Unions et intersections de modèles pour l'analyse des systèmes d'information

André Miralles, Marianne Huchard, Jessie Carbonnel, Clémentine Nebut

Unions et intersections de modèles pour l'analyse des systèmes d'information

André Miralles [1], **Marianne Huchard** [2], **Jessie Carbonnel** [2], **Clémentine Nebut** [2]

1. *Tetis/IRSTEA, France*
   andre.miralles@teledetection.fr
2. *LIRMM, CNRS & Université de Montpellier, France*
   marianne.huchard,jessie.carbonnel,clementine.nebut@lirmm.fr

RÉSUMÉ. *En système d'information, l'intégration de modèles consiste à regrouper au sein d'un unique modèle l'ensemble des entités métiers de plusieurs modèles connectés d'un point de vue thématique. Dans cette communication, cinq opérations sont proposées afin d'assister cette intégration et améliorer les modèles structurels : la première crée un modèle alignement montrant les correspondances entre les modèles, la seconde produit deux modèles union et la troisième produit deux modèles intersection avec les seuls éléments communs à tous les modèles. Les modèles union et intersection sont calculés suivant soit un mécanisme reposant sur la sémantique des entités métiers soit un mécanisme fondé sur la description binaire de ces entités. Quel que soit le mécanisme utilisé, les modèles union sont les plus petits des unions des modèles et les modèles intersection, noyau de tous les modèles, sont les plus grands des intersections des modèles. Ces cinq opérations sont réalisées à l'aide de l'Analyse Formelle de Concepts (AFC).*

ABSTRACT. *In information systems, model integration consists in grouping into a single model all the business entities of several thematically connected models. In this paper, five operations are proposed to assist this integration: an alignment model which highlights the correspondences between the models; two union models and two intersection models built up with the common elements. The union and intersection models are calculated according to either a mechanism based on the semantics of business entities or a mechanism based on the binary description of these entities. Whatever the mechanism used, the union models are the smallest of the model unions and the intersection models, kernel of all models, are the greatest intersections of the models. These five operations are achieved with the help of Formal Concept Analysis (FCA).*

MOTS-CLÉS : *Système d'information, UML, modèle de classes, appariement de modèles, intégration de modèles, union de modèles, intersection de modèles, Analyse Formelle de Concepts*

KEYWORDS: *Information System, UML, Class model, class model matching, class model integration, class model union, class model intersection, Formal Concept Analysis*

1. Introduction

La phase d'analyse du domaine et des besoins des utilisateurs d'un système d'information est probablement la plus délicate car d'elle dépend le succès ou l'échec d'un développement informatique. Au cours de cette phase, la capture des *entités métiers* [1] par le concepteur reste la tâche la plus sensible car ce dernier doit se les approprier afin de les retranscrire et de les décrire correctement dans le modèle de l'application.

L'ingénierie dirigée par les modèles a pour but d'automatiser via des transformations l'évolution des modèles depuis l'analyse jusqu'à la génération de code. Dans cette approche, le modèle d'analyse joue un rôle majeur et sa qualité est essentielle pour le bon déroulement du cycle de développement mais aussi pour que l'application finale ou le système d'information soit conforme aux besoins des utilisateurs. Une conséquence directe est qu'il doit être parfaitement structuré et avoir une qualité sémantique irréprochable. Dans la pratique, sauf pour des modèles de faible taille, il est impossible de satisfaire ces exigences de qualité en quelques séances d'analyse. Ce constat a conduit à la multiplication des méthodes de développement itératives ou agiles (Kruchten, 1999 ; Beck, 2000 ; Alliance, 2001) qui permettent d'améliorer progressivement la structuration ou la qualité sémantique des systèmes.

La réalisation de ce modèle "idéal" est d'autant plus difficile que le nombre d'acteurs impliqués est grand et que les domaines (thématiques, scientifiques, législatifs, etc.) concernés sont nombreux et variés. Dans ce contexte, (Miralles, 2016) préconise de faire l'analyse par petits groupes homogènes de 5 à 10 acteurs avec une démarche agile (Royce, 1970). Un autre avantage de cette analyse en groupes est que, si certains acteurs sont en situation de conflit potentiel (ex. agriculteur/écologiste), l'analyse sera plus sereine et donc de meilleure qualité. La contrepartie de ce processus d'analyse en groupes est la nécessité de disposer de méthodes et d'outils pour intégrer les modèles produits. Cette problématique d'intégration et de consolidation des modèles se rencontre en ingénierie mais également en réingénierie de systèmes patrimoniaux (Miralles, 2016). Lors de l'inventaire de l'existant, s'il existe déjà des applications ou des bases de données dans des domaines connexes, les ateliers de génie logiciel actuels permettent de reconstruire un modèle UML par ré-ingénierie du code ou de la base de données. Le concepteur dispose alors de plusieurs modèles UML à intégrer. Plusieurs auteurs ont développé des méthodes et des outils pour réaliser cette intégration.

Dans cet article, nous proposons une approche mettant en œuvre l'Analyse Formelle de Concepts pour mener à bien cette intégration en effectuant différentes transformations. Par rapport à un alignement de schémas qui produit habituellement seulement des correspondances entre éléments de modélisation, nous construisons trois types de modèles (schémas) : un modèle alignement qui est le plus proche de l'alignement de schémas standard, un modèle union, appelé LCM (Least Common Multiple Model) qui est l'intégration minimale et complète des modèles et un modèle inter-

1. Afin d'éviter toute confusion entre la notion de `Concept métier` utilisée en modélisation et celle de `Concept` d'un treillis, le terme `Concept métier` est remplacé par `Entité métier`.

section, appelé GCM (Greatest Common Model) qui est la factorisation maximale des modèles. Ces deux derniers modèles bornent au sens mathématique l'espace de correspondance des modèles sources. Ces trois modèles peuvent être construits indépendamment. De plus, nous proposons deux mécanismes pour construire les modèles union et intersection, en explorant deux pistes : l'une utilise les noms des entités (elle sera qualifiée de *sémantique*) ; l'autre utilise les descriptions des entités (elle sera qualifiée d'*ensembliste*).

Le contexte et la problématique ont été présentés dans cette première section. La section 2 présente le cas d'étude que nous utilisons pour illustrer l'approche. La section 3 rappelle quelques éléments de l'Analyse Formelle de Concepts, puis les principes de calcul de l'union et de l'intersection sémantique versus ensembliste sont présentés en section 4 ainsi que le processus de construction des modèles alignement, union et intersection. Les trois sections suivantes exposent successivement l'opération d'alignement (section 5), les opérations d'union (section 6) et les opérations d'intersection (section 7), ainsi que les cinq types de modèles qui en résultent. La section 8 décrit une ré-analyse approfondie du modèle alignement et donne quelques pistes d'amélioration de l'analyse en mobilisant les cinq types de modèles. La section 9 présente les travaux connexes qui ont inspiré et alimenté notre réflexion. Enfin, nous concluons et proposons quelques perspectives en section 10.

2. Cas d'étude

Dans le monde agricole, les prévisions météorologiques sont des informations majeures pour les agriculteurs car d'elles dépendent la croissance des plantes, mais aussi certains travaux agricoles à effectuer et, en particulier, les traitements phytosanitaires. Ces prévisions issues de mesures locales sont de plus en plus souvent « personnalisées » afin de bien répondre aux besoins locaux des agriculteurs. Face à la multiplication de produits proposés, le souhait des acteurs agricoles est de disposer d'un modèle unique issu des différents modèles de station météorologique existants. L'étude de cas portera sur la fusion des extraits de deux modèles. Elle est inspirée du système d'information sur les Pesticides développé à IRSTEA (Miralles *et al*., 2011).

2.1. Présentation des deux modèles à fusionner

La station météorologique, dont le modèle est celui de la FIGURE 1, est équipée d'un pluviomètre (`RainGauge`) pour effectuer des relevés ponctuels de pluie (`Rain`) et d'un anémomètre (`Anemometer`) pour mesurer les caractéristiques du vent (`Wind`). Le pluviomètre est identifié par un numéro de série (`serialNumber`) et la hauteur du tube (`tubeHeight`) recueillant l'eau de pluie, paramètre propre à ce matériel. L'anémomètre est quant à lui caractérisé par sa fréquence d'acquisition (`acquisitionFrequency`) ainsi que par sa classe de précision (`accuracyClass`). Les quantités de précipitation (`waterAmount`) ainsi que l'intensité et la direction du vent (`windStrength` et `windDirection` respectivement) sont datées (`timePoint`) et renseignées par un code de qualité globale de la mesure (`code-`

Quality). Cette station permet en outre de suivre la quantité de particules (particuleAmount) des pluies acides (AcidRain) qui sont un cas particulier de pluie. Pour reconstituer un événement pluvieux (RainEvent), il est nécessaire d'identifier, pendant la durée de l'épisode (timePeriod), les différents relevés ponctuels de pluie effectués, via le rôle storedRain.

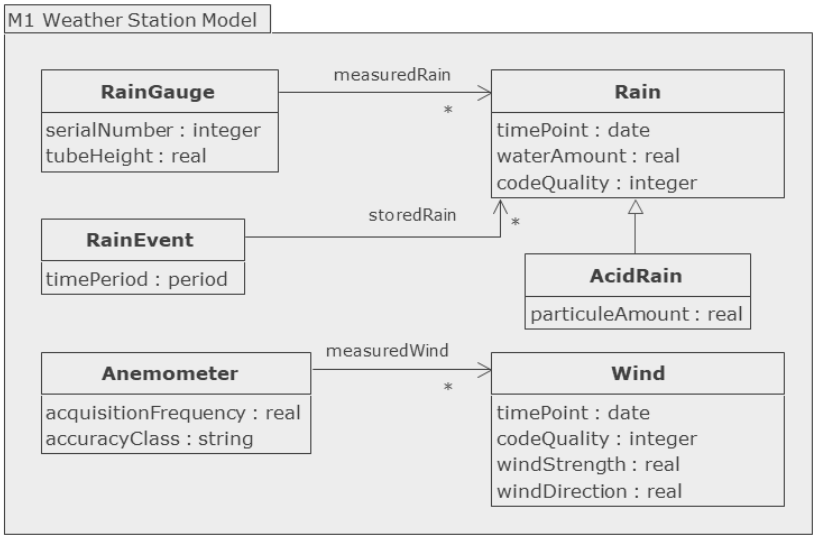

Figure 1. Modèle de la station météorologique M1

Le modèle de la FIGURE 2 est celui d'une station météorologique permettant en outre de relever les chutes de neige (SnowFall) au moyen d'un nivomètre (SnowGauge). Ce dernier instrument est un appareil qui donne l'équivalent en eau de la quantité de neige recueillie. De ce fait, comme pour le pluviomètre, la hauteur du tube (tubeHeight) recueillant la neige est une caractéristique intrinsèque de l'appareil. La similarité se retrouve aussi dans les caractéristiques relevées qui sont la quantité d'eau (waterAmount) et un code global de qualité (codeQuality) de cette mesure. Dans ce modèle, ces mesures ne sont pas datées (absence de timePoint) car, comme les chutes de neige sont rares, le choix des gestionnaires a été de reporter la date sur un cahier de terrain. Par ailleurs, la connaissance des événements pluvieux n'est pas nécessaire pour les utilisations envisagées (absence d'une entité RainEvent). En outre, cette station n'est pas équipée pour le suivi des pluies acides (modèle sans la classe AcidRain). Les autres différences importantes par rapport au modèle de la station M1 sont :

– l'utilisation d'entités sémantiquement différentes pour désigner la pluie et le vent ; dans le modèle M2, l'entité pluie (Rain) est remplacée par précipitation (Precipitation) et vent (Wind) par brise (Breeze [2]),

2. Pour un météorologue, une Breeze est souvent un vent de faible intensité. En termes de modélisation, l'entité Breeze pourrait être vue comme une spécialisation de l'entité Wind.

– le numéro de série du pluviomètre (`serialNumber`) n'est pas pris en compte,

– la direction du vent (`windDirection`) est ignorée ainsi que la qualité de la mesure (`codeQuality`),

– enfin, une information importante pour le suivi à long terme de cette station est le type d'anémomètre utilisé, type qui peut évoluer dans le temps et expliquer certains écarts de mesure.

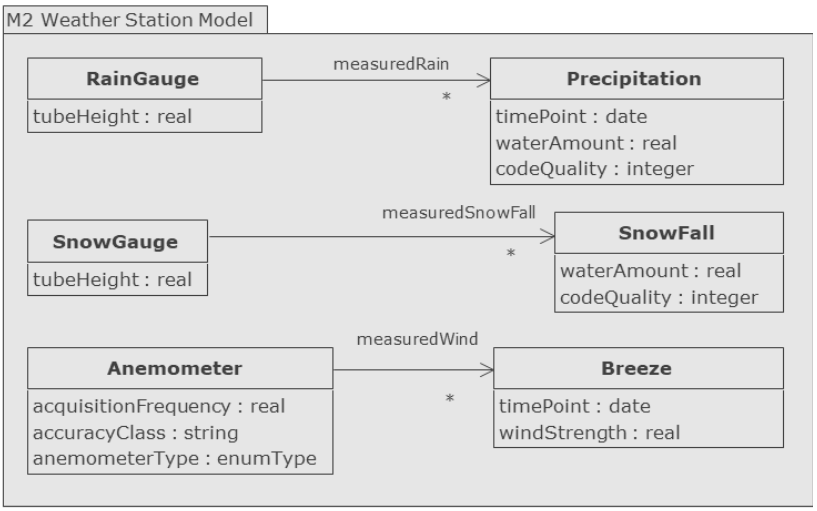

Figure 2. Modèle de la station météorologique M2

3. Eléments sur l'Analyse Formelle de Concepts

Notre approche prend sa source dans l'analyse par treillis (Barbut, Monjardet, 1970), également connue comme Analyse Formelle de Concepts (Ganter, Wille, 1999). L'AFC est une approche mixte permettant à la fois (1) l'extraction de connaissances et de règles dans des données et leur ordonnancement par spécialisation/généralisation, (2) la construction de classifications conceptuelles telles que des ontologies, des modèles entités-relations et des modèles de classes. Ces propriétés lui sont procurées par sa capacité à mettre en évidence des schémas communs et des différences dans les données soumises à l'analyse.

Dans sa forme la plus simple, un ensemble ordonné de concepts est formé à partir d'un *contexte formel* composé d'un ensemble d'entités décrites par des caractéristiques. Un *contexte formel* est ainsi un triplet $K = (G, M, I)$, où G est l'ensemble d'entités, M l'ensemble de caractéristiques et $I \subseteq G \times M$ une relation binaire dont chaque couple (g, m) associe une entité g à une caractéristique m qu'elle possède.

La TABLE 1 présente un contexte formel associant aux classes du modèle UML de la FIGURE 1 (modèle M1 [3]), leurs attributs et les rôles qu'elles possèdent dans des associations. Dans cette table, par exemple, la classe `AcidRain` est associée aux attributs `timePoint`, `codeQuality`, `waterAmount` et `particuleAmount`. Les trois premiers attributs sont hérités, tandis que le quatrième est déclaré dans la classe elle-même. La classe `RainEvent` est associée quant à elle à l'attribut `timePeriod` et au rôle `storedRain`.

Table 1. Contexte formel K_{M1} avec le concept $Concept\_M1\_6$ grisé

M1	**serialNumber**	**tubeHeight**	**timePoint**	**codeQuality**	**waterAmount**	**timePeriod**	**acquisitionFrequency**	**accuracyClass**	**windStrength**	**windDirection**	**particuleAmount**	**measuredRain**	**storedRain**	**measuredWind**
RainGauge	×	×										×		
Rain			×	×	×									
RainEvent						×							×	
Anemometer							×	×						×
Wind			×	×					×	×				
AcidRain			×	×	×						×			

Pour un contexte formel $K = (G, M, I)$ donné, un *concept formel* $C = (Extent(C), Intent(C))$ associe un ensemble maximal d'entités avec un ensemble maximal de caractéristiques qu'elles possèdent. Dans le contexte formel, un concept peut se repérer visuellement comme un plus grand pavé de cases cochées par des croix pour une permutation possible sur les lignes et sur les colonnes. L'extension du concept est l'ensemble $Extent(C) = \{g \in G | \forall m \in Intent(C), (g, m) \in I\}$ des entités couvertes par le concept, tandis que son intension $Intent(C) = \{m \in M | \forall g \in Extent(C), (g, m) \in I\}$ contient les caractéristiques partagées.

Pour la FIGURE 3, le concept $Concept\_M1\_6$ regroupe ainsi trois classes partageant les deux attributs de son intension : $Extent(Concept\_M1\_6)$=$\{Wind, Rain, AcidRain\}$ (les trois classes héritées des sous-concepts en remontant) et $Intent(Concept\_M1\_6)$)=$\{timePoint, codeQuality\}$) (les deux attributs introduits dans le concept).

Étant donnés deux concepts formels $C_1 = (E_1, I_1)$ et $C_2 = (E_2, I_2)$ d'un contexte formel K, un ordre de spécialisation/généralisation $\preceq_C$ peut être donné par $C_1 \preceq_C C_2$ si et seulement si $E_1 \subseteq E_2$ (et de manière équivalente $I_2 \subseteq I_1$). C_1 est une spécialisation (un sous-concept) de C_2. C_2 est une généralisation (un super-concept) de C_1. Par exemple $Concept\_M1\_5 = (\{Rain, AcidRain\}, \{timePoint, codeQuality, waterAmount\})$ est un sous-concept de $Concept\_M1\_6$.

Par ces définitions, C_1 hérite des caractéristiques de C_2, tandis qu'inversement, C_2 hérite des entités de C_1. Les caractéristiques et entités héritées sont souvent omises sur

3. Dans ce qui suit, M1 et M2 désignent respectivement les modèles `M1 Weather Station Model` et `M2 Weather Station Model`.

les schémas pour des raisons de lisibilité. Une caractéristique (resp. une entité) est dite *introduite* par un concept s'il s'agit du concept le plus général (resp. le plus spécifique) où elle apparaît. C'est ainsi que la représentation graphique de $Concept\_M1\_5$ ne présente que la caractéristique introduite $waterAmount$ car dans une représentation exhaustive, ce concept serait le plus haut concept où $waterAmount$ apparaîtrait (les sous-concepts en héritent). $Concept\_M1\_5$ introduit la classe $Rain$ car il s'agirait du concept le plus bas où apparaîtrait $Rain$ dans une représentation exhaustive (les super-concepts en héritent). Si $\mathcal{C}_K$ est l'ensemble de tous les concepts de K, le munir de la relation d'ordre $\preceq_C$ lui confère une structure de treillis.

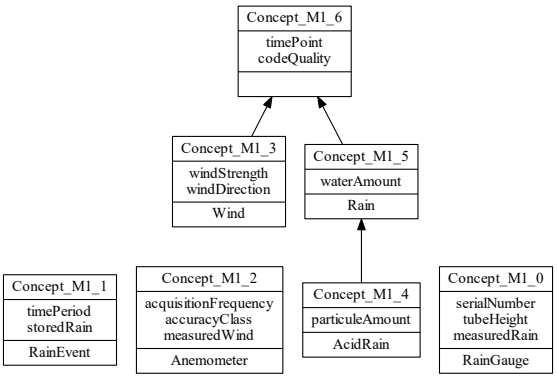

Figure 3. AOC-poset du contexte formel K_{M1} de station météorologique

Plusieurs applications de l'AFC considèrent uniquement le sous-ordre du treillis restreint aux concepts qui introduisent au moins une caractéristique ou au moins une entité. Ce sous-ordre porte le nom d'AOC-poset et se trouve souvent utilisé dans les applications relatives à la classification conceptuelle. La FIGURE 3 présente l'AOC-poset associé au contexte formel de la TABLE 1. L'interprétation d'un AOC-poset dans un tel contexte sera reprise plus en détail dans les prochaines sections. Pour en donner un premier aperçu ici, on peut constater que les concepts offrent une vue par spécialisation des classes. Ces dernières apparaissent dans les extensions des concepts (partie basse des boîtes), tandis que les attributs ou rôles introduits apparaissent dans les intensions des concepts (partie centrale des boîtes). La construction de l'AOC-poset met en évidence, grâce au $Concept\_M1\_6$, une possible super-classe de `Wind` et `Rain`, factorisant `timePoint` et `codeQuality`, et introduisant le concept métier de *mesure*. Elle montre aussi, ce qui était déjà connu, que `AcidRain`, introduite dans un sous-concept de celui qui introduit `Rain`, pourrait en être une sous-classe. Ces relations, qui ne sont pas toujours explicites dans le modèle, le deviennent dans l'AOC-poset. Dans les sections suivantes, nous étudions une nouvelle utilisation de l'AFC, cette fois en présence de plusieurs modèles UML, pour analyser leurs correspondances et les intégrer.

4. Union, intersection sémantiques vs Union, intersection ensemblistes

Dans cette section, nous décrivons le principe général des opérations d'union et d'intersection sémantique versus les mêmes opérations ensemblistes. Nous explicitons également le processus général de création de différents modèles qui seront présentés par la suite.

4.1. Principe

Nous avons vu qu'en Analyse Formelle de Concepts, un contexte formel K est souvent représenté comme une table binaire qui comprend quatre composantes :

– la première composante est la première cellule située en haut à gauche de la table, cellule contenant le nom de la relation dans notre exemple,

– la deuxième composante est la colonne de gauche de la table où sont listées, ligne par ligne, les entités concernées par la relation. Ces entités sont nommées par des termes issus du domaine métier d'où la nature fortement sémantique de cette colonne,

– la troisième composante est l'entête (première ligne) où sont répertoriées, colonne par colonne, les caractéristiques des entités. Celles-ci sont aussi définies par leur nom conférant également à cette composante une nature sémantique,

– la quatrième composante de la table est la partie centrale où la présence d'une croix à l'intersection d'une ligne et d'une colonne indique que l'entité de la ligne possède la caractéristique de la colonne. L'ensemble des croix d'une ligne constitue un tuple.

A gauche de la FIGURE 4 (resp. la FIGURE 5), on peut observer des contextes formels nommés $R1$ et $R2$ contenant des entités nommées mb (resp. ab, op, hi, rk) et des attributs nommés a, b, c (resp. u, v, w). $< a, b, c >$ est le tuple décrivant mb dans $R1$ et $< a, c >$ est le tuple décrivant mb dans $R2$.

Cette description montre que le tableau est constitué d'une partie sémantique, première colonne avec son entête, et d'une partie binaire, le reste du tableau. Aussi, chaque ligne comporte une partie sémantique, le nom de l'entité en première colonne, et une partie binaire qui est le tuple décrivant l'entité. Effectuer des opérations comme l'union et l'intersection sur ce type de table conduit à manipuler soit des lignes soit des colonnes. Comme notre objectif est de réduire l'espace d'analyse en regroupant soit des entités de même nom ayant potentiellement des tuples différents (descriptions différentes), soit des entités de même tuple ayant potentiellement des noms différents, les opérations ne porteront que sur les lignes de la table.

De ce fait, les opérations concernent soit la première colonne soit les colonnes binaires. Dans le premier cas, les opérations portent sur la chaîne de caractères constituant le nom (égalité des chaînes, fonctions de similarité comme Levenshtein, Jaro-Winkler, ou encore basées sur une ontologie ou sur un dictionnaire de données, un vocabulaire contrôlé, etc.). À ce titre, ces opérations seront qualifiées de ***sémantiques***.

Dans le second cas, les opérations portent sur des tuples d'où le qualitatif d'***ensemblistes***.

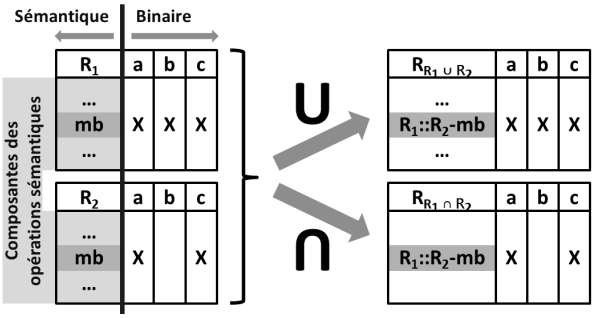

Figure 4. Principe de l'union sémantique et de l'intersection sémantique

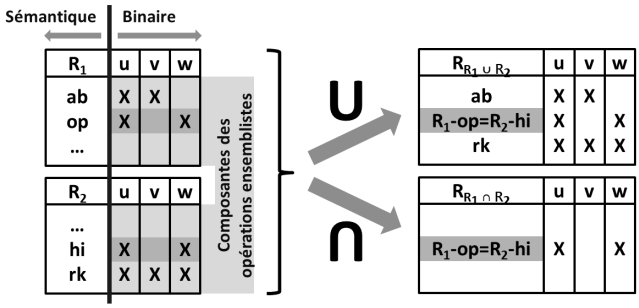

Figure 5. Principe de l'union ensembliste et de l'intersection ensembliste

La FIGURE 4 montre les mécanismes d'union et d'intersection sémantiques de contextes formels. Les opérations sémantiques portent sur la première colonne des tables (FIGURE 4). L'union sémantique des contextes formels R_1 et R_2 consiste à faire l'union binaire des tuples des entités de même nom dans les deux contextes (ex. mb). Pour conserver une traçabilité, le nom de l'entité résultante est préfixé par le nom des relations des contextes formels des modèles sources suivi d'un tiret '-' (ex. R_1::R_2-mb). Les autres lignes des contextes formels des modèles sources sont regroupées dans le contexte formel d'union $R_{R_1 \cup R_2}$. Le calcul de l'intersection sémantique est effectuée de manière similaire à celui de l'union à la variante près que seules les lignes correspondant à des entités de même nom appartiennent au contexte formel d'intersection $R_{R_1 \cap R_2}$, les autres lignes sont délaissées.

Les mécanismes d'union et d'intersection ensemblistes de contextes formels sont décrits en FIGURE 5. Comme le montre cette figure, les opérations ensemblistes portent sur l'ensemble des colonnes des tables, la première colonne n'étant pas prise en compte. L'union ensembliste des contextes formels R_1 et R_2 concerne les lignes qui ont des tuples identiques (lignes op et hi des contextes R_1 et R_2 respectivement qui ont le tuple $< u, w >$) et ce, quels que soient les noms des entités. L'union ensembliste

conduit à fusionner des lignes même si les noms des entités sont différents. Pour assurer la traçabilité, le nom de l'entité résultante est construit à partir du nom des entités préfixé (avec un '-') par les noms des relations des contextes formels des modèles sources séparés par le signe égal (ex. R_1-*op*=R_2-hi). Comme pour l'union sémantique, les autres lignes des contextes formels sources sont réunis dans le contexte formel d'union $R_{R_1 \cup R_2}$. De façon similaire à l'intersection sémantique, le contexte formel de l'intersection ensembliste $R_{R_1 \cap R_2}$ est constitué des seules lignes correspondant à des doublons de tuples, les autres lignes ne sont pas considérées.

4.2. Processus de création des modèles alignement, union et intersection

Le processus général de création des modèles alignement, union et intersection est décrit en FIGURE 6. Dans l'atelier de génie logiciel Objecteering, un profil UML, nommé *RCFT*, a été conçu et développé pour générer les fichiers (*.rcft) décrivant les contextes formels des modèles sources. C'est la transformation Tr. 1. Dans l'atelier de génie logiciel, l'utilisateur a la possibilité de sélectionner les éléments de modélisation sur lesquels il souhaite travailler (classe, attribut, etc.).

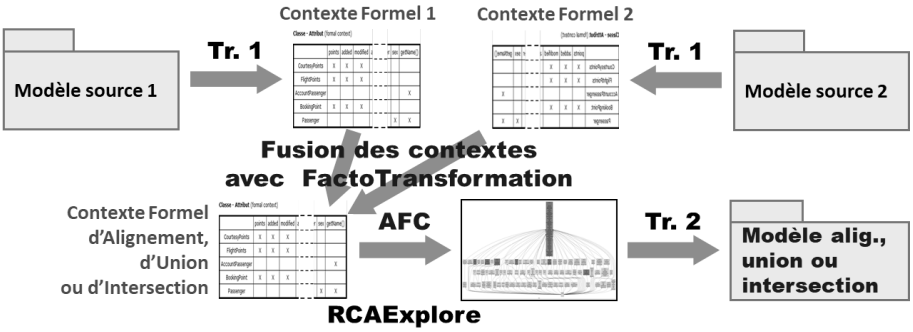

Figure 6. Processus de création des modèles alignement, union et intersection

Ensuite, les contextes formels d'alignement, d'union ou d'intersection, appelés contextes de fusion, sont produits avec l'application *FactoTransformations* dont la conception et le développement ont été effectués dans le cadre de cette recherche. L'Analyse Formelle de Concepts est réalisée avec *RCAExplore* [4] qui transforme les contextes de fusion en treillis. La transformation Tr. 2 est une transformation complexe qui altère un clone des modèles sources en fonction des nouveaux concepts et des concepts fusionnés du treillis. Suivant le treillis, l'altération du clone donne le modèle alignement, union ou intersection. Cette transformation décrite en détail dans (Miralles, 2016) établit des liens de traçabilité entre les éléments de modélisation sources et clonés. Bien évidemment, les nouveaux concepts n'ont pas de liens de traçabilité.

4. http://dolques.free.fr/rcaexplore/

5. Alignement de modèles

La section 4.1 a été dédiée à présenter les principes des opérations d'union et d'intersection qu'elles soient sémantiques ou ensemblistes et la section 4.2 décrit le processus de construction des modèles de fusion (alignement, union ou intersection). La présente section est consacrée à la description de l'opération d'alignement de deux modèles à partir des contextes formels. Les deux sections suivantes sont structurées à l'identique à celle-ci, mais concernent les opérations d'union et d'intersection.

5.1. Contexte Formel pour l'opération d'alignement

Le *contexte formel d'alignement*, présenté en TABLE 2, consiste à concaténer les contextes formels associés aux modèles M1 et M2, après avoir préfixé le nom des classes par le nom du modèle dont elles proviennent. Une ligne de ce contexte correspond donc à une paire (modèle d'origine, classe). Une colonne correspond à un attribut ou un rôle provenant de l'un ou de l'autre des contextes sources (ou des deux).

Table 2. Contextes formels pour l'alignement. Dans cet article, pour des raisons de place nous ne présentons pas séparément les contextes formels des modèles M1 et M2. Ils peuvent être facilement retrouvés à partir de la Table 2 : la partie supérieure (lignes préfixées par "M1-") correspond au contexte du modèle M1 alors que la partie inférieure (lignes préfixées par "M2-") est issue du modèle M2.

Alignement	**serialNumber**	**tubeHeight**	**timePoint**	**codeQuality**	**waterAmount**	**timePeriod**	**acquisitionFrequency**	**accuracyClass**	**windStrength**	**windDirection**	**particuleAmount**	**measuredRain**	**storedRain**	**measuredWind**	**anemometerType**	**measuredSnowFall**
M1-RainGauge	×	×										×				
M1-Rain			×	×	×											
M1-RainEvent						×							×			
M1-Anemometer							×	×						×		
M1-Wind			×	×					×	×						
M1-AcidRain			×	×	×						×					
M2-RainGauge		×										×				
M2-Precipitation			×	×	×											
M2-SnowFall				×	×											
M2-Anemometer							×	×						×	×	
M2-Breeze			×						×							
M2-SnowGauge		×														×

5.2. AOC-Poset d'alignement

L'AOC-poset du contexte formel d'alignement (TABLE 2) des modèles M1 et M2 est donné en FIGURE 7. Cet AOC-poset permet de reconstruire (grâce à la transformation Tr. 2) le modèle UML de la FIGURE 8 représentant l'alignement des modèles M1 et M2.

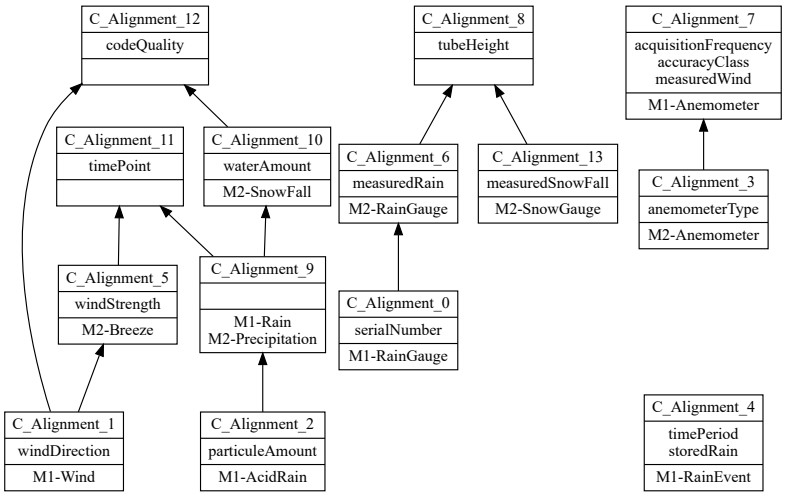

Figure 7. AOC-poset du contexte formel d'alignement

Une première analyse rapide de l'AOC-poset permet de constater la présence de trois concepts ($C\_Alignment\_8$, $C\_Alignment\_11$ et $C\_Alignment\_12$) qui ne correspondent pas à une classe d'origine (leur extension simplifiée est vide). Pour cette raison, nous les qualifions de « nouveau » dans la suite. Ces trois nouveaux concepts sont le résultat de la factorisation des attributs `tubeHeight`, `timePoint` et `codeQuality` respectivement. Ces concepts doivent être validés et nommés par un expert du domaine et, dans le cas présent, ils donnent naissance à trois nouvelles entités métiers dans le modèle de la FIGURE 8. Ces trois entités sont respectivement `PrecipitationDevice`, `TimeParameter` et `QualityParameter`.

L'AFC fusionne aussi les entités métiers `Rain` du modèle M1 et `Precipitation` du modèle M2 au sein du concept $C\_Alignment\_9$. Ce n'est pas surprenant au vu des attributs les décrivant. Ce concept correspond à l'entité métier `Rain/Precipitation` de la FIGURE 8.

Les concepts $C\_Alignment\_8$, $C\_Alignment\_11$, $C\_Alignment\_12$ et $C\_Alignment\_9$ révèlent ainsi quatre nouvelles entités métiers n'appartenant pas aux modèles M1 et M2.

Dans l'AOC-poset de la FIGURE 7, les entités métiers `Anemometer` de M1 et M2 apparaissent dans deux concepts distincts ($C\_Alignment\_3$ et $C\_Alignment\_7$ respectivement) donnant ainsi naissance à deux classes dans le modèle alignement de la FIGURE 8. L'entité métier `Anemometer` de M2 spécialise celle de M1 car elle possède en plus l'attribut `anemometerType`. Il en est de même pour les deux entités métiers `RainGauge` de M1 et de M2 qui partagent le rôle `measuredRain` ($C\_Alignment\_6$) alors que l'attribut `serialNumber` est propre à l'entité métier `RainGauge` de M1 ($C\_Alignment\_0$), d'où la relation de spécialisation en FIGURE 8. La présence des doubles occurrences de `Anemometer` et de `RainGauge`

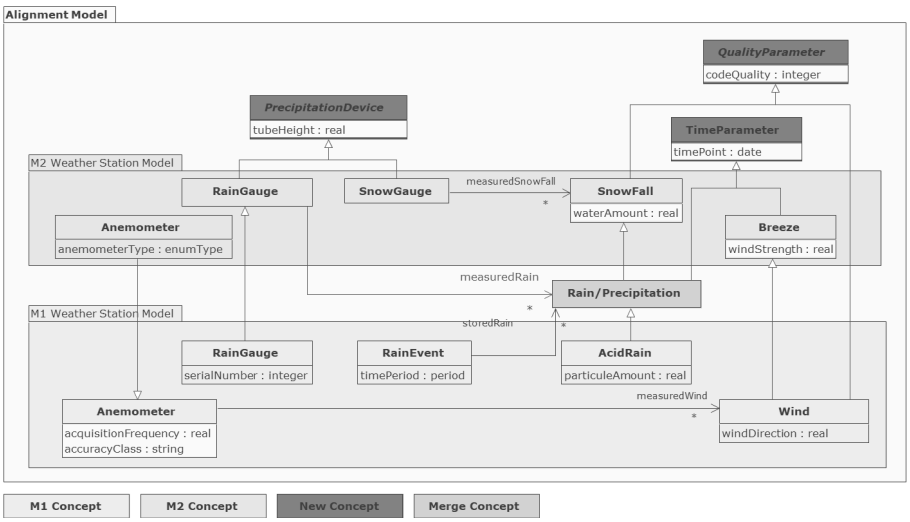

Figure 8. Modèle alignement des modèles M1 et M2

impose le maintien des modèles M1 et M2 au sein du modèle alignement (`Alignment Model`) de la FIGURE 8.

L'entité métier `Anemometer` est en relation avec `Wind` au travers d'une association dont le rôle est `measuredWind`. Le concept $C\_Alignment\_5$ factorise l'attribut `windStrength` qui est commun aux entités métiers `Wind` et `Breeze` de M1 et M2 respectivement. L'entité `Wind` spécialise `Breeze` puisque `Wind` a en plus l'attribut `windDirection` (FIGURE 8).

Enfin, les entités métiers `RainEvent` avec son attribut `timePeriod` et `AcidRain` qui a comme attribut `particuleAmount` sont propres au modèle M1. `RainEvent` a une association vers l'entité résultat de la fusion `Rain/Precipitation` qui a pour rôle `storedRain`. `AcidRain` spécialise l'entité `Rain/Precipitation` comme c'était le cas dans le modèle M1. Les entités métiers `SnowGauge` et `SnowFall` appartiennent au modèle M2. Elles sont reliées par une association dont le rôle de `SnowFall` est `measuredSnowFall`.

Ce modèle alignement, comme on le voit, est intéressant pour avoir une vue structurée sur le rapprochement des deux modèles. Il ne fait aucune hypothèse préalable sur des correspondances entre des éléments, au contraire, il prend les modèles tels qu'ils sont.

6. Union de modèles

De façon analogue à la section 5, nous abordons dans cette section la description des deux opérations d'union de modèles évoquées en section 4.

6.1. Contextes Formels pour les opérations d'union sémantique et ensembliste

Dans les contextes formels d'*union sémantique* (TABLE 3) et d'*union ensembliste* (TABLE 4), on retrouve les mêmes colonnes que dans le contexte formel d'alignement.

Pour l'union sémantique, les lignes du contexte d'alignement qui correspondent à des classes dont le nom est identique sont fusionnées au sein de la TABLE 3. Dans notre exemple, ce sera le cas d'une part, pour les lignes `M1-RainGauge` de M1 et `M2-RainGauge` de M2, fusionnées dans la ligne `M1::M2-RainGauge` et, d'autre part, pour `M1-Anemometer` et `M2-Anemometer` fusionnées dans la ligne `M1::M2-Anemometer`. Pour ces classes de même nom, la description est l'union des descriptions (attributs et rôles) des deux modèles comme cela a été décrit en section 4. Par exemple, bien que `M1-Anemometer` ne possède pas `anemometerType`, comme `M2-Anemometer` le possède, cet attribut est associé à `M1::M2-Anemometer` dans l'union.

Table 3. Contextes formels pour l'union sémantique

Union sémantique	serialNumber	tubeHeight	timePoint	codeQuality	waterAmount	timePeriod	acquisitionFrequency	accuracyClass	windStrength	windDirection	particuleAmount	measuredRain	storedRain	measuredWind	anemometerType	measuredSnowFall
M1::M2-RainGauge	×	×										×				
M1-Rain			×	×	×											
M1-RainEvent						×							×			
M1::M2-Anemometer							×	×						×	×	
M1-Wind			×	×					×	×						
M1-AcidRain			×	×	×						×					
M2-Precipitation			×	×	×											
M2-SnowFall				×	×											
M2-Breeze			×						×							
M2-SnowGauge		×														×

Pour l'union ensembliste, ce sont les lignes `M1-Rain` et `M2-Precipitation` qui fusionnent car elles ont le même tuple donc la même description : `timePoint`, `codeQuality` et `waterAmount`. Le résultat est la ligne fusionnée `M1-Rain=M2-Precipitation` dans le contexte formel d'union ensembliste. Dans ce cas précis, la fusion des deux lignes est sémantiquement acceptable car les deux entités métiers `Rain` de M1 et `Precipitation` de M2 modélisent le même phénomène.

6.2. AOC-Posets d'union sémantique et ensembliste

La FIGURE 9 montre l'AOC-poset du contexte formel d'union sémantique des modèles M1 et M2 (TABLE 3) et la FIGURE 11 celui du contexte formel d'union ensembliste (TABLE 4) des deux modèles.

L'analyse du contexte formel d'union sémantique (FIGURE 9) montre qu'il est quasiment identique à celui du contexte formel d'alignement de la FIGURE 7. Au

Table 4. Contextes formels pour l'union ensembliste

Union ensembliste	serialNumber	tubeHeight	timePoint	codeQuality	waterAmount	timePeriod	acquisitionFrequency	accuracyClass	windStrength	windDirection	particuleAmount	measuredRain	storedRain	measuredWind	anemometerType	measuredSnowFall
M1-RainGauge	×	×										×				
M1-Rain=M2-Precipitation			×	×	×											
M1-RainEvent						×							×			
M1-Anemometer							×	×						×		
M1-Wind			×	×					×	×						
M1-AcidRain			×	×	×						×					
M2-RainGauge		×										×				
M2-SnowFall				×	×											
M2-Anemometer							×	×						×	×	
M2-Breeze			×						×							
M2-SnowGauge		×														×

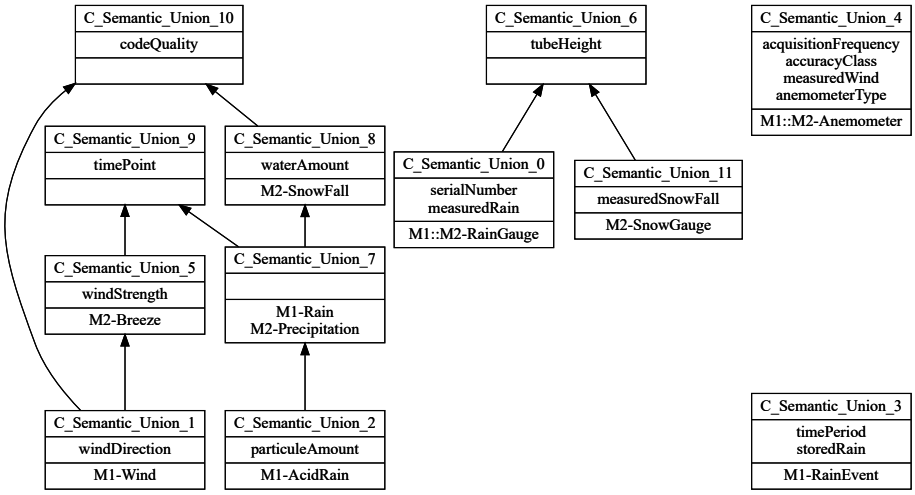

Figure 9. AOC-poset du contexte formel d'union sémantique

nom et à la numérotation des concepts près (identifiants générés automatiquement par *RCAExplore*), les seules différences résident dans :

– la fusion des concepts $C\_Alignment\_3$ et $C\_Alignment\_7$ de la FIGURE 7 pour donner le concept $C\_Semantic\_Union\_4$ de la FIGURE 9, concept où on retrouve les propriétés des entités métiers `Anemometer` des deux modèles M1 et M2,

– l'assimilation des deux concepts $C\_Alignment\_0$ et $C\_Alignment\_6$ de la FIGURE 7 au sein du concept $C\_Semantic\_Union\_0$ de la FIGURE 9 ; les propriétés des entités `RainGauge` des deux modèles M1 et M2 sont réunies dans le concept $C\_Semantic\_Union\_0$.

Au nom et à la numérotation des concepts près, l'analyse du contexte formel d'union ensembliste (FIGURE 11) montre qu'il est identique à celui du contexte formel

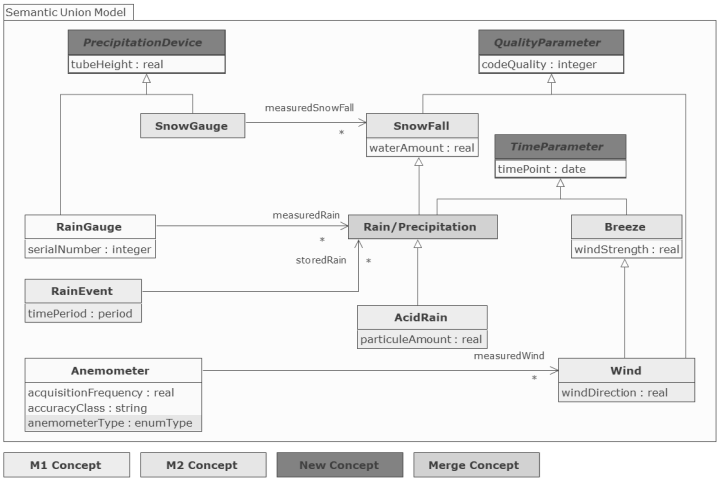

Figure 10. Modèle union sémantique des modèles M1 et M2

d'alignement de la FIGURE 7. C'est un hasard lié au cas d'étude. La seule différence est que le concept $C\_Alignment\_9$ de la FIGURE 7 est issu d'une *fusion des classes* `M1-Rain` et `M2-Precipitation` alors que l'union ensembliste crée un *nouveau concept* regroupant ces deux classes ($C\_Set\_Union\_0$ - FIGURE 11).

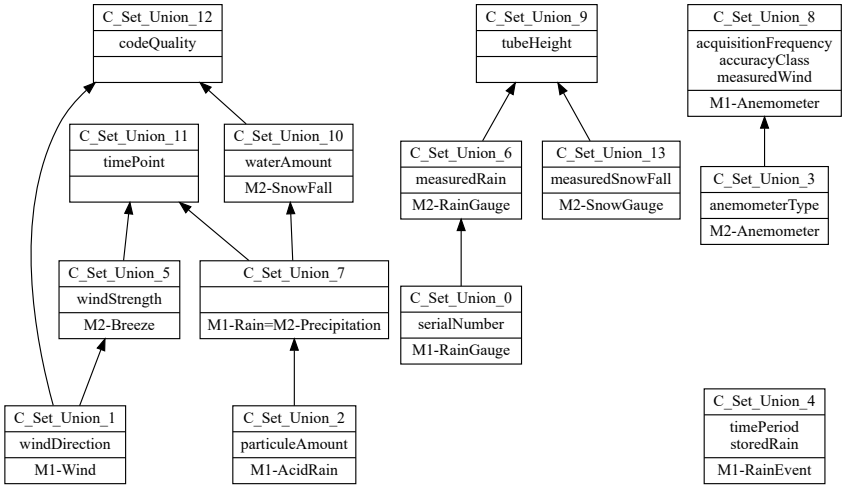

Figure 11. AOC-poset du contexte formel d'union ensembliste

De ce fait, le modèle de l'union ensembliste de la FIGURE 12 est identique au modèle alignement de la FIGURE 8, la seule différence est la classe `Rain/Precipitation` qui est de type *New Concepts* au lieu d'être de type *Merge Concepts*.

Le modèle union sémantique de la FIGURE 10, issu du calcul du contexte formel d'union sémantique, est plus simple et plus facile à analyser que le modèle alignement

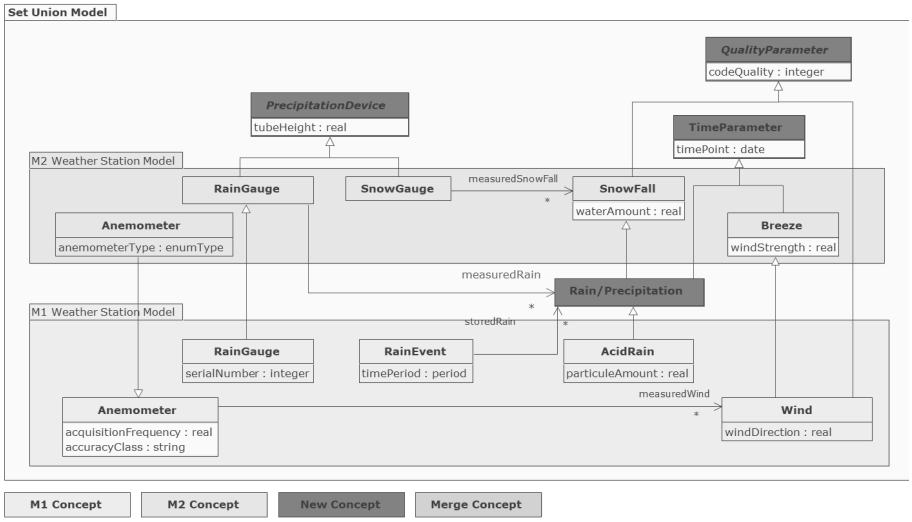

Figure 12. Modèle union ensembliste des modèles M1 et M2

de la FIGURE 8. Cela se traduit par un gain de productivité lors de l'analyse approfondie puisque celle-ci est plus rapide. Une analyse approfondie comme celle de modèle alignement (cf. 5.2) aboutirait plus rapidement au même modèle final (FIGURE 17), modèle qui sera présenté en section 8.1.

Pour le modèle union ensembliste de la FIGURE 12, l'analyse approfondie serait aussi conséquente que pour le modèle alignement puisque les AOC-posets sont identiques.

7. Intersection de modèles

Comme évoqué en section 5, nous étudions à présent les deux formes d'intersection de modèles décrites en section 4.

7.1. Contextes Formels pour les opérations d'intersection sémantique et ensembliste

Dans le *contexte formel intersection sémantique* de la TABLE 5, on ne garde d'une part, que les classes de même nom et, d'autre part, que les colonnes du contexte formel d'alignement communes aux deux modèles et ayant une croix. Ainsi, l'attribut `serialNumber`, qui n'est associé à la classe `RainGauge` que dans le modèle M1, n'apparaît pas. Il en est de même pour l'attribut `anemometerType` de la classe `Anemometer` de M2. Une classe n'est donc associée à un attribut ou à un rôle que si elle possède cette description dans les deux modèles.

Table 5. Contexte formel pour l'intersection sémantique

Intersection sémantique	**tubeHeight**	**acquisitionFrequency**	**accuracyClass**	**measuredRain**	**measuredWind**
M1::M2-RainGauge	×			×	
M1::M2-Anemometer		×	×		×

Pour le *contexte formel intersection ensembliste* (TABLE 6), le mécanisme de construction est plus simple puisque seules les classes ayant exactement les mêmes tuples sont des éléments de l'intersection.

Table 6. Contexte formel pour l'intersection ensembliste

Intersection ensembliste	**timePoint**	**codeQuality**	**waterAmount**
M1-Rain=M2-Precipitation	×	×	×

7.2. AOC-Posets d'intersection sémantique et ensembliste

L'AOC-poset du contexte formel d'intersection sémantique (TABLE 5) des modèles M1 et M2 est représenté en FIGURE 13. Il est composé des deux concepts $C\_Semantic\_Intersection\_0$ et $C\_Semantic\_Intersection\_1$. Le premier de ces concepts regroupe les deux entités métiers `RainGauge` des modèles M1 et M2 ainsi que les seules propriétés communes à ces deux entités : l'attribut `tubeHeight` et le rôle `measuredRain`. Par définition, l'attribut `serialNumber` n'est pas un élément de cette intersection, car il n'apparaît pas dans le modèle M2.

Le second concept correspond à la fusion des entités `Anemometer` des deux modèles. Comme pour l'entité `RainGauge`, les seules propriétés communes sont les attributs `acquisitionFrequency` et `accuracyClass` et le rôle `measuredWind`. `AnemometerType` étant une propriété de la seule entité `Anemometer` de M2, elle n'est pas un élément du modèle union sémantique.

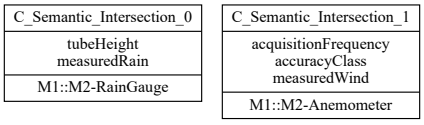

Figure 13. AOC-poset du contexte formel d'intersection sémantique

La FIGURE 14 montre le modèle union sémantique reconstruit à partir de l'AOC-poset de la FIGURE 13. Dans ce modèle, la classe `RainGauge` n'a que l'attribut

tubeHeight et une association vers une classe à définir et à nommer mais dont le rôle est measuredRain. De façon similaire, la classe Anemometer possède les deux attributs acquisitionFrequency et accuracyClass ainsi qu'une association vers une classe non définie ayant pour rôle measuredWind.

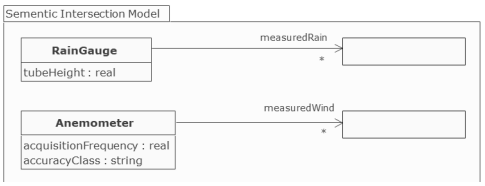

Figure 14. Modèle intersection sémantique des modèles M1 et M2

L'AOC-poset du contexte formel d'intersection ensembliste (TABLE 6) des deux modèles M1 et M2 est représenté en FIGURE 15. Il est simple puisqu'il est constitué du seul concept $C\_Set\_Intersection\_0$ qui regroupe les attributs timePoint, codeQuality et waterAmount communs aux deux classes Rain de M1 et Precipitation de M2.

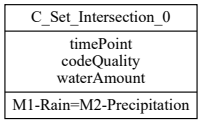

Figure 15. AOC-poset du contexte formel d'intersection ensembliste

Il en résulte que le modèle intersection ensembliste (FIGURE 16) n'a que la seule classe Rain/Precipitation munie des attributs listés dans l'AOC-poset du contexte formel d'intersection ensembliste.

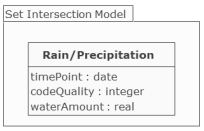

Figure 16. Modèle intersection ensembliste des modèles M1 et M2

8. Ré-analyse de modèle et pistes d'amélioration de la méthodologie de ré-analyse

Dans cette section, nous allons montrer dans un premier temps comment les modèles obtenus peuvent être améliorés au niveau sémantique via une ré-analyse du modèle alignement, ré-analyse qui aurait pu être effectuée de la même façon sur les modèles union. Dans un second temps, nous donnerons quelques pistes pour améliorer la méthodologie de ré-analyse des modèles.

8.1. Ré-analyse par un expert pour une meilleure fusion des modèles M1 et M2

Une fois terminées les étapes de reconstruction des modèles alignement, union sémantique ou ensembliste et intersection sémantique ou ensembliste, il est possible de poursuivre l'analyse des modèles pour les améliorer et surtout pour aboutir à un modèle intégré, plus générique et plus cohérent prenant mieux en compte les besoins et les connaissances des acteurs. Le modèle de la FIGURE 17 est le résultat de cette analyse approfondie, analyse qui ne peut se faire qu'avec le concours d'experts et d'acteurs des domaines concernés.

En premier lieu, l'analyse du modèle alignement remet en cause le choix des gestionnaires d'inscrire la date des relevés de neige sur un cahier de terrain. Cette pratique n'étant pas fiable à long terme, la décision est prise d'ajouter une date (`timePoint`) à l'entité métier `SnowFall`. Il est alors possible de fusionner les classes `Quality-Parameter` et `TimeParameter` et de les remplacer par l'entité `Data` qui a comme attributs `timePoint` et `codeQuality`.

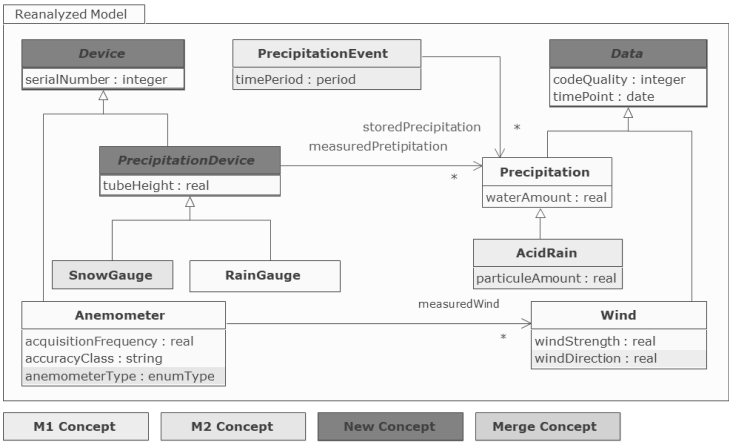

Figure 17. Modèle alignement réanalysé

La poursuite de l'analyse met en évidence que les entités `Anemometer` peuvent être regroupées en une seule classe. La même opération peut être réalisée pour les entités `RainGauge`. N'ayant plus de classe en doublon, il est alors possible de supprimer les paquetages M1 et M2. Comme, du point de vue sémantique, les entités métiers `Rain` et `SnowFall` sont des cas particuliers de l'entité `Precipitation`, on peut considérer de les fusionner en une seule et même entité `Precipitation`. L'entité métier `AcidRain` reste quant à elle inchangée et spécialise l'entité fusionnée `Precipitation`.

Afin que le modèle alignement ré-analysé soit sémantiquement cohérent, d'une part, la classe `RainEvent` est renommée `PrecipitationEvent`, entité de niveau d'abstraction plus élevé et, d'autre part, le rôle de l'association entre les classes `PrecipitationEvent` et `Precipitation` devient `storedPrecipitation`.

Le numéro de série (`serialNumber`) n'est pas une propriété propre à l'entité `RainGauge`. Elle est généralisable à tout instrument de mesure. À ce titre, elle s'applique à `Anemometer` et à `SnowGauge`. Cela conduit à créer une nouvelle classe `Device` contenant l'attribut `serialNumber` généralisant les classes `Anemometer` et `PrecipitationDevice`.

Les associations ayant pour rôle `measuredRain` et `measuredSnowFall` sont généralisées par une association entre les classes `PrecipitationDevice` et `Precipitation`. Le rôle de cette nouvelle association est remplacé par `measuredPrecipitation`, abstraction des précédents rôles.

Enfin, les entités métiers `Breeze` et `Wind` étant maintenant sémantiquement semblables, il est possible de les fusionner en une seule entité `Wind` ayant un niveau d'abstraction plus élevé que `Breeze`.

8.2. Pistes d'amélioration de la méthodologie d'analyse

À aucun moment, dans cette section, il n'a été nécessaire de recourir aux modèles union et intersection qu'ils soient sémantiques ou ensemblistes, car nous travaillions sur des modèles de petite taille.

Toutefois, comme cela a été dit précédemment, le *modèle union sémantique* contient d'une part, l'***ensemble*** des entités métiers des deux modèles M1 et M2 et, d'autre part, l'***ensemble*** des descriptions des entités contenues dans les deux modèles. Il est à noter que ce modèle est toujours plus simple que le modèle alignement car les entités métiers potentiellement fusionnables le sont par construction du modèle. Dans notre cas d'étude, le *modèle union ensembliste* n'a apporté aucune information déterminante pour la fusion de classes ou de descriptions mais il est possible que sur des modèles plus conséquents il puisse présenter un intérêt. Les deux *modèles intersection, sémantique et ensembliste,* ont chacun leur intérêt car ils constituent la plus grande intersection possible entre les modèles M1 et M2 soit du point de vue de la sémantique du nom des objets soit du point de vue ensembliste des tuples.

A contrario de l'union et de l'intersection sémantiques, l'union et l'intersection ensemblistes vont effectuer des fusions d'entités métiers qui ont des noms très différents. Dans certains cas, cette différence mettra en exergue des fusions d'entités thématiquement intéressantes. Elles pourraient en particulier révéler des fautes d'orthographe, des stratégies de nommage différentes ou des glissements de sens. Dans d'autres cas, les fusions n'auront absolument aucun sens du point de vue thématique. C'est pourquoi, l'intervention d'un expert est fortement conseillée pour valider ou non les fusions produites par l'union et l'intersection ensembliste.

Enfin, outre la factorisation automatique de certaines entités, les opérations d'union et d'intersection améliorent et facilitent la méthodologie d'analyse des concepteurs car elles permettent d'obtenir les bornes maximales et minimales soit dans l'espace sémantique soit dans l'espace ensembliste. Selon leur nature, les unions contiennent tous les éléments des modèles et les intersections les éléments du noyau commun.

L'exemple de l'article est volontairement de taille réduite pour exposer la problématique, décrire les mécanismes d'union et d'intersection, etc. mais ces techniques prennent toute leur puissance pour des modèles de tailles importantes. Le modèle SIE Pesticides dont est extrait l'exemple de l'article est constitué de plus de 200 classes. Dans ce contexte, l'expert est sollicité pour toutes les fusions possibles afin de valider celles qui sont thématiquement cohérentes et de refuser celles qui n'ont pas de sens dans le domaine. Pour les modèles conséquents, ces techniques permettent à l'expert de se focaliser sur les fusions proposées et de porter moins d'attention aux éléments non fusionnés. Cela se traduit par un gain de temps et donc de productivité.

9. Travaux connexes

L'analyse et l'intégration de modèles conceptuels UML par la construction de modèles alignement, union ou intersection sont des questions qui font écho à différentes problématiques voisines dans le domaines de bases de données, des ontologies et de l'ingénierie dirigée par les modèles.

En bases de données, les problèmes d'intégration ou d'alignement de schémas ont été largement étudiés (Batini *et al.*, 1986 ; Rahm, Bernstein, 2001 ; Shvaiko, Euzenat, 2005 ; Kashyap, Sheth, 1996), notamment dans l'optique de l'interopérabilité entre systèmes multi-bases de données (Parent, Spaccapietra, 1998). Dans le panorama présenté dans (Parent, Spaccapietra, 1998), plusieurs étapes clefs des approches d'intégration sont décrites. *Une première étape* consiste à transformer les données sur les plans syntaxiques, sémantiques et normalisation de la représentation. Dans notre approche, nous ne modifions pas les modèles dans une phase de pré-traitement pour accroître la découverte de correspondances. Nous nous proposons de les prendre tels qu'ils sont, de faire émerger des premières correspondances et des conflits potentiels, puis de laisser l'expert s'appuyer sur ces correspondances pour faire évoluer les modèles et mettre en valeur des parties communes plus larges. *La seconde étape* identifiée par (Parent, Spaccapietra, 1998) s'intéresse justement à l'identification des correspondances. Les auteurs indiquent qu'elle est assez difficile à réaliser sur le seul niveau schéma (noms et organisation des concepts) et concentrent leur réflexion au niveau des données, qui n'existent pas dans notre cas car nous étudions des modèles conceptuels en cours d'analyse, donc non peuplés d'instances à ce stade. Nous investigons des correspondances basées sur les noms ou sur des descriptions semblables, comme celles décrites dans le domaine des bases de données. *Une troisième étape* est relative aux stratégies d'intégration et nous pouvons y situer notre proposition. Nous présentons ici une approche semi-automatique, qui laisse une large place à l'expertise humaine pour construire de manière itérative un nouveau modèle. A chaque itération, le concepteur assisté par l'expert va pouvoir produire un incrément du nouveau modèle. L'expérience montre que les approches itératives sont particulièrement adaptées pour les modèles de taille conséquente. Notre approche d'alignement met en valeur les correspondances, tandis que les modèles intersection montrent les sous-systèmes strictement identiques et que les modèles union servent de support à une possible ré-

organisation car ces derniers contiennent l'*ensemble des entités* de tous les modèles sources ainsi que l'*ensemble des descriptions* de ces entités.

Les modèles union et intersection nous semblent entrer dans le cadre de la recherche de l'exhaustivité dans l'intégration selon (Parent, Spaccapietra, 1998) : identification de l'union, de l'intersection et des parties complémentaires. Cette vision exhaustive est supposée faciliter d'autres intégrations futures de modèles.

Le problème présenté ici se rapproche également de l'appariement ou de l'alignement d'ontologies (Shvaiko, Euzenat, 2013). Ce problème est souvent traité conjointement avec l'alignement de schémas, comme dans les systèmes COMA++ (Aumueller *et al*., 2005) ou YAM++ (Duchateau, Bellahsene, 2016 ; Ngo, Bellahsene, 2016). Certaines approches traitent ce problème avec l'Analyse Formelle de Concepts (AFC) (Kalfoglou, Schorlemmer, 2005 ; Bendaoud *et al*., 2008). L'approche présentée dans (Stumme, Maedche, 2001) utilise l'AFC et une analyse linguistique pour fusionner des ontologies dans le contexte du Web sémantique. Dans (Formica, 2006), l'alignement utilise une mesure de similarité basée sur l'AFC, tandis que dans (Tatsiopoulos, Boutsinas, 2009), l'alignement s'appuie sur la structure interne de l'ontologie et l'extraction de règles d'association.

Avec le développement du domaine de l'Ingénierie Dirigée par les Modèles, les mêmes problèmes de recherche de correspondances se sont déplacés sur d'autres types de modèles. L'identification de similarités entre modèles ou entre méta-modèles a ainsi été étudiée dans le domaine de la gestion de versions (Altmanninger *et al*., 2009), du développement distribué (Cicchetti *et al*., 2008), ou pour assister le développement de transformation de modèles (Falleri *et al*., 2008). L'algorithme UMLDiff extrait les changements entre deux modèles UML (présumés être des versions successives) dans un but de suivi de l'évolution et de planification de la maintenance (Xing, Stroulia, 2005). Plus généralement, un certain nombre de techniques de *matching* ont vu le jour, dont une bonne partie se focalisant sur les métamodèles (Lafi *et al*., 2014). On peut citer comme exemples l'approche Matchbox qui s'inspire de COMA++ (Voigt, Heinze, 2010), des approches inspirées par le Similarity Flooding (Melnik *et al*., 2002) comme celle de (Falleri *et al*., 2008), utilisant des heuristiques comme décrit dans (Kessentini *et al*., 2014), ou comme AtlanMod Model Weaver (Fabro, Valduriez, 2007), généraliste comme MadMatch qui oeuvre sur des modèles de toutes sortes (Kpodjedo *et al*., 2013), ou semi-automatique comme SAMT4MDE+ et ses extensions (Hammoudi, Lopes, 2005 ; Lafi *et al*., 2016) qui proposent une approche incluant une validation par un expert.

Toutes les approches d'appariement qui se focalisent sur la recherche de correspondances dans le domaine des ontologies ou de l'ingénierie dirigée par les modèles peuvent servir pour affiner la construction des contextes formels de l'alignement, des unions et des intersections. Comparativement à ces approches, à partir des contextes formels, format d'entrée de l'AFC, nous allons au-delà de la simple mise en évidence de correspondances 1-1 ou n-m de concepts, de la recherche de recalage de la sémantique des termes ou de la recherche de listes plates de différences entre les modèles. Nous proposons de nouvelles vues sur les concepts métiers et de nouveaux concepts

plus abstraits. Tous les concepts métiers analysés ou produits sont intégrés dans une structure de spécialisation/généralisation, plutôt que dans une simple liste.

L'approche que nous présentons exploite des qualités des treillis et de l'AFC déjà utilisés en génie logiciel et en base de données pour différents objectifs et, en particulier, pour construire, maintenir ou réorganiser des hiérarchies de classes ou des schémas de bases de données par refactorisation (Missikoff, Scholl, 1989 ; Rundensteiner, 1992 ; Godin, Mili, 1993 ; Snelting, Tip, 2000 ; Huchard, 2015). Certains autres travaux s'intéressent aussi à l'héritage multiple, au typage, à la multiplicité des propriétés (mono-valuées versus multi-valuées), à la redéfinition des propriétés (attributs et méthodes) et à la navigation des associations (Roume, 2004 ; Falleri, 2009 ; Osman-Guédi, 2013). Mais ici l'AFC est une analyse inter-modèles par opposition aux approches intra-modèle précédemment développées. Dans l'approche inter-modèles de cet article, nous conservons la traçabilité des entités en préfixant leur nom par l'identifiant du modèle d'origine. Une approche consistant à construire un modèle GCM sémantique (intersection sémantique) a été présentée dans (Amar *et al.*, 2012). Elle organise également les concepts métiers construits ou modifiés pour leur choix par un utilisateur. Elle a été testée sur des modèles de systèmes d'information décrivant l?impact des pesticides sur l?environnement. Ces modèles ont été la source d'inspiration de l'exemple simplifié choisi dans cet article pour expliquer la méthode. Ces vues union ou intersection ont aussi été partiellement étudiées dans le domaine de la modélisation par *feature model* pour les lignes de produits (Carbonnel *et al.*, 2017).

10. Conclusion

L'alignement est une technique utilisée depuis de nombreuses années pour factoriser, assembler ou fusionner plusieurs schémas de bases de données ou des modèles en un seul. Son automatisation via l'AFC produit des modèles alignement qui sont complexes car, en particulier, les entités métiers de même nom dans différents modèles ne sont pas fusionnées directement comme le montre la FIGURE 8, inconvénient qui augmente avec la taille des modèles.

Les modèles union sémantique et ensembliste réunissent, selon des mécanismes de calcul différents, toutes les entités métiers et toutes les descriptions de ces entités des modèles sources. Même si, pour des raisons de réutilisation du code, l'algorithme développé calcule le modèle LCM à partir du modèle alignement et le modèle GCM à partir du modèle LCM, les cinq modèles peuvent être construits indépendamment.

La ré-analyse d'un modèle union est normalement plus simple que l'analyse d'un modèle alignement. Dans notre cas d'étude, cela s'est vérifié pour le modèle union sémantique mais pas pour le modèle union ensembliste qui est identique au modèle alignement. Il faudra vérifier si ce dernier constat se reproduit pour d'autres modèles sources. Selon le mécanisme de fusion adopté, ces modèles sont toujours les plus petits des modèles union calculés à partir de modèles sources.

Selon le mécanisme sémantique ou ensembliste mis en œuvre pour les produire, les modèles intersection sont quant à eux constitués des seuls éléments communs aux modèles sources. Ce sont les plus grandes des intersections des modèles sources, c'est-à-dire le noyau de tous les modèles. À ce titre, ils regroupent le capital de connaissances commun à plusieurs modèles, capital qui peut être mobilisé pour le développement d'une nouvelle application, d'un nouveau système d'information, etc.

Le modèle alignement ainsi que les modèles union et intersection qu'ils soient sémantique ou ensembliste sont des modèles obtenus automatiquement (par les algorithmes développés dans le cadre de ce travail). Pour rendre utilisables ces modèles, l'expertise humaine est indispensable comme le montre la section 5 pour le nommage des nouvelles entités mais aussi la section 8 où la fusion ou la réorganisation de certaines entités ne peut être faite que par des experts des domaines mobilisés.

Une particularité de l'AFC dans ce contexte de factorisation inter-modèles est que cette méthode produit simultanément une factorisation intra-modèle et il est difficile de découpler ces deux mécanismes de factorisation. Dans ce contexte-là, la seule solution à notre connaissance est l'intervention d'experts pour faire des réarrangements intra ou inter-modèles. Toutefois, il est aussi possible d'assister les experts à l'aide d'un arbre de décision comme présenté dans (Amar *et al.*, 2012).

Il est à noter que ces modèles union et intersection qu'ils soient sémantiques ou ensemblistes peuvent être calculés de façon itérative. En effet, la factorisation d'un modèle union (résultat de la factorisation de deux premiers modèles par exemple) avec un nouveau modèle va donner un nouveau modèle union contenant l'ensemble de toutes les entités métiers et leurs descriptions de tous les modèles sources. Il en est de même pour le modèle intersection, l'intersection d'un modèle intersection avec un nouveau modèle va donner un nouveau modèle intersection qui ne sera composé que des entités métiers et leurs descriptions de l'ensemble des modèles sources.

Les travaux vont se poursuivre dans plusieurs directions. Tout d'abord, on peut définir des opérations d'union et d'intersection basées sur les n-uplets d'éléments (attributs et rôles) et non sur les noms des classes, ce qui peut faire émerger des informations complémentaires. De plus, la capacité de factorisation de l'AFC étant limitée, nous comptons mettre en œuvre l'Analyse Relationnelle de Concepts (ARC) qui étend l'analyse aux relations.

Dans cette étude de cas, il n'y a pas d'ambiguïté sur le nom des éléments de modélisation (classes, attributs, etc.) car le modèle est de petite taille et relève d'un domaine unique. Pour des modèles plus conséquents et/ou complexes impliquant plusieurs domaines, il est fort probable de rencontrer les ambiguïtés (par exemple la *Forêt* d'un domaine forestier et le *Foret* utilisé pour faire des perçages). Dans ce cas, nous envisageons de conditionner la fusion des contextes (FIGURE 6) à une analyse des éléments de modélisation environnants mais aussi d'utiliser des ontologies métiers, des ressources lexicales et des outils de Traitements Automatiques du Langage. De plus, nous proposerons une solution pour les attributs optionnels ou obligatoires s'ils s'en présentent.

Nous allons poursuivre l'étude sur des modèles déjà mobilisés dans des travaux antérieurs (Amar *et al*., 2012 ; Miralles *et al*., 2014) pour évaluer la pertinence des alignements en termes de précision et de rappel. Nous visons en effet d'utiliser dans ce but le modèle SIE Pesticides pour lequel nous disposons d'une quinzaine de versions du modèle d'analyse, suite à des évolutions et des enrichissements lors de séances d'analyse avec des acteurs. En fonction des versions, le nombre de classes évolue d'une quarantaine à un peu plus de deux cents.

Bibliographie

Alliance A. (2001). *Manifesto for agile software development* (vol. 2005) n° June.

Altmanninger K., Seidl M., Wimmer M. (2009). A survey on model versioning approaches. *International Journal of Web Information Systems*, vol. 5, n° 3, p. 271–304.

Amar B., Guédi A. O., Miralles A., Huchard M., Libourel T., Nebut C. (2012). Finding Semi-Automatically a Greatest Common Model Thanks to Formal Concept Analysis. In *Revised selected papers of ICEIS 2012, LNBIP vol. 141*, p. 72–91.

Aumueller D., Do H. H., Massmann S., Rahm E. (2005). Schema and ontology matching with COMA++. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, p. 906–908. Consulté sur http://doi.acm.org/10.1145/1066157.1066283

Barbut M., Monjardet B. (1970). *Ordre et classification (volume 2)*. Hachette.

Batini C., Lenzerini M., Navathe S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computer Survey*, vol. 18, p. 323–364.

Beck K. (2000). *extreme programming explained - embrace change*. Addison-Wesley.

Bendaoud R., Napoli A., Toussaint Y. (2008). Formal Concept Analysis: A unified framework for building and refining ontologies. In *EKAW 2008*, p. 156-171.

Carbonnel J., Huchard M., Miralles A., Nebut C. (2017). Feature model composition assisted by formal concept analysis. In *ENASE 2017 - Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering, Porto, Portugal, April 28-29, 2017*, p. 27–37. Consulté sur https://doi.org/10.5220/0006276600270037

Cicchetti A., Ruscio D., Pierantonio A. (2008). Managing model conflicts in distributed development. In *MoDELS 2008*, p. 311–325.

Duchateau F., Bellahsene Z. (2016). YAM: A step forward for generating a dedicated schema matcher. *Trans. Large-Scale Data- and Knowledge-Centered Systems*, vol. 25, p. 150–185. Consulté sur https://doi.org/10.1007/978-3-662-49534-6_5

Fabro M. D. D., Valduriez P. (2007). Semi-automatic model integration using matching transformations and weaving models. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC), Seoul, Korea, March 11-15, 2007*, p. 963–970. Consulté sur http://doi.acm.org/10.1145/1244002.1244215

Falleri J.-R. (2009). *Contributions à l'IDM : reconstruction et alignement de modèles de classes*. Thèse de doctorat, Université Montpellier 2.

Falleri J.-R., Huchard M., Lafourcade M., Nebut C. (2008). Metamodel Matching for Automatic Model Transformation Generation. In *MoDELS 2008*, p. 326–340.

Formica A. (2006). Ontology-based concept similarity in Formal Concept Analysis. *Information Sciences*, vol. 176, p. 2624–2641.

Ganter B., Wille R. (1999). *Formal concept analysis: Mathematical foundation*. Springer-Verlag Berlin.

Godin R., Mili H. (1993). Building and maintaining analysis-level class hierarchies using Galois lattices. In *OOPSLA*, p. 394–410.

Hammoudi S., Lopes D. (2005). From mapping specification to model transformation in MDA: conceptualization and prototyping. In *Web Services and Model-Driven Enterprise Information Services, Proceedings of the Joint Workshop on Web Services and Model-Driven Enterprise Information Services, WSMDEIS 2005, In conjunction with ICEIS 2005, Miami, USA, May 2005*, p. 110–121.

Huchard M. (2015). Analyzing inheritance hierarchies through formal concept analysis: A 22-years walk in a landscape of conceptual structures. In *MASPEGHI@ECOOP 2015, ACM Digital Library*, p. 8–13.

Kalfoglou Y., Schorlemmer M. (2005). Ontology mapping: The state of the art. In *Semantic interoperability and integration, Dagstuhl Seminar proceedings.*

Kashyap V., Sheth A. P. (1996). Semantic and schematic similarities between database objects: A context-based approach. *VLDB J.*, vol. 5, n° 4, p. 276–304. Consulté sur https://doi.org/10.1007/s007780050029

Kessentini M., Ouni A., Langer P., Wimmer M., Bechikh S. (2014). Search-based metamodel matching with structural and syntactic measures. *Journal of Systems and Software*, vol. 97, p. 1–14. Consulté sur https://doi.org/10.1016/j.jss.2014.06.040

Kpodjedo S., Ricca F., Galinier P., Antoniol G., Guéhéneuc Y. (2013). Madmatch: Many-to-many approximate diagram matching for design comparison. *IEEE Trans. Software Eng.*, vol. 39, n° 8, p. 1090–1111. Consulté sur https://doi.org/10.1109/TSE.2013.9

Kruchten P. B. (1999). *The rational unified process: An introduction* (3rd éd.). Addison-Wesley.

Lafi L., Feki J., Hammoudi S. (2014). Metamodel matching techniques: Review, comparison and evaluation. *IJISMD*, vol. 5, n° 2, p. 70–94. Consulté sur https://doi.org/10.4018/ijismd.2014040104

Lafi L., Feki J., Hammoudi S. (2016). Towards a hybrid semi-automatic technique for metamodel matching. In *MODELSWARD 2016 - proceedings of the 4rd international conference on model-driven engineering and software development, rome, italy, 19-21 february, 2016.*, p. 421–426. Consulté sur http://ieeexplore.ieee.org/document/7954389

Melnik S., Garcia-Molina H., Rahm E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, p. 117–128. Consulté sur https://doi.org/10.1109/ICDE.2002.994702

Miralles A. (2016). *Contribution à une conception rationnelle et malléable des systèmes d'information environnementaux*. Habilitation á diriger les recherches. (in french)

Miralles A., Dolques X., Huchard M., Le Ber F., Libourel T., et. al. (2014). Exploration de la factorisation d'un modèle de classes sous contrôle des acteurs. In *Actes du XXXIIème Congrès INFORSID, Lyon, France, 20-23 Mai 2014*, p. 245–261.

Miralles A., Pinet F., Carluer N., Vernier F., Bimonte S., Lauvernet C. *et al.* (2011). EIS-Pesticide: an information system for data and knowledge capitalization and analysis. In *Euraqua-peer scientific conference, 26/10/2011 - 28/10/2011*, p. 1. Montpellier, FRA.

Missikoff M., Scholl M. (1989). An Algorithm for Insertion into a Lattice: Application to Type Classification. In *Proceedings of the 3rd Int. Conf. FODD 1989*, p. 64–82.

Ngo D., Bellahsene Z. (2016). Overview of YAM++ - (not) yet another matcher for ontology alignment task. *J. Web Sem.*, vol. 41, p. 30–49. Consulté sur https://doi.org/10.1016/j.websem.2016.09.002

Osman-Guédi A. (2013). *Transformation automatisée de modèles de Systèmes d'Information*. Thèse de doctorat, Université Montpellier 2.

Parent C., Spaccapietra S. (1998, May). Issues and approaches of database integration. *Communication of the ACM*, vol. 41, p. 166–178.

Rahm E., Bernstein P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, vol. 10, p. 334–350.

Roume C. (2004). *Analyse et restructuration de hiérarchies de classes*. Thèse de doctorat, Université Montpellier 2.

Royce W. W. (1970, août). Managing the development of large software systems. In *Proceedings of IEEE WESCON*, p. 1-9.

Rundensteiner E. A. (1992). *A Class Classification Algorithm For Supporting Consistent Object Views*. Rapport technique. University of Michigan.

Shvaiko P., Euzenat J. (2005). A Survey of Schema-Based Matching Approaches Journal on Data Semantics IV. In *Journal on data semantics IV*, vol. 3730, p. 146–171.

Shvaiko P., Euzenat J. (2013). Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, vol. 25, n° 1, p. 158–176. Consulté sur https://doi.org/10.1109/TKDE.2011.253

Snelting G., Tip F. (2000). Understanding class hierarchies using concept analysis. *ACM Trans. Program. Lang. Syst.*, vol. 22, n° 3, p. 540–582. Consulté sur http://doi.acm.org/10.1145/353926.353940

Stumme G., Maedche A. (2001). Ontology merging for federated ontologies on the semantic web. In *Int. work. foundations of models for information integration (FMII)*, p. 413–418.

Tatsiopoulos C., Boutsinas B. (2009). Ontology mapping based on association rule mining. In *ICEIS 2009*, p. 33-40.

Voigt K., Heinze T. (2010). Metamodel matching based on planar graph edit distance. In *ICMT 2010*, p. 245–259.

Xing Z., Stroulia E. (2005). Umldiff: An algorithm for object-oriented design differencing. In *Proceedings of the 20th ieee/acm international conference on automated software engineering*, p. 54–65. New York, NY, USA, ACM. Consulté sur http://doi.acm.org/10.1145/1101908.1101919