes correspondantes**

Le nœud 7 décrit les concepts ayant une extension vide de M_i et/ou M_j. Cela est incohérent : par définition, une extension de concept fusion contient au moins deux éléments (cf. définition 7.6).

7.4.2.2 Analyse des nouveaux concepts : nouvelles abstractions créées

Dans cette partie, nous analysons les nouvelles abstractions créées, qui correspondent à un nouveau concept d'après la définition 7.7), et au nœud 8. Dans le cas où l'extension contient des éléments des deux modèles M_i et M_j (nœud 9), le nouveau concept doit être mis dans la liste L_{pGCM}

car il s'agit d'une factorisation de concepts définis dans les modèles M_j et M_i. C'est potentiellement un concept commun et les experts auront à décider si cette factorisation est sémantiquement cohérente et souhaitable. Si oui, ce nouveau concept devra être inclus dans le Plus Grand Modèle Commun. Dans la figure 7.5, le concept `Concept_39` est un exemple de ce type de concept. Dans notre cas d'étude, l'expert valide ce concept comme étant un concept commun appartenant au Plus Grand Modèle Commun.

Ensuite, dans le cas de figure où l'extension ne contient que des classes d'un seul modèle, il peut être ajouté dans la liste L_{nM_i} (nœud 10). Ce concept correspond à une factorisation intra-modèle. Un exemple de la factorisation intra-modèle est présenté dans la figure 7.9. Le concept `Concept_47` factorise l'attribut `att_Date` des deux classes `cl_PesticideMeasurement` et `cl_WindMeasurement`, classes toutes deux appartenant au modèle M_2. Ce type de concept n'est pas un concept commun et ne doit pas être inclus dans le Plus Grand Modèle Commun. Il peut être présenté au concepteur du modèle M_i afin d'améliorer la qualité de son modèle avant une nouvelle factorisation.

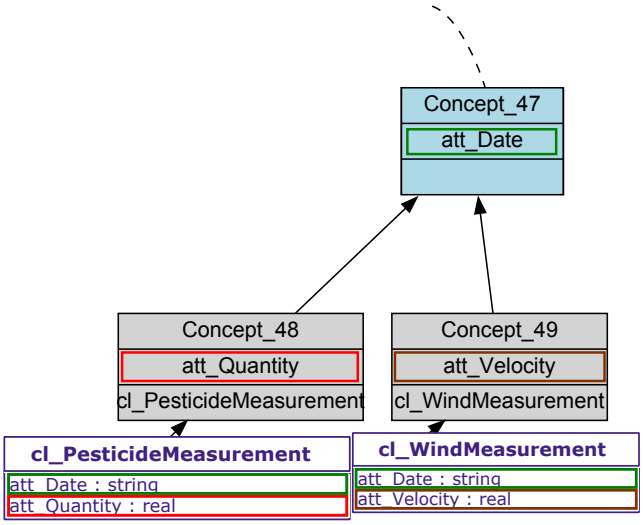


FIGURE 7.9: **Extrait du treillis de concept classe/attribut et les classes correspondantes**

Enfin, dans le cas où l'extension du nouveau concept ne contient pas de classes de M_1 ni de M_2 (nœud 11), cela signifie qu'il s'agit d'un concept Bottom. Les concepts `Concept_2`, `Concept_37` et `Concept_17`, situés au niveau le plus bas des différents treillis, appartiennent à cette catégorie. Ils représentent des éléments qui possèdent tous les attributs. Par contre, le concept racine ne peut pas être déduit seulement par une analyse étendue et il peut apparaître dans chacune des branches de l'arbre. Selon les modèles analysés, le concept Top (racine) peut être pertinent et il doit être classé comme les autres concepts.

7.4.2.3 Analyse des concepts immuables ou pérennes

Le nœud 13 de l'arbre de décision traite les concepts pérennes qui ont dans leur extension complète des classes de chacun des modèles M_i et M_j. Le `Concept_35` (cf. figure 7.5) en est un exemple. Factorisant `Concept_36` et `Concept_42`, ce dernier est un candidat potentiel au Plus Grand Modèle Commun (liste L_{pGCM}.). De ce fait, ce concept doit être présenté à l'expert pour validation. Dans notre exemple, le concepteur peut décider d'adopter *cl_Limnimeter* comme super-classe de *cl_Piezometer* et *cl_RainGauge*. Comme un piézomètre *n'est pas un limnimètre* particulier mais un instrument de mesure au même titre que limnimètre, la hiérarchisation sémantique proposée par l'Analyse Formelle de Concepts n'est pas cohérente. L'expert du domaine a refusé d'inclure *cl_Limnimeter* dans le Plus Grand Modèle Commun. L'analyse du treillis issu du contexte formel décrivant les classes par les noms de leurs attributs et de rôles (cf. figure 7.6) montre qu'il est préférable de créer une nouvelle super-classe *cl_Device* (`Concept_15`) pour factoriser les trois classes *cl_Limnimeter*, *cl_Piezometer* et *cl_RainGauge*. Dans ce cas, ce treillis est utile pour aider le concepteur à prendre une décision.

Ensuite, dans le cas où l'extension du concept pérenne ne contient que des classes de M_i (nœud 14) alors il s'agit d'un concept spécifique au domaine M_i. Ce concept doit être ajouté à la liste L_{M_i}. Par exemple, les concepts `Concept_7`, `Concept_8`, `Concept_48` et `Concept_49` sont des concepts propres aux domaines pour M_i. Les concepts `Concept_48` et `Concept_49` sont présentés dans la figure 7.9 avec leurs classes correspondantes *PesticideMeasurement* et *WindMeasurement*. Ces classes ne sont pas factorisées et elles restent des concepts spécifiques du modèle M_2.

Enfin, un concept pérenne ne peut pas avoir une extension vide (nœud 15) : la définition 7.8 spécifie qu'un concept pérenne possède un (et un seul) élément dans son extension.

Les listes L_{GCM} et L_{pGCM} doivent ensuite être analysées par l'expert qui doit sélectionner les concepts communs qui seront inclus dans le Plus Grand Modèle Commun (GCM).

7.5 Démarche suivie et mise en œuvre

Notre démarche s'articule autour de quatre grandes transformations schématisées en figure 7.10. Dans un premier temps, une première transformation Tr. 1 convertit les modèles UML en contextes formels, un par caractéristique étudiée. Ils constituent l'entrée de l'Analyse Formelle de Concepts. Ensuite, la seconde (Tr. FCA) applique l'Analyse Formelle de Concepts (FCA en anglais) aux contextes formels et produit un treillis par caractéristique. La transformation suivante (Tr. 3) génère les différentes listes de concepts du domaine conformément à l'arbre de décision. Enfin, la dernière transformation (Tr. 4), appelée *Tr. GCM*, effectue la construction finale du Plus Grand Modèle Commun. Cette dernière utilisera la classification des concepts dans les listes de l'arbre de décision pour proposer un Plus Grand Modèle Commun au concepteur. Ce dernier aura à valider les concepts proposés. Pour les modèles *MeasuringStation* (cf. figure 7.1), le Plus Grand Modèle Commun est réalisé manuellement en prenant en compte la classification issue de l'arbre de décision. Cela est possible car ces modèles sont de petite taille. Pour des modèles de taille plus importante, par exemple le modèle *SIE-Pesticide* (plus de 170 classes), l'extraction manuelle du Plus Grand Modèle Commun nécessitera des outils d'assistance plus avancés.

Mis à part la transformation Tr. 4, les autres sont implémentées sous forme de profil dans

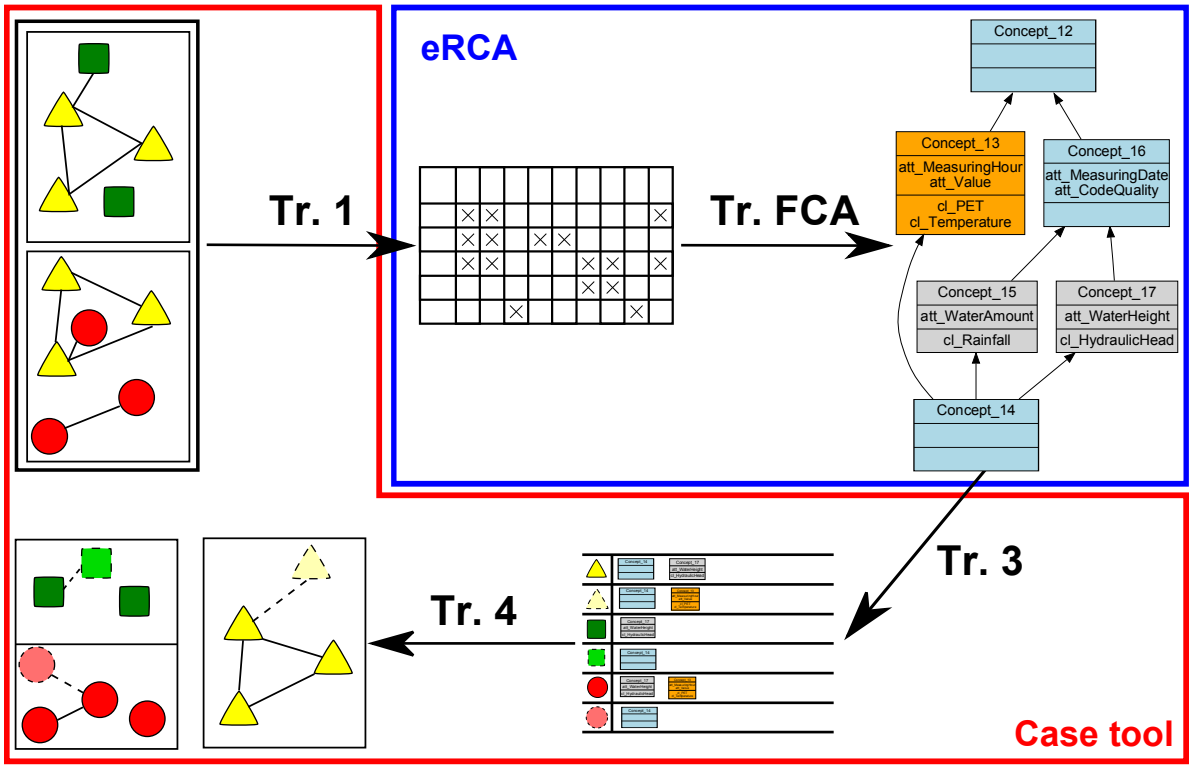


FIGURE 7.10: **Mise en œuvre détaillée de notre approche**

l'Atelier de Génie Logiciel Objecteering 4.4.1 [2]. Dans ce profil, nous avons utilisé le framework eRCA [3] 4.4.2 (cf. figure 7.10 présentant l'association de ces deux composants).

L'Atelier de Génie Logiciel (AGL) Objecteering (cf. 4.4.1) est un modeleur UML commercialisé par Objecteering Software pour faciliter la modélisation avec ce langage. Ses principales fonctionnalités sont : la création et l'édition de modèles UML, la génération de code et la rétro-ingénierie de projets existants. S'inscrivant dans les méthodes d'ingénierie dirigée par les modèles, cet atelier propose des outils pour créer des profils UML, étendant son métamodèle soit avec de nouveaux concepts métiers, soit avec de nouvelles fonctionnalités. Ces profils sont encapsulés dans Component Model Driven Approach (MDAC), sorte de pluging propre à Objecteering. Ces MDAC manipulent les modèles grâce à une API qui réifie le métamodèle UML. Il existe aussi des outils pour créer des interfaces d'utilisation ergonomiques.

Le framework eRCA implémente les algorithmes permettant l'AFCA et l'ARC. Conçu à l'origine comme un pluging d'Eclipse, il est également disponible en version autonome grâce à fichier *jar* de façon à être utilisé dans les projets Java classiques. C'est sous cette forme qu'il est couplé à Objecteering.

Dans la pratique, les modèles *MeasuringStation* et *SIE-Pesticides* ont été réalisés dans Objec-

2. http://www.objecteering.com/
3. http://code.google.com/p/erca/

teering. Grâce au profil, il est possible de parcourir ces modèles et générer automatiquement les contextes formels. La transformation tr.FCA appliquée sur les contextes formels produit les treillis associés. Le framework eRCA inclut une API pour manipuler les treillis générés qui sont exploités par le MDAC pour classer les concepts formels dans les différentes listes conformément à l'arbre de décision.

7.6 Analyse des résultats obtenus

La figure 7.11 montre le Plus Grand Modèle Commun des modèles M1 et M2 (cf. figure 7.1). Il prend aussi en compte l'interprétation des nouveaux concepts effectuée par un expert du domaine. Ce modèle est donc validé. Nous avons annoté les classes modifiées par les concepts formels des treillis dont elles sont issues (cf. figures 7.4, 7.5 et 7.6).

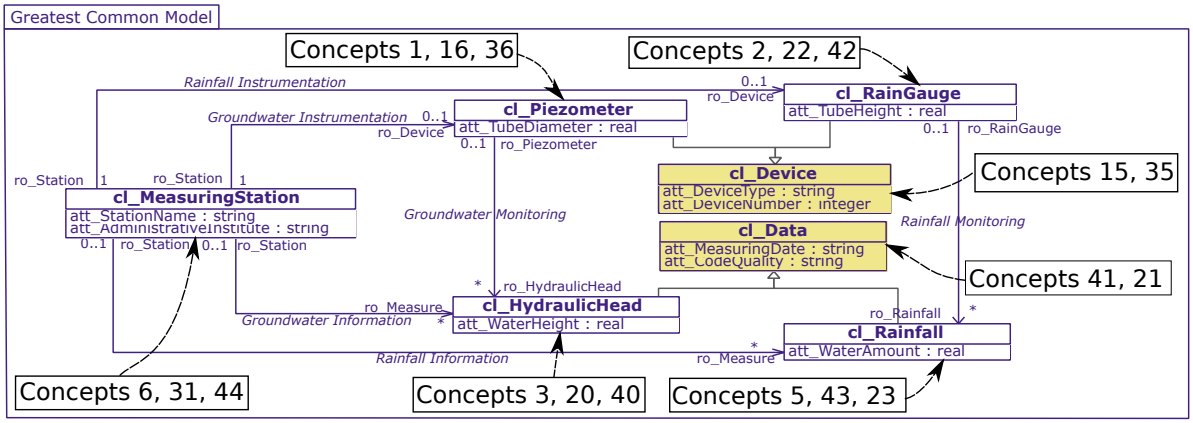


FIGURE 7.11: **Plus Grand Modèle Commun des modèles M1 et M2 (cf. Figure 7.1)**

Comme prévu, les concepts identiques de M1 et M2 sont présents dans le Plus Grand Modèle Commun *GCM* : *cl_MeasuringStation*,*cl_Piezometer*, *cl_HydraulicHead*, *cl_RainGauge* et *cl_Rainfall*. Ils constituent les concepts communs des modèles $M1$ et $M2$. Ils représentent les cinq concepts ajoutés automatiquement dans la liste L_{GCM}.

Notre approche propose une liste de factorisation de concepts potentiels du domaine dans la liste L_{pGCM}. L'expert a dû valider la pertinence de ces concepts (cf. Tableau de la classification de concepts formels 7.3).

Dans cet exemple, sur les cinq concepts potentiellement factorisables dans le Plus Grand Modèle Commun seulement deux ont été considérés comme pertinents par l'expert. Ils sont colorés dans la figure 7.11.

Le premier correspond aux concepts formels `Concept_15` (cf. figure 7.6) et `Concept_35` (cf. figure 7.5) des treillis. Ils factorisent les attributs *att_DeviceType* et *att_DeviceNumber*. Ce concept a été validé par les experts comme étant la nouvelle classe *cl_device*.

Le deuxième provient des concepts formels `Concept_41` et `Concept_21` des treillis. Il factorise

	L_{GCM}	L_{pGCM}	L_{nM1}	L_{nM2}	L_{M1}	L_{M2}
classe/nom classe	5	1	0	0	3	4
classe/nom attribut	5	5	1	1	1	2
classe/nom attribut-rôle	4	7	1	1	2	4

TABLE 7.2: **Nombre de concepts classés dans chacune des listes pour les modèles *MeasuringStation***

les deux attributs *att_MeasuringDate* et *att_CodeQuality*. De même que pour le premier, l'expert a validé ce concept comme une nouvelle classe *cl_Data*.

Le tableau 7.2 donne le nombre de concepts formels de chacune des principales listes de l'arbre de décision.

Afin de valider l'évolution de notre approche, des tests ont été effectués sur deux versions du modèle complet du projet *SIE-Pesticides* (environ 125 classes [4]). Le tableau 7.4 donne le nombre de concepts par liste de l'arbre de décision.

Suivant le type de description des classes par leurs caractéristiques (noms et noms d'attributs), les experts ont à analyser et à valider entre 34 et 39 concepts présents dans la liste des concepts potentiels L_{pGCM}. Ils peuvent obtenir plus de précision avec le treillis résultant de la description des classes par les attributs et les rôles, où 119 concepts potentiels sont proposés pour être candidats au Plus Grand Modèle Commun. Nous travaillons aussi pour aider l'expert dans cette tâche d'analyse[Osman Guédi *et al.*, 2011]. Nous pouvons également déduire de ce tableau que les deux versions du modèle *SIE-Pesticides* sont très proches. En effet, les listes de concepts spécifiques L_{M1} et L_{M2} ne contiennent qu'un seul concept formel pour la relation décrivant les classes par leur nom. Lorsque les classes sont décrites par les attributs et les rôles, ces deux listes contiennent 8 et 9 concepts formels respectivement.

7.7 Travaux connexes

Ces travaux ont des points d'accroches avec plusieurs autres types de travaux de la littérature. Nous les mentionnons dans cette partie.

Certains travaux s'intéressent à la comparaison de modèles. Une première approche consiste à représenter les modèles de classes sous forme de graphes [Xing et Stroulia, 2005] pour calculer la similarité des noms des éléments et par la suite déterminer des distances de similarité structurelle. Cette approche se concentre sur le contexte de chaque élément de modélisation, c'est-à-dire son voisinage. Par exemple, elle s'intéresse aux éléments appartenant au même paquetage. Dans son aspect pratique, elle offre à l'utilisateur plusieurs fonctionnalités de déplacement et de modification du nom des éléments de modélisation tout en opérant l'évolution de leur changement. D'autres méthodes s'intéressent à extraire les différences entre deux modèles de classes. C'est le cas de l'algorithme **Generic Difference Algorithm for UML** [Kelter *et al.*, 2005]. Comme pour la méthode précédente, ce dernier est fondé sur la théorie des graphes. Il transforme dans un premier temps les modèles en graphes pour ensuite extraire les mesures qui définissent des différences. Ces

4. Selon les versions du modèle *SIE-Pesticides.*, le nombre de classes varie de 36 à 171

Listes de l'arbre de décision	classe/nom classe	classe/nom attribut	classe/nom attribut-rôle
L_{GCM}	– Concept_1 – Concept_3 – Concept_5 – Concept_6	– Concept_36 – Concept_40 – Concept_42 – Concept_43 – Concept_44	– Concept_16 – Concept_20 – Concept_22 – Concept_23
L_{pGCM}	– Concept_0	– Concept_34 – Concept_35 – Concept_38 – Concept_39 – Concept_41	– Concept_14 – Concept_15 – Concept_18 – Concept_19 – Concept_21 – Concept_24 – Concept_30
L_{nM1}	– Vide	– Concept_45	– Concept_25
L_{nM2}	– Vide	– Concept_47	– Concept_27
L_{M1}	– Concept_7 – Concept_8 – Concept_9	– Concept_46	– Concept_26 – Concept_31
L_{M2}	– Concept_10 – Concept_11 – Concept_12 – Concept_13	– Concept_48 – Concept_49	– Concept_28 – Concept_29 – Concept_32 – Concept_33

TABLE 7.3: **Classification des concepts des treillis(cf. figure 7.4, 7.5 et 7.6)**

	L_{GCM}	L_{pGCM}	L_{nM1}	L_{nM2}	L_{M1}	L_{M2}
classe/nom classe	111	34	0	0	1	1
classe/nom attribut	43	39	0	0	1	2
classe/nom attribut-rôle	68	119	0	0	8	9

TABLE 7.4: **Nombre de concepts classés dans chacune des listes pour deux versions du modèle *SIE-pesticides***

graphes sont représentés sous forme d'arbre contenant les relations entre les éléments de modélisation et leur parcours en profondeur détermine la similarité sous forme d'une table d'éléments similaires. Les tables issues des modèles sont utilisées pour déterminer les différences entre ces modèles. La gestion des similitudes et des différences entre les modèles a également été étudiée dans la gestion de versionning des modèles [Altmanninger *et al.*, 2009]. *Smover Tool*[5] est utilisé comme outil de comparaison directe entre un modèle et sa version précédente pour détecter les conflits syntaxiques et sémantiques [Altmanninger *et al.*, 2010]. Afin de gérer les conflits de modèles dans un contexte de développement distribué, le travail présenté dans [Cicchetti *et al.*, 2008] propose l'utilisation d'un modèle (*de différence*) pour stocker les différences entre deux versions d'un même modèle [Cicchetti *et al.*, 2007]. Cette méthode permet de montrer les différences entre les modèles, mais ils n'ont pas pour but de proposer ni de détecter automatiquement le concept concerné. Dans l'approche décrite dans [Ohst *et al.*, 2003] , les modèles et les diagrammes sont considérés comme des arbres de syntaxe, ce qui permet aux auteurs de concevoir une opération de différence entre les modèles. En se basant sur le domaine de la gestion des versions du modèle, nous visons à présenter le Plus Grand Commum de Modèle dans une forme normale (factorisée). C'est pourquoi nous pensons que l'Analyse Formelle de Concepts est mieux adaptée à notre problème.

Depuis le début des années 80, le domaine des bases de données s'est penché sur l'étude des problèmes liés à l'intégration de schéma et au couplage des données. L'objectif du cadre d'intégration de base de données est de produire le schéma global d'une collection de bases de données [Batini *et al.*, 1986; Rahm et Bernstein, 2001; Shvaiko et Euzenat, 2005]. En règle générale, l'intégration est composée de différentes étapes : 1) transformation de schéma, 2) étude de correspondance et enfin, intégration de schéma proprement dite. Notre travail se concentre sur l'étude de la correspondance et de l'intégration du schéma [Parent et Spaccapietra, 1998].

L'approche de [Bouzeghoub et Comyn-Wattiau, 1990] propose une méthode d'intégration de vue originale, basée sur le principe de l'unification des structures par rapport à l'unification logique utilisée en Prolog. Le modèle de données est un modèle entité-relation dont la représentation interne est un réseau sémantique. La notion de substitution est redéfinie et un raisonnement par analogie basé sur les vecteurs de similarité est proposé. La comparaison d'objets est suivie par des transformations de structures qui conduisent à une forme canonique de vue. Cet algorithme est inséré dans une méthodologie de conception de base de données incrémentielle. Il pourrait probablement être utilisé comme un outil de base dans la conception de base de données avancée qui intègre notamment des objets structurés ou complexes.

Enfin l'approche décrite par [Métais *et al.*, 1997] traite du problème de l'intégration dans un environnement CASE qui vise à l'élaboration d'un schéma conceptuel d'une application. Les outils d'intégration précédents étaient principalement basés sur la syntaxe et les comparaisons de la structure. Une nouvelle génération d'outils intelligents découle désormais, en supposant que les vues d'intégration doivent également capturer la sémantique profonde des objets représentés dans les vues. Traiter avec la sémantique des objets est désormais un objectif réaliste grâce aux résultats de recherche obtenus dans le domaine du langage naturel. Cette approche présente une définition d'une vue algorithmique d'intégration renforcée par l'utilisation de connaissances linguistiques. Cet algorithme se compose principalement d'une unification sémantique de vues qui sont décrites en utilisant un modèle entité-relation. Il est combiné avec des techniques de langage

5. fromhttp://smover.tk.uni-linz.ac.at

naturel comme les sémantiques de Fillmore et les graphes conceptuels de Sowa, soutenus par les dictionnaires sémantiques.

Le schéma intégré comprend le plus grand modèle commun tel que nous l'avons défini et la partie spécifique des schémas initiaux. Deux groupes de solutions principaux sont distingués dans la littérature pour trouver de manière semi-automatiquement : d'une part, une solution basée sur des règles et, d'autre part, une solution basée sur l'apprentissage. Notre approche est similaire à la solution basée sur les règles. En effet, nous recherchons les similitudes entre plusieurs éléments du modèle en fonction de leurs caractéristiques [Doan et Halevy, 2005]. De manière complémentaire à ces approches, l'utilisation du AFC permet de choisir avec finesse la façon de décrire les caractéristiques que nous considérons. Ce domaine rejoint actuellement le domaine de l'alignement et de la fusion d'ontologies où l'on trouve également des approches utilisant des treillis de concepts [Stumme et Maedche, 2001].

7.8 Conclusion

Lors de la modélisation d'un domaine, plusieurs équipes ayant des compétences différentes interviennent pour modéliser leur domaine respectif. Chaque équipe participe essentiellement à construire la partie du modèle qui relève de sa spécialité tout en donnant son point de vue sur les relations de son domaine avec les autres domaines de spécialité. Le modèle global du système est ainsi consolidé par l'intervention de l'ensemble des équipes. Lorsque le nombre de participants aux séances d'analyse est trop élevé, il est souvent préférable de travailler par spécialité. Dans ce type de démarche, souvent des concepts métiers sont dupliqués. Il est alors nécessaire d'identifier les concepts communs.

La principale contribution dans ce chapitre est une approche [Amar *et al.*, 2012] pour assister le concepteur dans la tâche de fusionner plusieurs diagrammes de classes décrivant le domaine étudié. La méthodologie proposée est basée sur l'Analyse Formelle de Concepts couplée à un arbre de décision. Cette contribution permet la production du Plus Grand Modèle Commun qui peut être assimilé à une forme normale (factorisée) de l'ensemble des modèles sources. L'approche proposée distingue deux principaux types de concepts candidats au Plus Grand Modèle Commun : les concepts du domaine qui sont sans ambiguïté dans le Plus Grand Modèle Commun et ceux qui potentiellement peuvent y appartenir mais qui nécessitent une validation préalable par des experts du domaine. En outre, cette approche identifie de nouveaux concepts spécifiques à chacun des modèles qui factorisent des concepts propres à chacun des modèles. L'approche a été validée sur deux versions du modèle *SIE-Pesticides* contenant environ 125 classes. Les résultats de notre approche ont été analysés, validés et utilisés par A. Miralles, encadrant de la thèse, qui a une double compétence : en informatique et dans les techniques d'application de pulvérisation de pesticides [Miralles *et al.*, 1994; Miralles et Polvêche, 1998; Miralles *et al.*, 2011]. La contribution présentée est une étape importante vers la capitalisation des connaissances, que nous avons appliquée ici au domaine agricole (projet *SIE-Pesticides*).

L'une des principales perspectives de notre travail est d'améliorer la détermination du Plus Grand Modèle Commun grâce à l'utilisation de l'Analyse Relationnelle de Concepts (cf. chapitre 3), qui est une extension de l'Analyse Formelle de Concepts. Elle nous permettra d'expliciter de façon plus précise les relations (associations en UML) entre les concepts du domaine. Dans notre

exemple de la figure 7.1, l'utilisation de l'Analyse Relationnelle de Concepts permettrait de factoriser les associations *Rainfall Instrumentation* et *Groundwater Instrumentation* en proposant une nouvelle association reliant les classes *cl_device* avec *cl_MeasuringStation*. De même, l'Analyse Relationnelle de Concepts va extraire une nouvelle association entre la nouvelle classe *cl_Data* et *cl_MeasuringStation*, factorisant deux associations *RainFall Information* et *GroundWater Information*.

Une autre perspective importante est l'utilisation de techniques de traitement automatique du langage naturel pour affiner la description des caractéristiques basée simplement sur le nom d'éléments (classes, attributs, rôles, etc.). La connaissance de relations sémantiques telles que l'hyperonymie, la synonymie ou l'homonymie entre termes va enrichir l'analyse des concepts du domaine. Dans cette perspective, l'utilisation de thésaurus spécialisés du domaine serait opportune. Ces thésaurus définissent des relations sémantiques d'intérêt général dans des bases de données lexicales telles que wordnet[6]. Nous envisageons aussi la possibilité de construire un réseau spécifique au domaine lexical grâce à une approche basée sur les techniques de jeu[7].

6. http://wordnet.princeton.edu/
7. http://www2.lirmm.fr/jeuxdemots-english/jdm-accueil.php

Quatrième partie

Conclusion et Perspectives

CHAPITRE

8 Conclusion générale

Comme nous avons évoqué dans l'introduction, la réalisation d'une application ou d'un système d'information complexe comme ceux auxquels nous sommes confrontés dans le domaine universitaire et/ou de l'environnement nécessite de mobiliser différents types d'acteurs issus de domaines d'expertise bien distincts.

De ce fait, l'analyse du système d'information et le développement de l'application avec l'ensemble des acteurs s'avèrent rapidement difficiles et même dans certains cas conflictuels. Ainsi, la qualité du résultat final peut différer des attentes de l'utilisateur final.

Au cours de l'évolution de ces systèmes d'information dus entre autres à l'innovation technique apportée par la nouvelle technologique, la concurrence ou la réglementation au cours de leur cycle de vie. La préservation d'une conception correcte doit être une quête permanente. Cependant les systèmes impliquant un grand nombre de classes sont soumis à de fréquentes modifications et ils deviennent de plus en plus complexes tout en prenant en compte des informations de plus en plus volumineuses. L'utilisation de la détection automatique et des outils de correction peut être utile pour cette tâche.

Face à ce type de situation, d'une part, nous nous inspirons d'un corpus de concepts, de méthodes et d'outils, mais aussi de langages pour créer, organiser, restructurer et transformer des modèles issus de l'Ingénierie Dirigée par les Modèles (IDM) [Kent, 2002] qui prône l'utilisation des modèles pour guider le développement. De ce fait, le modèle constitue une référence parmi les outillages disponibles pour les utilisateurs pour développer des applications. Ces modèles représentent une vision naturelle du contexte de l'utilisateur.

Et d'autre part, l'Analyse Formelle de Concepts [Ganter et Wille, 1999] et l'Analyse Relationnelle de Concepts sont des méthodes d'analyse de données basées sur la théorie des treillis [Birkhoff, 1940] et des treillis de Galois [Barbut et Monjardet, 1970]. Les deux méthodes extraient des concepts formels qui associent un ensemble maximal d'entités à un ensemble maximal de caractéristiques partagées. Ces concepts formels sont structurés dans un ordre partiel de spécialisation qui les munit d'une structure de treillis. L'Analyse Relationnelle de Concepts permet de compléter la description des entités par des relations entre entités.

Un processus interactif qui est implémenté dans un Atelier de Génie Logiciel Objecteering (un

AGL 4.4.1) est conçu pour répondre à la problématique de la thèse. En effet, ce processus utilise le framework eRCA 4.4.2 permettant la construction et la manipulation de treillis de concepts généré par l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts.

Dans ce manuscrit de thèse, nous avons présenté trois contributions pour analyser l'évolution de modèle tout en étudiant les comportements des approches mises en œuvre tel que l'Analyse Relationnelle de Concepts afin d'extraire le modèle commun.

Analyse de l'évolution des modèles

Tout d'abord, la méthode d'analyse de l'évolution des modèles est basée sur des métriques portant d'une part, sur les éléments de modélisation du modèle de classe UML et, d'autre part, sur les treillis générés à partir de l'Analyse Formelle de Concepts et de l'Analyse Relationnelle de Concepts. L'application systématique aux quinze versions du modèle *SIE-Pesticides*, nous a permis de proposer des métriques et plusieurs recommandations qui constituent le début d'une méthode d'analyse.

Cette méthode s'appuie sur la capacité de l'Analyse Formelle de Concepts et de l'Analyse Relationnelle de Concepts à faire émerger au sein d'un modèle des abstractions thématiques de niveau supérieur, améliorant ainsi la sémantique des modèles. Nous montrons que ces méthodes peuvent aussi être employées pour suivre l'évolution du processus d'analyse avec des acteurs. Une approche nouvelle et originale de normalisation est introduite, elle permet à l'acteur d'analyser le suivi de l'évolution du modèle au cours de son cycle de vie. L'étude de l'évolution du processus d'analyse d'un modèle par application systématique des métriques aux quinze versions du modèle *SIE-Pesticides* a permis de proposer les premières recommandations qui constituent également le début d'une méthode, utilisant cette fois l'ARC.

Étude comportementale de l'Analyse Relationnelle de Concepts

Ensuite, pour assister le concepteur dans le contrôle de la complexité des résultats, une étude approfondie du comportement de l'Analyse Relationnelle de Concepts est expérimentée sur les quinze versions de modèles du projet SIE-Pesticides. Dans ce comportement, le gain en nombre d'étapes, en nombre de concepts créés, en temps de calcul et en espace mémoire occupé est mesuré à l'aide de plusieurs métriques. Les premières recommandations que nous en tirons concernent les paramètres de la factorisation, en d'autres termes, l'usage de la navigabilité, qui est recommandé, et l'usage des rôles *undefineds*, qui n'est pas recommandé.

Nous avons également étudié la structure des graphes sous-jacents aux modèles, et des indicateurs sur ces graphes, tels que les nombres de sommets et d'arcs, la densité, la longueur d'un plus long chemin simple, le degré ou encore le nombre de circuits. Ainsi, un ensemble de métriques et de recommandations utiles est défini pour comprendre et définir un cadre d'utilisation de l'Analyse Relationnelle de Concepts et maîtriser les résultats d'une factorisation avec l'Analyse Relationnelle de Concepts.

Factorisation inter-modèles

Les différentes perceptions qu'ont les concepteurs sur un système d'information entraînent parfois plusieurs points de vue. Ces perceptions relèvent des spécialités spécifiques à chaque acteur. Ainsi, le modèle global du système est consolidé par l'intervention de l'ensemble de ces acteurs. Il est alors nécessaire d'identifier les concepts communs.

Pour ce faire, une nouvelle approche est conçue pour assister le concepteur dans cette tâche de fusionner plusieurs diagrammes de classes décrivant le domaine étudié. Cette approche proposée est basée sur l'Analyse Formelle de Concepts couplée à un arbre de décision. Cette contribution permet la production du Plus Grand Modèle Commun qui peut être assimilé à une forme normale (factorisée) de l'ensemble des modèles sources.

En outre, cette approche identifie les concepts identiques entre plusieurs modèles tout en faisant émerger les nouveaux concepts spécifiques à chacun des modèles qui factorisent des concepts internes propres à chacun des modèles.

Ces trois contributions ont été validées sur les quinze versions du modèle *SIE-Pesticides*. Les expérimentations ont été effectuées à l'aide d'un profil UML implémenté sous Objecteering. Ce profil intègre le framework ERCA [Falleri, 2010] « Eclipse's Relational Concept Analysis » permettant la construction et la manipulation de treillis de concepts générés à partir de l'Analyse Formelle de Concepts ou l'Analyse Relationnelle de Concepts. Ainsi, il a permis de définir des transformations qui permettent de convertir les modèles en une famille de contextes formels et relationnels (entrée de ERCA) et d'adapter, d'automatiser et de contrôler de manière itérative les expérimentations et l'extraction des métriques d'analyse.

L'ensemble des résultats ont été analysés, validés et ils représentent une étape importante vers la capitalisation des connaissances que nous avons appliquée ici au domaine agricole (projet *SIE-Pesticides* 4.2.1).

Enfin, l'ensemble de ces contributions et, plus précisément, l'approche qui consiste à effectuer une factorisation inter-modèles permettra d'extraire les parties communes aux deux applications de gestion du personnel et des étudiants utilisées au sein de l'université de Djibouti. Le but est d'analyser les parties communes des différentes applications au niveau des bases de données mais aussi des codes d'implémentation. Ensuite, une validation de Plus Grand Modèle Commun sera effectuée lors de l'analyse avec différents acteurs ou groupes d'acteurs.

CHAPITRE

9 Perspectives

Préambule

Les contributions de la thèse ont pour sens de mettre à disposition des concepteurs des méthodes et des outils permettant de :

1. *suivre et analyser l'évolution des modèles depuis l'analyse jusqu'à l'implémentation, en mettant l'accent sur la factorisation.*
2. *définir un meilleur cadre d'utilisation de l'Analyse Relationnelle de Concepts pour l'obtention de modèles avec un meilleur niveau d'abstraction.*
3. *produire le Plus Grand Modèle Commun d'un ensemble de modèles sources.*

Dans ce chapitre, nous dessinons quelques perspectives de ces travaux.

Sommaire

9.1 Introduction

Cette thèse a été l'occasion d'aborder tout un ensemble de pistes de réflexions et d'expérimentations dont certaines n'ont pas encore été exploitées. Dans un premier temps, nous présentons les perspectives de la méthode d'analyse de l'évolution. Ensuite, nous abordons d'une part, les perspectives liées à la factorisation inter-modèle et d'autre part, le potentiel de l'introduction de techniques exploitant mieux la sémantique au travers d'information sur les termes. Enfin, nous présentons les perspectives sur le processus global d'une factorisation maximale.

9.2 Perspectives sur l'analyse de l'évolution

9.2.1 La contextualisation de concepts

Une première perspective concerne l'assemblage de modèles suivant leur contexte. Par exemple, il faut selon le contexte distinguer le département d'une région et le département d'une administration.

Les changements de contexte des différents concepts du modèle créent le besoin de connaître à tout moment l'origine d'un concept ou son état initial, d'où la nécessité de mettre en place une traçabilité des concepts en les rattachant aux concepts initiaux. La traçabilité permet à tout moment du processus itératif de savoir l'origine de chaque concept (cf. figure 9.7).

9.2.2 Concepts Parents, Ascendants, Fils et Descendants

Une autre perspective est d'approfondir la méthode d'analyse de l'évolution en utilisant les métriques pour évaluer la "performance" des factorisations et "contrôler" le processus de factorisation. En effet, plusieurs métriques ont été calculées lors des différentes expérimentations mais certaines n'ont pas été exploitées.

Par exemple, le nombre de concepts parents c'est-à-dire les parents directs, ou le nombre de concepts fils autrement dit le nombre de fils directs, nous permettrait de définir l'importance du concept dans une hiérarchie (cf. figure 9.3). Un autre exemple est le nombre de concepts ascendants ou le nombre de concepts descendants. Ils nous permettraient de définir leur niveau d'abstraction du concept dans la hiérarchie(cf. figure 9.6). Ces métriques rejoignent certaines des métriques vues dans le chapitre 2.

9.3 Perspectives sur la factorisation inter-modèles avec l'Analyse Relationnelle de Concepts

L'une des principales perspectives de ce travail est d'améliorer l'extraction du Plus Grand Modèle Commun grâce à l'utilisation de l'Analyse Relationnelle de Concepts (cf. chapitre 3), qui est une extension de l'Analyse Formelle de Concepts. Elle permettrait d'une part, d'expliciter de façon plus précise les relations entre les concepts du domaine (associations UML en particulier) et, d'autre part, de faire émerger de nouvelles abstractions relationnelles issues des relations (par exemple les associations) entre les éléments de modélisation.

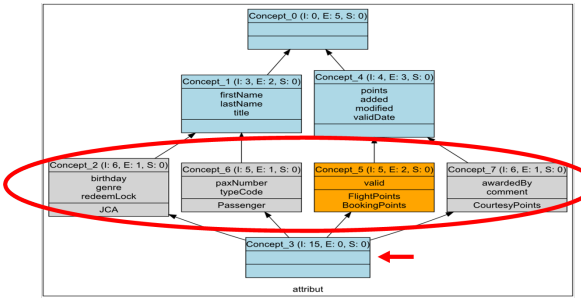


FIGURE 9.1: **Concepts parent**

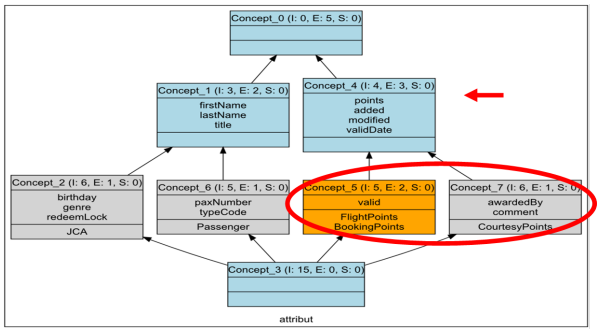


FIGURE 9.2: **Concepts fils**

FIGURE 9.3: **Concepts parent et fils**

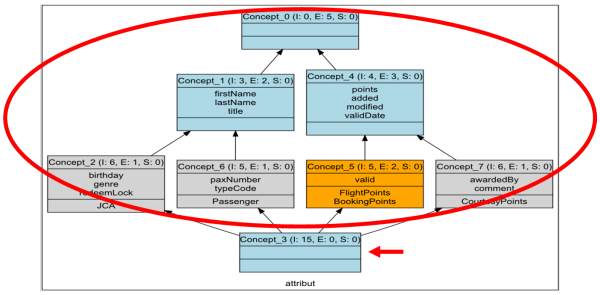


FIGURE 9.4: **Concepts ascendants *"les ancêtres"***

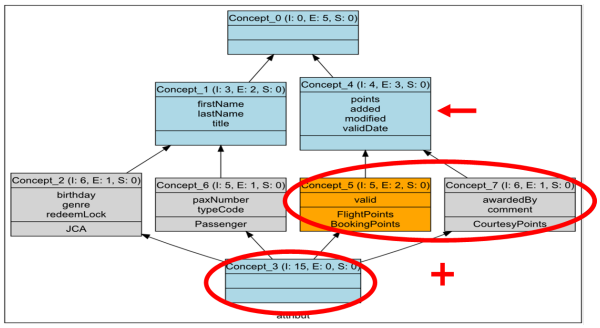


FIGURE 9.5: **Concepts descendants *"les héritiers"***

FIGURE 9.6: **les ascendants les descendants**

Le contrôle de l'émergence de nouveaux concepts grâce à l'Analyse Relationnelle de Concepts, l'automatisation de certaines phases des transformations des modèles UML en treillis mais aussi la phase de reconstruction du modèle UML seront au programme de travaux à venir.

9.4 Perspectives sur l'étude des relations lexicales : Rapprochement sémantique de concepts

Dans l'optique de réduire au maximum le nombre de concepts créés avec la factorisation inter-modèles, le rapprochement sémantique de concepts sera exploré.

Les relations se baseront sur le nom de l'élément de modélisation c'est-à-dire sur la chaîne de caractères qui permet de le désigner. Le choix des noms (termes, identificateurs) pour identifier et/ou décrire un élément de modélisation est donc fondamental. Ce travail de capture par le concepteur des concepts thématiques manipulés par les acteurs est une activité fondamentale de la phase d'analyse d'un projet. Dans le cadre de cette thèse, nous avons débuté un travail de réflexion sur l'usage d'un réseau lexical pour assister les concepteurs et/ou guider la méthode de

factorisation. Ce travail n'a pas été poursuivi car la reconstruction a été estimée prioritaire. Il fera l'objet d'une prochaine étape des recherches.

Après une réflexion sur les indicateurs "sémantiques", il faudra sûrement reproduire les mêmes étapes, à savoir l'expérimentation des indicateurs et l'évaluation des résultats de factorisation correspondants.

Dans cette perspective, l'utilisation d'un thésaurus [1] spécialisé du domaine serait opportun. Le thésaurus constitue un vocabulaire normalisé. Il ne fournit qu'accessoirement des définitions, les relations des termes et leur sélection l'emportant sur la description des significations. Ces thésaurus définissent des relations sémantiques d'intérêt général dans des bases de données lexicales telles que wordnet [2] [Fellbaum, 2010].

Nous envisageons aussi la possibilité de construire un réseau spécifique au domaine lexical grâce à une approche basée sur les techniques de jeu [3]. Pour ce faire, il est important de disposer d'une ontologie du domaine des pesticides. Des travaux préliminaires ont été entamés sur la base des recherches de Mathieu Lafourcade [4] et de Catherine Roussey [5]. Ils permettent de construire l'ontologie suivant deux approches différentes : d'une part par émergence et d'autre part, par négociation. Ces deux approches complémentaires devraient faire émerger une ontologie de meilleure qualité.

9.5 Perspectives sur le processus global d'une factorisation maximale

Lors de la factorisation, l'AFC et l'ARC produisent des treillis regroupant les concepts thématiques ou les caractéristiques de façon la plus "extrême" possible. Cette factorisation "extrême" n'est pas forcément sémantiquement cohérente pour les acteurs. Il est parfois nécessaire de dégrader cette cohérence. La factorisation la plus intéressante est la factorisation qui regroupe le plus de concepts ou de caractéristiques mais qui reste cohérente pour les acteurs.

9.5.1 La traçabilité des concepts

Dans un processus de développement itératif, les acteurs valident l'évolution de l'application au terme de chaque itération : modèles, fonctionnement d'un composant informatique, etc. De ce fait, les nouveaux concepts issus de la factorisation devront être appréhendés et validés par les acteurs.

Cela implique de contrôler l'émergence de nouveaux concepts, d'automatiser certaines phases des transformations des modèles UML en treillis mais aussi la phase de reconstruction du modèle UML qui constituera une prochaine étape.

Lors de cette opération, des liens de traçabilité (cf. chaînettesfigure 9.7) seront établis entre les concepts homologues à plusieurs modèles d'acteurs et ceux du modèle commun. Ces liens de traçabilité constituent une information car les concepts qui en seront démunis (cours, véhicule, etc.)

1. Un descripteur issu d'un type de langage documentaire qui consiste en une liste de termes sur un domaine de connaissance, reliés entre eux par des relations synonymiques, hiérarchiques et associatives.
2. http://wordnet.princeton.edu/
3. http://www2.lirmm.fr/jeuxdemots-english/jdm-accueil.php
4. http://www.lirmm.fr/~lafourcade/index2.html
5. http://motive.cemagref.fr/people/catherine.roussey

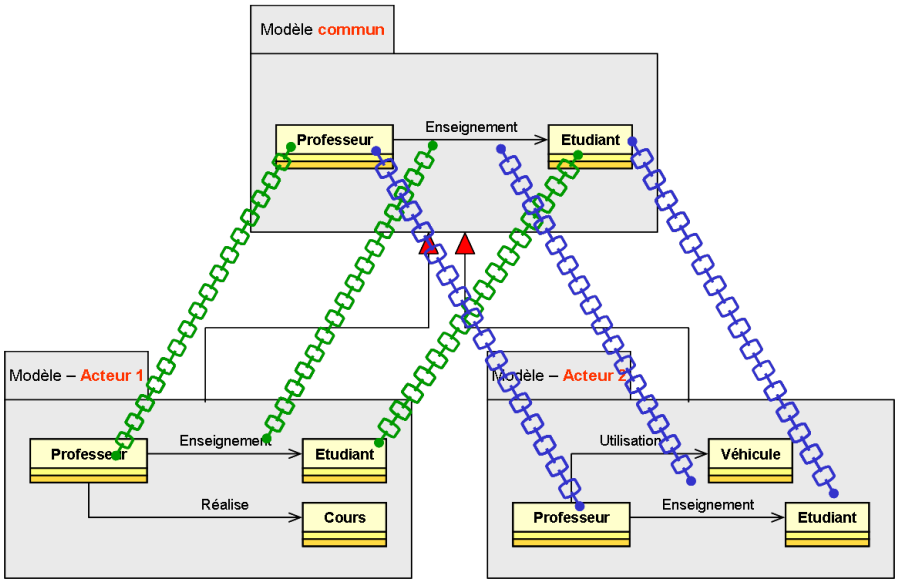


FIGURE 9.7: **Les liens de traçabilité**

sont uniques. L'analyse effectuée au début de chaque itération avec les acteurs devra approfondir si ces concepts uniques sont indispensables dans le modèle commun ou non.

9.5.2 Définition d'un processus de construction

Cette perspective de recherche consiste à savoir comment définir une méthodologie de parcours pour atteindre le modèle final. Plus précisément, l'idée est de travailler dans un processus itératif qui génère un treillis à chaque étape où le nombre de concepts factorisés sera moindre ce qui devrait rendre la validation des modèles par les acteurs beaucoup plus facile pour définir le *chemin optimal de la factorisation* (cf. figures 9.8 et 9.9). Les métriques engendrées permettent aussi d'évaluer l'étape en question en fonction de la configuration choisie afin de répondre à notre objectif principal qui est d'obtenir la factorisation maximale en minimisant le nombre de concepts créés ou fusionnés à chacune des étapes.

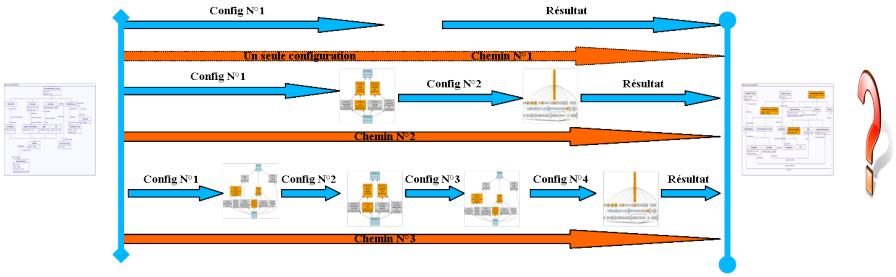


FIGURE 9.8: **Plusieurs chemins de factorisation**

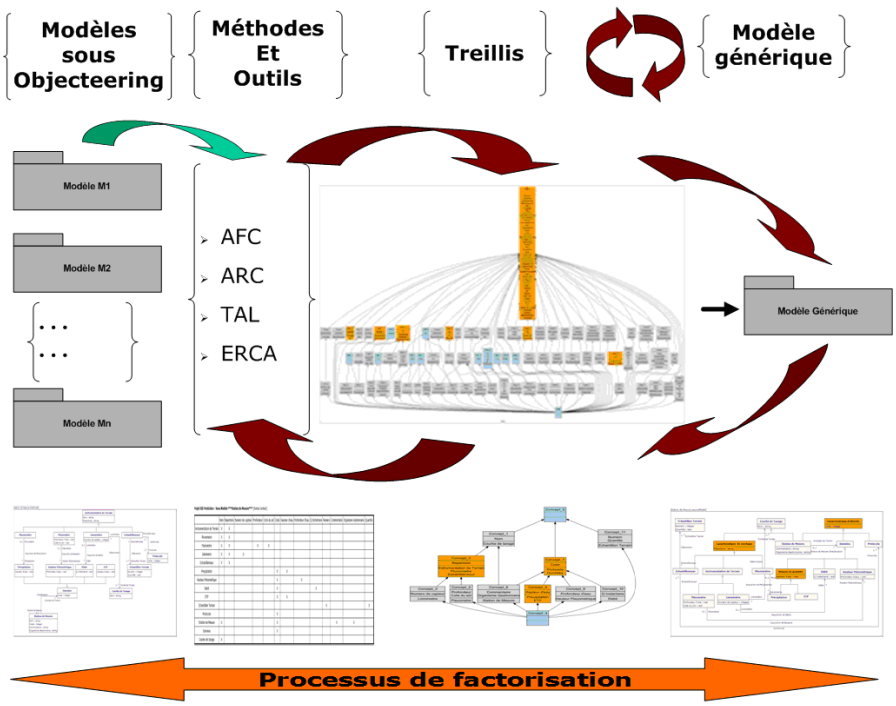


FIGURE 9.9: **Processus global de la factorisation**

ANNEXE

A Annexe

A.1 Table de contexte formel correspondant à la description des classes par les attributs et les rôles (voir section 4.3.1.5)

Classe × Attributs + Rôle	deviceType	deviceNumber	tubeDiameter	waterLevel	measuringDate	codeQuality	tubeHigh	waterAmount	stationName	administrativeInstitute	HydraulicHead	Station	Piezometer	Rainfall	RainGauge	Device	Measure
Piezometer	×	×	×								×	×					
HydraulicHead				×	×	×						×	×				
RainGauge	×	×					×					×		×			
Rainfall					×	×		×				×			×		
MeasuringStation									×	×						×	×

TABLE A.1: **Table de contexte formel correspondant à la description des classes par les attributs et les rôles**

Classe × Attribut + Rôle + Opération	deviceType	deviceNumber	tubeDiameter	waterLevel	measuringDate	codeQuality	tubeHigh	waterAmount	stationName	administrativeInstitute	HydraulicHead	Station	Piezometer	Rainfall	RainGauge	Device	Measure	getType
Piezometer	×	×	×								×	×						×
HydraulicHead				×	×	×						×	×					
RainGauge	×	×					×					×		×				×
Rainfall					×	×		×				×			×			
MeasuringStation									×	×						×	×	

TABLE A.2: **Table de contexte formel correspondant à la description des classes par les attributs, les opérations et les rôles**

A.2 Table de contexte formel correspondant à la description des classes par les attributs, les opérations et les rôles (voir section 4.3.1.6)

A.3 Algorithme de Calcul du Plus Long Chemin

LISTING A.1: **Algorithme de Calcul du Plus Long Chemin**

```
1     public void getPath(ArrayList<String> listArcs, ArrayList<String> path
          String currentArc = "";
3         int sizeListArcs = listArcs.size();
          int sizePath = 1;
5         for (int i = 0; i < sizeListArcs; i++) {
              currentArc = listArcs.get(i);
7
              if (!path.contains(currentArc)) {
9                 path.add(currentArc);

11                sizePath = path.size();
                  if (maxSizePath < sizePath) {
13                    maxSizePath = sizePath;
                      longestPath = new ArrayList<String>(path);
15                }
              } else {
17                //  Is Contains;
                  if (currentArc == path.get(0)) {
19                    int j;
                      for (j = 0; j < nbcycle; j++)
21                        if (cyclePath.get(j).containsAll(path))
                              break;
23                    if (j >= nbcycle) {
                          cyclePath.add(nbcycle, new ArrayList<String>(path)
25                        nbcycle++;
                      }
27                }
                  continue;
29            }

31            ArrayList<String> succArc = getSuccArc(currentArc);
              if (!succArc.isEmpty()) {
33                getPath(succArc, path);
              path.remove(currentArc);
35
          }
37    }
```

A.4 Le plus long chemin dans le graphe de configuration C2 sans la navigabilité

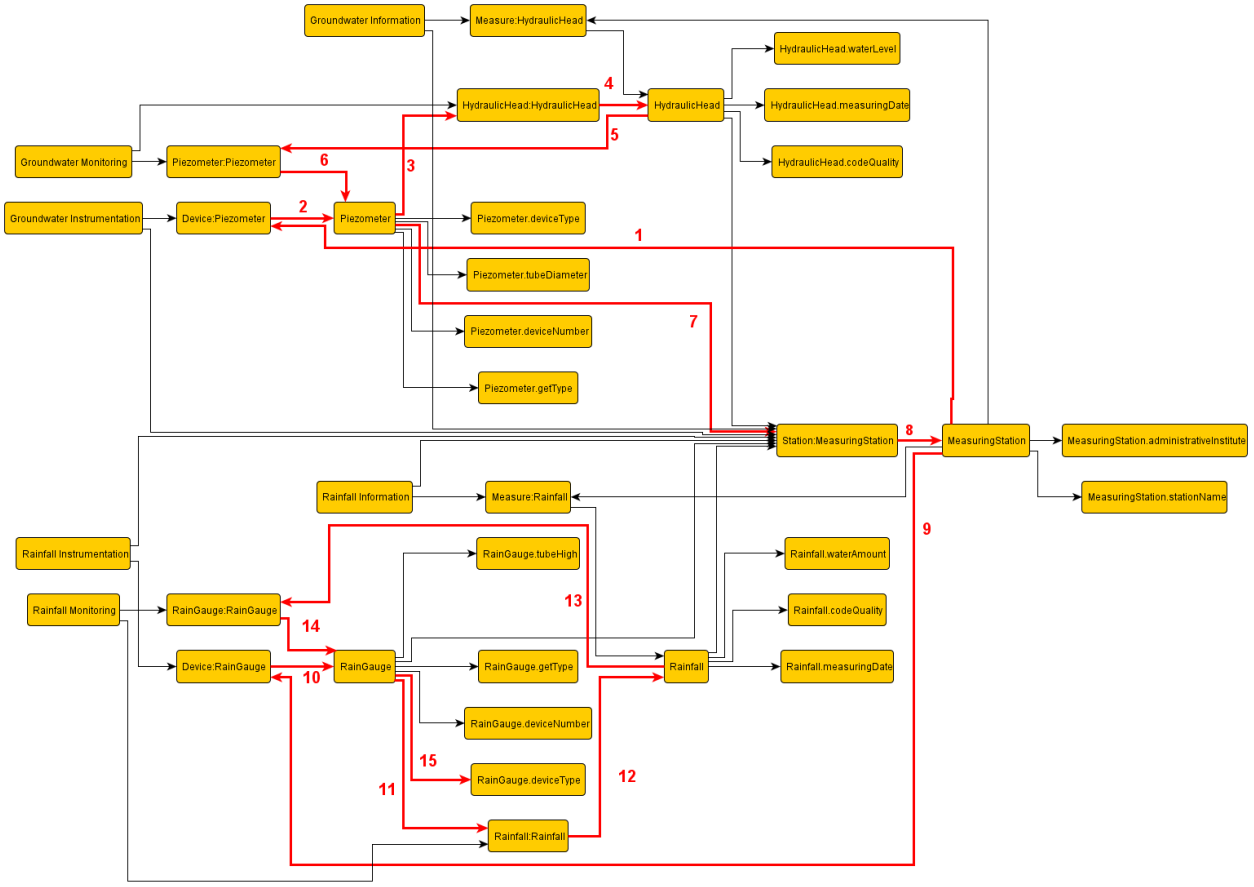


FIGURE A.1: **Le plus long chemin dans le graphe de configuration C2 sans la navigabilité**

A.5 Le degré de chaque sommet du graphe de configuration C2 sans la navigabilité

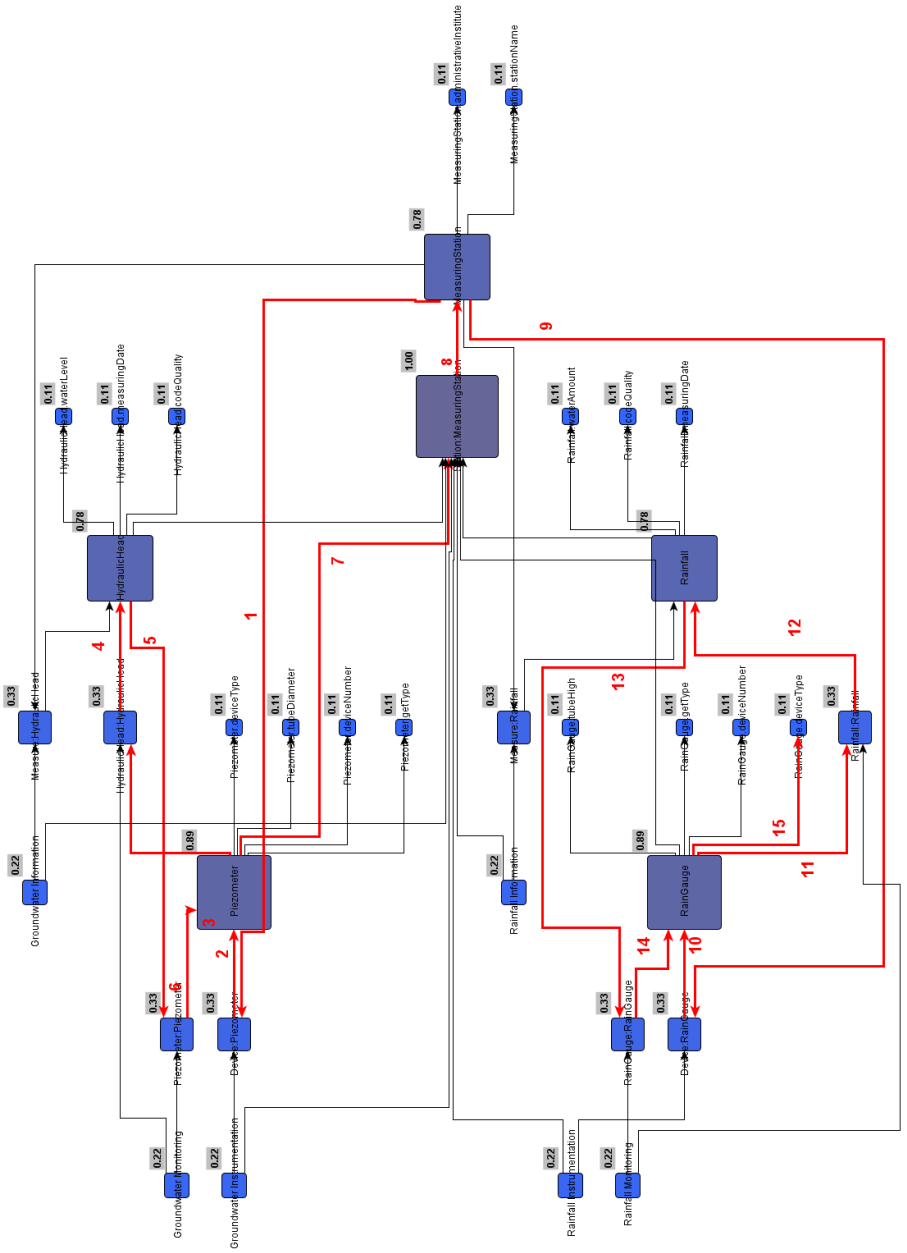


FIGURE A.2: **Le degré de chaque sommet du graphe de configuration C2 sans la navigabilité**

A.6 Treillis des rôles de la configuration C2 sans la navigabilité

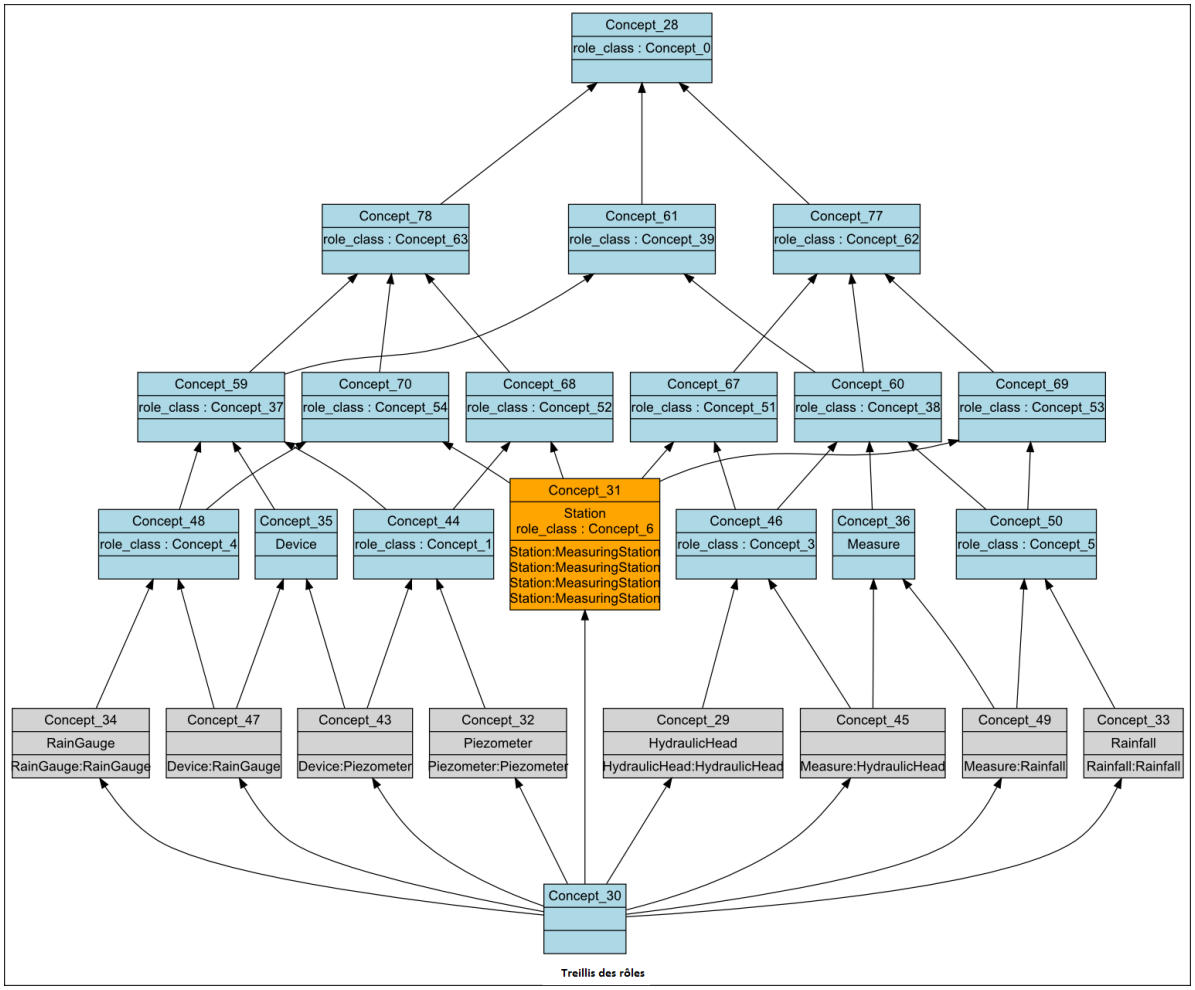


FIGURE A.3: **Treillis des rôles de la configuration C2 sans la navigabilité**

A.7 Pseudo-code de métrique

LISTING A.2: **Weighted Methods per Class**

```
metric  WMC is
    description ( Weighted Methods per Class )
    elements satisfy " this . isInstance ( CtClass )"
    reference is " Methods "
endMetric
```

LISTING A.3: **Depth in Inheritance Tree**

```
metric  DIT is
    description ( Depth in Inheritance Tree per Class )
    x satisfy " this . isInstance ( CtClass )"
    references followed ( Superclass )
endMetric
```

LISTING A.4: **Number of Children**

```
metric  NOC is
    description ( Number of Children of a Class )
    x satisfy " this . isInstance ( CtClass )"
    elements satisfy " this . Superclass "
endMetric
```

LISTING A.5: **Coupling Between Object Classes**

```
metric CBO is
    description ( Coupling Between Object Classes )
    x satisfy " this . isInstance ( CtClass ) "
    input metric SetOfEdges CBO is
            target satisfy " this . isInstance ( CtTypeReference )
            and this . QualifiedName.SimpleName "
          endMetric
endMetric
```

LISTING A.6: **Response For a Class**

```
metric SetOfElementsPerX RFC is
    description " Response For a Class "
    x satisfy " this . isInstance ( CtExecutableReference )"
    elements satisfy " this . isInstance ( CtClass )"
    references followed " Methods ,Body , Statements ,
    Expression , AssertExpression , CaseExpression ,
    ThenStatement , Condition , ElseStatement , Selector ,
    Cases ,Block , Finalizer , Catchers , AssertExpression ,
    CaseExpression , Finalizer , Executable ,DeclaringExecutable"
    endMetric
```

A.8 Dimensionnement de la structure sous-jacente de la factorisation d'un modèle de classe : leçons tirées du modèle de classe d'un système d'information

Dans cet article, nous étudions systématiquement une application pratique de l'Analyse Relationnelle de Concepts sur plusieurs versions d'un modèle de classe. Ce dernier est issu d'un cas réel et ces expérimentations permettent de donner d'abord des chiffres précis sur le comportement de L'Analyse Relationnelle de Concept, d'autre part pour détecter les configurations sont dociles et troisièmement ce genre de post-traitements méritent être conçus à l'avenir d'avoir une méthodologie efficace.

Sizing the underlying factorization structure of a class model: lessons learned from the class model of an information system

Abdoulkader Osman Guédi*, Marianne Huchard‡,
André Miralles†, Clémentine Nebut‡
*Université de Djibouti, Avenue Georges Clemenceau BP: 1904 Djibouti (REP)
Email: guedi@univ.edu.dj
†Tetis/IRSTEA, Maison de la télédétection, 500 rue J.-F. Breton 34093 Montpellier Cdx 5, France
Email: first.last@teledetection.fr
‡LIRMM, Univ. Montpellier 2 et CNRS, 161, rue Ada, F-34392 Montpellier Cdx 5, France
Email: first.last@lirmm.fr

***Abstract*—The design of class models for information systems, databases or programming is a delicate process in which experts of the domain and designers of the class model have to identify and agree on the domain concepts. Formal Concept Analysis (FCA) has been proposed, for a long time now, for supporting this collaborative work and fostering the emergence of higher level entities and the factorization of descriptions and behaviors. More recently, an extension of FCA, Relational Concept Analysis (RCA), has been designed to take into account relations in class models and extending the scope of FCA to the emergence of higher level domain relations and further improving the class model design. FCA and RCA build a kind of normal form for models, in which the factorization is exhaustive, and the specialization order is adequate. The counterpart of these strong properties is a worst-case exponential theoretical complexity. In practical cases, the complexity depends on the data, extracted from the class model, on which the method is applied, *e.g.* a simple data configuration may consider only classes and attributes. FCA and RCA have been applied in several previous works, testing only one data configuration of application on one class model and often not giving detailed figures about the results. In this paper, we systematically study a practical application of RCA on several versions of a real class model for an information system in order to firstly give precise figures about RCA, secondly to detect which configurations are tractable and thirdly what kind of post-treatments deserve to be designed in the future to have an efficient methodology.**

***Keywords*-Class model, class model factorization, Formal Concept Analysis**

I. Introduction

Designing class models for information systems, databases or programs is a recurrent activity, that usually involves domain experts and designers, that have to identify and organize domain concepts. One difficulty is to capture the concepts to introduce, and organize them in a relevant structure introducing adequate abstractions. Formal Concept Analysis (FCA) and its variant Relational Concept Analysis (RCA) are classification techniques that have been proposed to assist this elaboration phase, so as to introduce new abstractions emerging from the identified domain concepts, and to set up a factorization structure avoiding duplication. FCA classifies entities having characteristics, and RCA also takes into account the fact that entities are linked by relations. Both FCA and RCA result in a lattice (called a Concept lattice) that reveals the entities (the already existing ones and the introduced ones), ordered by a partial order of specialization-generalization. This lattice can be exploited so as to obtain the generalization structure for the class model.

Nevertheless, while this whole factorization structure of a class model is a mathematical object with strong theoretical properties, its practical use might suffer from limitations due to the large size of the obtained lattice. A well known risk using it, as it is currently the case with data mining methods is to generate too many abstractions. In such a case, domain experts might be overwhelmed by the information produced making it difficult (or even impossible) to assess the relevance of all these new abstractions. Among techniques for facing this complexity, we can

use various strategies, including strategies for presenting the information (most interesting abstractions are presented first), working on sub-models or using only part of the model elements. But the definition of a strategy has to be based on observations of real case studies to be adequate. Until now, this has only been done on specific class models and generally with only one way of configuring the factorization process. Here, we want to do it more systematically. In this paper we want to assess, in a case study, the size of the factorization results, in order to have a solid foundation for proposing practical recommendations, tools or approaches. We work with a kind of "worst case" of RCA application, by using all the modeling elements and not limiting our investigation to some elements (like classes and attributes, or classes and operations). We show, via various selected graphics, how RCA behaves and we show which configurations are tractable, admitting that some tools present results in a fine way, and which configurations lead to quite unusable results.

The rest of the paper is structured as follows. Section II summarizes Formal and Relational Concept Analysis and explains how they can contribute to a class model design. Section III introduces our case study, a class model for an environmental information system. Section IV settles the environment for our experiments, Section V presents the obtained results, and Section VI discusses the results. Finally, Section VII presents and discusses related work, and Section VIII concludes.

II. Concept lattices in class model refactoring

In this section, we explain how concept lattices implement and reveal the underlying factorization structure of a class model. We also show how this property can be exploited for class model refactoring, and in particular: generating new reusable abstractions that improve the class organization and understanding, especially for domain experts, limiting attribute and role duplication.

A. Brief introduction to Formal and Relational Concept Analysis

Formal Concept Analysis (FCA): Formal Concept Analysis [1] is a mathematical framework that groups entities sharing characteristics: entities are described by characteristics, and FCA builds (formal) concepts from this description. A concept groups a maximal set of entities sharing a maximal set of characteristics. The concepts are (partially) ordered by a lattice structure. For example, let us consider several animals described by several characteristics, as specified in Table I. In such a table (called a formal context) a cross is put in a cell of row `r` and column `c` to indicate that the entity described by row `r` has the characteristic of column `c`. Here, an ostrich has no characteristic, whereas a hoopoe has a crest, migrates, and flies.

	flying	nocturnal	migratory	with crest	with membrane
flying squirrel	x				x
bat	x	x			x
ostrich					
flamingo	x		x		
hoopoe	x		x	x	

Table I
Context for the animals

Applying FCA on such formal context results in the concept lattice given in Figure 1. In the lattice, concepts are represented by rectangles, and the partial order is represented by edges. The upper part of a concept is its name (here, the names are generated from the tool and are meaningless). The middle part of a concept contains its set of characteristics, that is called the intent of the concept. The bottom part of a concept contains its set of entities, that is called the extent of the concept. In the lattice, the concepts are structured following a generalization mechanism from the most general, at top, to the most specialized, at bottom. The intent of a concept includes the characteristics from upper concepts, while the extent of a concept includes the entities from the concepts below.

The lattice of Figure 1 shows a simplified form of the lattice of the animals, in which only the elements (entities or characteristics) introduced by the concept are shown: in other words, we show only the simplified extent and simplified intent. For example, `Concept_Animals_4` represents migratory and flying animals: the flamingo and the hoopoe. The flying characteristic is inherited from `Concept_Animals_1` , while the hoopoe entity is included from `Concept_Animals_5` . In the simplified form, characteristics are maximally factorized, and the

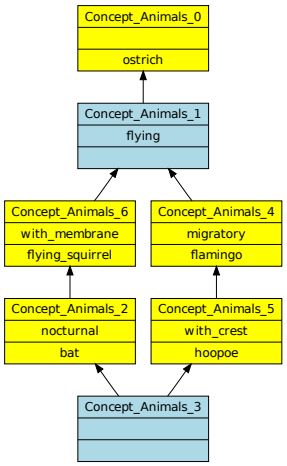


Figure 1. Concept lattice for the animals (simplified concept)

	group leader	**song writer**
John	×	×
Georges		×
Paul	×	×
Ringo		

Table II
CONTEXT FOR THE SINGERS

partial order corresponds to inclusion between sets of characteristics or in other terms, concept specialization.

Relational Concept Analysis (RCA): Relational Concept Analysis is a variant of FCA that aims at grouping several kinds of entities that are linked by different relations. The main principle of RCA is to iterate on FCA application, and the concepts learnt during one iteration for one kind of entity are propagated through the relations to the other kinds of entities. As an example, let us consider as entities not only animals, but also singers, that can be song writers and group leaders; they are described in the formal context of Table II.

Applying FCA to the singers results in the lattice presented in Figure 2 and allows to discover concepts like `Concept_Singers_1` that represents singers that are group leaders and that write songs. In the extent of that concept, we find *Paul* and *John*.

Singers and animals are linked by the relation "likes": singers like (or not) animals. This relation is also represented by a binary table (Table III), that is

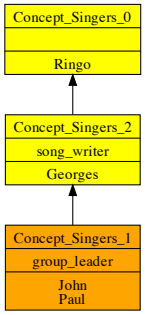


Figure 2. Lattice for the singers using FCA

likes ↗	**flying squirrel**	**bat**	**ostrich**	**flamingo**	**hoopoe**
John				x	
Georges			x		
Paul		x			x
Ringo	x				

Table III
RELATIONAL CONTEXT FOR THE RELATION *likes*

called a relational context, and that links entities to entities. The idea is to use the relation linking the singers to the animals to discover new concepts. For example, we can discover the concept of group leaders and song writers that like at least one migratory and flying animal. The migratory and flying animals are grouped in `Concept_4` in the lattice of the animals. This information is propagated into the context of the singers, introducing relational characteristics stating which singer is in relation "likes" with which concepts of animals. For example, a relational characteristic `likes Concept_4` is introduced, and it has to be decided which singer likes `Concept_4`. This propagation step is called the relational scaling, and can be done using several scaling operators. Here, we use the existential one, that states that an entity has a relational characteristic for relation `r` and concept `c` (i.e. is in relation `r` with `c`) if the entity is in relation `r` with at least one entity of the extent of `c`. Using different scaling operators, we could find concepts of singers liking all or only the migratory and flying animals. This propagation allows new concepts for entities to be discovered. Here, we discover new concepts for the singers: singers are grouped not only based on their characteristics, but on the fact that they like or not

groups of animals.

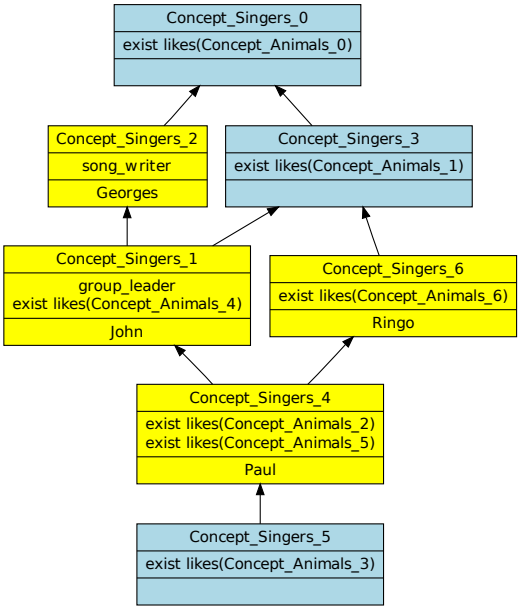


Figure 3. Lattice for the singers using RCA, after 2 FCA iterations

The process stops when a fix point is reached, i.e. when no new concepts are introduced by an iteration. In Figure 3, we show the result of the application of RCA, with the scaling operator `exists`, to our example of singers and animals (here, there were two FCA iterations). `Concept_1` represents the leaders and song writers that like at least one migratory and flying animal: *John* and *Paul*. `Concept_6` represents the singers that like at least one flying animal with membrane (*Ringo* and *Paul*).

In the obtained concept lattices, we distinguish *merged concepts* and *new concepts*. A ***merged concept*** is a concept that has more than one entity in its simplified extent.For example, in Figure 2, `Concept_Singers_1` is a merged Concept, since its simplified extent contains *John* and *Paul*. That means that *John* and *Paul* are described by exactly the same set of characteristics. A ***new concept*** is a concept that has an empty simplified extent.For example, in Figure 3, `Concept_Singers_3` is a new concept: it represents the concept of singers that like at least one flying animal, and corresponds to a new domain abstraction that avoids the duplication of some characteristics.

B. Applying FCA and RCA to class model refactoring

To apply FCA to class models, we encode the elements of a class model into formal contexts. For example, we provide a context describing the classes by their attributes. Then the FCA approach reveals part of the factorization structure and supports part of the refactoring process by using a straightforward description of UML elements. For example, we can discover new concepts for classes interpreted as new super-classes, factorizing two attributes.

Nevertheless this approach does not fully exploit the deep structure of the class model. Let us take the example of Figure 4(a). The attribute `name` is duplicated in classes `B1` and `B2`, and FCA can generate the model of Figure 4(b) that introduces a new class (here manually named `NamedElement`) that factorizes this attribute. However, FCA does not compute the factorization that can be found in Figure 4(c), in which a class called `SuperA` factorizes the two associations from `A1` to `B1` and from `A2` to `B2`, being given that now `B1` and `B2` have an ancestor `NamedElement`.

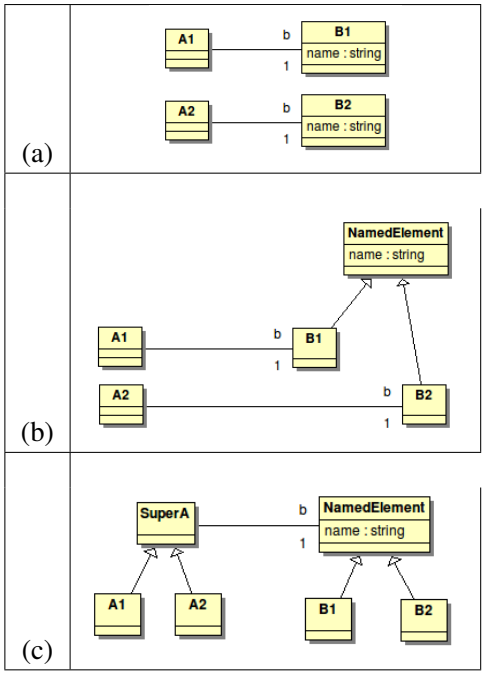


Figure 4. Example of factorization in class models

Extracting abstractions using this deep structure can be done with RCA, which builds the entire factorization structure, including information on the elements

and their relations. RCA models the fact that classes are linked through associations. In the first iteration steps, RCA computes the factorization in Figure 4(b), and then propagates the new concept `NamedElement` through the association between classes. Then the factorization of Figure 4(c) is computed during the next iteration steps. The process then stops since a fixpoint is found (no new abstractions can be found).

RCA is thus able to generate a maximal factorization structure for a class model, taking into account the relations between the different model elements. The obtained structure contains no duplication, and improves the organization of the model. However, when applied on large data, RCA may result in the introduction of many new concepts, that may be too abstract, and/or too many to be analyzed. That is why in the next sections, we study on a case study the behavior of RCA for a large class model corresponding to an actual information system. Our objective is to determine if RCA remains suitable for large class models, and how to configure RCA to obtain exploitable results.

III. The *Pesticides* class model

The case study we use in this paper is a project from the Irstea institute, called Environmental Information System for Pesticides (EIS-Pesticides) which aims at designing an information system centralizing knowledge and information produced by two teams: a Transfer team in charge to study the pesticide transfers from plots to rivers and a Practice team which mainly works on the agricultural practices of the farmers. The domain analysis has been carried on during series of meetings with one team or both teams.

The first model (V0) was designed by a master student in computer science with the expertise of the Transfer team, studying a set of documents and data, and with punctual short interviews with experts. This model was exempt from superclasses. The other models has been produced by experts on computer science. The V1 model is the result of an analysis with the Transfer team. The hydrographic domain concepts from V0 model have been better refined by taking into account part of the specifications of the chosen database. In the V2 model, discussions with the Practice team lead to add many superclasses. The V3 model results from a structuration in three models within packages (landscape objects, agricultural activity, metrology to study the transfer of pesticides). Inappropriate concepts of metrology of the V0 model were also removed and replaced in a well-formed hierarchy. In this structuring effort, several associations have been removed. Then the model of agricultural activity was finely detailed producing a strong increase of the number of classes (V4 model). In the V5 model, the activity of metrology was entirely analyzed again. To do this, a copy of the package was made in order to avoid the loss of concepts, producing thus a strong increase of the model elements. The V6 model is the result of a classical analysis during which the domain concepts were refined. In the V7 model, the project leader has removed the remaining concepts following the copying of the activity model of metrology, decreasing strongly the model elements. In the V8 model, all the attributes were given a type, and the associations were detailed. In the V9 model, the use of a pictogrammic language to annotate the spatiality and temporality components of the domain concepts has resulted in the deletion of attributes and their substitution by stereotypes. The V10 model results from a classical process of analysis. The V11 model is the result of strong modifications of the design of agricultural activities and of the description of the soil and subsoil. In this model, specific groups of classes with no attribute and all belonging to the same hierarchy of classes have been replaced by an enum. Consequently, several classes and associations were deleted. In V12, a forgotten class with its attribute wad added. The inability with UML to specialize an enumeration into two enumerations led to get back into the V13 model to a classical hierarchy of classes.The V14 model comes from a small reorganization resulting from the addition of a class. Figure 5 shows in function of the model version the number of Classes, Attributes, Associations and the sum of all model elements.

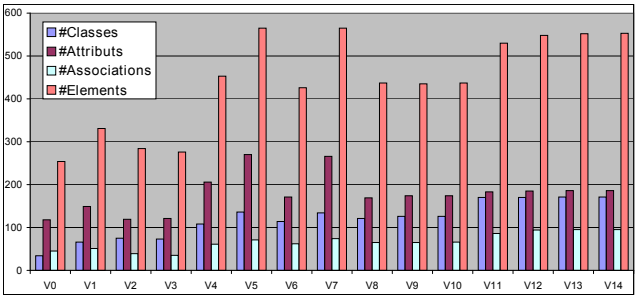


Figure 5. The number of model elements over the various versions

IV. Experimental setup

Our tool is based on the Modeling Tool Objecteering[1] and the framework eRCA[2] that are connected via a Java programming API. eRCA has been extended for computing metrics. In this paper we focus on a configuration (part of the meta-model) including the following entities described in formal contexts: classes, associations, operations (very few in the *Pesticides* model), roles and attributes. Their characteristics are their names. Let us notice that in the Objecteering meta-model, there is an `EndAssociation` metaclass distinct from the `Attribute` metaclass, contrarily to the current UML meta-model which represents roles and attributes in the `Property` meta-class. The relational contexts describe: which class owns which attribute, which class owns which operation, which class owns which role, which association owns which role and which type (class) has a role. We also consider four parameterizations for this configuration depending on whether we take into account navigability and undefined elements. If a navigability is indicated on an association, meaning that objects from the source know objects from the target (not the reverse), taking into account navigability (denoted by `Nav`) results in the following encoding: the source class owns the corresponding role, but the target class does not own any role corresponding to that association. Not taking into account navigability (denoted by `noNav`) means that the source class and the target class own their respective role in the association. In the modeling tool, unnamed roles are named "undefined". We can choose to include this "undefined" name in the contexts (denoted by `Undef`) or not (denoted by `noUndef`) meaning that unnamed elements have no name.

V. Results

In this section, we report the main results that we obtain when we apply RCA on the *Pesticides* class model using the configuration and the parameterizations defined in Section IV. We consider two special iteration steps: step 1 (close to FCA application) and step 6 (where paths of length 6 in the model are followed). At step 1 for example, common name attributes are used to find new superclasses. At step 4, new superclasses can be found as shown in Figure 4(c), and 2 steps later, new super-associations can be found from the class concepts found at step 4. We examine, for classes and associations, which are the main elements of the model, metrics on *new* class concepts and *new* associations concepts (Section V-A), then on *merge* class and association concepts (Section V-B). Execution time is presented in Section V-C, and we conclude this part by giving indications about the number of steps when the process reaches the fix-point.

A. New abstractions

We focus first on the concepts that appear in the class lattice and in the association lattice and which represent new abstractions: they will be interpreted as new superclasses or as new generalizations of associations. In a collaborative work, these concepts are presented to the experts that use some of them to improve the higher levels of the class model because they recognize domain concepts not explicit until now. This is why their number is important; if too many new abstractions are presented to the experts, these experts might be overwhelmed by the quantity, preventing a relevant and efficient use of the method.

Figure 6 shows the *new* concepts in the class lattice (thus the new superclasses) at step 1, when paths of size 1 have been traversed. For example, this means that if some classes have attributes (or roles) of the same name in common, those attributes (or roles) will certainly be grouped in a *new* class concept. This new class concept can be presented to the expert to control if this corresponds to a new relevant class abstraction (or superclass). We notice that `Nav` parameterizations produce less new concepts than `noNav` ones. The number of new concepts decreases as the analysis process progresses.

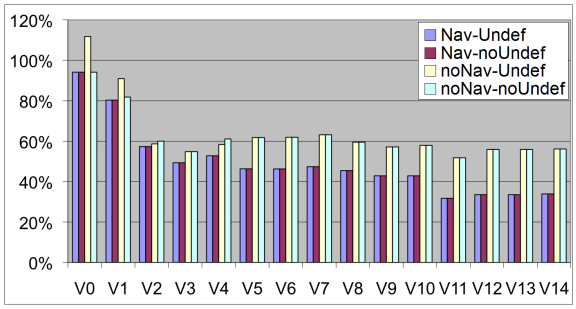


Figure 6. New class abstractions created at step 1 versus the number of initial classes.

[1] http://www.objecteering.com/
[2] http://code.google.com/p/erca/

Table IV focuses on best and worst cases. It indicates that in the best case (of percentage of new superclasses), 32% of new potential superclasses will be presented to the experts, for the model V11 which contains 170 classes, giving 54 new potential superclasses. In the worst case, we have 112% of new potential superclasses, for V0 model, which has 34 classes, thus only 38 new potential superclasses are found. At this stage, we do not see a serious difference between the four parameterizations.

Difficulties and differences are evident at step 6. Figure 7 shows that the two parameterizations `noNav` (generating more cycles in data) give results that will need serious filtering to separate relevant *new* concepts from the large set of *new* concepts. Table IV shows that `Nav` parameterizations will produce less than one and a half the initial number of classes, while `noNav` parameterizations can produce up to 10998 class concepts, really requiring either additional post-treatments or avoiding to generate all the concepts.

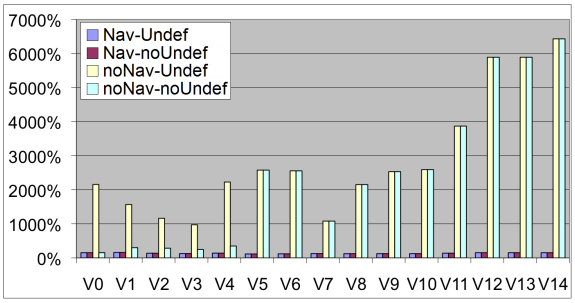


Figure 7. New class abstractions created at step 6 versus the number of initial classes

Figure 8 shows the *new* concepts in the association lattice at step 1. They represent associations that are at a higher level of abstraction. Experts can examine them, to see if they can replace a set of low-level associations. Table V gives concrete figures on new association concepts: in `Nav` parameterizations, at most 15 higher level associations are presented to experts; in `noNav` parameterizations, the number grows until 32, remaining very reasonable to analyze.

Figure 9 shows the *new* concepts in the association lattice at step 6. It highlights the fact that, at this step, the number of these concepts may exponentially grow, and it is especially high in the last versions of the class model, in which we initially have many associations. Table V focuses on details of these figures. This confirms that the number of new association concepts, in `Nav` parameterizations still remains reasonable (even it is higher than in step 1), but that in `noNav` parameterizations it dramatically grows and may reach about 9500 concepts.

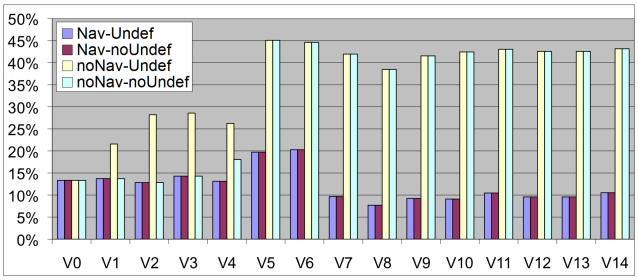


Figure 8. New association abstractions created at step 1 versus the number of initial associations

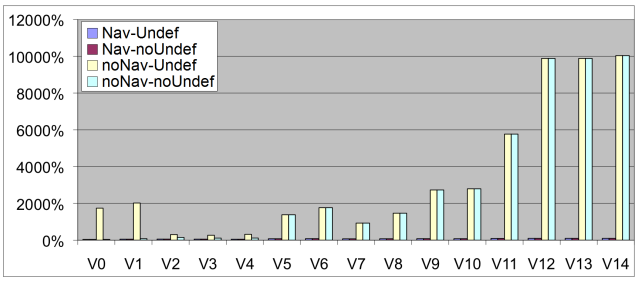


Figure 9. New association abstractions created at step 6 versus the number of initial associations

B. Merged concepts

Merge concepts are concepts which introduce several entities (*e.g* classes or associations) in their extent. Such entities share exactly the same description in the model. For example, a merge class concept can group classes that have exactly the same name attributes. This common description is first detected at step 1, then it does not change because the following steps refine the description by adding new relational characteristics and concepts; entities remain introduced in the same concepts. Thus we only present merge concept metrics at step 1. For classes and associations, the merge concept number is the same for the four analysis configurations. For experts, analyzing a merge concept consists in reviewing the simplified extent and examining if the entities (class or association) have been exhaustively described or effectively correspond to a same domain concept.

Step	Parameters	% Min	Version	# classes	# new class concepts	% Max	Version	# classes	# new class concepts
STEP 1	Nav-noUndef/Undef	32	V11	170	54	94	V0	34	32
	noNav-Undef	52	V11	170	88	112	V0	34	38
	noNav-noUndef	52	V11	170	88	94	V0	34	32
STEP 6	Nav-noUndef/Undef	116	V5	136	158	161	V1	66	106
	noNav-Undef	973	V3	73	710	6432	V14	171	10998
	noNav-noUndef	153	V0	34	52	6432	V14	171	10998

Table IV
STEP 1 AND STEP 6 - NEW CLASS CONCEPTS

Step	Parameters	% Min	Version	# assoc.	# new assoc. concepts	% Max	Version	# assoc.	# new assoc. concepts
STEP 1	Nav-noUndef/Undef	8	V8	65	5	20	V6	74	15
	noNav-noUndef/Undef	13	V0	45	6	45	V5	71	32
STEP 6	Nav-noUndef/Undef	38	V0	45	17	98	V14	95	93
	noNav-Undef	266	V3	35	93	10037	V14	95	9535
	noNav-noUndef	38	V0	45	17	10037	V14	95	9535

Table V
STEP 1 AND STEP 6 - NEW ASSOCIATION CONCEPTS

Figure 10 presents metrics for merge class concepts. V5 and V6 have a higher percentage of merge concepts because during analysis, a package has been duplicated at step 5 for refactoring purpose. The duplicated classes have been removed at step V7. In the other cases, there are not so much merge class concepts to be presented to the experts, between 0% and 2%, giving a maximum of two classes. This often corresponds to classes with incomplete description, that the experts should develop into more details. The low number of such cases makes easy the task of experts.

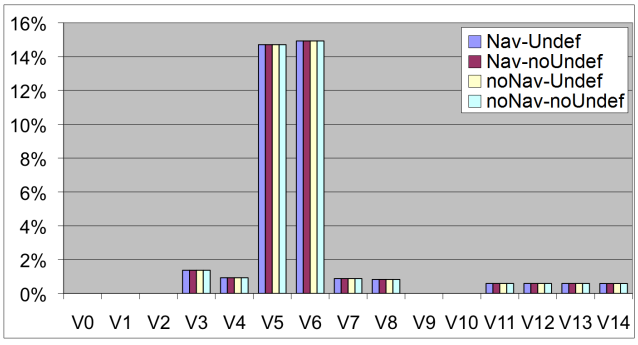


Figure 10. Ratio # merge classes on # initial classes

Figure 11 presents metrics for merge association concepts. The percentage of merge association concepts is higher than the percentage of merge class concepts. This is explained by the fact that associations are only described by roles, that occasionally share the same names (identical to some class names). It varies between about 2% and 18%, meaning that at most 10 merge association concepts are presented to the experts for evaluation, making a little bit more complicated the analysis task compared to the case of classes, but it remains very reasonable.

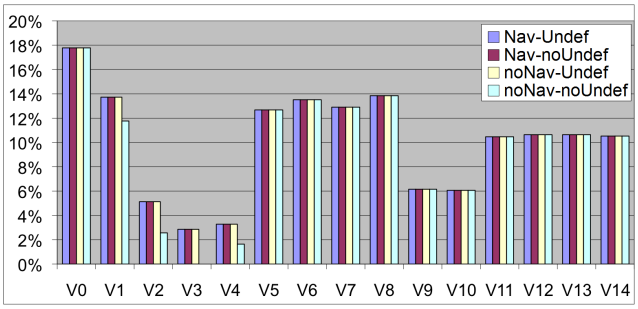


Figure 11. Ratio # merge associations on # initial associations

C. Execution time, used RAM and total number of steps

Experimentations have been performed on a cluster composed of 9 nodes, each one having 8 processors Intel (R) Xeon (R) CPU E5335 @ 2.00GHz with 8 GO of RAM. The operating system was Linux (64bits version) and the programs are written in Java.

Figures 12 and 13 show the execution time in seconds, at step 1 and at step 6. At step 1, the execution time for the two `Nav` parameterizations are below 6

seconds, while for the two noNav parameterizations, for some versions (especially when there are more associations, like in the last versions) it may reach about 13 seconds. At step 6, the execution time for the two Nav parameterizations are below 8 seconds. But for the noNav parameterizations, we notice longer executions, up to 10 minutes. Considering that the task does not require an instantaneous answer, even if it occurs during an expert meeting, spending a few minutes for constructing the concept lattices can be admitted.

Table VI
FIGURES ON USED MEMORY (IN MEGABYTES)

Step	Parameters	min	max	average
Step 1	Nav-Undef	39	453	237
	Nav-noUndef	17	471	205
	noNav-Undef	41	969	480
	noNav-noUndef	24	969	532
Step 6	Nav-Undef	44	471	213
	Nav-noUndef	6	403	140
	noNav-Undef	33	1846	656
	noNav-noUndef	33	1147	520

Table VI shows the RAM used during execution, here again, noNav parameterizations are the worst, reaching about 2 GigaBytes of used memory. In the case of Nav parameterizations, it is interesting to observe that there is not a significative difference between step 1 and step 6.

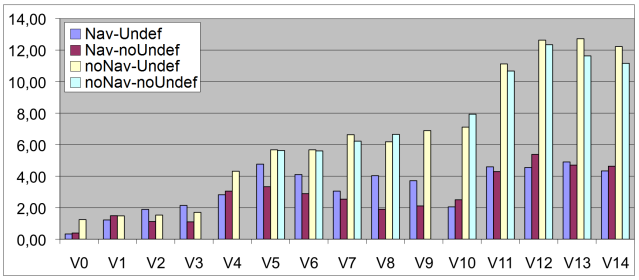


Figure 12. Execution time at step 1 (in seconds)

Figure 14 shows, for the Nav-noUndef parameterization the total number of steps needed to reach the fix-point, and the size of a longest simple path (may contain cycles, but no repetition of arc). We observe that the step number (from 6 to 16) is always below the size of the longest simple path which gives in our context a practical upper bound to the number of

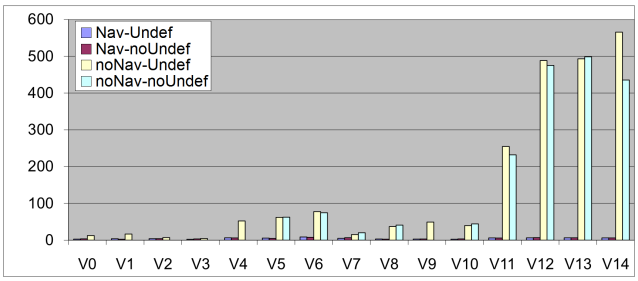


Figure 13. Execution time at step 6 (in seconds)

steps. This means that if we dispose in the future, of relevant filtering strategies, we can envisage studying new concepts appearing after step 6.

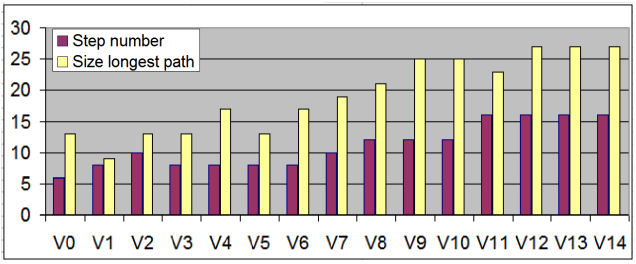


Figure 14. Step number (first) and longest simple path size (second) in C2, Nav-noUndef parameterization

VI. DISCUSSION

From the observed results, we can learn several lessons. Execution time is correct on the cluster, and the used memory is acceptable. Both the concepts and the structure of the lattice is useful for the experts to determine which relevant modifications should be applied. The strength of the lattice representation is that it provides a structure adapted to the task of choosing the right new abstractions, since concepts can be presented following the specialization order, depending of what is the demand of experts. Analyzing *merge* class concepts and *merge* associations concepts is not a problematic task, due to their small numbers. The feasibility of analyzing *new* class concepts and *new* associations concepts deserves more attention. Nav parameterizations produce exploitable results with a maximum of about 50 class concepts (resp. about 30 new association concepts) to be examined at step 1. At step 6, more analysis work has to be done because experts may have to face from one to three hundreds of new class concepts (resp. about one hundred of

new association concepts). In this case, a simple initial advice is not to try to understand as a whole all the concepts, but to use the possibility of looking, at each step between step 1 and 6 the appearing concepts. When a concept C is considered as not important, it has to be marked and this offers the possibility to remove from the list of presented concepts at the next steps all the concepts that are built because of the existence of C (via relational attributes).

Here, we examine successive versions of a model, and there are less and less *new* concepts meaning that as the design of the model progresses, the experts themselves add more abstractions. Our analysis is done *a posteriori*: the objective of our method would be to insert between each model release, an RCA-based analysis, in order to accelerate the discovery of new abstractions, and the completion of elements (highlighting of the *merge* concepts).

As our problem is a kind of data-mining problem (finding similarities and common patterns in a class model), we face similar questions, mainly similar explosion of results, and we may exploit similar approaches for reducing the complexity. In the context of FCA, a particular sub-order of the concept lattice (the AOC-poset), induced by the concepts that introduce an entity or a characteristic, might offer an important reduction of the produced concept numbers, without loosing main information about factorization.

Another track is limiting input data, for example by removing attributes that have limited semantic value. To go in that direction, we would propose to the experts a preliminary study on terms that often appear in the model, with the intuition that a frequent term, like "name", or "code", does not carry main information to be factorized. To limit the number of concepts to be examined by experts, it would be fruitful to group new concepts that declare few attributes (a group being a connected part of the lattice).

Classical visualization techniques for large result exploitation can be used, for example presenting concepts grouped by abstraction level, or using relevancy metrics. Such metrics can use the place in the lattice (from most concrete to more abstract concept), an importance degree between the attributes (present first concepts that mainly contain non relational attributes, opposed to concepts that mainly or only contain relational attributes), or stability of the concepts [2].

There are some limits to the generalization of our conclusions. The class model is mainly a data model (very few operations), destined to build a database schema and we study various versions of a same class model. Nevertheless, the *Pesticides* model is a classical model, representative of the models that are built in the environmental field.

VII. Related Work

The use of FCA in the domain of class model refactoring has a long and rich history in the literature. As far as we know, it has been introduced in the seminal paper of Godin et al. [3] for extracting abstract interfaces from a Smalltalk class hierarchy and extensions of the work have been published in [4]. Other approaches have been proposed, that take into account more information extracted from source code like super calls and method signatures in [5]. Similar goals and lattice-based techniques are found in type systems and object-oriented database schema refactoring [6], [7], [8]. The ancestor method of RCA has been introduced for generalizing the approach to take into account relations between analyzed elements in [9].

The first practical case of application of RCA was reported in [10]. It has been conducted on several medium-size class models of France Télécom (now Orange Labs)[3]. The RCA configuration was composed of classes, methods, attributes, and associations. Classes have no description; attributes are described by name, multiplicity and initial value; methods are described by name and method body; associations are described by names, role name and information like multiplicity and navigability. Associations are connected to classes that are origin and destination, classes are connected to the attributes and operations they own, attributes are connected to classes that are their type, etc. The class models contain a few dozens of classes and the new concepts to be examined by experts varies from a few concepts to several hundreds. In [11] detailed figures about the experiment are given. In the largest project (57 classes), the number of *new* concepts was 110 for the classes, 9 for the associations and 59 for the properties. Among these concepts, experts have been able to select relevant ones. In this paper, we have several class models that are geater in termes of

[3]Joint RNTL project supported by french department of research and industry

number of classes and we reify the role notion, rather than encoding it in the association description, with the risk of having more produced concepts. We discard technical description (like multiplicity which has no strong semantics). We also analyze the *merge* concepts, another interesting product of RCA.

In [12], RCA has been experimented on an excerpt of the class model of the Jetsmiles software of JETSGO society (http ://www.jetsgo.net/). We give details about this experiment because it has not yet published in english. The class model excerpt is composed of only 6 classes, connected by 9 associations, and about 30 attributes. Attributes and roles are mixed, and classes are connected by a relational context to attributes+roles, while attributes+roles are connected by another relational context to their type (when it is a class). The UML elements are described by many technical features: multiplicity, visibility, being "abstract", initial value, etc. This technical description is important to rebuild new UML elements at the last step of the process, but it adds complexity during RCA application and many non relevant concepts, *e.g.* the superclass of all abstract classes. A post-treatment analyzes the built concepts in order to keep the most relevant ones. The class concept lattice contains about 35 concepts while the attribute+role concept lattice has about 25 concepts. In [12], the size of the class model is very small and not realistic. Using the configuration with many technical elements as they do would not be scalable in the case of the *Pesticides* model.

RCA has been experimented on two Ecore models, two Java programs and five UML models in [13]. The used configuration is composed of the classes described by their names and the properties (including attributes and roles) described by their names; classes are connected to their properties and properties are connected to their type (when it is a class). To report some representative results of this experiment, in Apache Common Collections, which is composed of 250 classes, RCA finds 34 new class concepts and new property concepts; in UML2 metamodel, which is composed of 246 classes and 615 properties, RCA extracts 1534 new class concepts and 998 new property concepts.In this experiment, associations were not encoded, contrarily to what we do in the *Pesticides* model. Nevertheless the possible explosion of the concept number already appears. In our case, we introduce associations in the configuration, and we show, that with some precautions as annotating by navigability and naming the roles, refactoring with data including the associations may remain feasible.

The more recent study [14] compared three strategies of FCA/RCA application to part of the open-source Java Salome-TMF software[4] (test management, 37 classes, 142 attributes). We also describe into details this experiment because it has not yet published in english. The strategies were: AFC-NAME, AFCC-NAME and ARC-NAME. In AFC-NAME, entities are the classes and characteristics are the attribute names; A semantic analysis of identifiers allows to assign to a class its own attribute names and their hyperonyms. In AFCC-NAME, classes are described by composite characteristics containing the attribute name, its type, if it is static or not; hyperonymy is also encoded in the relation. Operations were not taken into account because they were very technical and the author estimated that the attributes were bringing the main information. In ARC-NAME, a first formal context includes the classes (no description), a second formal context describes the attributes by their names and the hyperonyms of their names, a first relational context associates to a class its attribute names, a second relational context associates to an attribute its type when it is a class. AFC-NAME produces 26 new class concepts, and 3 merge concepts; AFCC-NAME produces 17 new class concepts, and 3 merge concepts; ARC-NAME produces 33 new class concepts, and 3 merge concepts. Besides, ARC-NAME produces 21 new attribute concepts and 13 merge attribute concepts. All these results are very acceptable for analysis purpose by experts, but Java softwares do not have associations and it is difficult to generalize these results to the general case of class models. Compared to this work, here we do not use linguistic information, this will be done in a future work, nevertheless the terms in the model are not technical identifiers but rather domain terms carefully selected by the expert group, thus there are less problems in using them directly.

To conclude, this paper proposes the most advanced study of the application of RCA to class models: avoiding non relevant, too technical, features; using the model elements as described in their meta-model, without changing their structure and connections; and using realistic class models.

[4] http://wiki.ow2.org/salome-tmf/

VIII. Conclusion and perspectives

In this paper, we study the possible application of Relational Concept Analysis (a variant of Formal Concept Analysis, a data analysis method) on class models, so as to obtain a relevant factorization structure. We applied RCA on several versions of the same information system (from 40 to 170 classes), and we studied the impact of several parameters in the application (taking (or not) into account navigability and undefined roles in associations). The objective was to study the behaviour of RCA on real-sized class models, so as to draw conclusions on its application, in particular, a major concern was the scalability of the approach, since RCA may generate too many very abstract concepts in certain cases. The experiment shows that taking into account the navigability, the results remain possible to analyze, the explosion of concepts does not appear. Consequently, RCA can be considered to scale to real-size models, if it is adequately parameterized. However, the produced results remains quite large to analyze, and new strategies can be settled to face the number of concepts to analyze. The main strategy we will explore is an incremental exploration approach: instead of presenting to the expert the final result, with hundreds of concepts to analyze, the abstractions will be presented step by step to the expert, so as to "cut" branches of abstractions that are not relevant.

References

[1] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundation*. Springer-Verlag Berlin, 1999.

[2] S. O. Kuznetsov, "On stability of a formal concept," *Ann. Math. Artif. Intell.*, vol. 49, no. 1-4, pp. 101–115, 2007.

[3] R. Godin and H. Mili, "Building and maintaining analysis-level class hierarchies using Galois lattices," in *Proceedings of OOPSLA'93, Washington (DC), USA*, ser. special issue of ACM SIGPLAN Notices, 28(10), 1993, pp. 394–410.

[4] R. Godin, H. Mili, G. Mineau, R. Missaoui, A. Arfi, and T. Chau, "Design of Class Hierarchies Based on Concept (Galois) Lattices," *Theory And Practice of Object Systems*, vol. 4, no. 2, 1998.

[5] H. Dicky, C. Dony, M. Huchard, and T. Libourel, "On automatic class insertion with overloading," in *Proceedings of OOPSLA'96, San Jose (CA), USA*, ser. special issue of ACM SIGPLAN Notices, 31(10), 1996, pp. 251–267.

[6] M. Missikoff and M. Scholl, "An Algorithm for Insertion into a Lattice: Application to Type Classification," *Proceedings of the 3rd Int. Conf. FODD'89*, pp. 64–82, 1989.

[7] E. A. Rundensteiner, "A Class Classification Algorithm For Supporting Consistent Object Views," University of Michigan, Tech. Rep., 1992.

[8] A. Yahia, L. Lakhal, R. Cicchetti, and J. Bordat, "iO2, An Algorithmic Method for Building Inheritance Graphs in Object Database Design," in *Proceedings of the 15th International Conference on Conceptual Modeling ER'96*, 1996.

[9] M. Huchard, C. Roume, and P. Valtchev, "When concepts point at other concepts: the case of uml diagram reconstruction," in *FCAKDD 2002, Advances in Formal Concept Analysis for Knowledge Discovery in Databases, Int. workshop ECAI 2002*, V. Duquenne, B. Ganter, M. Liquire, E. M. Nguifo, and G. Stumme, Eds., Lyon, juillet 2002, pp. 32–43.

[10] M. Dao, M. Huchard, M. R. Hacene, C. Roume, and P. Valtchev, "Improving generalization level in uml models iterative cross generalization in practice," in *ICCS*, ser. Lecture Notes in Computer Science, K. E. Wolff, H. D. Pfeiffer, and H. S. Delugach, Eds., vol. 3127. Springer, 2004, pp. 346–360.

[11] C. Roume, "Analyse et restructuration de hiérarchies de classes," Ph.D. dissertation, Université Montpellier 2, 2004.

[12] M. R. Hacène, "Relational concept analysis, application to software re-engineering," Ph.D. dissertation, Universit du Québec Montréal, 2005.

[13] J.-R. Falleri, M. Huchard, and C. Nebut, "A generic approach for class model normalization," in *ASE*. IEEE, 2008, pp. 431–434.

[14] J.-R. Falleri, "Contributions à l'IDM : reconstruction et alignement de modèles de classes," Ph.D. dissertation, Université Montpellier 2, 2009.

Index

Bibliographie

ALSHAYEB, M. et LI, W. (2003). An empirical validation of object-oriented metrics in two different iterative software processes. *Software Engineering, IEEE Transactions on*, 29(11):1043–1049. Cité page 26.

ALTMANNINGER, K., SCHWINGER, W. et KOTSIS, G. (2010). Semantics for accurate conflict detection in smover : Specification, detection and presentation by example. *International Journal of Enterprise Information Systems*, 6(1):68–84. Cité page 150.

ALTMANNINGER, K., SEIDL, M. et WIMMER, M. (2009). A survey on model versioning approaches. *International Journal of Web Information Systems*, 5(3):271–304. Cité page 150.

AMAR, B., OSMAN GUÉDI, A., MIRALLES, A., HUCHARD, M., LIBOUREL, T. et NEBUT, C. (2012). Using Formal Concept Analysis to Extract a Greatest Common Model. *In* MACIASZEK, L. A., CUZZOCREA, A. et CORDEIRO, J., éditeurs : *ICEIS 2012 : 14th International Conference on Enterprise Information Systems*, volume 1, pages 27–37, Wroclaw, Pologne. SciTePress. Cité page 151.

A. MIRALLES et LIBOUREL, T. (2009). Application of a model transformation paradigm in agriculture : A simple environmental system case study. *In Advances in Modelling Agricultural Systems*, volume 25, pages 37–54. Springer. Cité page 74.

ARÉVALO, G., FALLERI, J.-R., HUCHARD, M. et NEBUT, C. (2006). Building abstractions in class models : Formal concept analysis in a model-driven approach. *In MoDELS*, pages 513–527. Cité pages 52 et 57.

ARNOLD ROCHFELD, J. M. (1989). *Merise*. Paris : les Ed. d'Organisation. Mémentos E.O., ISSN 0764-9851. Cité page 13.

AZMEH, Z., HUCHARD, M., TIBERMACINE, C., URTADO, C. et VAUTTIER, S. (2008). WSPAB : A Tool for Automatic Classification and Selection of Web Services Using Formal Concept Analysis. *In ECOWS'08 : The 6th IEEE European Conference on Web Services*, page 10, Irlande. Cité page 38.

BANSIYA, J., DAVIS, C. et ETZKORN, L. (1999). An entropy-based complexity measure for object-oriented designs. *Theory and Practice of Object Systems*, 5(2):111–118. Cité page 25.

BARBUT, M. et MONJARDET, B. (1970). *Ordre et classification : algèbre et combinatoire*, volume 2. Hachette Paris. Cité pages 7, 30 et 155.

BASILI, V. R., BRIAND, L. C. et MELO, W. L. (1996). A validation of object-oriented design metrics as quality indicators. *Software Engineering, IEEE Transactions on*, 22(10):751–761. Cité page 24.

BATINI, C., LENZERINI, M. et NAVATHE, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computer Survey*, 18:323–364. Cité page 150.

BECK, K. (2000). extreme programming explained : embrace change addison-wesley. *Reading, Ma*. Cité page 2.

BECK, K. et ANDRES, C. (2004). *Extreme programming explained : embrace change*. Addison-Wesley Professional. Cité page 21.

BELLMAN, R. (1958). On a routing problem. *In Quarterly of Applied Mathematics*, volume 16, page 87–90. Quarterly of Applied Mathematics. Cité page 122.

BÉNARD, J.-L., BOSSAVIT, L., MÉDINA, R. et WILLIAMS, D. (2002). L'extreme programming. *Paris : Eyrolles*. Cité pages 2 et 21.

BIRKHOFF, G. (1940). *Lattice theory*. American Mathematical Society. Cité pages 7, 30, 136 et 155.

BOEHM, B. (2006). A view of 20th and 21st century software engineering. *In Proceedings of the 28th international conference on Software engineering*, pages 12–29. ACM. Cité page 12.

BOEHM, B. et TURNER, R. (2004). Balancing agility and discipline : evaluating and integrating agile and plan-driven methods. *In Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 718–719. IEEE. Cité page 2.

BOEHM, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5): 61–72. Cité page 19.

BOOCH, G., RUMBAUGH, J. et JACOBSON, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley. Cité page 105.

BOOCH, G., RUMBAUGH, J. et JACOBSON, I. (2000). *UML : Le guide de l'utilisateur*. Technologies objet. Référence. Eyrolles. Cité page 13.

BOUZEGHOUB, M. et COMYN-WATTIAU, I. (1990). View integration by semantic unification and transformation of data structures. *In ER*, pages 413–430. Cité page 150.

BRIAND, L. C., MORASCA, S. et BASILI, V. (1996). Property-based software engineering measurement. *Transactions on Software Engineering*, 22(1):68–86. Cité page 25.

CAPRILE, C. et TONELLA, P. (1999). Nomen est omen : Analyzing the language of function identifiers. *In Reverse Engineering, 1999. Proceedings. Sixth Working Conference on*, pages 112–122. IEEE. Cité page 37.

carpers JONES (1991). *Applied software measurement : assuring productivity and quality*. McGraw-Hill. Cité page 23.

CHERFI, S. S.-S., AKOKA, J. et COMYN-WATTIAU, I. (2003). Conceptual modeling quality-from eer to uml schemas evaluation. pages 414–428. Cité page 24.

CHIDAMBER, S. R. et KEMERER, C. F. (1994a). A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493. Cité page 24.

CHIDAMBER, S. R. et KEMERER, C. F. (1994b). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493. Cité page 24.

CICCHETTI, A., RUSCIO, D. et PIERANTONIO, A. (2008). Managing model conflicts in distributed development. *In Model Driven Engineering Languages and Systems (MoDELS)*, pages 311–325. Cité page 150.

CICCHETTI, A., RUSCIO, D. D. et PIERANTONIO, A. (2007). A metamodel independent approach to difference representation. *Journal of Object Technology*, 6(9):165–185. Cité page 150.

CODD, E. F. (1970). A relational model for large shared data banks. *In Comm. of ACM 13(6)*. Cité page 35.

COMYN-WATTIAU, I., AKOKA, J. et BERTI-EQUILLE, L. (2010). La qualité des systèmes d'information : Vers une vision plus intégrée. *Ingénierie des systèmes d'information*, 15(6):9–32. Cité page 25.

CYRIL ROUME (2004). *Analyse et restructuration de hiérarchies de classes*. Thèse de doctorat, Université Montpellier 2. Cité pages 39 et 40.

DAO, M., HUCHARD, M., HACENE, M. R., ROUME, C. et VALTCHEV, P. (2004a). Improving generalization level in uml models iterative cross generalization in practice. *In ICCS*, pages 346–360. Cité page 39.

DAO, M., HUCHARD, M., LIBOUREL, T., ROUME, C. et LEBLANC, H. (2002). A new approach to factorization - introducing metrics. *In IEEE METRICS*, pages 227–236. Cité page 26.

DAO, M., HUCHARD, M., ROUANE, M. H., ROUME, C. et VALTCHEV, P. (2004b). Improving Generalization Level in UML Models : Iterative Cross Generalization in Practice. *In Proc. of the 12th Intl. Conf. on Conceptual Structures (ICCS'04)*, volume 3127 de *LNCS*, pages 346–360, Huntsville, AL. SV. Cité page 52.

DICKY, H., DONY, C., HUCHARD, M. et LIBOUREL, T. (1996). On automatic class insertion with overloading. *In oopsla96*, siacmsn no31-10, pages 251–267. Cité page 39.

DOAN, A. et HALEVY, A. Y. (2005). Semantic integration research in the database community : A brief survey. *AI Magazine*, 26:83–94. Cité page 151.

DOLQUES, X., FALLERI, J.-R., HUCHARD, M. et NEBUT, C. (2008). A model-driven engineering based RCA process for bi-level models elements/meta-elements - application to description logics. *In CLA'08 : 6th International Conference on Concept Lattices and Their Applications*, pages 21–23. Cité page 57.

DOLQUES, X., HUCHARD, M. et NEBUT, C. (2009). Génération de transformation de modèles par application de l'ARC sur des exemples. *In LMO'09 : Langages et Modèles à Objets*, pages 61–75. Cité page 57.

DOWSON, M. (1987). Iteration in the software process ; review of the 3rd international software process workshop. *In Proceedings of the 9th international conference on Software Engineering*, pages 36–41. IEEE Computer Society Press. Cité page 17.

e ABREU, F. B. et CARAPUÇA, R. (1994). Candidate metrics for object-oriented software within a taxonomy framework. *Journal of Systems and Software*, 26(1):87–96. Cité page 25.

EGYED, A. et KRUCHTEN, P. B. (1999). Rose/architect : a tool to visualize architecture. *In System Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, pages 10–pp. IEEE. Cité page 2.

FALLERI, J. (2009). *Contributions à l'IDM. : reconstruction et alignement de modèles de classes*. Thèse de doctorat, Ph. D. thesis (written in French), University of Montpellier 2. Cité pages 5, 27, 39, 41, 52, 57 et 82.

FALLERI, J.-R. (2010). Eclipse's Relational Concept Analysis ; http ://erca.googlecode.com. Cité pages 66 et 157.

FALLERI, J.-R., HUCHARD, M. et NEBUT, C. (2008). A generic approach for class model normalization. *In ASE*, pages 431–434. Cité pages 39 et 41.

FALLERI, J.-R., HUCHARD, M., NEBUT, C. et AND, G. A. (2007). A model driven engineering approach for making generic fca/rca tools. *In Proceedings of the Fifth International Conference on Concept Lattices and Their Applications, CLA 07*, volume 331, pages 229–240. CEUR. Cité page 52.

FAVRE, J.-M., ESTUBLIER, J. et BLAY-FORNARINO, M. (2006). L'ingénierie dirigée par les modèles. au-delà du mda (traité ic2, série informatique et systèmes d'information). Cité pages 13, 14 et 196.

FELLBAUM, C. (2010). *WordNet*. Springer. Cité page 162.

FENTON, N. E. et NEIL, M. (1999). A critique of software defect prediction models. *Software Engineering, IEEE Transactions on*, 25(5):675–689. Cité page 23.

FENTON, N. E. et PFLEEGER, S. L. (1991). *Software metrics*, volume 1. Chapman & Hall London. Cité page 23.

GALL, H., JAZAYERI, M. et RIVA, C. (1999). Visualizing software release histories : The use of color and third dimension. *In ICSM*, pages 99–108. Cité page 26.

GANTER, B., STUMME, G. et WILLE, R., éditeurs (2005). *Formal Concept Analysis, Foundations and Applications*, volume 3626 de *Lecture Notes in Computer Science*. Springer. Cité page 107.

GANTER, B. et WILLE, R. (1999). *Formal Concept Analysis, Mathematical Foundations*. Springer-Verlag. Cité pages 5, 7, 30, 81, 136 et 155.

GODIN, A. et VALTCHEV, P. (2005). Formal concept analysis-based class hierarchy design in object-oriented software development. *In Formal Concept Analysis*, pages 304–323. Cité pages 35, 89 et 105.

GODIN, R. et MILI, H. (1993). Building and maintaining analysis-level class hierarchies using galois lattices. *In OOPSLA*, pages 394–410. Cité pages 39, 52 et 81.

GODIN, R., MILI, H., MINEAU, G., MISSAOUI, R., ARFI, A. et CHAU, T. (1998). Design of Class Hierarchies Based on Concept (Galois) Lattices. *Theory And Practice of Object Systems*, 4(2). Cité page 39.

GROUP, S. *et al.* (2001). Extreme chaos. *The Standish Group International Inc. http ://www. standishgroup. com/chaos. html*, pages 1–12. Cité page 2.

GUIMOND, L.-É. (2005). Conception d'un environnement de découverte des besoins pour le développement de solutions solap. *Résumé*, page 01. Cité pages 2 et 3.

GYIMÓTHY, T., FERENC, R. et SIKET, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Trans. Software Eng.*, 31(10):897–910. Cité page 25.

HACÈNE, M. R. (2005). *Relational concept analysis, application to software re-engineering*. Thèse de doctorat, Université du Québec à Montréal. Cité pages 39 et 40.

HACENE, M. R., DAO, M., HUCHARD, M. et VALTCHEV, P. (2007a). Analyse formelle de données relationnelles pour la réingénierie des modèles uml. *In LMO*, pages 151–166. Cité pages 52, 53 et 57.

HACENE, M. R., DAO, M., HUCHARD, M. et VALTCHEV, P. (2007b). Analyse formelle de données relationnelles pour la réingénierie des modèles UML. *In Langages et modèles à objets (LMO'2007)*, pages 151–166, Toulouse,France. Hermes. Cité page 107.

HACENE, M. R., HUCHARD, M., NAPOLI, A. et VALTCHEV, P. (2013). Relational concept analysis : mining concept lattices from multi-relational data. *Ann. Math. Artif. Intell.*, 67(1):81–108. Cité page 30.

HIGHSMITH, J. A. (2002). *Agile software development ecosystems*. Addison-Wesley Professional. Cité page 2.

HUCHARD, M. (2003). *Classification de classes dans les approches à objets, algorithmes et applications*. Mémoire d'habilitation à diriger des recherches, avril 2003, Université Montpellier 2. 154 pages. Cité pages 5, 7, 52 et 107.

HUCHARD, M., HACENE, M. R., ROUME, C. et VALTCHEV, P. (2007). Relational concept discovery in structured datasets. *Ann. Math. Artif. Intell.*, 49(1-4):39–76. Cité page 81.

JACOBSON, I., BOOCH, G. et RUMBAUGH, J. (1999). The unified software development process–the complete guide to the unified process from the original designers. *Rational Software Corporation, US*. Cité page 2.

KAY, R. (2002). Quickstudy : System development life cycle. *Computerworld*, 14. Cité page 18.

KELTER, U., WEHREN, J. et NIERE, J. (2005). A generic difference algorithm for uml models. *Software Engineering*, 64(105-116):4–9. Cité page 148.

KENT, S. (2002). Model driven engineering. *In Integrated formal methods*, pages 286–298. Springer. Cité pages 12, 28 et 155.

KLEPPE, A. G., WARMER, J. B. et BAST, W. (2003). *MDA explained, the model driven architecture : Practice and promise*. Addison-Wesley Professional. Cité pages 13 et 14.

KPODJEDO, S., RICCA, F., GALINIER, P., GUÉHÉNEUC, Y.-G. et ANTONIOL, G. (2011). Design evolution metrics for defect prediction in object oriented systems. *Empirical Software Engineering*, 16(1): 141–175. Cité page 26.

LANZA, M. et DUCASSE, S. (2002). Understanding software evolution using a combination of software visualization and software metrics. *L'OBJET*, 8(1-2):135–149. Cité page 26.

LARMAN, C. (2002). *Applying UML and Patters : An introduction to Object-oriented analysis and design and the Unified Process*, volume 130925691. Prentice Hall, ISBN. Cité page 17.

LORENZ, M. et KIDD, J. (1994). *Object-Oriented Software Metrics : A Practical Guide*. Prentice-Hall. Cité page 24.

MARTIN, J. (1991). *Rapid application development*. Macmillan Publishing Co., Inc. Cité page 2.

MCCABE, T. J. (1976). A complexity measure. *Software Engineering, IEEE Transactions on*, 4(4):308–320. Cité page 25.

MCDERMID, J. A. et RIPKEN, K. (1984). *Life cycle support in the ADA environment*. CUP Archive. Cité page 17.

MÉTAIS, E., KEDAD, Z., COMYN-WATTIAU, I. et BOUZEGHOUB, M. (1997). Using linguistic knowledge in view integration : toward a third generation of tools. *Data & knowledge engineering*, 23(1):59–78. Cité page 150.

MILLER, B. K., HSIA, P. et KUNG, D. C. (1999). Object-oriented architecture measures. *In HICSS*. Cité page 25.

MILLER, J. et MUKERJI, J. (2003). Mda guide version 1.0. 1 omg. Cité page 15.

MIRALLES, A. (2006). *Ingénierie des modèles pour les applications environnementales*. Thèse de doctorat, I2S. Cité pages 2, 47 et 72.

MIRALLES, A., GORRETTA, N., MILLER, P. C., WALKLATE, P., VAN ZUYDAM, R. P., PORSKAMP, H. A., GANZELMEIER, H., RIETZ, S., ADE, G., BALSARI, P., VANNUCCI, D. et PLANAS, S. (1994). Orchard sprayers : an european program to compare testing methods. *In International symposium on fruit nut and vegetable production production engineering, Valencia Zaragoza, ESP, 22-26 mars 1993*, pages 117–122. Cité page 151.

MIRALLES, A., GOUY, V., VERNIER, F., LAUVERNET, C., PINET, F., CARLUER, N., BERNARD, S., PETIT, K., FAIDIX, K., MOLLA, G., CHANET, J.-P., SOUSA, G. D. et BIMONTE, S. (2009). Un sie pesticides pour la réduction de l'impact des produits phytosanitaires sur l'environnement - rapport 1° année. Rapport scientifique interne, Istea. Cité page 48.

MIRALLES, A. et LIBOUREL, T. (2006). Cycle de développement revisité par mda. . Cité page 21.

MIRALLES, A. et LIBOUREL, T. (2009). A new methodology to automate the transformation of gis models in an iterative development process. *In* SPRINGER, éditeur : *Advances in Modelling Agricultural Systems*, volume 25, pages 19–36. Papajorgji, Petraq - Pardalos, Panos. Cité pages 21 et 74.

MIRALLES, A., PINET, F., CARLUER, N., VERNIER, F., BIMONTE, S., LAUVERNET, C. et GOUY, V. (2011). EIS-Pesticide : an information system for data and knowledge capitalization and analysis. *In Euraqua-PEER Scientific Conference, 26/10/2011 - 28/10/2011*, page 1, Montpellier, FRA. Cité pages 132 et 151.

MIRALLES, A. et POLVÊCHE, V. (1998). Effects of the agrochemical products and adjuvants on spray quality and drift potential. *In 5th International Symposium on Adjuvants for Agrochemicals - ISAA '98*, volume 1, pages 426–432, Memphis (USA). Cité page 151.

MIRALLES, A., VERNIER, F., CARLUER, N., PINET, F., FAIDIX, K., LAUVERNET, C., MOLLA, G., PETIT, K., BERNARD, S., GOUY, V., BIMONTE, S., SOUSA, G. D. et CHANET, J.-P. (2010). Un sie pesticides pour la réduction de l'impact des produits phytosanitaires sur l'environnement - rapport 2° année. Rapport scientifique interne, Istea. Cité page 48.

MISSIKOFF, M. et SCHOLL, M. (1989). An Algorithm for Insertion into a Lattice : Application to Type Classification. *Proceedings of the 3rd Int. Conf. FODD'89*, pages 64–82. Cité page 39.

MOHA, N. (2008). *DECOR : Détection et correction des défauts dans les systemes orientés objet*. Thèse de doctorat, Université des Sciences et Technologie de Lille-Lille I. Cité page 38.

MOHA, N., HACENE, A. M. R., VALTCHEV, P. et GUÉHÉNEUC, Y.-G. (2008a). Refactorings of design defects using relational concept analysis. *In Formal Concept Analysis*, pages 289–304. Springer. Cité page 38.

MOHA, N., REZGUI, J., GUÉHÉNEUC, Y.-G., VALTCHEV, P. et EL BOUSSAIDI, G. (2008b). Using fca to suggest refactorings to correct design defects. *In Concept Lattices and Their Applications*, pages 269–275. Springer. Cité page 38.

MONK, S. et SC, S. M. B. (1993). A model for schema evolution in object-oriented database systems. Cité page 27.

NAGAPPAN, N. et BALL, T. (2005). Use of relative code churn measures to predict system defect density. *In ICSE*, pages 284–292. Cité page 26.

OHST, D., WELLE, M. et KELTER, U. (2003). Differences between versions of uml diagrams. *SIGSOFT Software Engineering Notes*, 28:227–236. Cité page 150.

OSMAN GUÉDI, A., MIRALLES, A., AMAR, B., NEBUT, C., HUCHARD, M. et LIBOUREL, T. (2012). How Relational Concept Analysis Can Help to Observe the Evolution of Business Class Models. *In* LASZLO SZATHMARY, U. P., éditeur : *CLA'2012 : 9th International Conference on Concept Lattices and Applications*, pages 139–150, Fuengirola (Málaga), Espagne. Universidad de Malaga. Cité pages 95, 107, 109, 115 et 119.

OSMAN GUÉDI, A., MIRALLES, A., NEBUT, C. et HUCHARD, M. (2011). Analyse de l'évolution d'un modèle : vers une méthode basée sur l'analyse formelle de concepts. *In XXIXème Congrès INFORSID*. Cité pages 107 et 148.

PARENT, C. et SPACCAPIETRA, S. (1998). Issues and approaches of database integration. *Communication of the ACM*, 41:166–178. Cité page 150.

PERRENOUD, P. (1994). Travailler en équipe pédagogique, c'est partager sa part de folie. *Cahiers pédagogiques*, 325:68–71. Cité page 2.

PINET, F., MIRALLES, A., BIMONTE, S., VERNIER, F., CARLUER, N., GOUY, V. et BERNARD, S. (2010). The use of uml to design agricultural data warehouses. *In International Conference on Agricultural Engineering (AgEng 2010)*, pages 1–10. Cité page 132.

PINZGER, M., GALL, H., FISCHER, M. et LANZA, M. (2005). Visualizing multiple evolution metrics. *In SOFTVIS*, pages 67–75. Cité page 26.

PRÉGENT, R. (1990). Quelques habiletés de communication" and "la formation des étudiants au travail en équipe". *La préparation d'un cours*, pages 196–202. Cité page 2.

RAHM, E. et BERNSTEIN, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–350. Cité page 150.

ROIDOR, H. et MAUVILAIN, M. (2007). Le rapport de mission (rapport roidor). Rapport technique, Académie de Bordeaux et Ministère de l'Éducation nationale et de la Formation professionnelle de la République de Djibouti. Cité page 50.

ROUME, C. (2004). *Analyse et restructuration de hiérarchies de classes*. Thèse de doctorat, PhD thesis, Université Montpellier 2. Cité page 5.

ROYCE, W. W. (1970). Managing the development of large software systems. *In proceedings of IEEE WESCON*, volume 26, page 8. Los Angeles. Cité pages 2, 16 et 17.

RUNDENSTEINER, E. A. (1992). A Class Classification Algorithm For Supporting Consistent Object Views. Rapport technique, University of Michigan. Cité page 39.

SCHRAMM, W. et ROBERTS, D. F. (1971). The process and effects of mass communication. *Dans The Process and Effects of Communication, Urbana (Illinois). University of Illinois Press*, pages 3–26. Cité page 3.

SCHWABER, K. et BEEDLE, M. (2001). Agile software development with scrum. 2001. *Upper Saddle River, NJ*. Cité page 2.

SELIC, B. (2003). The pragmatics of model-driven development. *Software, IEEE*, 20(5):19–25. Cité page 12.

SHVAIKO, P. et EUZENAT, J. (2005). A Survey of Schema-Based Matching Approaches Journal on Data Semantics IV. *In* SPACCAPIETRA, S. et SPACCAPIETRA, S., éditeurs : *Journal on Data Semantics IV*, volume 3730 de *Lecture Notes in Computer Science*, chapitre 5, pages 146–171. Springer Berlin / Heidelberg, Berlin, Heidelberg. Cité page 150.

SIFF, M. et REPS, T. (1999). Identifying modules via concept analysis. *Software Engineering, IEEE Transactions on*, 25(6):749–768. Cité page 37.

SKARRA, A. H., ZDONIK, S. B. et UNIVERSITP, B. (1986). The management of changing types in an object-oriented database. *In In Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA*. Cité page 27.

SNELTING, G. (1998). Concept analysis—a new framework for program understanding. *In ACM SIGPLAN Notices*, volume 33, pages 1–10. ACM. Cité page 37.

STUMME, G. et MAEDCHE, A. (2001). Ontology merging for federated ontologies on the semantic web. *In International Workshop for Foundations of Models for Information Integration (FMII-2001)*, pages 413–418. Cité page 151.

VICKOFF, J.-P. (2003). *Systèmes d'information et processus agiles*. Hermès science publ. Cité page 20.

WILLE, R. (1982a). Restructuring lattice theory : An approach based on hierarchies of concepts. *In* RIVAL, I., éditeur : *Ordered Sets*, pages 445–470. Reidel, Dordrecht-Boston. Cité page 30.

WILLE, R. (1982b). Restructuring the lattice theory : An approach based on hierarchies of concepts. *In* RIVAL, I., éditeur : *Ordered sets*, pages 445–470, Dordrecht-Boston. Reidel. Cité page 107.

XING, Z. et STROULIA, E. (2005). Umldiff : an algorithm for object-oriented design differencing. *In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 54–65. ACM. Cité page 148.

YAHIA, A., LAKHAL, L., CICCHETTI, R. et BORDAT, J. (1996). iO2, An Algorithmic Method for Building Inheritance Graphs in Object Database Design. *In Proceedings of the 15th International Conference on Conceptual Modeling ER'96*. Cité page 39.

YI, T., WU, F. et GAN, C. (2004). A comparison of metrics for uml class diagrams. *ACM SIGSOFT Software Engineering Notes*, 29(5):1–6. Cité page 23.

ZUSE, H. (1998). *A framework of software measurement*. Walter de Gruyter. Cité page 23.

Table des figures

Liste des tableaux

Listings

Abstract

The rapidly changing needs among other things due to technical innovation, competition and regulation often leads to describe the context for the study of conceptual models in information systems to facilitate the evolution of operating systems. The development of these models is carried out in several phases during which several working teams of different nature, providing each participant's perception of the system to be built is limited to the part of his area of specialization. It must then reconcile the different perceptions.

The main objective of the thesis is to design mechanisms to obtain a share of the model factoring concepts common to several models and, secondly, to provide designers with a methodology for monitoring the evolution of factorization.

To perform the factorization, we have implemented the Formal Concept Analysis (FCA) and Relational Concepts Analysis (RCA), which are methods of analysis based on the theory of lattice data. In a set of entities described by features, both methods extract formal concepts that combine a maximum of entities to a maximum set of shared characteristics together. These formal concepts are structured in a partial order of specialization that provides with a lattice structure. The RCA can complement the description of the entities by relationships between entities.

The first contribution of the thesis is a **method a model for analyzing the evolution of the factorization based on the FCA and the RCA**. This method builds the capacity of the AFC and the RCA to emerge in a model of thematic abstractions higher level, improving semantic models. We show that these methods can also be used to monitor the analytical process with stakeholders. We introduce metrics on the design elements and the concept lattices which are the basis for the development of recommendations. We conduct an experiment in which we study the evolution of the 15 versions of the model class of information-Pesticides EIS system.

The second contribution of this thesis is a **depth study of the behavior of the RCA on UML models.** We show the influence of model structure on different variables studied (such as execution time and memory used) through several experiments on 15 versions of the EIS-Pesticides model. For this, we study several configurations (choice of elements and relations in the meta-model) and several parameters (choice of using unnamed elements, choice of using airworthiness). Metrics are introduced to guide the designer in managing the process of factoring and recommendations on the preferred configurations and settings are made.

The last contribution is a **approach to inter-model factorization** to group in a model all the concepts common to different source models designed by different experts. In addition to the consolidation of common concepts, this analysis produces new abstractions generalizing existing thematic concepts. We apply our approach on 15 versions of the model EIS-Pesticides.

All this work is part of a research framework which aims to factor thematic concepts within a model and control metrics by the profusion of concepts produced by the FCA and especially by RCA.

Keywords: *Factorization, Evolution, Processing, Modeling, Model Driven Engineering (MDE), Unified Modeling Language (UML), Lattices, Formal Concept Analysis (FCA), Relational Concepts Analysis (RCA), Meta-modeling, metrics, Profile, Model Driven Architecture (MDA).*

Université de Djibouti — Avenue Georges Clemenceau — BP: 1904 — République de Djibouti
TETIS — Maison de la télédétection – 500, rue J.-F. Breton – 34093 Montpellier cedex 5 — France
LIRMM — Université Montpellier 2 — CNRS — 161, rue Ada — 34095 Montpellier cedex 5 — France

Résumé

L'évolution rapide des besoins dus entre autres à l'innovation technique, la concurrence ou la réglementation conduit souvent à décrire le cadre d'étude des systèmes d'information dans des modèles conceptuels, pour faciliter l'évolution du fonctionnement des systèmes. La mise au point de ces modèles s'effectue en plusieurs phases au cours desquelles collaborent plusieurs équipes de nature différente, chaque intervenant apportant sa perception du système à construire en se limitant à la partie de son domaine de spécialisation. Il faut alors concilier les différentes perceptions.

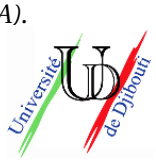
L'objectif essentiel de la thèse est de concevoir les mécanismes permettant d'une part d'obtenir le modèle factorisant les concepts communs à plusieurs modèles et, d'autre part, de proposer aux concepteurs une méthodologie de suivi de l'évolution de la factorisation. Pour réaliser la factorisation, nous avons mis en œuvre l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts (ARC) qui sont des méthodes d'analyse de données basées sur la théorie des treillis. Dans un ensemble d'entités décrites par des caractéristiques, les deux méthodes extraient des concepts formels qui associent un ensemble maximal d'entités à un ensemble maximal de caractéristiques partagées. Ces concepts formels sont structurés dans un ordre partiel de spécialisation qui les munit d'une structure de treillis. L'ARC permet de compléter la description des entités par des relations entre entités.

La première contribution de la thèse est une **méthode d'analyse de l'évolution de la factorisation d'un modèle basée sur l'AFC et l'ARC**. Cette méthode s'appuie la capacité de l'AFC et de l'ARC à faire émerger au sein d'un modèle des abstractions thématiques de niveau supérieur, améliorant ainsi la sémantique des modèles. Nous montrons que ces méthodes peuvent aussi être employées pour suivre l'évolution du processus d'analyse avec des acteurs. Nous introduisons des métriques sur les éléments de modélisation et sur les treillis de concepts qui servent de base à l'élaboration de recommandations. Nous effectuons une expérimentation dans laquelle nous étudions l'évolution des 15 versions du modèle de classes du système d'information SIE-Pesticides.

La seconde contribution de la thèse est une **étude approfondie du comportement de l'ARC sur des modèles UML**. Nous montrons l'influence de la structure des modèles sur différentes variables étudiées (comme les temps d'exécution et la mémoire occupée) au travers de plusieurs expérimentations sur les 15 versions du modèle SIE-Pesticides. Pour cela, nous étudions plusieurs configurations (choix d'éléments et de relations dans le méta-modèle) et plusieurs paramètres (choix d'utiliser les éléments non nommés, choix d'utiliser la navigabilité). Des métriques sont introduites pour guider le concepteur dans le pilotage du processus de factorisation et des recommandations sur les configurations et paramétrages à privilégier sont faites.

La dernière contribution est une **approche de factorisation inter-modèles** afin de regrouper au sein d'un modèle l'ensemble des concepts communs à différents modèles sources conçus par différents experts. Outre le regroupement des concepts communs, cette analyse produit de nouvelles abstractions généralisant des concepts thématiques existants. Nous appliquons notre approche sur les 15 versions du modèle du SIE-Pesticides.

Mots clefs : *Factorisation, Évolution, Transformation, Modélisation, Ingénierie Dirigée par les Modèles (IDM), Unified Modeling Language (UML), Treillis, Analyse Formelle de Concepts (AFC), Analyse Relationnelle de Concepts (ARC), Méta-modélisation, Métriques, Profil, Model Driven Architecture (MDA).*



Université de Djibouti — Avenue Georges Clemenceau — BP: 1904 — République de Djibouti
TETIS — Maison de la télédétection – 500, rue J.-F. Breton – 34093 Montpellier cedex 5 — France
LIRMM — Université Montpellier 2 — CNRS — 161, rue Ada — 34095 Montpellier cedex 5 — France