Sizing the underlying factorization structure of a class model

Abdoulkader Osman Guédi*, Marianne Huchard‡,
André Miralles†, Clémentine Nebut‡
*Université de Djibouti, Avenue Georges Clemenceau BP: 1904 Djibouti (REP)
Email: first.last@univ.edu.dj
†Tetis/IRSTEA, Maison de la télédétection, 500 rue J.-F. Breton 34093 Montpellier Cdx 5, France
Email: first.last@teledetection.fr
‡LIRMM, Univ. Montpellier 2 et CNRS, 161, rue Ada, F-34392 Montpellier Cdx 5, France
Email: first.last@lirmm.fr

***Abstract*—The design of class models for information systems, databases or programming is a delicate process in which experts of the domain and designers have to identify and agree on the domain concepts. Formal Concept Analysis (FCA) has been proposed for supporting this collaborative work and fostering the emergence of higher level entities and the factorization of descriptions and behaviors. More recently, an extension of FCA, Relational Concept Analysis (RCA), has been designed to extend the scope of FCA to the emergence of higher level domain associations. FCA and RCA build a kind of normal form for models, in which the factorization is exhaustive, and the specialization order is adequate. The counterpart of these strong properties is a worst-case exponential theoretical complexity. In this paper, we study a practical application of RCA on several versions of a real class model in order to give precise figures about RCA and to detect which configurations are tractable.**

***Keywords*-Class model factorization, Formal Concept Analysis**

I. Introduction

Designing class models for information systems, enterprise systems, databases or programs is a recurrent activity, that usually involves domain experts and designers that have to identify and organize domain concepts. Formal Concept Analysis (FCA) and its variant Relational Concept Analysis (RCA) have been proposed to assist this phase, so as to introduce new abstractions emerging from the identified domain concepts. FCA classifies entities having characteristics, and RCA also takes into account links between the entities. Both FCA and RCA result in a concept lattice with a specialization-generalization order. This lattice can be exploited so as to obtain the generalization structure for the class model.

Nevertheless, while this whole factorization structure of a class model is a mathematical object with strong theoretical properties, its practical use might suffer from limitations due to the large size of the obtained lattice. A well known risk using it, as it is currently the case with data mining methods is to generate too many abstractions. In such a case, domain experts might be overwhelmed by the information produced making it difficult (or even impossible) to assess the relevance of all these new abstractions.

	flying	nocturnal	migratory	with crest	with membrane
flying squirrel	x				x
bat	x	x			x
ostrich					
flamingo	x		x		
hoopoe	x		x	x	

Table I
Context for the animals

In this paper we want to assess, in a case study taken from a real-world system, the size of the factorization results, in order to have a solid foundation for proposing practical recommendations, tools or approaches. We work with a kind of "worst case", by using all the modeling elements.

We show, through the obtained results on classes and associations how RCA behaves for improving abstraction. We highlight tractable and non tractable configurations.

The rest of the paper is structured as follows. Section II summarizes FCA and RCA. Section III presents the experiment and its results, which are discussed in Section IV. Finally, Section V presents related work, and Section VI concludes [1].

II. Background on FCA and RCA

Formal Concept Analysis (FCA): Formal Concept Analysis [1] is a mathematical framework that groups entities sharing characteristics into concepts. A concept groups a maximal set of entities sharing a maximal set of characteristics. The concepts are (partially) ordered by a lattice structure. For example, let us consider several animals described by several characteristics, as specified in Table I. Here, an ostrich has no characteristic, whereas a hoopoe has a crest, migrates, and flies.

Applying FCA on such formal context results in the concept lattice given in Figure 1. In the lattice, concepts are represented by rectangles containing the name, the intent

[1]This work has been funded by ANR 2010 CUTTER BLAN 021902

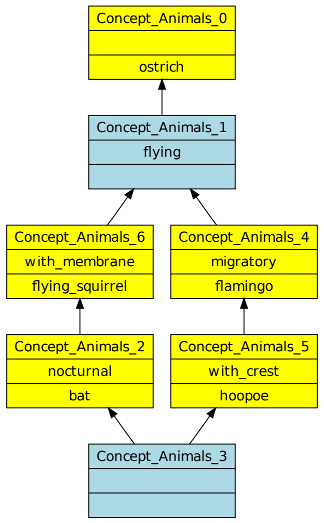

Figure 1. Concept lattice for the animals (simplified concept)

(set of shared characteristics) and the extent (set of covered entities) of the concept, and the partial order is represented by edges. In the lattice, the concepts are structured following a generalization mechanism from the most general, at top, to the most specialized, at bottom. The intent of a concept includes the characteristics from upper concepts, while the extent of a concept includes the entities from the concepts below. The lattice of Figure 1 shows a simplified form of the lattice of the animals, in which only the elements (entities or characteristics) introduced by the concept are shown: in other words, we show only the simplified extent and simplified intent. For example, `Concept_Animals_4` represents migratory and flying animals: the flamingo and the hoopoe. The flying characteristic is inherited from `Concept_Animals_1` , while the hoopoe entity is included from `Concept_Animals_5` . In the simplified form, characteristics are maximally factorized, and the partial order corresponds to inclusion between sets of characteristics or in other terms, concept specialization.

Relational Concept Analysis (RCA): Relational Concept Analysis is a variant of FCA that aims at grouping several kinds of entities that are linked by different relations. The main principle of RCA is to iterate on FCA application, and the concepts learnt during one iteration for one kind of entity are propagated through the relations to the other kinds of entities. As an example, let us consider as entities not only animals, but also singers, that can be song writers and group leaders (formal context of Table II).

Applying FCA to the singers results in the lattice presented in Figure 2 and allows to discover concepts like `Concept_Singers_1` that represents singers who are group leaders and who write songs. In the extent of that concept, we find *Paul* and *John*.

	group leader	song writer
John	×	×
George		×
Paul	×	×
Ringo		

Table II
CONTEXT FOR THE SINGERS

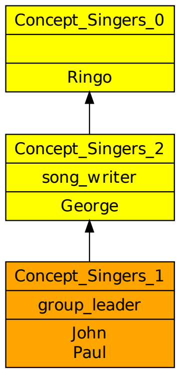

Figure 2. Lattice for the singers using FCA

Singers and animals are linked by the relation “likes”: singers like (or not) animals. This relation is also represented by a binary table (Table III), that is called a relational context, and that links entities to entities. The idea is to use the relation linking the singers to the animals to discover new concepts. For example, we can discover the concept of group leaders and song writers that like at least one migratory and flying animal. The migratory and flying animals are grouped in `Concept_4` in the lattice of the animals. This information is propagated into the context of the singers, introducing relational characteristics stating which singer is in relation “likes“ with which concepts of animals. For example, a relational characteristic `likes Concept_4` is introduced, and it has to be decided which singer likes `Concept_4`. This propagation step is called the relational scaling, and can be done using several scaling operators. Here, we use the existential one, that states that an entity has a relational characteristic for relation `r` and concept `c` (i.e. is in relation `r` with `c`) if the entity is in relation `r`

likes $\nearrow$	flying squirrel	bat	ostrich	flamingo	hoopoe
John				x	
George			x		
Paul		x			x
Ringo	x				

Table III
RELATIONAL CONTEXT FOR THE RELATION *likes*

with at least one entity of the extent of `c`. Using different scaling operators, we could find concepts of singers liking all or only the migratory and flying animals. This propagation allows new concepts for entities to be discovered. Here, we discover new concepts for the singers: singers are grouped not only based on their characteristics, but on the fact that they like or not groups of animals.

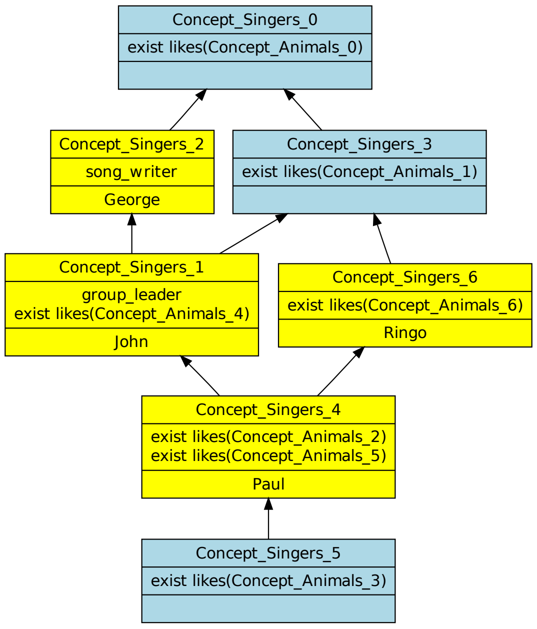
Concept_Singers_0
exist likes(Concept_Animals_0)
Concept_Singers_2
song_writer
George
Concept_Singers_3
exist likes(Concept_Animals_1)
Concept_Singers_1
group_leader
exist likes(Concept_Animals_4)
John
Concept_Singers_6
exist likes(Concept_Animals_6)
Ringo
Concept_Singers_4
exist likes(Concept_Animals_2)
exist likes(Concept_Animals_5)
Paul
Concept_Singers_5
exist likes(Concept_Animals_3)

Figure 3. Lattice for the singers using RCA, after 2 FCA iterations

The process stops when a fixpoint is reached, i.e. when no new concept is introduced by an iteration. In Figure 3, we show the result of the application of RCA, with the scaling operator `exists`, to our example of singers and animals (here, there were two FCA iterations). `Concept_1` represents the leaders and song writers that like at least one migratory and flying animal: *John* and *Paul*. `Concept_6` represents the singers that like at least one flying animal with membrane (*Ringo* and *Paul*).

In the obtained concept lattices, we distinguish *merged concepts* and *new concepts*. A ***merged concept*** is a concept that has more than one entity in its simplified extent. For example, in Figure 2, `Concept_Singers_1` is a merged Concept, since its simplified extent contains *John* and *Paul*. That means that *John* and *Paul* are described by exactly the same set of characteristics. A ***new concept*** is a concept that has an empty simplified extent. For example, in Figure 3, `Concept_Singers_3` is a new concept: it represents the concept of singers who like at least one flying animal, and corresponds to a new domain abstraction that avoids the duplication of some characteristics.

III. Experiment

In this section, we present the used configurations and parameterizations, then we report the main results that we obtain when we apply RCA on a real class model.

A. Experimental framework

Our tool is based on the Modeling Tool Objecteering[2] and the already existing framework eRCA[3] that are connected via a Java programming API. In this paper we focus on a configuration (part of the meta-model) including the following entities described in formal contexts: classes, associations, operations (very few in the *Pesticides* model), roles and attributes. Their characteristics are their names. Let us notice that in the Objecteering meta-model, there is an `EndAssociation` metaclass distinct from the `Attribute` metaclass, in contrast to the current UML meta-model which represents roles and attributes in the `Property` meta-class. The relational contexts describe: which class owns which attribute, which class owns which operation, which class owns which role, which association owns which role and which type (class) has a role. We also consider four parameterizations for this configuration depending on whether we take into account navigability and undefined elements. If a navigability is indicated on an association, meaning that objects from the source know objects from the target (not the reverse), taking into account navigability (denoted by `Nav`) results in the following encoding: the source class owns the corresponding role, but the target class does not own any role corresponding to that association. Not taking into account navigability (denoted by `noNav`) means that the source class and the target class own their respective role in the association. Our modeling tool does not allow to omit the roles names: when a role has no name, the tool names it "undefined". We can choose to include this "undefined" name in the contexts (denoted by `Undef`) or not (denoted by `noUndef`). The `Undef` option means that we keep this name in the description of the corresponding property. The `NoUndef` option means that we remove all those names set by the tool.

The case study we use is a project from the Irstea institute, that develops actual systems in the environmental fields. The project is called Environmental Information System for Pesticides (EIS-Pesticides), it aims at designing an information system centralizing knowledge and information produced by two teams: a Transfer team in charge to study the pesticide transfers from plots to rivers and a Practice team which mainly works on the agricultural practices of the farmers. The domain analysis has been carried on during series of meetings with one team or both teams.

Figure 4 shows in function of the model version the number of Classes, Attributes, Associations and the overall

[2]http://www.objecteering.com/
[3]http://code.google.com/p/erca/

number of model elements.

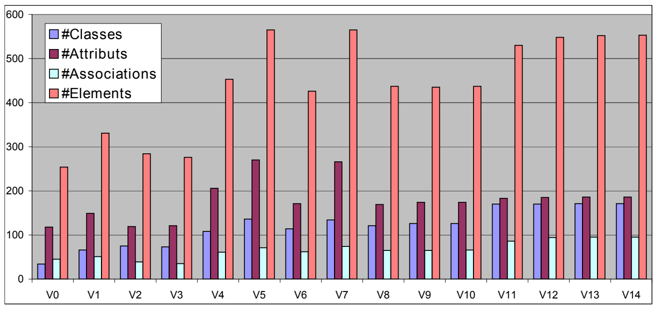

Figure 4. The number of model elements over the various versions

We consider two special iteration steps in RCA process: step 1 (close to FCA application) and step 6 (where paths of length 6 in the model are followed). At step 1 for example, common name attributes are used to find new superclasses. At step 4, new superclasses can be found, and 2 steps later, new super-associations can be found from the class concepts found at step 4. Indeed, RCA iterates on steps, each step being an application of FCA followed by a scaling. Results at step 1 thus correspond to the application of FCA rather than RCA, no relational information is used. Results at steps n propagate n times the abstractions found by FCA through the relations. Step 6 is particular for the used metamodel (UML) since it corresponds to the number of steps necessary to loop in the model (seen as an instance of its metamodel) from a class to itself, through a relation. Classes are classified: at step 1 wrt to its own characteristics, at step 2 also wrt its owned properties, at step 3 also wrt its properties linked to associations, etc, until step 6, in which classes are classified using all the relational information at least once.

In the section that follows, we examine metrics on *new* class and association concepts.

B. New abstractions

Due to space reasons, we focus on the concepts (in the class lattice and in the association lattice) which represent new abstractions: they will be interpreted as new superclasses or as new generalizations of associations. In a collaborative work, these concepts are presented to the experts that use some of them to improve the higher levels of the class model because they recognize domain concepts not explicit until now. This is why their number is important; if too many new abstractions are presented to the experts, these experts might be overwhelmed by the quantity, preventing a relevant and efficient use of the method.

Table IV shows indicative metrics about the *new* concepts in the class lattice (thus the new superclasses) at step 1, when paths of size 1 have been traversed. For example, this means that if some classes have attributes (or roles) of the same name in common, those attributes (or roles) will certainly be grouped in a *new* class concept. In the best case (of percentage of new superclasses), 32% of new potential superclasses will be presented to the experts, for the model V11 which contains 170 classes, giving 54 new potential superclasses. In the worst case, we have 112% of new potential superclasses, for V0 model, which has 34 classes, thus only 38 new potential superclasses are found. At this stage, we do not see a serious difference between the four parameterizations.

Difficulties and differences are evident at step 6. The two parameterizations `noNav` (generating more cycles in data) give results that will need serious filtering to separate relevant *new* concepts from the large set of *new* concepts. Table IV shows that `Nav` parameterizations will produce less than one and a half the initial number of classes, while `noNav` parameterizations can produce up to 10998 class concepts, really requiring either additional post-treatments or avoiding to generate all the concepts.

Table V gives concrete figures on new association concepts at step 1: in `Nav` parameterizations, at most 15 higher level associations are presented to experts; in `noNav` parameterizations, the number grows until 32, remaining very reasonable to analyze.

Table V also focuses on new association concepts at step 6. This confirms that the number of new association concepts, in `Nav` parameterizations still remains reasonable (even it is higher than in step 1), but that in `noNav` parameterizations it dramatically grows and may reach about 9500 concepts.

C. Execution time

Experimentations have been performed on a cluster composed of 9 nodes, each one having 8 processors Intel (R) Xeon (R) CPU E5335 @ 2.00GHz with 8 GO of RAM. The operating system was Linux (64bits version) and the programs are written in Java.

At step 1, the execution time for the two `Nav` parameterizations is below 6 seconds, while for the two `noNav` parameterizations, for some versions (especially when there are more associations, like in the last versions) it may reach about 13 seconds. At step 6, the execution time for the two `Nav` parameterizations are below 8 seconds. But for the `noNav` parameterizations, we notice longer executions, up to 10 minutes. Considering that the task does not require an instantaneous answer, even if it occurs during an expert meeting, spending a few minutes for constructing the concept lattices can be admitted.

IV. Discussion

From the observed results, we can learn several lessons. Execution time is correct on the cluster. Both the concepts and the structure of the lattice are useful for the experts to determine which relevant modifications should be applied. The strength of the lattice representation is that it provides

Step	Parameters	% Min	Version	# classes	# new class concepts	% Max	Version	# classes	# new class concepts
STEP 1	Nav-noUndef/Undef	32	V11	170	54	94	V0	34	32
	noNav-Undef	52	V11	170	88	112	V0	34	38
	noNav-noUndef	52	V11	170	88	94	V0	34	32
STEP 6	Nav-noUndef/Undef	116	V5	136	158	161	V1	66	106
	noNav-Undef	973	V3	73	710	6432	V14	171	10998
	noNav-noUndef	153	V0	34	52	6432	V14	171	10998

Table IV
STEP 1 AND STEP 6 - NEW CLASS CONCEPTS

Step	Parameters	% Min	Version	# assoc.	# new assoc. concepts	% Max	Version	# assoc.	# new assoc. conc.
STEP 1	Nav-noUndef/Undef	8	V8	65	5	20	V6	74	15
	noNav-noUndef/Undef	13	V0	45	6	45	V5	71	32
STEP 6	Nav-noUndef/Undef	38	V0	45	17	98	V14	95	93
	noNav-Undef	266	V3	35	93	10037	V14	95	9535
	noNav-noUndef	38	V0	45	17	10037	V14	95	9535

Table V
STEP 1 AND STEP 6 - NEW ASSOCIATION CONCEPTS

a structure adapted to the task of choosing the right new abstractions, since concepts can be presented following the specialization order, depending of what is the demand of experts. We also analyzed *merge* class concepts and *merge* associations concepts (details are not reported here) and this study shown that this is not a problematic task, due to their small numbers. The feasibility of analyzing *new* class concepts and *new* associations concepts deserves more attention. `Nav` parameterizations produce exploitable results with a maximum of about 50 class concepts (resp. about 30 new association concepts) to be examined at step 1. At step 6, more analysis work has to be done because experts may have to face from one to three hundreds of new class concepts (resp. about one hundred of new association concepts). In this case, a simple initial advice is not to try to understand as a whole all the concepts, but to use the possibility of looking, at each step between step 1 and 6 the appearing concepts until a certain rank.

Besides, in the context of FCA, a particular sub-order of the concept lattice (the AOC-poset), induced by the concepts that introduce an entity or a characteristic, might offer an important reduction of the produced concept numbers, without loosing main information about factorization.

Another track is limiting input data, for example by removing attributes that have limited semantic value. To go in that direction, we would propose to the experts a preliminary study on terms that often appear in the model, with the intuition that a frequent term, like ”name”, or ”code”, does not carry main information to be factorized. To limit the number of concepts to be examined by experts, it would be fruitful to group new concepts that declare few attributes (a group being a connected part of the lattice).

There are some limits to the generalization of our conclusions. The class model is mainly a data model (very few operations), destined to build a database schema and we study various versions of a same class model. Nevertheless, the *Pesticides* model is a classical model, representative of the models that are built in the environmental field.

V. RELATED WORK

The use of FCA in the domain of class model refactoring has a long and rich history in the literature. As far as we know, it has been introduced in the seminal paper of Godin et al. [2] for extracting abstract interfaces from a Smalltalk class hierarchy and extensions of the work have been published in [3]. Other approaches have been proposed, that take into account more information extracted from source code like super calls and method signatures in [4].

The first practical case of application of RCA was reported in [5]. It has been conducted on several medium-size class models of France Télécom (now Orange Labs)[4]. The RCA configuration was composed of classes, methods, attributes, and associations and links between these elements. The class models contain a few dozens of classes and the new concepts to be examined by experts vary from a few concepts to several hundreds. In [6] detailed figures about the experiment are given. In the largest project (57 classes), the number of *new* concepts was 110 for the classes, 9 for the associations and 59 for the properties. Among these concepts, experts have been able to select relevant ones. In this paper, we have several class models that are greater in terms of number of classes and we reify the role notion, rather than encoding it in the association description, with the risk of having more produced concepts. We discard technical description (like multiplicity which has no strong semantics).

In [7], RCA has been experimented on an excerpt of the class model of the Jetsmiles software of JETSGO society

[4]Joint RNTL project supported by french department of research and industry

(http ://www.jetsgo.net/), that is composed of only 6 classes, connected by 9 associations, and about 30 attributes. Attributes and roles are mixed, and classes are connected by a relational context to attributes+roles, attributes+roles are connected by another relational context to their type (when it is a class). The UML elements are described by many technical features such as visibility, that add complexity during RCA application and generate many non relevant concepts. A post-treatment analyzes the built concepts in order to keep the most relevant ones. The class concept lattice contains about 35 concepts while the attribute+role concept lattice has about 25 concepts. The size of the class model is very small and not realistic. Using the configuration with many technical elements would not be scalable in the case of the *Pesticides* model.

RCA has been experimented on two Ecore models, two Java programs and five UML models in [8]. The used configuration is composed of the classes described by their names and the properties (attributes and roles) described by their names; classes are connected to their properties and properties are connected to their type (when it is a class). In this experiment, associations were not encoded, contrarily to what we do in the *Pesticides* model. Nevertheless the explosion of the concept number already appears. In our case, we introduce associations in the configuration, and we show, that with some precautions as annotating by navigability and naming the roles, refactoring with data including the associations may remain feasible.

The more recent study [9] compared three strategies of FCA/RCA application to part of the open-source Java Salome-TMF software[5] (test management, 37 classes, 142 attributes). Several strategies are experimented, in particular one with a first formal context that includes the classes (no description), a second formal context that describes the attributes by their names and the hyperonyms of their names, a first relational context that associates to a class its attribute names, a second relational context that associates to an attribute its type when it is a class. The experiments show reasonable numbers of concepts, but Java software does not have associations and it is difficult to generalize these results to the general case of class models. Also, compared to this work, here we do not use linguistic information.

To conclude, this paper proposes the most advanced study of the application of RCA to class models: avoiding non relevant, too technical, features; using the model elements as described in their meta-model, without changing their structure and connections; and using realistic class models.

VI. Conclusion and perspectives

In this paper, we study the possible application of Relational Concept Analysis (a variant of Formal Concept Analysis, a data analysis method) on class models, so as to obtain a relevant factorization structure. We applied RCA on several versions of the same information system class model which contains from 40 to 170 classes, and we studied the impact of several parameters in the application (taking (or not) into account navigability and undefined roles in associations). The objective was to study the behaviour of RCA on real-sized class models, so as to draw conclusions on its application. In particular, a major concern was the scalability of the approach, since RCA may generate too many very abstract concepts in certain cases. The experiment shows that taking into account the navigability, the results remain possible to analyze, the explosion of concepts does not appear. Consequently, RCA can be considered to scale to real-size models, if it is adequately parameterized. However, the produced results remain quite large to analyze, and new strategies have to be settled to face the number of concepts to analyze. The main strategy we will explore is an incremental exploration approach: instead of presenting to the expert the final result, with hundreds of concepts to analyze, the abstractions will be presented step by step to the expert, so as to "cut" branches of abstractions that are not relevant.

[5]http://wiki.ow2.org/salome-tmf/

References

[1] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundation.* Springer-Verlag Berlin, 1999.

[2] R. Godin and H. Mili, "Building and maintaining analysis-level class hierarchies using Galois lattices," in *Proceedings of OOPSLA'93, Washington (DC), USA*, ser. special issue of ACM SIGPLAN Notices, 28(10), 1993, pp. 394–410.

[3] R. Godin, H. Mili, G. Mineau, R. Missaoui, A. Arfi, and T. Chau, "Design of Class Hierarchies Based on Concept (Galois) Lattices," *Theory And Practice of Object Systems*, vol. 4, no. 2, 1998.

[4] H. Dicky, C. Dony, M. Huchard, and T. Libourel, "On Automatic Class Insertion with Overloading," in *Proceedings of OOPSLA'96, San Jose (CA), USA*, ser. special issue of ACM SIGPLAN Notices, 31(10), 1996, pp. 251–267.

[5] M. Dao, M. Huchard, M. R. Hacene, C. Roume, and P. Valtchev, "Improving Generalization Level in UML Models Iterative Cross Generalization in Practice," in *ICCS*, ser. Lecture Notes in Computer Science, K. E. Wolff, H. D. Pfeiffer, and H. S. Delugach, Eds., vol. 3127. Springer, 2004, pp. 346–360.

[6] C. Roume, "Analyse et restructuration de hiérarchies de classes," Ph.D. dissertation, Université Montpellier 2, 2004.

[7] M. R. Hacène, "Relational Concept Analysis, application to software re-engineering," Ph.D. dissertation, Université du Québec à Montréal, 2005.

[8] J.-R. Falleri, M. Huchard, and C. Nebut, "A generic approach for class model normalization," in *ASE*. IEEE, 2008, pp. 431–434.

[9] J.-R. Falleri, "Contributions à l'IDM : reconstruction et alignement de modèles de classes," Ph.D. dissertation, Université Montpellier 2, 2009.