Correction des défauts de généralisation dans les diagrammes de cas d'utilisation UML

X. Dolques, L. M. Hakik, M. Huchard, C. Nebut, P. Reitz

LIRMM, Univ Montpellier 2 et CNRS
161 rue Ada, 34395 MONTPELLIER CEDEX 5
{dolques, hakik, huchard, nebut, reitz}@lirmm.fr,

RÉSUMÉ. *Dans le contexte d'un processus de développement basé sur la notation* UML*, les diagrammes de cas d'utilisation et les modèles ou documents qui les accompagnent pour traduire les besoins utilisateurs contribuent à formaliser la définition des besoins. Ils guident également toutes les étapes de construction du système depuis l'établissement du cahier des charges jusqu'à la réalisation des dossiers de test. Nous étudions la faisabilité d'une approche permettant la correction de certains défauts des diagrammes de cas d'utilisation* UML *(liés à une mauvaise utilisation des concepteurs) par introduction d'acteurs et de cas plus généraux, ainsi que de relations de spécialisation/généralisation appropriées. Les différents cas de refactorisation par introduction de généralisations sont considérés de manière systématique, en utilisant la structure du diagramme et une segmentation des noms des cas. Cette étude débouche sur une utilisation de l'Analyse Formelle de Concepts pour traiter la refactorisation de manière globale. Une chaîne d'outils permet de partir d'un diagramme de cas d'utilisation* UML *et d'analyser différentes généralisations possibles qui aboutissent le plus souvent à une simplification du diagramme.*

ABSTRACT. *During a software development process using the* UML *modeling language, use case diagrams, as well as their companions (documents and other dynamic models), help in describing the user requirements and guiding the whole system construction. We study the feasability of an approach that aims at fixing some of the defaults of the use case diagrams. More general actors and use cases are introduced, while suitable specialization/generalization relations are added. Refactoring schemas are proposed, using the diagram structure and a basic analysis of use case identifiers. In order to deal with refactoring in a global way, we introduce an approach based on Formal Concept Analysis. A tool chain implements our proposal.*

MOTS-CLÉS : *Diagrammes de cas d'utilisation, refactorisation, Analyse Formelle de Concepts*

KEYWORDS: *Use case diagrams, refactoring, Formal Concept Analysis*

1. Introduction

Dans le cadre d'un processus de développement de mieux en mieux encadré par des méthodologies, des modèles et des outils, les cas d'utilisation [JAC 92] permettent de recenser et de structurer les besoins des utilisateurs et les scénarios d'utilisation. Différentes variantes en ont été proposées, parmi lesquelles l'approche d'A. Cockburn qui oriente la définition par les buts [COC 97a, COC 97b] et définit un patron de description de cas d'utilisation précis [COC 00]. Les diagrammes de cas d'utilisation d'UML [OMG09] tâchent de formaliser une partie des informations relatives aux cas d'utilisation. On peut leur reprocher certaines ambiguïtés (comme l'absence de nom pour les rôles des acteurs), le manque de place pour la description textuelle [HEN 02] ou pour les aspects chronologiques, et le peu de formalisation des relations liant les cas d'utilisation entre eux [GEN 02]. Ils présentent cependant l'intérêt d'offrir une vue graphique, plutôt basée sur la logique d'inclusion des cas les uns dans les autres, qui constitue une brique pertinente de la définition des cas d'utilisation. Leur réalisation demande un soin particulier, d'autant plus qu'ils vont orienter une grande partie de la conception du système. Comme les autres diagrammes, ils peuvent être sujets à toutes sortes de défauts tels qu'une mauvaise utilisation des relations, un nommage discutable des cas, des relations manquantes, une absence de structuration, etc. Certains défauts portent plus précisément sur la généralisation : faible structuration des liaisons entre les acteurs et les rôles, manque de spécialisation entre les acteurs, complexité due à l'absence de cas plus généraux, etc. Ainsi que mentionné dans le manuel d'utilisation d'UML [BOO 05], « organiser des cas d'utilisation en extrayant des comportements communs (via des relations d'inclusion) et distinguer des variantes (via des relations d'extension) est un point important pour créer pour un système donné un ensemble de cas d'utilisation qui soit simple, équilibré et compréhensible. »

Nous nous concentrons dans cet article sur la découverte d'acteurs ou de cas par abstraction des éléments existants, sur l'introduction de relations de spécialisation/généralisation entre cas ou entre acteurs, ainsi que sur la factorisation des différentes relations impliquées. Nous étudions de manière systématique les différentes situations de refactorisation qui peuvent présenter un intérêt pour la correction des défauts de généralisation, correction qui conduit le plus souvent à une simplification du diagramme. Nous spécifions une approche basée sur l'Analyse Formelle de Concepts afin de traiter de manière globale ces opportunités de refactorisation. Une chaîne d'outils, intégrant les plugins pour Eclipse UML2 Tools[1] pour la syntaxe graphique d'UML, UML2[2] pour la syntaxe abtraite d'UML et eRCA[3] pour l'application de l'analyse formelle de concepts, est proposée et mise en œuvre sur quelques cas d'étude.

1. http ://www.eclipse.org/modeling/mdt/ ?project=uml2tools
2. http ://www.eclipse.org/uml2/
3. http ://code.google.com/p/erca/

2. Illustration de l'approche proposée

Cette section présente la problématique de la restructuration de diagrammes de cas d'utilisation sur un exemple que nous utiliserons tout au long de ce papier : le site d'une chocolaterie permettant la consultation des produits et l'inscription à des stages. La figure 1 présente un diagramme de cas d'utilisation dessiné avec le plugin UML2 Tools lors d'une première lecture de l'ensemble des besoins de la chocolaterie (voir en annexe A pour une description synthétique du problème). Il contient les principaux éléments de cette vue que propose UML pour exprimer les usages d'un système. Les acteurs, dessinés sous forme d'un personnage stylisé ou d'une boîte avec le stéréotype «actor», représentent les rôles des personnes ou des systèmes extérieurs interagissant avec le système que l'on décrit. Dans notre exemple, quatre acteurs ont été identifiés : l'administrateur, l'utilisateur standard, le professionnel du chocolat et la base de données de la chambre de commerce et d'industrie. Les cas, qui apparaissent dans des ellipses, modélisent les fonctionnalités, par exemple, `AddPage` représente la fonctionnalité d'ajout de page sur le site par un administrateur, `Register for internship` permet aux professionnels du chocolat de s'inscrire à des stages. Acteurs et cas sont reliés par des associations (traits pleins sur les schémas) qui indiquent qu'un acteur intervient dans un cas d'utilisation : l'utilisateur standard (`User`) intervient ainsi dans `Free Consult`, tandis que la base de données `DB CCI` participe au cas d'utilisation `Verify Id`. Deux relations, l'extension et l'inclusion, permettent de structurer les cas d'utilisation. Un cas, comme `Browse by Keyword`, inclut (`include`) un autre cas, comme `Input Keyword`, si toute réalisation du premier comporte obligatoirement une réalisation du second. L'extension n'a pas ce caractère obligatoire : un cas, par exemple `Bill`, étend (`extend`) un autre, par exemple `Registration Classic Internship`, si une réalisation du premier peut optionnellement venir s'ajouter à une réalisation du second. Dans l'exemple qui nous préoccupe, l'émission de la facture n'est en effet pas systématique. Des conditions et des points d'extension peuvent compléter la relation `extend`, nous en discuterons dans la prochaine section. UML offre également la possibilité d'utiliser la relation de spécialisation entre acteurs ou entre cas d'utilisation. Dans notre exemple, `Registration Classic Internship` décrit une forme particulière de `Register for internship`, comme le traduit la relation de spécialisation placée entre ces deux cas.

Le diagramme de la figure 1 présente néanmoins plusieurs défauts.

– Les cas d'utilisation attachés à l'acteur `Administrator`, soit `Addpage`, `Removepage`, `Modifypage`, `Compilecomment` et `Attachcomment`, appartiennent en réalité à deux catégories : les cas traitant des pages du site et les cas correspondant au traitement des commentaires.

– Les cas attachés par des relations à `Free consult` sont également tous attachés au cas `Browse by keyword`, créant une complexité visuelle et gênant la compréhension.

– L'acteur `Professional of chocolate` peut visiblement réaliser toutes les opérations des utilisateurs ordinaires, il peut donc être considéré comme une sous-

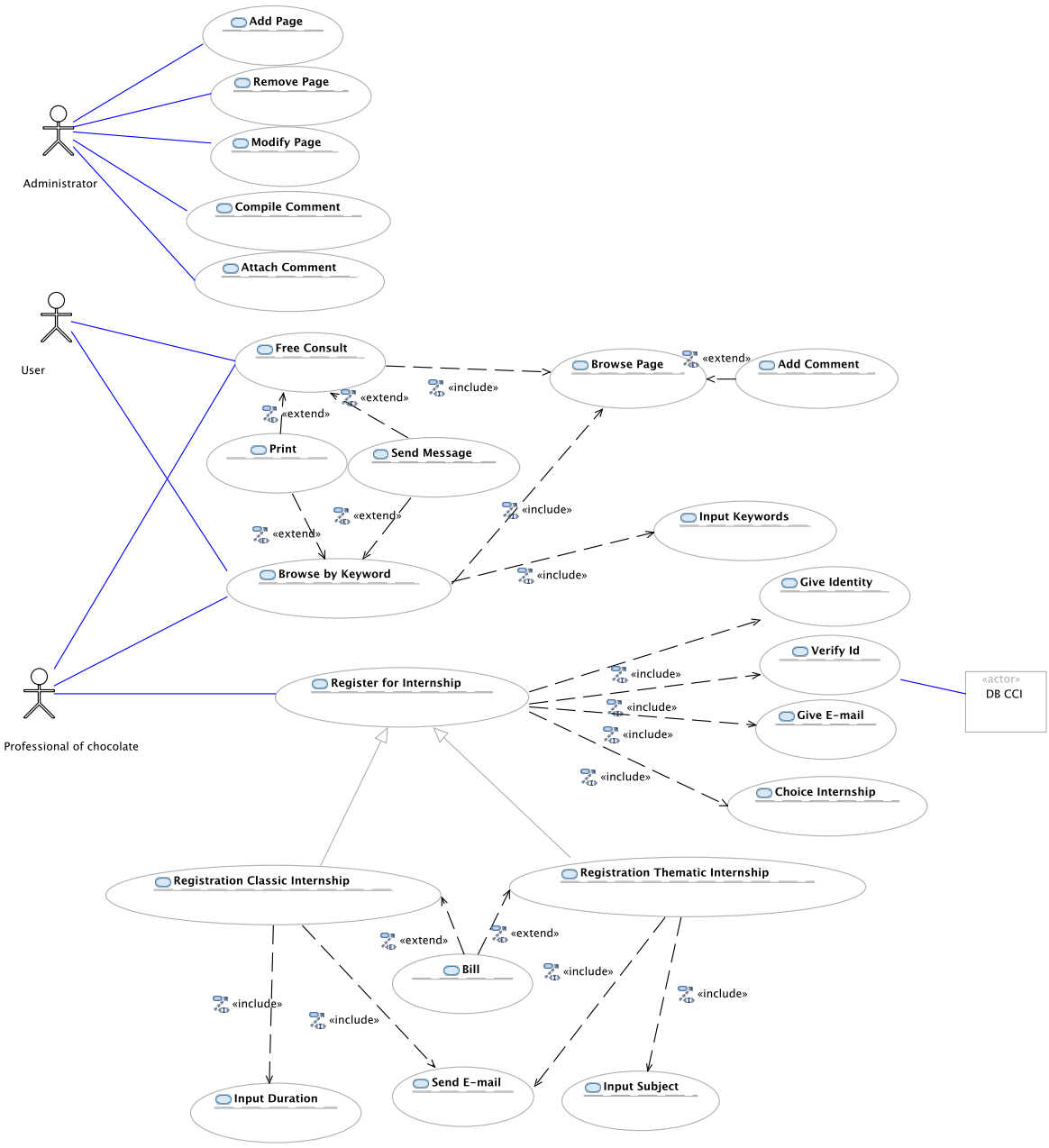


Figure 1. *Diagramme de cas d'utilisation de la chocolaterie (forme initiale)*

catégorie d'utilisateur.

– Le cas `Bill` étend deux cas d'inscription qui incluent également tous les deux le même cas `send email`.

Le diagramme de la figure 2 propose des solutions pour corriger ces défauts. En ce qui concerne les cas attachés à l'acteur `Administrator`, une analyse des segments constituant les identifiants permet de découvrir les deux catégories qui se traduisent par deux cas plus généraux, `Page management` et `Comment management`.

La description de `Free consult` et `Browse by keyword` montre que `Browse by keyword` est simplement un cas plus spécialisé que `Free consult`. L'héritage des relations attachées à `Free consult` nettoie alors le diagramme de trois relations incidentes à `Browse by keyword`.

`Professional of chocolate` disposant de toutes les opérations de `User`, l'introduction d'une relation de spécialisation permet (là encore par héritage) de faire disparaître une partie des associations sortantes de `Professional of chocolate`.

`Bill` et `send email` peuvent être factorisés sur `Register for internship`.

Dans toutes ces situations, la solution est apportée par la découverte d'éléments plus généraux basés sur des caractéristiques partagées :

– par plusieurs acteurs qui ont les mêmes associations incidentes ou des associations vers des cas portant des noms avec une intersection commune,

– par plusieurs cas qui ont les mêmes relations `extend` ou `include` entrantes ou sortantes.

Ceci nous amène à considérer trois points.

– L'Analyse Formelle de Concepts [GAN 99], qui construit des groupes d'entités basés sur des caractéristiques communes, peut sans doute nous aider à former les éléments plus généraux et ce, de manière systématique.

– Les situations dans lesquelles elle peut être appliquée doivent être étudiées soigneusement pour s'assurer de la sémantique du résultat obtenu.

– Le résultat est toujours destiné à être retravaillé par un expert en modélisation. L'approche que nous proposons consiste à présenter à l'expert des opportunités de refactorisation et à lui simplifier le travail d'application de schémas de refactorisation.

3. Schémas de refactorisation

Les défauts que nous nous intéressons à corriger ici portent sur la factorisation de caractéristiques communes, cependant le fait que plusieurs cas d'utilisation partagent certaines relations incidentes ne garantit pas toujours la pertinence de l'ajout d'un *super cas*, ou cas plus général. C'est également le cas pour les acteurs, car les diagrammes de cas d'utilisation sont relativement imprécis et ne portent notamment pas de noms de rôles. Dans cette partie, nous envisageons différentes situations afin de dé-

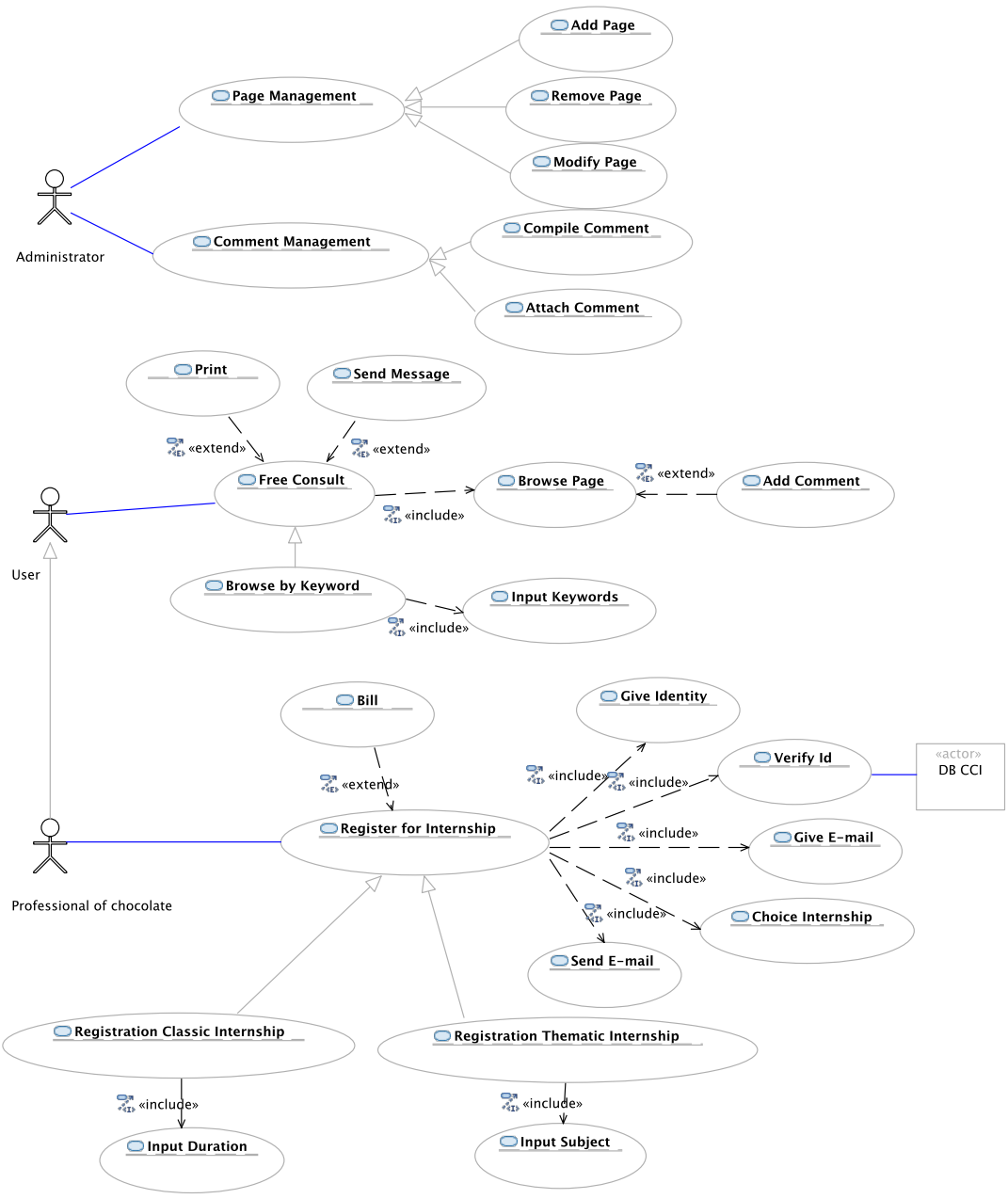


Figure 2. *Diagramme de cas d'utilisation de la chocolaterie (forme refactorisée)*

terminer celles dans lesquelles il peut être pertinent ou non de factoriser des caractéristiques communes. Dans tous les cas, la pertinence se fonde sur une sémantique sous-jacente des diagrammes de cas d'utilisation. Nous considérons ici une sémantique ensembliste simple, qui associe à un diagramme de cas d'utilisation l'ensemble de ses acteurs et pour chaque acteur les séquences réalisables de cas d'utilisation feuilles. Les cas qui possèdent des spécialisations seront supposés abstraits, c'est-à-dire remplacés par une de leurs spécialisations lors d'un usage du système. Ce n'est pas une limitation car tout cas qui possède des spécialisations qui ne serait pas abstrait peut être remplacé théoriquement par un cas spécialisé abstrait plus une feuille représentant la forme non abstraite. Dans notre exemple (Figure 1), l'acteur `User` comptera, parmi les séquences de cas d'utilisation réalisables, la séquence `Browse by keyword`, `Input Keyword`, `Browse Page`, `Add Comment`, `Print`. Dans les deux sous-sections suivantes, nous envisageons des schémas de refactorisation, en ne détaillant que quelques exemples. Il est à noter que certains schémas ne sont pas très pertinents seuls mais le deviennent couplés à d'autres schémas : cela est dû au fait que nous nous plaçons dans le cadre d'une restructuration globale des diagrammes de cas d'utilisation.

3.1. *Refactorisation entre cas d'utilisation*

Les principaux schémas de refactorisation entre cas d'utilisation sont présentés dans la Table 1. Considérons par exemple la situation I (`include` sortant). Deux cas `A` et `B` incluent un cas `C`. Cela signifie entre autres que toute séquence réalisable par un acteur et contenant `A` (resp. `B`) contiendra également `C`. Regardons à présent la transformation proposée (à droite). Une séquence réalisable par un acteur et contenant `A` continue à contenir obligatoirement `C`, puisque `A` spécialise `N` qui contient `C`.

La situation II de la Table 1 (`include` entrant) est moins évidente. Deux cas `A` et `B` sont inclus par un cas `C`. Cela s'interprète ainsi : toute séquence réalisable par un acteur et contenant `C` contiendra également `A` et `B`. Dans la transformation proposée (à sa droite), une séquence réalisable par un acteur et contenant `C` contiendra `N`, c'est-à-dire `A` **ou** `B`, ce qui est donc une sémantique différente.

Dans les deux situations suivantes III et IV (`extend` sortant, `extend` entrant), lorsque les deux conditions sont identiques et que les deux points d'extension sont égaux, une généralisation est possible dans les deux cas grâce à la sémantique optionnelle de la relation `extend`. Dans les cas fréquents d'absence de précision sur les conditions et les points d'extension, on appliquera également la refactorisation. Par contre nous proposons dans cette première étude que toute divergence des conditions ou des points d'extension rende la refactorisation caduque.

À ces cas s'ajoute une refactorisation par spécialisation (cas V de la Table 1). La situation initiale comprend un cas `A` dont l'ensemble des relations `include` sortantes et des relations `extend` entrantes est strictement inclus dans celui d'un autre cas `B`. La refactorisation consiste à dériver `B` de `A` et à enlever les relations à présent héritées.

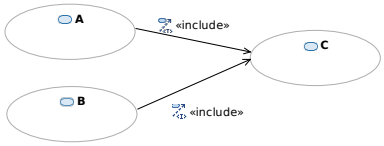
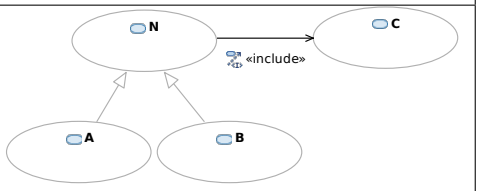
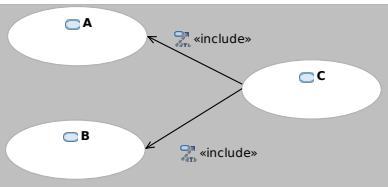
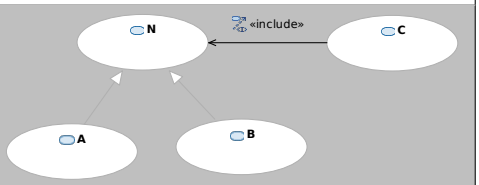
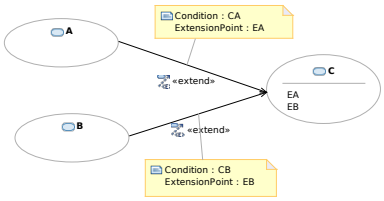
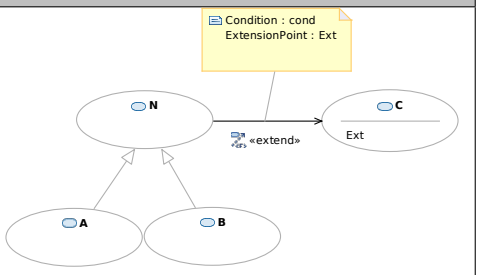
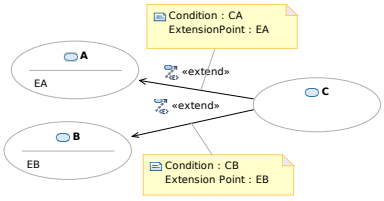
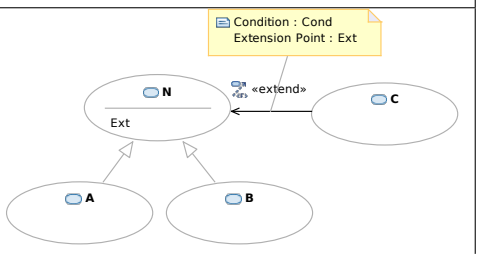
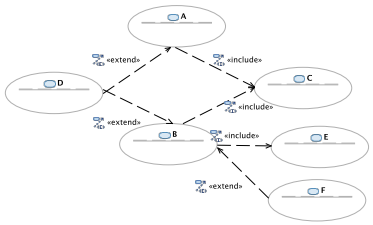
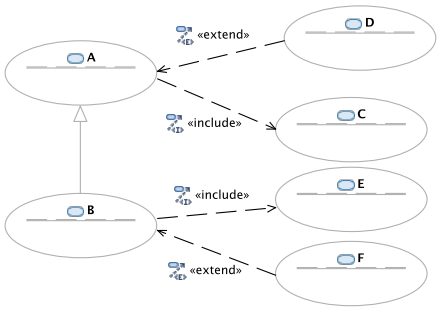
	motif source	motif refactorisé
I	 include sortant	
II	 include entrant	
III	 extend sortant	
IV	 extend entrant	
V	 extend et include partagés	

Tableau 1. *Schémas de refactorisation entre cas d'utilisation (en gris : un schéma envisagé mais non retenu)*

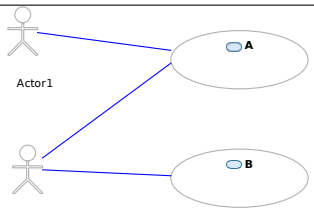
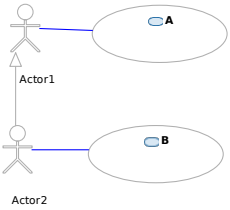
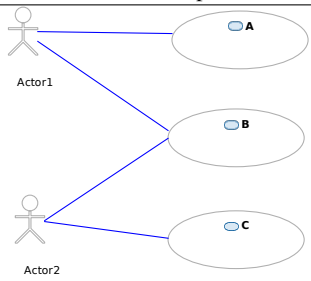
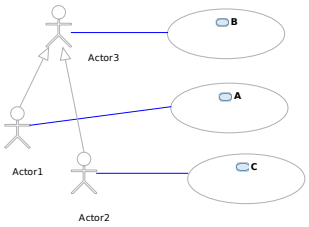
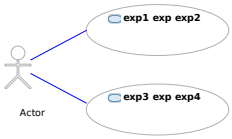
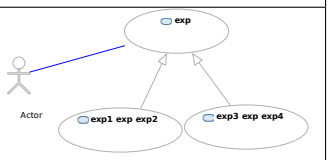
	motif source	motif refactorisé
I	 Acteurs et cas avec spécialisation	
II	 Acteurs et cas avec factorisation	
III	 Acteurs et cas avec factorisation de segments de noms	

Tableau 2. *Schémas de refactorisation entre acteurs et cas d'utilisation*

3.2. *Refactorisation entre acteurs et cas*

Le tableau 2 présente trois schémas de refactorisation que nous avons identifiés comme de bonnes opportunités d'amélioration du diagramme. Le premier (I) introduit une relation de spécialisation entre acteurs lorsque l'ensemble des associations de l'un (`Actor 1`) est strictement inclus dans l'ensemble des associations de l'autre (`Actor 2`), puis retire les associations héritées. Le deuxième (II) ajoute un acteur abstrait lorsque les ensembles d'associations de deux acteurs (`Actor 1` et `Actor 2`) ont une intersection non vide mais que chacun a des associations que l'autre n'a pas. Le troisième (III) utilise une segmentation des identifiants des cas d'utilisation auxquels participe un acteur. Lorsque des segments communs apparaissent (qui sont des substantifs ou des actions), cela peut correspondre à une sémantique commune. On introduit alors un cas plus général correspondant à cette sémantique.

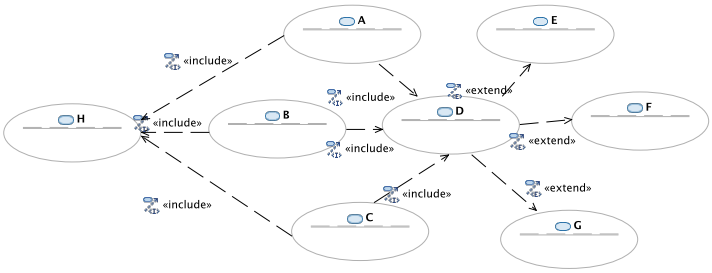


Figure 3. *Un diagramme à refactoriser*

4. Analyse Formelle de Concepts pour une refactorisation globale

Les schémas de refactorisation présentés à la section précédente procurent des solutions tout à fait adaptées à la simplification des diagrammes par généralisation, mais n'offrent pas de méthode pour traiter un diagramme dans son ensemble. Si on considère le simple exemple présenté à la figure 3, où les cas `D` et `H` sont inclus par trois autres cas `A`, `B` et `C` et où `D` étend les trois cas `E`, `F` et `G`, une application avec un point de vue limité sur le diagramme peut conduire un outil ou un concepteur à proposer une solution de refactorisation (Figure 4) dans laquelle l'inclusion de `D` et `H` par `A` et `B` a été considérée indépendamment de l'inclusion de `D` et `H` par `B` et `C`, menant aux deux nouveaux cas `N1` et `N2`. Une solution symétrique de refactorisation a été appliquée aux cas que `D` étend. Cette solution, quoique peut-être pertinente dans certains cas particuliers, est plutôt une complexification du schéma initial.

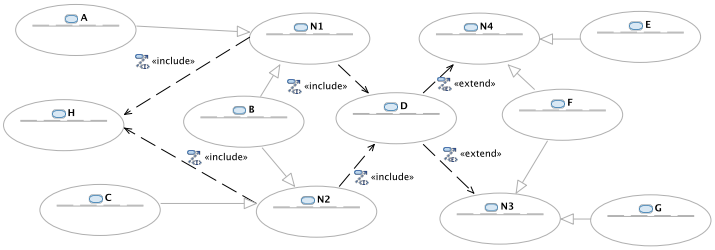


Figure 4. *Une application des schémas de refactorisation avec un point de vue local*

Il existe pourtant une solution minimale, présentée par la figure 5, dans laquelle une vision globale de la refactorisation a mené à introduire une seule généralisation (`N1`) pour les trois cas qui incluent `D` et `H` et une seule généralisation (`N2`) pour les trois cas que `D` étend. Ces situations peuvent être bien plus complexes encore par exemple lorsque trois cas ont en commun des caractéristiques et parmi ces trois cas, deux ont également des caractéristiques communes que le troisième ne possède pas, etc.

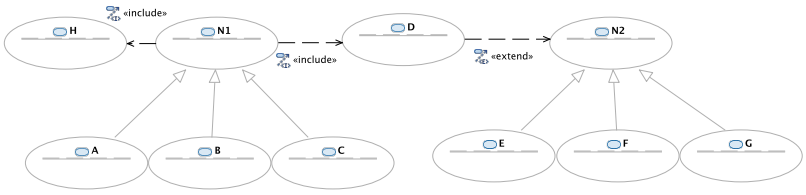


Figure 5. *Une application des schémas de refactorisation avec un point de vue global*

L'Analyse Formelle de Concepts (AFC) [GAN 99], qui effectue des regroupements systématiques d'objets partageant des caractéristiques, offre un bon support pour trouver ces solutions minimales grâce aux propriétés des classifications produites. Nous l'utilisons ici avec les éléments descriptifs des diagrammes de cas d'utilisation : acteurs et cas d'utilisation forment les objets que l'on cherche à regrouper, tandis que les associations acteur-cas, les relations d'inclusion et d'extension entrantes ou sortantes et les segments des noms des cas d'utilisation sont les caractéristiques de ces objets. Les données sur lesquelles travaille l'AFC sont des tables indiquant quels objets ont quelles caractéristiques. Pour notre exemple, nous définissons une table partielle (Tableau 3) décrivant quelques éléments du diagramme de cas d'utilisation de la figure 1 en utilisant les caractéristiques pertinentes d'après l'analyse des schémas locaux de refactorisation. Dans cette table, l'acteur `User` est par exemple décrit par l'association vers `Free Consult`, le cas `Browse By Keyword` est décrit par le fait d'être étendu par `Print` ou d'inclure `Browse page`. Les groupes (ou concepts) sont formés en associant les ensembles d'objets maximaux partageant des ensembles maximaux de caractéristiques communes et sont organisés dans une classification qui dispose d'une structure de treillis (Figure 6). Dans la classification, les caractéristiques sont héritées du haut vers le bas, tandis que les objets décrits sont hérités du bas vers le haut. Par exemple, l'ensemble (ou extension du concept) formé par les deux cas d'utilisation `Reg. Classic Internship` et `Reg. Thematic Internship` associé à l'ensemble (ou intension du concept) de toutes leurs caractéristiques communes (`extended by Bill`, `include Send Email`) forme un concept ($Concept_6$). Dans la classification, les concepts sont partiellement ordonnés par inclusion des intensions (en descendant dans la classification) ou par l'inclusion des extensions (en remontant dans la classification). Le concept $Concept_7 = (\{$ `Reg. Classic Internship`$\}, \{$ `extended by Bill`, `include Send Email`, `include Input Duration`$\})$ est ainsi un concept plus petit (plus spécialisé) que $Concept_6$.

Les concepts et leur capacité à factoriser les caractéristiques sont utilisés pour définir des acteurs ou des cas plus généraux. Dans notre exemple, le concept $Concept_6$ déduit de la factorisation des caractéristiques communes aux cas `Reg. Classic Internship` et `Reg. Thematic Internship`, serait interprété comme un cas plus général représentant l'inscription à un stage (même s'il n'avait pas été présent dans le diagramme initial) étendu par `Bill` et incluant `Send Email`. De l'ordre de spécialisation entre concepts nous déduisons la spécialisation entre acteurs (par exemple `Pro-`

`fessional of Chocolate` spécialise `User`) ou la spécialisation entre cas (par exemple `Browse By Keyword` spécialise `Free Consult`).

	assoc. to Free Consult	assoc. to Browse By Keyword	assoc. to Register Internship	extended by Print	extended by Send Message	include Browse Page	include Input Keyword	extended by Bill	include Send Email	include Input Duration	include Input Subject
User	x	x									
Prof. of Choco	x	x	x								
Free Consult				x	x	x					
Browse By Keyword				x	x	x	x				
Reg. Classic Internship								x	x	x	
Reg. Thematic Internship								x	x		x

Tableau 3. *Table partielle pour le diagramme initial de la chocolaterie*

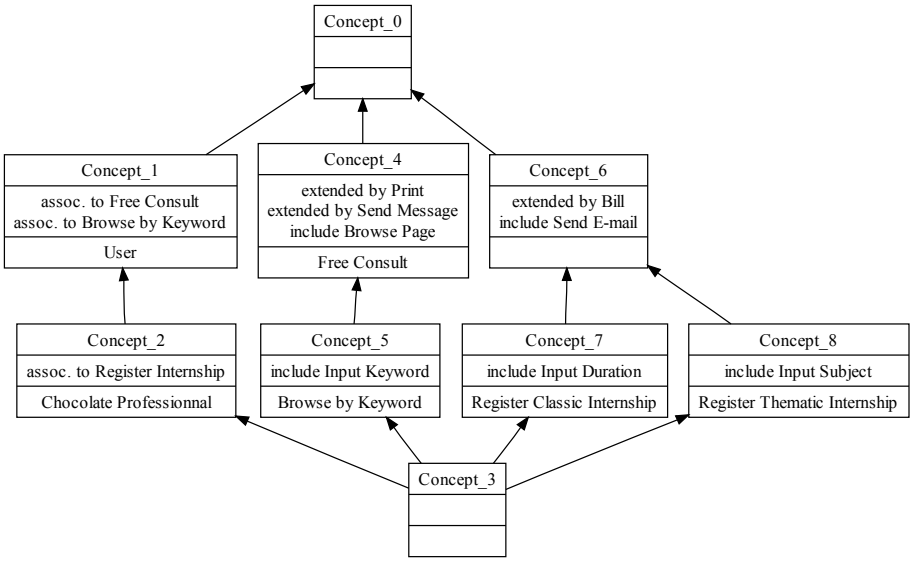


Figure 6. *Treillis associé au tableau 3*

5. Une chaîne d'outils pour la refactorisation

L'approche est implémentée par connexion de différents outils sous Eclipse. Les diagrammes de cas d'utilisation à étudier sont conçus avec l'éditeur graphique du plug-in UML2 Tools. Ils sont sauvegardés au format XMI et sont manipulables avec

l'API fournie par le plugin UML2, lui-même créé en utilisant le framework EMF. Une procédure d'import analyse les fichiers XMI dans lesquels sont sauvegardés les diagrammes et en extrait les informations sur lesquelles va porter la refactorisation. Ces informations sont placées dans des contextes formels qu'exploite ensuite le plug-in eRCA pour produire les treillis. Une procédure d'export reconstruit des diagrammes de cas d'utilisation au format XMI afin que ceux-ci soient lisibles dans UML2 Tools et utilisables par les concepteurs. Les treillis peuvent être consultés ou rester totalement cachés à l'utilisateur final. Différentes options sont proposées dans l'outil actuel. Plutôt que de recevoir uniquement un diagramme complètement transformé, le concepteur du diagramme peut ainsi demander de traiter séparément :

– les cas d'utilisation décrits d'une part par les relations `include` et `extend` auxquelles ils participent, d'autre part par les acteurs auxquels ils sont associés,

– les acteurs humains ou machine décrits par les cas d'utilisation auxquels ils sont associés,

– les cas d'utilisation en association directe avec les acteurs, décrits par les segments qui forment leurs identificateurs.

Ces différentes vues permettent au concepteur d'analyser des résultats partiels de refactorisation et de comprendre leur combinaison dans le diagramme final. Par exemple, le contexte formel des acteurs humains décrits par les cas d'utilisation auxquels ils participent permet de se concentrer sur les relations de spécialisation entre ces acteurs et sur une éventuelle création de nouveaux acteurs plus abstraits, sans se plonger dans les détails des cas d'utilisation. Ces constructions partielles se plongent sans difficulté dans le diagramme final, grâce aux propriétés de l'Analyse Formelle de Concepts : quel que soit l'ordre dans lequel on combine ces vues, le résultat global est toujours le même, l'ajout de nouvelles descriptions sur les cas d'utilisation ou sur les acteurs se traduisant par un raffinement du treillis. Plus de détails sur l'exemple de la chocolaterie, l'outil et son application sont présentés dans [HAK 09][4].

6. Travaux connexes

La restructuration manuelle de cas d'utilisation fait l'objet de nombreux travaux, lesquels s'appuient sur des spécifications plus ou moins formelles ; ces spécifications montrent les points sur lesquels il est possible d'agir pour modifier un cas d'utilisation, et sous quelles contraintes. Dans [RUI 03], un méta-modèle des cas d'utilisation est décrit, reprenant et étendant des modèles existants ; des relations entre cas d'utilisation sont exhibées, permettant de relier en particulier deux versions d'un même cas d'utilisation : composition, utilisation/dépendance, précédence, prérequis, extension, généralisation/spécialisation, ressemblance, équivalence. Dans [ISS 07], une classification des différents moyens d'agir sur un cas d'utilisation conduit l'auteur à distinguer les modifications d'ordre comportemental (essentiellement liées à la définition des tâches d'un cas) de celles d'ordre structurel (liées à la définition des acteurs et à

4. Disponible à l'URL http ://www.lirmm.fr/~huchard/Documents/Papiers/MasterLalaHakik.pdf

leurs liens avec les tâches). Différentes versions d'un même cas d'utilisation peuvent aussi être comparées via des métriques appropriées ; ainsi [HEN 02] présente quelques métriques, qui restent toutefois assez générales (issues de métriques générales sur les graphes, pour l'essentiel des statistiques élémentaires).

Les propriétés de généralisation de l'AFC ont été utilisées dans de nombreux travaux en génie logiciel, dont certains sont présentés dans [TIL 05]. On peut notamment citer les travaux de [GOD 93] sur la restructuration et maintenance des hiérarchies de classes, mis en application par la suite dans une approche dirigée par les modèles dans [ARÉ 06]. On peut aussi trouver dans la littérature des travaux appliquant l'AFC sur des cas d'utilisation textuels : dans [RIC 02] des cas d'utilisation écrits dans un langage naturel restreint sont classifiés dans un treillis en fonction des termes qu'ils contiennent dans un but de visualisation et dans [DüW 00] les cas d'utilisation sont classifiés en fonction des termes importants, choisis par l'utilisateur, qu'ils contiennent dans le but de faire émerger les notions essentielles qui deviendront des classes. Mais aucun de ces travaux à notre connaissance ne proposait l'utilisation de l'AFC dans le but de restructurer des cas d'utilisation.

7. Conclusion

Nous avons posé les bases d'une approche permettant de corriger des défauts de généralisation apparaissant dans les diagrammes d'utilisation UML. Cette approche s'appuie sur plusieurs schémas de refactorisation et sur l'Analyse Formelle de Concepts pour combiner leur application. L'outil proposé a été testé sur plusieurs cas d'école, d'une complexité similaire à l'étude de cas de la chocolaterie, mais les diagrammes de cas d'utilisation sont rarement plus complexes, sinon ils doivent être décomposés. Une prochaine étape consistera à réaliser une expérience sur une plus grande quantité de diagrammes et à déterminer des métriques pertinentes pour évaluer la qualité des résultats obtenus. La description du problème invite à penser que nous pourrions utiliser l'Analyse Relationnelle de Concepts (variante de l'AFC) pour affiner les résultats, mais dans un premier temps, nous avons préféré ne pas créer un résultat trop complexe afin d'évaluer la faisabilité de l'approche. Nous souhaitons ensuite intégrer divers autres éléments dans la description, notamment la notion d'acteur primaire ou secondaire, les noms des rôles, ou encore des documents complémentaires tels que les diagrammes de séquences et approfondir l'analyse des identifiants des cas d'utilisation par des techniques de traitement automatique de la langue.

8. Bibliographie

[ARÉ 06] ARÉVALO G., FALLERI J.-R., HUCHARD M., NEBUT C., « Building Abstractions in Class Models : Formal Concept Analysis in a Model-Driven Approach », NIERSTRASZ O., WHITTLE J., HAREL D., REGGIO G., Eds., *MoDELS*, Lecture Notes in Computer Science, Springer, 2006, p. 513-527.

[BOO 05] BOOCH G., RUMBAUGH J., JACOBSON I., « *Unified Modeling Language User Guide, The (Addison-Wesley Object Technology Series)* », chapitre 16, Addison-Wesley Professional, 2005.

[COC 97a] COCKBURN A., « Goals and Use Cases », *JOOP*, vol. 10, n° 5, 1997, p. 35-40.

[COC 97b] COCKBURN A., « Using Goal-Based Use Cases », *JOOP*, vol. 10, n° 7, 1997, p. 56-62.

[COC 00] COCKBURN A., *Writing Effective Use Cases*, Addison-Wesley Professional, January 2000.

[DüW 00] DÜWEL S., HESSE W., « Bridging the gap between use case analysis and class structure design by formal concept analysis », *Proceedings of Modellierung 2000*, Fölbach-Verlag, 2000, p. 27–40.

[GAN 99] GANTER B., WILLE R., *Formal Concept Analysis : Mathematical Foundations*, Springer, 1999.

[GEN 02] GENOVA G., LLORENS J., QUINTANA V., « Digging into Use Case Relationships », *UML '02 : Proceedings of the 5th International Conference on The Unified Modeling Language*, London, UK, 2002, Springer-Verlag, p. 115–127.

[GOD 93] GODIN R., MILI H., « Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices », *OOPSLA*, 1993, p. 394-410.

[HAK 09] HAKIK L. M., « Simplification de diagrammes de cas d'utilisation », Master's thesis, Université Montpellier 2, Montpellier, June 2009.

[HEN 02] HENDERSON-SELLERS B., ZOWGHI D., KLEMOLA T., PARASURAM S., « Sizing Use Cases : How to Create a Standard Metrical Approach », BELLAHSENE Z., PATEL D., ROLLAND C., Eds., *OOIS*, vol. 2425 de *Lecture Notes in Computer Science*, Springer, 2002, p. 409-421.

[ISS 07] ISSA A., « Utilising Refactoring To Restructure Use-Case Models », AO S. I., GELMAN L., HUKINS D. W. L., HUNTER A., KORSUNSKY A. M., Eds., *Proceedings of the World Congress on Engineering, WCE'07, London, UK, 2-4 July, 2007*, Lecture Notes in Engineering and Computer Science, Newswood Limited, 2007, p. 523-527.

[JAC 92] JACOBSON I., CHRISTERSON M., JOHNSSON P., OVERGAARD G., *Object-Oriented Software Engineering : A Use Case Driven Approach*, Addison-Wesley Professional, June 1992.

[OMG09] OMG, « Unified Modeling Language Superstructure, version 2.2 », 2009.

[RIC 02] RICHARDS D., BÖTTGER K., AGUILERA O., « A Controlled Language to Assist Conversion of Use Case Descriptions into Concept Lattices », MCKAY B., SLANEY J. K., Eds., *Australian Joint Conference on Artificial Intelligence*, Lecture Notes in Computer Science, Springer, 2002, p. 1-11.

[RUI 03] RUI K., BUTLER G., « Refactoring Use Case Models : The Metamodel », OUDSHOORN M. J., Ed., *Twenty-Sixth Australasian Computer Science Conference (ACSC'03)*, vol. 16 de *CRPIT*, Adelaide, Australia, 2003, ACS, p. 301-308.

[TIL 05] TILLEY T., COLE R., 0002 P. B., EKLUND P. W., « A Survey of Formal Concept Analysis Support for Software Engineering Activities », GANTER B., STUMME G., WILLE R., Eds., *Formal Concept Analysis*, Lecture Notes in Computer Science, Springer, 2005, p. 250-271.

A. Annexe - Etude de cas chocolaterie

Une chocolaterie prévoit de créer un site de présentation de ses produits à d'éventuels gourmets et d'inscription à des stages de formation thématiques (moulage ou bouchée). Le système doit permettre à un administrateur d'ajouter ou de retirer des pages web de présentation des produits ainsi que de modifier leur contenu. Tous les usagers du site pourront consulter les pages des produits proposés par la chocolaterie. Une consultation libre permettra de parcourir les pages du catalogue des produits proposés. Chaque page pourra faire l'objet d'un commentaire donné par le gourmet sur le produit présenté. À intervalles réguliers, ces commentaires seront compilés par l'administrateur du site et attachés à la page du produit. Une consultation par mots-clefs demande de saisir un ensemble de mots-clefs saisis par l'utilisateur ; le système construit la liste des pages du site contenant ces mots-clefs que l'utilisateur peut alors consulter. Toute consultation (libre ou par mots-clefs) pourra être suivie d'une impression dans un format amélioré et une autre option de la consultation permettra d'en envoyer les résultats à des adresses email. En plus du grand public, le site s'adresse aux professionnels de la chocolaterie qui peuvent, en plus de la consultation, s'inscrire à des stages proposés par la chocolaterie. Pour s'inscrire à un stage, un professionnel doit donner son identité, qui est vérifiée dans une base de données externe au système (celle des chambres de commerce et d'industrie), son email, puis doit choisir entre deux formes de stage, le stage classique ou le stage thématique (chocolats de Pâques, pour Noël, La St Valentin ou la fête des femmes). Choisir le stage classique demande d'indiquer la durée souhaitée, tandis que choisir le stage thématique demande de préciser le thème. A l'issue d'une inscription un email récapitulant son choix est obligatoirement envoyé au futur stagiaire. Une facture peut être émise optionnellement au moment de l'inscription, si le stage n'est pas pris en charge par la chambre de commerce et d'industrie.