Practical Application of Relational Concept Analysis to class model factorization: lessons learned from a thematic information system

Abdoulkader Osman Guédi, Marianne Huchard, André Miralles, Clémentine Nebut

A practical application of Relational Concept Analysis to class model factorization: lessons learned from a thematic information system

A. Osman Guédi[1,2,3], A. Miralles[2], M. Huchard[3], and C. Nebut[3]

[1]Université de Djibouti, Avenue Georges Clémenceau BP: 1904 Djibouti (REP)
[2]Tetis/Irstea, Maison de la télédétection, 500 rue JF Breton 34093 Montpellier Cdx 5
[3]LIRMM (CNRS et Univ. Montpellier), 161, rue Ada, F-34392 Montpellier Cdx 5

Abstract. During the design of class models for information systems, databases or programming, experts of the domain and designers discuss to identify and agree on the domain concepts. Formal Concept Analysis (FCA) and Relational Concept Analysis (RCA) have been proposed, for fostering the emergence of higher level domain concepts and relations, while factorizing descriptions and behaviors. The risk of these methods is overwhelming the designer with too many concepts to be analyzed. In this paper, we systematically study a practical application of RCA on several versions of a real class model for an information system in order to give precise figures about RCA and to identify which configurations are tractable.

Keywords: Class model, class model factorization, Formal Concept Analysis, Relational Concept Analysis

1 Introduction

Designing class models for information systems, databases or programs is a common activity, that usually involves domain experts and designers. Their task consists in capturing the domain concepts, and organizing them in a relevant specialization structure with adequate abstractions and avoiding redundant concepts. Formal Concept Analysis (FCA) and its variant Relational Concept Analysis (RCA) have been proposed to assist this elaboration phase, so as to introduce new abstractions emerging from the identified domain concepts, and to set up a factorization structure avoiding duplication. FCA classifies entities having characteristics[1], while RCA also takes into account the fact that entities are linked by relations. The concept lattices produced 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

[1] The usual terms in the FCA domain are "objects" and "attributes"; we prefer not using them here because they conflict with the vocabulary of class models.

suffer from limitations due to the large size of the obtained lattices. In such a case, domain experts might be overwhelmed by the produced information making it difficult (or even impossible) to use it to improve the class model.

In this paper we want to assess, in a real case study, the size of the factorization results, in order to have a solid foundation for proposing practical recommendations, tools or approaches. We work with a kind of "worst case" of RCA application, by using all the modeling elements and not limiting our investigation to some elements (like classes and attributes, or classes and operations). We show, via various selected graphics, how RCA behaves. Our experiments indicate which configurations are tractable, admitting that some tools present results in a fine way, and which configurations lead to quite unusable results.

The rest of the paper is structured as follows. Section 2 briefly explains how FCA and RCA can contribute to a class model design. Section 3 settles the environment for our experiments and introduces our case study. Section 4 presents and discusses the obtained results. Section 5 presents related work, and section 6 concludes.

2 Concept lattices in class model refactoring

In this section, we explain how concept lattices implement and reveal the underlying factorization structure of a class model. We also show how this property can be exploited for class model refactoring, and in particular: generating new reusable abstractions that improve the class organization and understanding, especially for domain experts, limiting attribute and role duplication.

Formal Concept Analysis [5] is a mathematical framework that groups entities sharing characteristics: entities are described by characteristics (in a Formal Context), and FCA builds (formal) concepts from this description. In Relational Concept Analysis (RCA) [10], the data description consists of several Formal Contexts and 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. The concepts are provided with a partial order which is a lattice. 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. This means that the entities of the simplified extent share the same description. A *new concept* is a concept that has an empty simplified extent. This means that no entity has exactly the simplified intent of the concept as set of characteristics: Entities of the whole extent own the characteristics of the simplified intent in addition to other characteristics.

To apply FCA to class models, we encode the elements of a class model into formal contexts. For example, we provide a context describing the classes by their attributes. The FCA approach then reveals part of the factorization structure and supports part of the refactoring process by using a straightforward description of UML elements. For example, we can discover *new concepts* for classes interpreted as new super-classes, factorizing two attributes.

Nevertheless this approach does not fully exploit the deep structure of the class model. Let us take the example of Figure 1(a). The attribute `name` is duplicated in classes `B1` and `B2`, and FCA can generate the model of Figure 1(b) that introduces a new class (here manually named `NamedElement`) that factorizes this attribute. However, FCA does not compute the factorization that can be found in Figure 1(c), in which a class called `SuperA` factorizes the two associations from `A1` to `B1` and from `A2` to `B2`, being given that now `B1` and `B2` have an ancestor `NamedElement`.

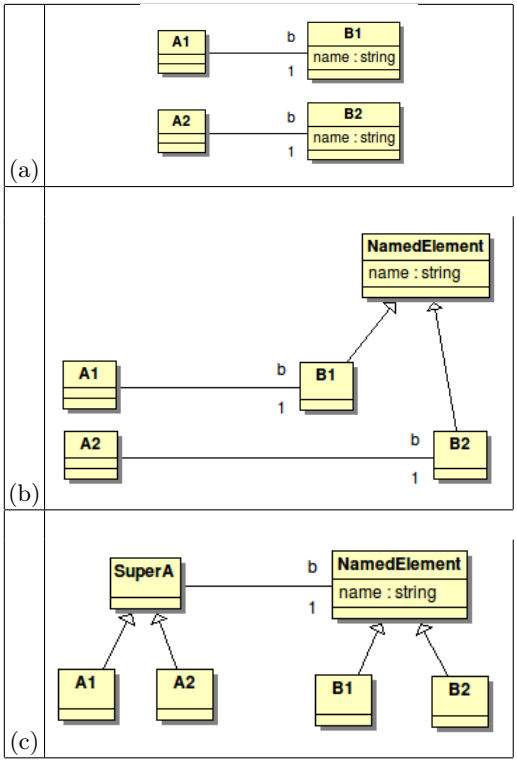


Fig. 1. Example of factorization in class models

Extracting abstractions using this deep structure can be done with RCA, which builds the entire factorization structure, including information on the elements (classes, attributes, associations) and their relations. RCA uses the fact that classes are linked through associations. In the first iteration step, RCA computes the factorization in Figure 1(b), and then propagates the *new concept* `NamedElement` through the association between classes. Then the factorization of Figure 1(c) is computed during the next iteration steps. The process stops there since a fixpoint is found (no new abstractions can be found).

The obtained structure contains no duplication, and improves the organization of the model. However, when applied on large data, RCA may result in the introduction of many *new concepts*, that may be too abstract, and/or too many to be analyzed. That is why in the next sections, we investigate on a case study the behavior of RCA for a large class model corresponding to an actual information system. Our objective is to determine if RCA remains suitable for large class models, and how to configure RCA to obtain exploitable results.

3 The Pesticides class model and experimental setup

Our case study is a class model which is part of a project from the Irstea institute, 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 of studying 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. Fifteen versions of this class model have been created during this analysis. Figure 2 shows the number of model elements over the versions.

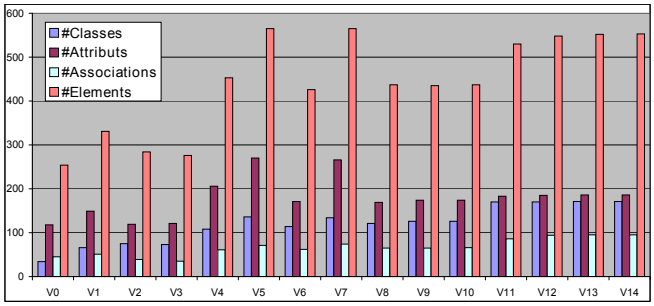


Fig. 2. The number of model elements over the various versions

Our tool is based on the Modeling Tool Objecteering[2] and the framework eRCA[3]. eRCA has been extended for computing metrics. In this paper we focus on a configuration (part of the meta-model) including the following entities described in formal contexts: classes, associations, operations (very few in the *Pesticides* model), roles and attributes. Their characteristics are their names. The relations 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. When applying RCA to this configuration, we obtain 5 concept lattices, one for each formal context. We also consider four

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

parameterizations for this configuration depending on whether we take into account navigability and undefined elements. If a navigability is indicated on an association, meaning that objects from the source know objects from the target (not the reverse), taking into account navigability (denoted by `Nav`) results in the following encoding: the source class owns the corresponding role, but the target class does not own any role corresponding to that association. Not taking into account navigability (denoted by `noNav`) means that the source class and the target class own their respective role in the association. In the modeling tool, unnamed roles are named "undefined". We can choose to include this "undefined" name in the contexts (denoted by `Undef`) or not (denoted by `noUndef`).

4 Results

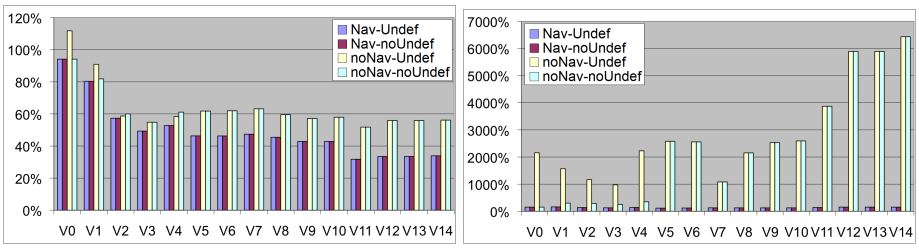
In this section, we report the main results that we obtain. We consider two special iteration steps: step 1 (close to FCA application) and step 6 (where paths of length 6 in the model are followed, meaning that abstractions on classes created at step 1 have been exploited to create other class abstractions through roles and associations). At step 1 for example, common name attributes are used to find new superclasses. At step 4, new superclasses can be found as shown in Figure 1(c), and 2 steps later, new super-associations can be found from the class concepts found at step 4. We examine, for classes and associations, which are the main elements of the model, metrics on *new* class concepts and *new* associations concepts (Section 4.1), then on *merge* class and association concepts (Section 4.2). Execution time is presented in Section 4.3, and we conclude this part by giving indications about the number of steps when the process reaches the fixpoint.

4.1 New abstractions

We focus first on the *new concepts* that appear in the class lattice and in the association lattice. They will be interpreted as new superclasses or as new generalizations of associations. In a collaborative work, these concepts are presented to the experts who use some of them to improve the higher levels of the class model with domain concepts not explicit until then. This is why their number is important; if too many new abstractions are presented to the experts, these experts might be overwhelmed by the quantity, preventing a relevant and efficient use of the method.

Figure 3 (left-hand side) shows the *new concepts* in the class lattice (thus the new superclasses) at step 1, when paths of size 1 have been traversed. For example, this means that if some classes have attributes (or roles) of the same name in common, those attributes (or roles) will certainly be grouped in a *new* class concept. This new class concept can be presented to the expert to control if this corresponds to a new relevant class abstraction (or superclass). We notice that `Nav` parameterizations produce less *new concepts* than `noNav` ones. This is due to the fact that `noNav` parameterizations induce much more circuits in the

analyzed data, increasing the number of RCA steps and the number of generated concepts. The number of *new concepts* decreases as the analysis process progresses.



STEP 1 STEP 6

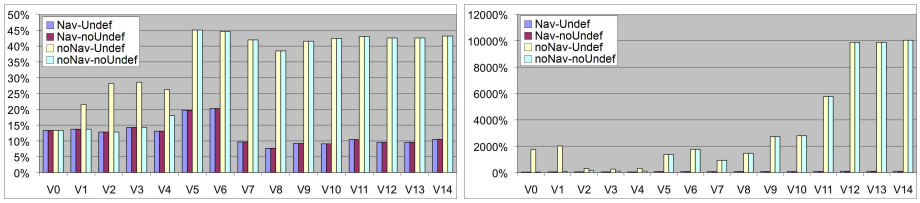
Fig. 3. New class abstractions created at step 1 and 6 v.s. the number of initial classes

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.

Results obtained at step 6 are much more difficult to deal with. Figure 3 (right-hand-side) shows that the two parameterizations `noNav` (generating more cycles in data) give results that will need serious filtering to separate relevant *new concepts* from the large set of *new concepts*. `Nav` parameterizations will produce less than one and a half the initial number of classes, while `noNav` parameterizations can produce up to 10998 class concepts, really requiring either additional post-treatments or avoiding to generate all the concepts.

Figure 4 (left-hand side) shows the *new concepts* in the association lattice at step 1. They represent associations that are at a higher level of abstraction. Experts can examine them, to see if they can replace a set of low-level associations. In `Nav` parameterizations, at most 15 higher level associations are presented to experts; in `noNav` parameterizations, the number grows until 32, remaining very reasonable to analyze.

Figure 4 (right-hand side) shows the *new concepts* in the association lattice at step 6. It highlights the fact that, at this step, the number of these concepts may explode, and it is especially high in the last versions of the class model, in which we initially have many associations. The number of new association concepts, in `Nav` parameterizations, is less than a hundred, and it still remains reasonable (even if it is higher than in step 1), but in `noNav` parameterizations it dramatically grows and may reach about 9500 concepts.



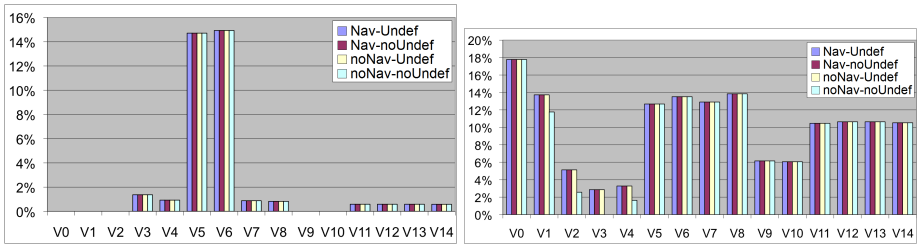
STEP 1 STEP 6

Fig. 4. New association abstractions created at step1 and 6 v.s. the number of initial associations

4.2 Merged concepts

Merged concepts are concepts which introduce several entities (*e.g* classes or associations) in their extent. Such entities share exactly the same description in the model. For example, a merge class concept can group classes that have exactly the same name attributes. This common description is first detected at step 1, then it does not change because the following steps refine the description by adding new relational characteristics and concepts; entities remain introduced in the same concepts. For classes and associations, the merged concept number is the same for the four analysis configurations. For experts, analyzing a *merged concept* consists in reviewing the simplified extent and examining if the entities (class or association) have been exhaustively described or effectively correspond to a same domain concept.

Figure 5 (left-hand side) presents metrics for merge class concepts. V5 and V6 have a higher percentage of *merged concepts* because during analysis, a package has been duplicated at step 5 for refactoring purpose. The duplicated classes have been removed at step V7. In the other cases, there are not so much merge class concepts to be presented to the experts, between 0% and 2%, giving a maximum of two classes. This often corresponds to classes with incomplete description, that the experts should develop into more details. The low number of such cases makes the task of experts easy.



Ratio # merge class concepts on # initial classes

Ratio # merge association concepts on # initial associations

Fig. 5. Merge class concept and merge association concepts vs. initial elements

Figure 5 (right-hand side) presents metrics for merge association concepts. The percentage of merge association concepts is higher than the percentage of merge class concepts. This is explained by the fact that associations are only described by roles, that occasionally share the same names (identical to some class names). It varies between about 2% and 18%, meaning that at most 10 merge association concepts are presented to the experts for evaluation, making a little bit more complicated the analysis task compared to the case of classes, but it remains very reasonable.

4.3 Execution time, used RAM and total number of steps

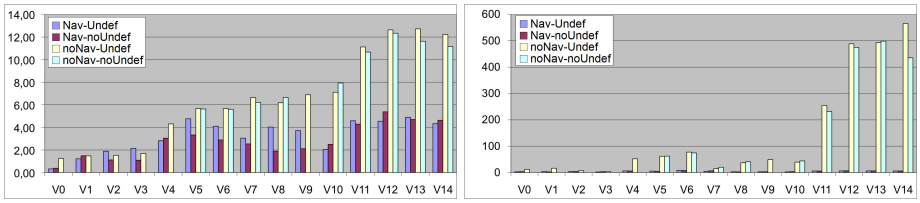
Experimentations have been performed on a cluster composed of 9 nodes, each one having 8 processors Intel (R) Xeon (R) CPU E5335 @ 2.00GHz with 8 Go of RAM. The operating system was Linux (64 bits) and the programs are written in Java.

Figure 6 shows the execution time in seconds, at step 1 and at step 6. At step 1, the execution time for the two `Nav` parameterizations are below 6 seconds, while for the two `noNav` parameterizations, for some versions (especially when there are more associations, like in the last versions) it may reach about 13 seconds. At step 6, the execution time for the two `Nav` parameterizations are below 8 seconds. But for the `noNav` parameterizations, we notice longer executions, up to 10 minutes. However, such a task does not require an instantaneous answer, and has not to be carried out too many times. Even if it occurs during an expert meeting, it can be admitted to spend a few minutes for constructing the concept lattices.

Table 1. Figures on used memory (in MegaBytes)

Step	Parameters	min	max	average
Step 1	Nav-Undef	39	453	237
	Nav-noUndef	17	471	205
	noNav-Undef	41	969	480
	noNav-noUndef	24	969	532
Step 6	Nav-Undef	44	471	213
	Nav-noUndef	6	403	140
	noNav-Undef	33	1846	656
	noNav-noUndef	33	1147	520

Table 1 shows the RAM used during execution, here again, `noNav` parameterizations are the worst, reaching about 2 GigaBytes of used memory. In the case of `Nav` parameterizations, it is interesting to observe that there is not a significant difference between step 1 and step 6.



STEP 1 STEP 6

Fig. 6. Execution time at step 1 and 6 (in seconds)

Figure 7 shows, for the `Nav-noUndef` parameterization the total number of steps needed to reach the fix-point, and the size of a longest path with no repeated arcs (such a path can be a cycle). We observe that the step number (from 6 to 16) is always below the size of the longest simple path which gives in our context a practical upper bound to the number of steps. This means that if we dispose in the future of relevant filtering strategies, we can envisage studying *new concepts* appearing after step 6.

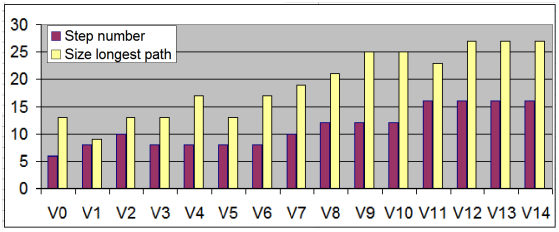


Fig. 7. Step number (first) and longest simple path size (second) in C2, `Nav-noUndef` parameterization

4.4 Discussion

During this study, we observed that analyzing *merge* class concepts and *merge* association concepts was a feasible task in all parameterizations. The analysis of *new* class concepts and *new* associations concepts is more difficult. `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, experts may have to face from one to three hundreds of new class concepts (resp. about one hundred of new association concepts). Execution time and used memory are not problematic issues and we get a practical upper bound to the number of steps in this kind of data, which is given by the size of a longest simple path.

These observations may be the starting point of an efficient steering method for a collaborative class model construction. The objective of such a method

would be to insert between each model release, an RCA-based analysis, in order to accelerate the discovery of new abstractions, and the completion of elements (highlighting of the *merged concepts*). 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 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.

Known effects of some parameterizations can serve to favor different steering strategies. The longest simple path size gives a bound on the step number, giving an heuristic to decide when to stop the RCA process, with an idea about how far we are from the convergence. `Nav` parameterizations can be easily controlled by looking, at each step, the appearing concepts (and marking the non-relevant ones to avoid finding them again at next steps). If information is expected from `NoNav` parameterizations, experts have to be very careful because many concepts will be created. 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 and to group concepts declaring few attributes.

There are of course 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.

5 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. [6] for extracting abstract interfaces from a Smalltalk class hierarchy and extensions of the work have been published in [7]. Other approaches have been proposed, that take into account more information extracted from source code like super calls and method signatures in [2].

In [1], authors report an application of RCA to several medium-size class models of France Télécom (now Orange Labs). The RCA configuration was composed of classes, methods, attributes, and associations. Classes have no description; attributes are described by name, multiplicity and initial value; methods are described by name and method body; associations are described by names, role name and information like multiplicity and navigability. Associations are connected to their origin and destination classes, classes are connected to the attributes and operations they own, attributes are connected to classes that are their type, etc. The class models contain a few dozens of classes and the *new concepts* to be examined by experts varies from a few concepts to several hundreds.

In [11] detailed figures 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. 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). We also analyze the *merged concepts*, another interesting product of RCA.

In [9], RCA has been applied on an excerpt of the class model of the Jetsmiles software of JETSGO society[4]. The class model excerpt is composed of only 6 classes, connected by 9 associations, and about 30 attributes. Attributes and roles are mixed, and classes are connected by a relational context to attributes+roles, while attributes+roles are connected by another relational context to their type (when it is a class). The UML elements are described by many technical features: multiplicity, visibility, being "abstract", initial value, etc. A post-treatment analyzes the built concepts in order to keep the most relevant ones. The class concept lattice contains about 35 concepts while the attribute+role concept lattice has about 25 concepts. In [9], the size of the class model is very small. We suspect that 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 [4]. The used configuration is composed of the classes and the properties (including attributes and roles) described by their names and their connections. To report some representative results of this experiment, in Apache Common Collections, which is composed of 250 classes, RCA finds 34 new class concepts and less than 80 new property concepts; in UML2 metamodel, which is composed of 246 classes and 615 properties, RCA extracts 1534 new class concepts and 998 new property concepts. In this experiment, associations were not encoded, contrary to what we do. Nevertheless an explosion of the concept number yet 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.

A more recent study [3] compared three strategies of FCA/RCA application to part of the open-source Java Salome-TMF software which comprises 37 classes and 142 attributes[5]. In the RCA strategy (ARC-Name), a formal context includes the classes (no description), a formal context describes the attributes by their names and the hyperonyms of their names, and relations connect classes and their attributes (and reversely). ARC-Name produces 33 new class concepts, and 3 merge class concepts, 21 new attribute concepts and 13 merge attribute concepts. Java softwares do not have associations and it is difficult to generalize these results to the general case of class models. Compared to this work, here we do not use linguistic information, this will be done in a future work, nevertheless the terms in the model are not technical identifiers but rather do-

[4] http://www.jetsgo.net/
[5] http://wiki.ow2.org/salome-tmf/

main terms carefully selected by the expert group, thus there are less problems in using them directly.

Here we use the same class models as in [8] where RCA based metrics were proposed to assist a designer during the evolution of the class model in indicating him the evolution of the level of description and the level of abstraction.

6 Conclusion and perspectives

In this paper, we describe the most advanced study of the application of RCA on class models, so as to obtain a relevant factorization structure. We apply RCA on several versions of the model of the same information system (from 40 to 170 classes), and we study the impact of several parameters in the application. The objective was to observe RCA on real-sized class models, so as to draw conclusions, mainly on its scalability. The experiment shows that taking into account the navigability, it is still possible to analyze the newly introduced abstractions. 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 can be settled to face the number of concepts to analyze.

References

1. Dao, M., Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Improving Generalization Level in UML Models Iterative Cross Generalization in Practice. In: ICCS 2004. pp. 346–360 (2004)
2. Dicky, H., Dony, C., Huchard, M., Libourel, T.: On automatic class insertion with overloading. In: OOPSLA 96. pp. 251–267 (1996)
3. Falleri, J.R.: Contributions à l'IDM : reconstruction et alignement de modèles de classes. Ph.D. thesis, Université Montpellier 2 (2009)
4. Falleri, J.R., Huchard, M., Nebut, C.: A generic approach for class model normalization. In: ASE 2008. pp. 431–434 (2008)
5. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundation. Springer-Verlag Berlin (1999)
6. Godin, R., Mili, H.: Building and maintaining analysis-level class hierarchies using Galois lattices. In: OOPSLA 93. pp. 394–410 (1993)
7. Godin, R., Mili, H., Mineau, G., Missaoui, R., Arfi, A., Chau, T.: Design of Class Hierarchies Based on Concept (Galois) Lattices. Theory And Practice of Object Systems 4(2) (1998)
8. Guédi, A.O., Miralles, A., Amar, B., Nebut, C., Huchard, M., Libourel, T.: How relational concept analysis can help to observe the evolution of business class models. In: CLA 2012. pp. 139–150 (2012)
9. Hacene, M.R.: Relational concept analysis, application to software re-engineering. Ph.D. thesis, Universite du Quebec A Montreal (2005)
10. Hacene, M.R., Huchard, M., Napoli, A., Valtchev, P.: Relational concept analysis: mining concept lattices from multi-relational data. Ann. Math. Artif. Intell. 67(1), 81–108 (2013)
11. Roume, C.: Analyse et restructuration de hiérarchies de classes. Ph.D. thesis, Université Montpellier 2 (2004)