Alignement, union et intersection de modèles : 3 transformations pour l'analyse des systèmes d'information

André Miralles, Marianne Huchard, Jessie Carbonnel, Clémentine Nebut

Alignement, union et intersection de modèles : 3 transformations 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, trois transformations sont proposées afin d'assister cette intégration et d'améliorer les modèles : la première produit un modèle d'alignement montrant les correspondances entre les modèles, la seconde produit un modèle union (Least Common Multiple Model ou LCM) et la troisième produit un modèle intersection (Greatest Common Model ou GCM) constitué des seuls éléments communs à tous les modèles. Le LCM est la plus petite des unions des modèles et le GCM, noyau de tous les modèles, est la plus grande des intersections des modèles. Ces transformations sont réalisées à l'aide de l'Analyse Formelle de Concepts (AFC) en basant les transformations sur des opérations portant sur les contextes formels.*

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, three transformations are proposed to assist this integration: an alignment model which highlights the correspondences between the models; a union model (Least Common Multiple Model or LCM) and the intersection model (Greatest Common Model or GCM) built up with the common elements. The LCM is the smallest union of the models and the GCM, kernel of all models, is the greatest intersection of the models. These transformations 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

Quelle que soit l'application informatique à développer et a fortiori le système d'information, l'analyse du système et des besoins des utilisateurs est probablement la phase 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 s'approprier les entités pour le capturer 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 ces conditions, le modèle est alors le principal produit de l'analyse. Une conséquence directe est que le modèle 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. Cela 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.

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 d'acteurs. Un autre avantage de cette analyse en groupe 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 l'analyse en groupe est la nécessité de disposer de méthodes et d'outils pour fusionner les modèles produits en un seul. Cette problématique d'intégration se rencontre dans d'autres situations (Miralles, 2016) et en particulier 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 factorisation soit par des techniques d'alignement de schémas (Batini *et al.*, 1986; Rahm, Bernstein, 2001; Shvaiko, Euzenat, 2005) soit, plus récemment, en mettant en œuvre l'Analyse Formelle de Concepts (Stumme, Maedche, 2001; Amar *et al.*, 2012). 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) : le modèle d'alignement qui est le plus proche de l'alignement de schémas standard, le LCM (Least Common Multiple Model) qui est l'intégration minimale et complète

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`.

des modèles et le 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.

Le contexte et la problématique sont présentés en section 1. La section 2 rappelle le principe et les produits résultant de l'Analyse Formelle de Concepts qui permettent de reconstruire des modèles d'alignement, d'union et d'intersection obtenus à partir des contextes formels associés, contextes qui sont définis en section 3. La section 4 illustre l'approche par un cas d'étude. Enfin, avant d'aborder la conclusion en section 6, la section 5 présente les travaux connexes qui ont inspiré et alimenté notre réflexion.

2. 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 2 (modèle M1[2]), 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`.

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. 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. Par exemple le concept $Concept\_M1\_6$=({-

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

Table 1. Contexte formel K_{M1}

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

$Wind, Rain, AcidRain\}, \{timePoint, codeQuality\}$) regroupe trois classes partageant les deux attributs de son intension (les exemples sont extraits de la Figure 1).

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 $E_2 \subseteq E_1$ (et de manière équivalente $I_1 \subseteq I_2$). 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 les schémas pour des raisons de lisibilité. C'est ainsi que la représentation graphique de $Concept\_M1\_6$ ne présente que les caractéristiques introduites $\{timePoint, codeQuality\}$ et celle de $Concept\_M1\_5$ ne présente que la caractéristique introduite $\{waterAmount\}$ et l'entité introduite $\{Rain\}$. 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. 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.

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 1 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 la section 4. 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`, doit en être une sous-classe. Quelquefois ces

relations ne sont pas explicites dans le modèle et elles le deviennent dans l'AOC-poset. Dans la section suivante, nous étudions une nouvelle utilisation de l'AFC, cette fois en présence de plusieurs modèles UML, pour analyser leurs correspondances et leur possible intégration.

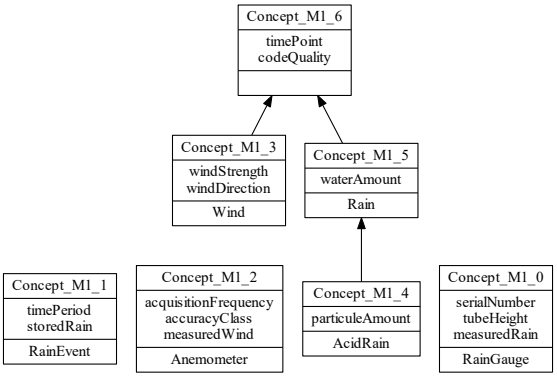

Figure 1. AOC-poset du contexte formel K_{M1} de station météorologique

3. Contextes formels pour l'alignement, l'union et l'intersection de modèles

Trois contextes formels vont nous permettre d'analyser respectivement l'alignement, l'union et l'intersection de modèles. Différentes hypothèses de construction de ces contextes peuvent être faites. Ici, nous supposons que la terminologie des modèles a été uniformisée au préalable. Ainsi deux classes (resp. deux attributs) portant le même nom seront supposées représenter les mêmes notions et avoir une même sémantique.

Le *contexte formel d'alignement*, présenté en haut de la Table 2, consiste à concaténer les contextes formels associés aux modèles M1 et M2, après avoir préfixé les noms des classes du nom du modèle dont elles proviennent. Dans cet article, pour des raisons de place nous ne présentons pas séparément le contexte formel du modèle M2. Il peut être retrouvé facilement à partir de la partie inférieure de la Table 2 (partie haute) : il correspond à toutes les lignes préfixées par "M2-". 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 (ou des deux).

Dans le *contexte formel union*, présenté au centre de la Table 2, on retrouve les colonnes du contexte formel d'alignement. Par contre, les lignes du contexte d'alignement qui correspondent à des classes portant des noms identiques sont fusionnées. Dans notre exemple, ce sera le cas pour `M1-RainGauge` et `M2-RainGauge`, fusionnées dans la ligne `M1::M2-RainGauge` ou pour `M1-Anemometer` et `M2-Anemometer`, fusionnées dans la ligne `M1::M2-Anemometer`. Pour ces classes "com-

Table 2. Contextes formels pour l'alignement, l'union et l'intersection sémantique

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		×														×

LCM (Union)	**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		×														×

GCM (Intersection)	**tubeHeight**	**acquisitionFrequency**	**accuracyClass**	**measuredRain**	**measuredWind**
M1::M2-RainGauge	×			×	
M1::M2-Anemometer		×	×		×

munes", la description est l'union des descriptions (attributs et rôles) des deux modèles. 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.

Dans le *contexte formel intersection*, présenté au bas de la Table 2, on ne garde que les classes communes et les colonnes du contexte formel d'alignement qui sont associées à une classe commune aux deux modèles et ceci dans les deux modèles. Ainsi, l'attribut `serialNumber`, qui n'est associé à `RainGauge` que dans le modèle M1,

n'apparaît pas. Une classe n'est donc associée à un attribut ou à un rôle que si elle le possède dans les deux modèles.

4. 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 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 cas portera sur la fusion de deux d'entre eux.

4.1. Présentation des deux modèles à fusionner

La station météorologique, dont le modèle est celui de la Figure 2, 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 (`codeQuality`). 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`.

Le modèle de la Figure 3 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. Pour cette station, 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

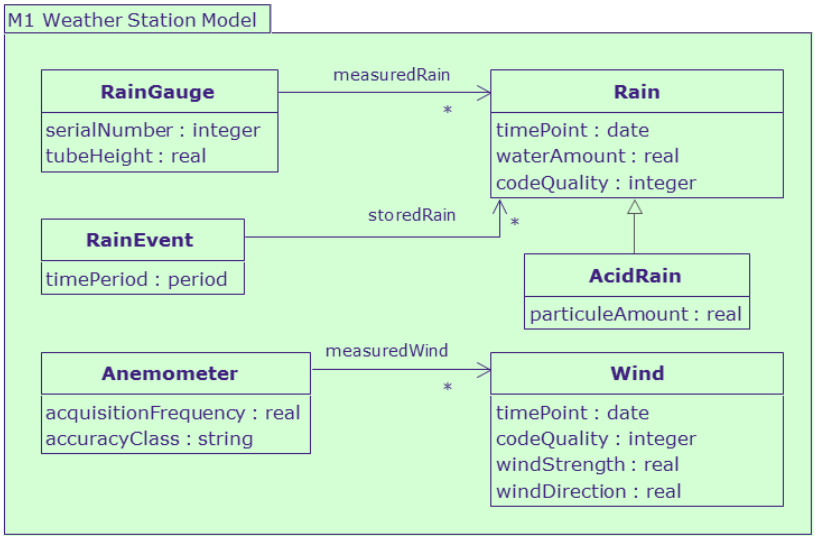

Figure 2. Modèle de la station météorologique M1

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`),

– 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.

4.2. Alignement des modèles M1 et M2

L'AOC-poset du contexte formel d'alignement (Table 2) des modèles M1 et M2 est donné en Figure 4. Cet AOC-poset permet de reconstruire le modèle UML de la Figure 5 représentant l'alignement des modèles M1 et M2.

Une première analyse rapide de l'AOC-poset permet de constater la présence de trois nouveaux concepts ($Concept\_Alignment\_8$, $Concept\_Alignment\_11$ et $Concept\_Alignment\_12$) issus du calcul de l'AFC. 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

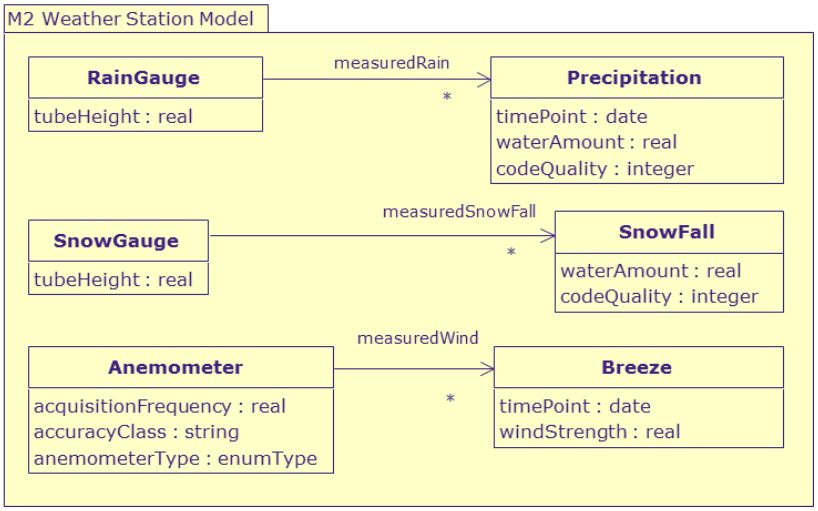

Figure 3. Modèle de la station météorologique M2

métiers qui sont `PrecipitationDevice`, `TimeParameter` et `QualityParameter` respectivement (cf. Figure 5).

L'AFC fusionne aussi les entités métiers `Rain` du modèle M1 et `Precipitation` du modèle M2 au sein du concept $Concept\_Alignment\_9$. Ce n'est pas surprenant au vu des attributs les décrivant. Ce concept représente l'entité métier `Rain/Precipitation` de la Figure 5. Les quatre entités métiers évoquées ci-dessus sont extérieures aux modèles M1 et M2 car elles sont nouvelles.

Dans l'AOC-poset de la Figure 4, les entités métiers `Anemometer` de M1 et M2 apparaissent dans deux concepts distincts ($Concept\_Alignment\_3$ et $Concept\_Alignment\_7$) donnant ainsi naissance à deux classes dans le modèle d'alignement de la Figure 5. 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` ($Concept\_Alignment\_6$) alors que l'attribut `serialNumber` est propre à l'entité métier `RainGauge` de M1 ($Concept\_Alignment\_0$), d'où la relation de spécialisation en Figure 5. La présence des doubles occurrences de `Anemometer` et de `RainGauge` impose le maintien des modèles M1 et M2 au sein du modèle d'alignement (`Alignment Model`) de la Figure 5.

L'entité métier `Anemometer` est en relation avec `Wind` au travers d'une association dont le rôle est `measuredWind`. Le concept $Concept\_Alignment\_5$ 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` (cf. Figure 5).

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 fusionnée `Rain/Precipitation` qui a pour rôle `storedRain`. `AcidRain` spécialise l'entité fusionnée `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`.

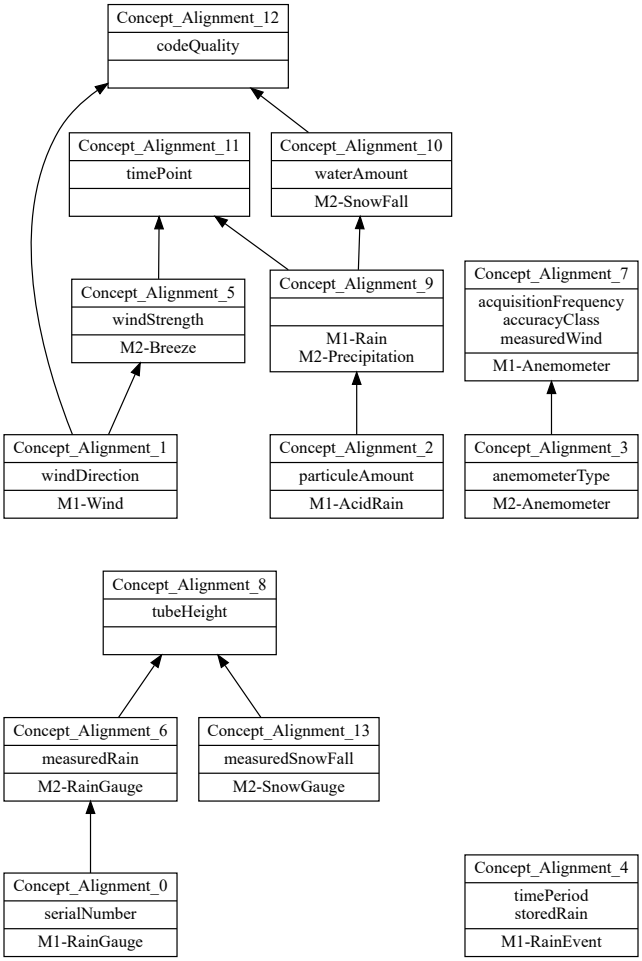

Figure 4. AOC-poset du contexte formel d'alignement

4.3. Union des modèles M1 et M2

La Figure 6 montre l'AOC-poset du contexte formel d'union (Table 2) des modèles M1 et M2. Par construction (cf. section 3), le contexte formel d'union est quasi-

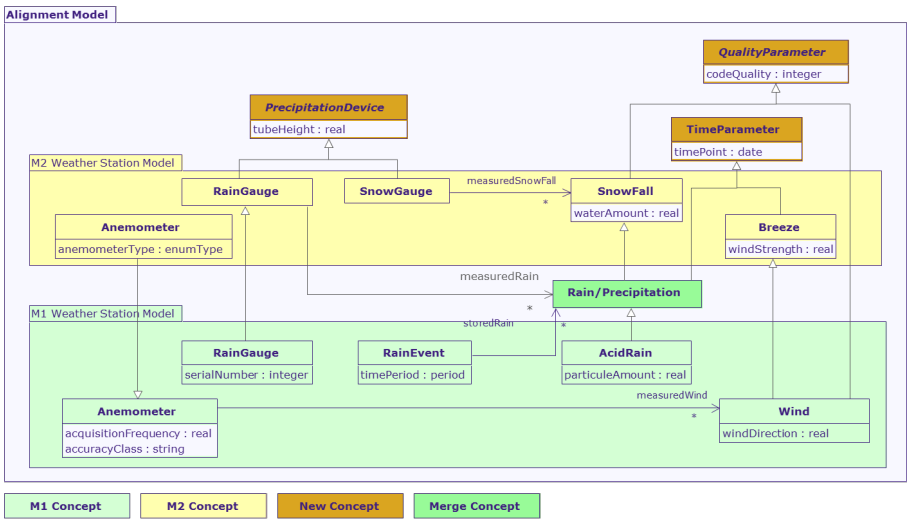

Figure 5. Modèle d'Alignement des modèles M1 et M2

ment identique à celui du contexte formel d'alignement (cf. Figure 5). Au nom et à la numérotation des concepts près (identifiants générés automatiquement), les seules différences résident dans :

– la fusion des concepts $Concept\_Alignment\_3$ et $Concept\_Alignment\_7$ de la Figure 5 pour donner le concept $Concept\_Union\_4$ de la Figure 6, concept où on retrouve les propriétés des entités métiers `Anemometer` des deux modèles M1 et M2,

– l'assimilation des deux concepts $Concept\_Alignment\_0$ et $Concept\_Alignment\_6$ de la Figure 5 au sein du concept $Concept\_Union\_0$ de la Figure 6 ; les propriétés des entités `RainGauge` des deux modèles M1 et M2 sont réunies dans le concept $Concept\_Union\_0$.

Il en résulte que le modèle LCM de la Figure 7, issu du calcul du contexte formel d'union, est plus simple et plus facile à analyser que le modèle d'alignement de la Figure 5. Cela se traduit par un gain de productivité lors de l'analyse approfondie puisque celle-ci est plus rapide. Une analyse approfondie comme celle du modèle d'alignement (cf. 4.2) aboutirait plus rapidement au même résultat comme ce sera présenté en section 4.5.

4.4. Intersection des modèles M1 et M2

L'AOC-poset du contexte formel d'intersection (Table 2) des modèles M1 et M2 est représenté en Figure 8. Il est composé des deux concepts $Concept\_Intersection\_0$ et $Concept\_Intersection\_1$. 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`.

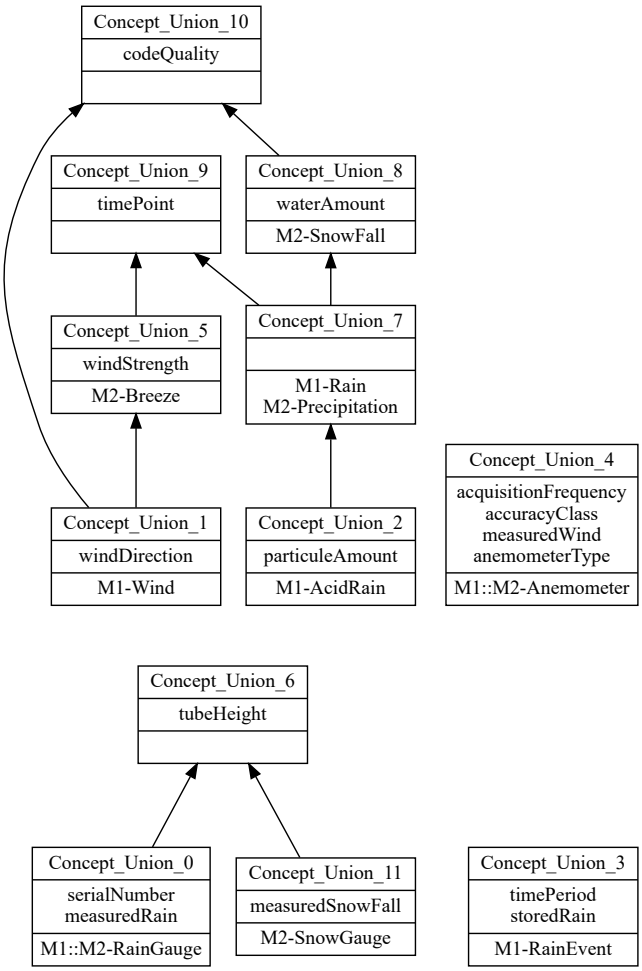

Figure 6. AOC-poset du contexte formel d'union

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 GCM.

La Figure 9 montre le GCM reconstruit à partir de l'AOC-poset de la Figure 8. Dans le GCM, 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 `acquisitionFre-`

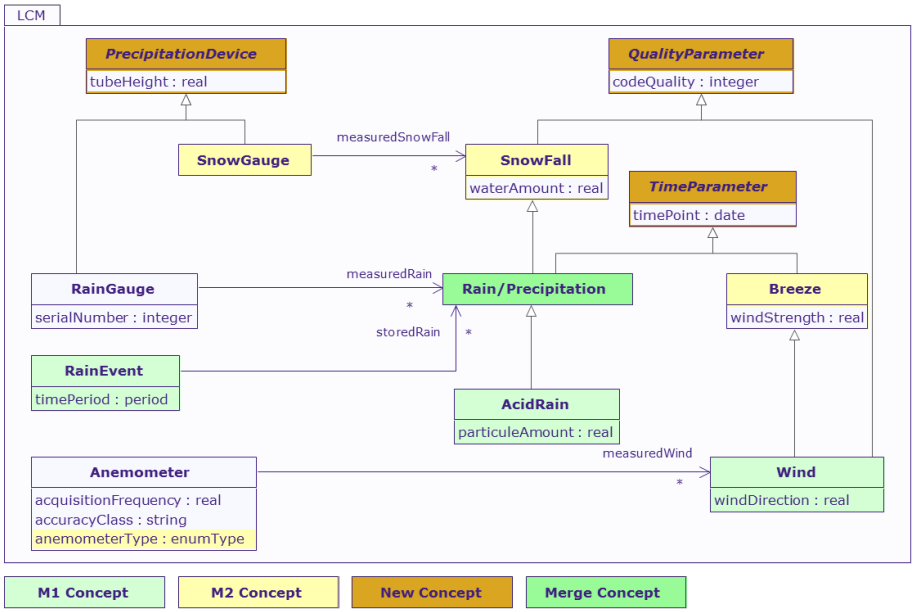

Figure 7. LCM des modèles M1 et M2

`quency` et `accuracyClass` ainsi qu'une association vers une classe non définie ayant pour rôle `measuredWind`.

Même si, pour des raisons de réutilisation du code, l'algorithme développé calcule le modèle LCM à partir du modèle d'alignement et le modèle GCM à partir du modèle LCM, ces trois modèles peuvent être construits indépendamment.

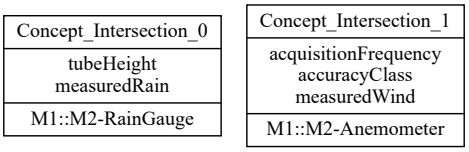

Figure 8. AOC-poset du contexte formel d'intersection

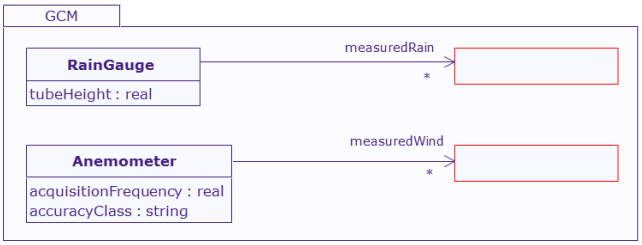

Figure 9. GCM des modèles M1 et M2

4.5. Ré-analyse pour l'intégration des modèles M1 et M2

Une fois les étapes de reconstruction des modèles d'alignement, d'union et d'intersection terminées, 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. Le modèle de la Figure 10 est le résultat de cette analyse approfondie, analyse qui ne peut se faire qu'avec le concours d'un expert du domaine. En premier lieu, l'analyse du modèle d'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 `QualityParameter` et `TimeParameter` et de les remplacer par l'entité `Data` qui a comme attributs `timePoint` et `codeQuality`. 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`, il est normal 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 d'alignement réanalysé 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 `RainGauge`. Elle est généralisable à tout instrument de mesure. A 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 sémantiquement semblables, il est possible de les fusionner en une seule entité `Wind` ayant un niveau d'abstraction plus élevé que `Breeze`.

5. Travaux connexes

Les treillis et l'AFC ont été 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 au typage et à la multiplicité des propriétés et à la navigation des associations (Roume, 2004; J.-R. Falleri, 2009; Osman-Guédi, 2013).

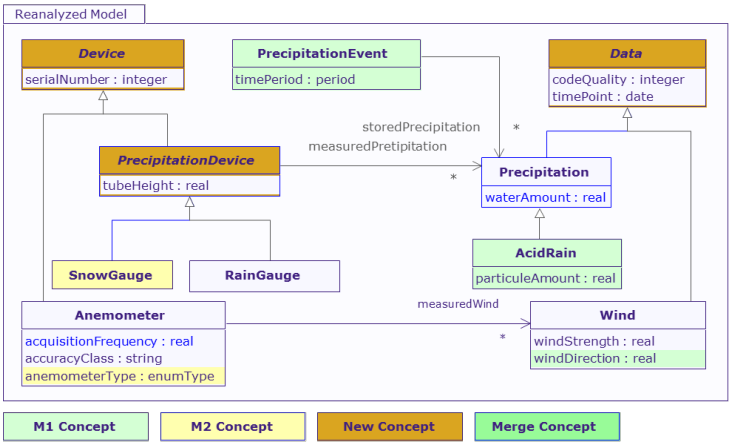

Figure 10. Modèle d'alignement réanalysé

La présente approche est différente puisqu'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 le modèle GCM 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 portant sur les Pesticides. Ici, nous proposons, en plus du modèle GCM, une vue sur l'alignement et l'union de modèles.

Les problèmes d'intégration ou d'alignement de bases de données ont été largement étudiés, notamment pour produire un schéma global (Batini *et al.*, 1986; Rahm, Bernstein, 2001; Shvaiko, Euzenat, 2005). Toutefois, les travaux d'interopérabilité des données entre systèmes multi-bases de données posent des problèmes de même nature (Parent, Spaccapietra, 1998). Pour ces deux types de problématiques, l'intégration des bases et l'interopérabilité des données consistent à trouver des correspondances entre concepts des différents schémas et à les ré-organiser dans une même structure. Dans notre approche, nous nous focalisons au niveau modèle/schéma alors que (Parent, Spaccapietra, 1998) concentrent leur réflexion au niveau des données. Cette approche d'alignement met en valeur les correspondances, tandis que le GCM montre les sous-systèmes strictement identiques et que le LCM sert de support à une possible ré-organisation.

L'identification de similarités entre modèles ou entre méta-modèles a 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 (J. Falleri *et al.*, 2008; Voigt, Heinze, 2010).

Le problème présenté ici se rapproche également de l'appariement ou de l'alignement d'ontologies, que certaines approches traitent avec l'AFC (Kalfoglou, Schorlemmer, 2005; Bendaoud *et al.*, 2008). L'approche présentée dans (Stumme, Maed-

che, 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. Toutes ces approches se focalisent sur la recherche de correspondances, elles peuvent servir pour affiner la construction des contextes formels d'alignement, de LCM ou de GCM. A partir des contextes formels, nous allons au-delà de la simple mise en évidence de correspondances, et nous proposons de nouvelles vues sur les concepts métiers et de nouveaux concepts plus abstraits. Tous les concepts métiers sont intégrés dans une structure de spécialisation.

6. 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 d'alignement qui sont complexes car, en particulier, les entités métiers de même nom dans différents modèles ne sont pas fusionnées directement. Le modèle LCM qui réunit tous les éléments des modèles sources élimine cet inconvénient et rend plus facile la ré-analyse du modèle. C'est la plus petite des unions des modèles sources. Le modèle GCM est quant à lui constitué des seuls éléments communs aux modèles sources. C'est la plus grande des intersections des modèles sources, c'est-à-dire le noyau de tous les modèles. Il est incontournable puisque tous les acteurs, codes, schémas de bases, etc. utilisent les éléments du GCM. Ce noyau est le capital de base pour le développement d'une nouvelle application ou d'un nouveau système d'information.

Les modèles d'alignement, LCM et GCM sont des modèles obtenus automatiquement (par les algorithmes développés dans le cadre de ce travail). Pour rendre utilisables ces modèles (obtenir le modèle ré-analysé), l'expertise humaine est indispensable comme le montre la section 4.2 pour le nommage des nouvelles entités mais aussi la section 4.5 où la fusion ou la réorganisation de certaines entités ne peut être faite que par un expert du domaine.

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

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. Enfin, lorsque plus de deux modèles doivent être intégrés, nous envisageons de définir un mode itératif d'intégration.

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 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 factorisation 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. Dans (Carbonnel *et al.*, 2017), nous utilisons une variante de l'approche présentée ici en construisant les modèles d'alignement, LCM et GMC non plus sur la sémantique des noms des éléments de modélisation mais en s'appuyant sur les caractéristiques définissant les entités.

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.

References

Alliance A. (2001). *Manifesto for agile software development* (Vol. 2005) No. June.

Altmanninger K., Seidl M., Wimmer M. (2009). A survey on model versioning approaches. *International Journal of Web Information Systems*, Vol. 5, No. 3, pp. 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*, pp. 72–91.

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, pp. 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 *To appear in 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE) 2017, April 28 - 29, 2017, Porto, Portugal.*

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

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

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

Formica A. (2006). Ontology-based concept similarity in Formal Concept Analysis. *Information Sciences*, Vol. 176, pp. 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*, pp. 394–410.

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*, pp. 8–13.

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

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

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

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*, pp. 245–261.

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*, pp. 64–82.

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, pp. 166–178.

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

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

Rundensteiner E. A. (1992). *A Class Classification Algorithm For Supporting Consistent Object Views*. Technical report. 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, pp. 146–171.

Snelting G., Tip F. (2000). Understanding class hierarchies using concept analysis. *ACM Trans. Program. Lang. Syst.*, Vol. 22, No. 3, pp. 540–582. Retrieved from 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)*, pp. 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*, pp. 245–259.