Class Model Normalization

Outperforming Formal Concept Analysis approaches with AOC-posets

A. Miralles[1,2], G. Molla[1], M. Huchard[2], C. Nebut[2],
L. Deruelle[3], and M. Derras[3]

(1) Tetis/IRSTEA, France
andre.miralles@teledetection.fr, guilhem.molla@irstea.fr
(2) LIRMM, CNRS & Université de Montpellier, France
huchard,nebut@lirmm.fr
(3) Berger Levrault, France
laurent.deruelle@berger-levrault.com,mustapha.derras@berger-levrault.com

Abstract. Designing or reengineering class models in the domain of programming or modeling involves capturing technical and domain concepts, finding the right abstractions and avoiding duplications. Making this last task in a systematic way corresponds to a kind of model normalization. Several approaches have been proposed, that all converge towards the use of Formal Concept Analysis (FCA). An extension of FCA to linked data, Relational Concept Analysis (RCA) helped to mine better reusable abstractions. But RCA relies on iteratively building concept lattices, which may cause a combinatorial explosion in the number of the built artifacts. In this paper, we investigate the use of an alternative RCA process, relying on a specific sub-order of the concept lattice (AOC-poset) which preserves the most relevant part of the normal form. We measure, on case studies from Java models extracted from Java code and from UML models, the practical reduction that AOC-posets bring to the normal form of the class model.

Keywords: Inheritance hierarchy, class model normalization, class model reengineering, Formal Concept Analysis, Relational Concept Analysis

1 Introduction

In object-oriented software or information systems, the specialization-generalization hierarchy is a main dimension in class model organization, as the *is-a* relation in the design of domain ontologies. Indeed, it captures a classification of the domain objects which is structuring for human comprehension and which makes the representation efficient. Designing or reengineering class models in the domain of programming or in the domain of modeling still remains a tricky task. It includes the integration of technical and domain concepts sometimes with no clear semantics, and the definition of the adequate abstractions while avoiding the duplication of information. In many aspects, this task corresponds to a kind of class model normalization, focusing on specialization and redundancy, by analogy to the database schema normalization. Normalization is important to assist forward engineering of a reliable and maintainable class model. It is also useful to address the erosion of the specialization-generalization hierarchy during software evolution.

After the seminal paper of R. Godin and H. Mili at OOPSLA'93 [1], several approaches have been proposed to address this normalization, that all converged to the implicit or explicit use of Formal Concept Analysis (FCA [2]) techniques. In this context, FCA was used to mine descriptions that are common to groups of classes and to suggest re-factorizing and creating more reusable super-classes. Several approaches more specifically used a specific sub-order of the concept lattice which captures the most relevant new super-classes (the AOC-poset, for Attribute-Object Concept poset [3]). Then, Relational Concept Analysis (RCA [4]), an extension of FCA to linked data, was proposed to find deeper re-factorizations. However, RCA iterates on building concept lattices, that leads to a combinatorial explosion of the number of the built artifacts (including classes).

In this paper, we investigate the use of an alternative version of RCA, relying on AOC-posets. With AOC-posets, RCA might not converge, thus we have to carefully handle the iteration mechanism, but when it converges, we expect more efficiency. We measure, on case studies from UML models and from Java models rebuilt from industrial Java code, the reduction brought in practice by this approach to the normal form of the class model. We also show that, on realistic tuning, the reasonable number of the new artifacts allows the designer to analyze them and decide which of them should be kept in the model.

In Section 2, the bases for FCA and RCA in the context of class models are outlined. Then we present current work in this domain, and the motivation for this study (Section 3). In Section 4, we give the setup of the experiment, as well as the results. We conclude in Section 5 with a few perspectives of this work.

2 Class Model Normalization

FCA: A classical approach for applying FCA to class model normalization involves building a formal context **K-class**$=(G, M, I)$, where the classes (in G) are associated with their characteristics (attributes, roles, operations, in M) through $I \subseteq G \times M$. There are many variations in this description of classes. For example, Fig. 1 (right-hand side) shows such a formal context for the class model presented in Fig. 2(a). A *concept* is a pair $(Extent, Intent)$ where $Extent = \{g \in G | \forall m \in Intent, (g, m) \in I\}$ and $Intent = \{m \in M | \forall g \in Extent, (g, m) \in I\}$. The concept extent represents the objects that own all the characteristics of the intent; the concept intent represents the characteristics shared by all objects of the extent. The specialization order between two formal concepts is given by: $(Extent\_1, Intent\_1) < (Extent\_2, Intent\_2) \Leftrightarrow Extent\_1 \subset Extent\_2$. It provides the concepts with a lattice structure.

In the concept lattice, there is an ascending inheritance of objects and a descending inheritance of characteristics. The simplified intent of a formal concept

Device	recommendedHeight	windVaneDimension	cupNumber	vaneType	plateDimension	tubeLength
CupAnemometer	×		×			
VaneAnemometer	×	×		×		
PlateAnemometer	×	×			×	
PitotAnemometer		×				×

K-class	tubeHeight	measureInterval	accuracy	measuringDate	codeQuality	waterAmount	windStrength	windDirection	rainfall	wind
RainGauge	×								×	
Anemometer		×	×							×
Rainfall				×	×	×				
Wind				×	×		×	×		

Fig. 1. Anemometer Formal Context (left-hand side), Formal context **K-class**=(G, M, I) with G is the set of classes of Fig. 2(a) associated by I with the set M of their attribute and role names (right-hand side)

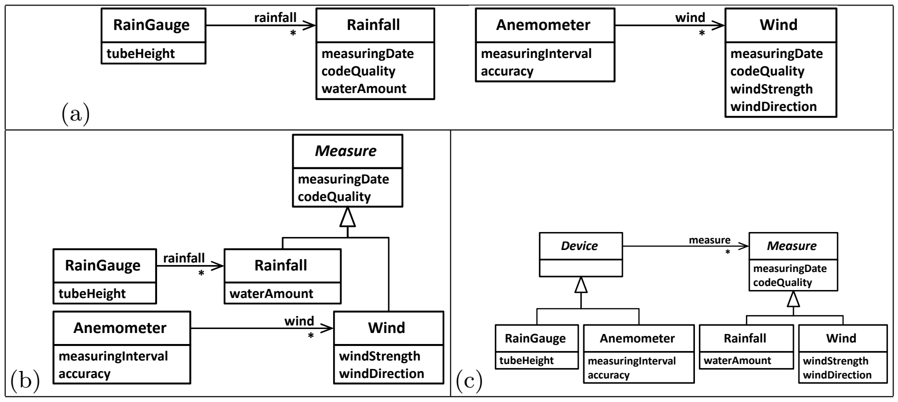

Fig. 2. Example of class model normalization with FCA and RCA [5]: (a) initial class model ; (b) (resp. (c)) class model refactored with FCA (resp. RCA).

is its intent without the characteristics inherited from its super-concept intents. The simplified extent is defined in a similar way. In our example, among the formal concepts that can be extracted from **K-class**, Concept C = ({`Rainfall, Wind`}, {`measuringDate, codeQuality`}) highlights two classes that share the two attributes `measuringDate` and `codeQuality`. This concept C is interpreted as a new super-class of the classes of its extent, namely `Rainfall` and `Wind`. The new super-class, called here `Measure`, appears in Fig. 2(b). New super-classes are named by the class model designer.

AOC-posets: In the framework of class model analysis with FCA, often AOC-posets, rather than concept lattices, are used. Formal context of Fig. 1 (left-hand side) is used to illustrate the difference between the concept lattice and the AOC-poset. The concept lattice like in Fig. 3(a) contains all the concepts from the formal context. Some concepts, like `Concept_Device_2`, inherit all their characteristics from their super-concepts and their objects from their sub-concepts. In the particular case of object-oriented modeling, they would correspond to empty description, with no correspondence with an initial class description and be rarely considered. In the AOC-poset like in Fig. 3(b), only concepts that introduce one characteristic or one object are kept, simplifying drastically the structure in case of large datasets. The number of concepts in the concept lattice can increase up to $2^{min(|G|,|M|)}$, while it is bounded by $|G| + |M|$ in the AOC-poset. The Iceberg lattice, such as introduced in [6], is another well known sub-set of the concept lattice which is used in many applications. The iceberg lattice is induced by the sub-set of concepts which have an extent support greater than a given threshold. In our case, this would mean only keeping new super-classes that have a minimum number of sub-classes, which is not relevant in modeling and programming: a super-class may only have one sub-class.

RCA: RCA helps to go further and get the class model of Fig. 2(c). In this additional normalization step, `RainGauge` and `Anemometer` have a new super-class which has been discovered because both have a role towards a sort of `Measure` (resp. `Rainfall` and `Wind`). To this end, the class model is encoded in a Relational Context Family (RCF) as the one in Table 1, composed of several formal contexts that separately describe classes (`K-class`), attributes (`K-attribute`), and roles (`K-role`) and of several relations including relation between classes and attributes (`r-hasAttribute`), relation between classes and roles (`r-hasRole`), or relation between roles and their type `r-hasTypeEnd`. Here again, this encoding can vary and integrate other modeling artifacts, like operations or associations.

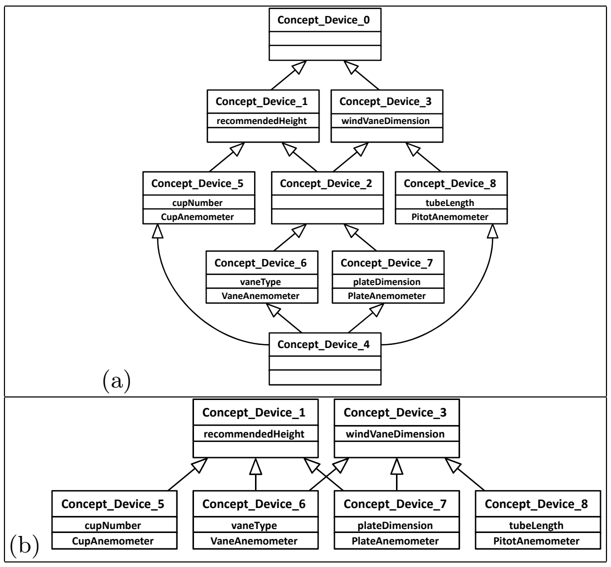

Fig. 3. Concept lattice (a) and AOC-poset (b) for anemometers (Fig. 1, left) [5]

RCA is an iterative process where concepts emerge at each step. Relations and concepts discovered at one step are integrated into contexts through relational attributes for computing concept lattices at the next step. At step 0, attributes with the same name (resp. the two attributes `measuringDate` or the two attributes `codeQuality`) are grouped. At step 1, classes that share attributes from an attribute group are grouped into a concept that produces a new super-class (e.g. `Wind` and `Rainfall` are grouped to produce the super-class `Measure`). At step 2, roles `rainfall` and `wind` share the fact that they end at a sub-class of `Measure`, thus they are grouped into new role shown under the name `measure` in Fig. 2(c). At step 3, the current context of classes (extended with relational attributes) shows that both classes `RainGauge` and `Anemometer` have a role ending to `Measure`. Then a new super-class, called `Device` by the designer, is extracted.

Table 1. Context family for the set of classes of Fig. 2(a)

Krole	rainfall	wind
rainfall	×	
wind		×

Kattribute	tubeHeight	measureInterval	accuracy	measuringDate	codeQuality	waterAmount	windStrength	windDirection
RG::tubeHeight	×							
A::measureInterval		×						
A::accuracy			×					
R::measuringDate				×				
W::measuringDate				×				
R::codeQuality					×			
W::codeQuality					×			
R::waterAmount						×		
W::windStrength							×	
W::windDirection								×

Kclass
RainGauge
Anemometer
Rainfall
Wind

$r_{hasAttribute}$	RG::tubeHeight	A::measureInterval	A::accuracy	R::measuringDate	W::measuringDate	R::codeQuality	W::codeQuality	R::waterAmount	W::windStrength	W::windDirection
RainGauge	×									
Anemometer		×	×							
Rainfall				×		×		×		
Wind					×		×		×	×

$r_{hasRole}$	rainfall	wind
RainGauge	×	
Anemometer		×
Rainfall		
Wind		

$r_{hasTypeEnd}$	RainGauge	Anemometer	Rainfall	Wind
rainfall			×	
wind				×

3 Previous work on RCA and Class Model Normalization

RCA has been first assessed on small [7] or medium [8] class models, encoding technical information (multiplicity, visibility, being abstract, initial value) in the RCF, which was the source of many non-relevant concepts.

In [9], the authors assessed RCA on Ecore models, Java programs and UML models. The encoding was similar to the one presented in Section 2 to illustrate RCA (classes, attributes, roles, described by their names and their relationships).

While for Java models, the number of discovered class concepts (about 13%) was very reasonable, for UML class models, the increase (about 600%) made the post-analysis impossible to achieve.

Recently, we systematically studied various encodings of the design class model of an information system about Pesticides [10, 11]. We noticed a strong impact of association encoding, and that encoding only named and navigable ends of associations was feasible, while encoding all ends (including those without a name and those that are not navigable) led to an explosion of the number of concepts. Restricting to named ends and to navigable ends means that we give strong importance to the semantics used by designer, thus the lost concepts have a greater chance to be uninteresting.

Guided by the intuition of the model designer, we recently proposed a controlled approach with progressive concept extraction [5]. In this approach, the designer chooses at each step of the RCA process the formal contexts and the relations he wants to explore. For example, he may begin with classes and attributes, then add roles, then associations, then remove all information about classes and consider only classes and operations, etc. Such a choice is memorized in a factorization path. In [5], we used AOC-posets, but we did not evaluate the difference between AOC-posets and concept lattices in the controlled process. The objective was to evaluate the number of discovered concepts at each step and to observe trends in their progress. It was worth noting that the curves of the same factorization path applied to different models had the same shape.

In this paper, we will use the 15 versions of the analysis class model of the same information system on Pesticides, as well as a dataset coming from industrial Java models. Contrarily to the experiments made by authors in [9–11], our objective is to evaluate the benefits of using the variant of RCA, which builds AOC-posets (rather than concept lattices) during the construction process. Studying the concepts discovered at each step, we can also evaluate what was called the *automatic factorization path* in the controlled approach of [5]. It is clear that AOC-posets are smaller than concept lattices, thus the results we expect focus on assessing the amount of the reduction in the number of discovered concepts that will be brought to the designer for analysis.

4 Case study

Experimental setup: Figure 4 presents the metamodel used in practice to define the RCF for our experiments. Object-Attribute contexts are associated to classes, attributes, operations, roles and associations. Attributes, operations, roles and associations are described by their names in UML and Java models. Classes are described by their names in UML models.

Object-object contexts correspond to the meta-relations `hasAttribute`, `hasOperation`, `hasRole` (between `Class` and `Role` and between `Association` and `Role`), `hasTypeEnd`, and `isRoleOf`. This last meta-relation is not used in the Java model RCF, because from Java code we can only extract unidirectional associations (corresponding to Java attributes whose type is a class).

All the roles extracted from Java code have a name. In UML models, we only consider the named roles, and the navigable ends of associations to focus on the most meaningful elements. We do not consider multiplicities in role description, nor the initializer methods (constructors).

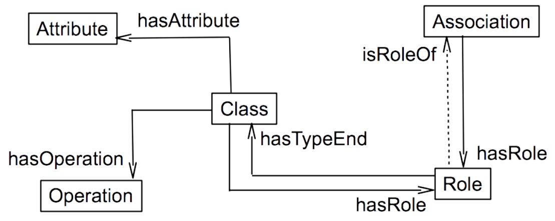

Fig. 4. Metamodel used to define the RCF.

Each of the 15 UML models corresponds to a version of an information system on Pesticides. These models, described in [10], were collected during the Pesticides information system development and then the evolution of the project throughout 6 years. The RCF for UML models contains all the UML model elements. We also used 15 Java models coming from Java programs developed by the company Berger-Levrault in the domain of human resource management. These 15 Java models come from 3 Java programs: Agents, Chapter and Appraisal Campaign. For each program, we determined a central meaningful class and navigated from this central class through association roles at several distances: 1, 2, 4, 8 and 16. For the 5 Java models, we could not get results due to the size of the models. The Java programs are the Java counterpart of the database accesses, thus we focused on Java attributes, that are encoded as attributes when their type is primitive (integer, real, boolean, etc), and as roles of a unidirectional associations when their type is a class. The operations were access and update operations associated to these attributes, thus they do not bring any meaningful information. For the sake of space, we only present results on 4 representative Java models, from the `Chapter` program (distances 1, 2, 4, 8). Depending on the version, 254 to 552 model elements are involved in the Pesticides models and 34 to 171 of these model elements are classes. The Chapter models are composed of 204 to 3979 model elements of which 37 to 282 are classes.

The experiments have been made with the help of UML profiles of Objecteering[1] to extract the RCF from the UML models and Java modules to extract the RCF from the Java models. RCAExplore[2] is used to compute the lattices and

[1] www.objecteering.com/
[2] dolques.free.fr/rcaexplore/

the AOC-posets, and Talend[3] to extract information from RCAExplore outputs, to integrate data, to build the curves and analyze the results.

Results: We computed various metrics on the built lattices and AOC-posets, such as the number of several categories of concepts: merged concepts (simplified extents have a cardinal strictly greater than 1), perennial concepts (simplified extents have a cardinal equals to 1) and new concepts (simplified extents are empty). A merged concept corresponds to a set of model elements that have the same description, for example a merged attribute concept groups attributes with a same name. A perennial concept corresponds to a model element which has at least one characteristic that makes it distinct from the others. A new concept corresponds to a group of a model elements that share part of their description, and no model element has exactly this description (it always has some additional information). We focus in this paper on the classes and on the number of new class concepts, because they are the predominating elements that reveal potential new abstractions and a possible lack of factorization in the current model. To highlight the observed increase brought by the conceptual structures (lattice and AOC-poset), we present the ratio of new class concepts on the number of initial classes in the model (Fig. 5, 6, 7, 8). To compare lattices and AOC-posets, we compute the ratio: $\frac{\#New\ class\ concepts\ in\ lattice}{\#New\ class\ concepts\ in\ AOC{-}poset}$ (Fig. 9, 10). For Chapter models, lattices are computed for the steps 0 to 6 and, for all other cases, lattices or AOC-posets are determined up to step 24.

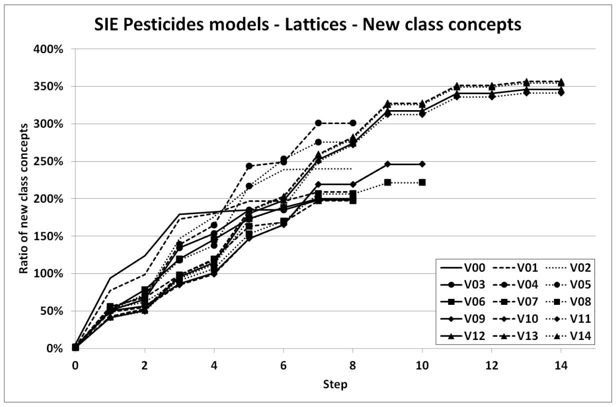

Fig. 5. New class concept number in lattices for Pesticides models

In lattices for the Pesticides models (Fig. 5), the process always stops between steps 6 and 14 depending on the models. At step 6, the ratio of new class concepts on the number of classes varies between 165% (V10) and 253% (V05). Results

[3] www.talend.com/

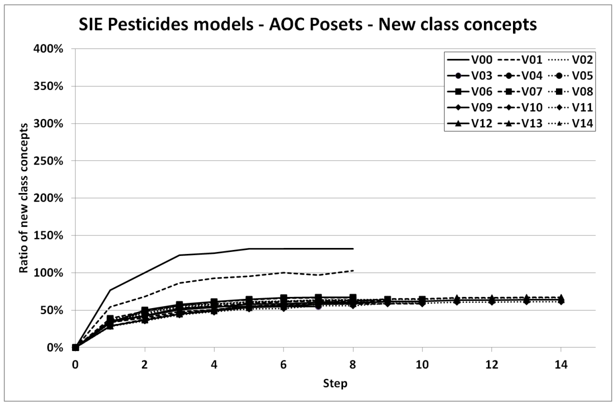

Fig. 6. New class concept number in AOC-posets for Pesticides models

are hard to analyze for a human designer without any further filtering. In lattices for the Chapter models (Fig. 7), we stopped the process at step 6, because we observed a high complexity: for example, for distance 8, at step 6, we get 43656 new class concepts. It is humanly impossible to analyze the obtained new class concepts. At distance 16, Chapter model could not even be processed. For the model at distance 1 (resp. 2), at step 6, the ratio is 11 % (resp. 62%) remaining reasonable (resp. beginning to be hard to analyze). We also observe stepping curves, that are explained by the metamodel: at step 1, new class concepts are introduced due to attribute and role concept creation of step 0, then they remain stable until step 2, while role and association concepts increase, etc...

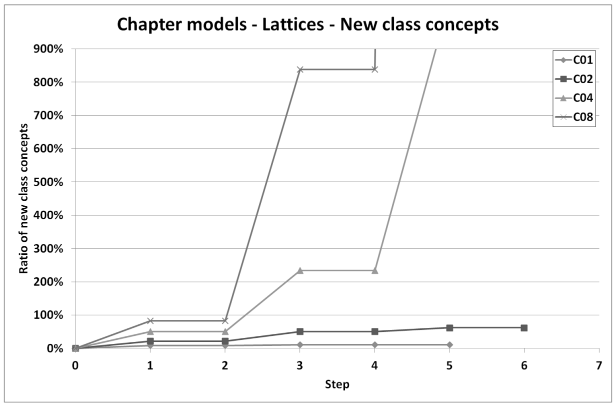

Fig. 7. New class concept number in lattices for Chapter models

Curves for AOC-posets for Pesticides models are shown in Fig. 6. The ordering on Pesticides AOC-posets roughly corresponds to the need of factorization: the highest curves correspond to V00 and V01 where few inheritance relations (there is none in V00) have been used. This shows the factorization work done by the designer during model versioning. The ratio at step 6 varies between 56% (V10) and 132% (V00). In Pesticides lattices, we observed many curve crossings, and curves have different shapes, while with Pesticide AOC-posets, the curves have a regular shape and globally they decrease.

For Chapter models (Fig. 8), convergence is obtained for all AOC-posets (distance 1 to 8) and the process stops between steps 5 and 23. The curves are also ordered and, as for lattices, the highest curve corresponds to the highest distance. The curves reveal many opportunities for factorization. The ratio at step 6 varies between 5% (distance 1) and 161% (distance 8).

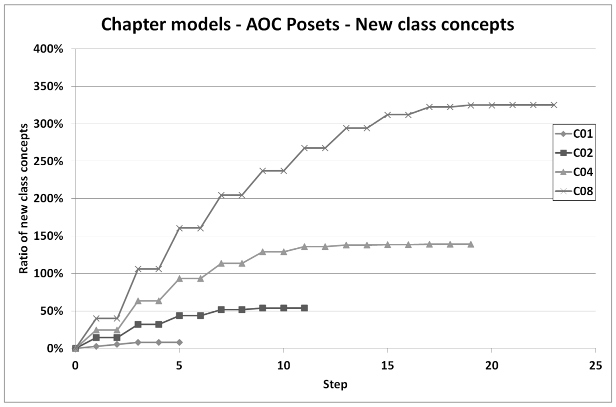

Fig. 8. New class concept number in AOC-posets for Chapter models

Fig. 9 and 10 allow lattices and AOC-posets to be compared. We always have more concepts in lattices than in AOC-posets, which was expected. In the AOC-poset of the Chapter model at distance 1, there is only one new class concept, while in the lattice they are three (including the top and bottom concepts), explaining the beginning of the curve (Fig. 10). For version V00, the behavior of the lattice-based process and the AOC-based process are similar. Except for version V04 (at version V05 a duplication of a part of the model has been made for working purpose), the highest curves correspond to the last models, which have been better factorized, and we here notice the highest difference between the two processes. We may hypothesize that lattices may contain many uninteresting factorizations compared to AOC-posets in these cases. This experiment shows that, in practice, the AOC-posets generally produce results that can be analyzed, while lattices are often difficult to compute in some cases and are often too huge to be used for our purpose.

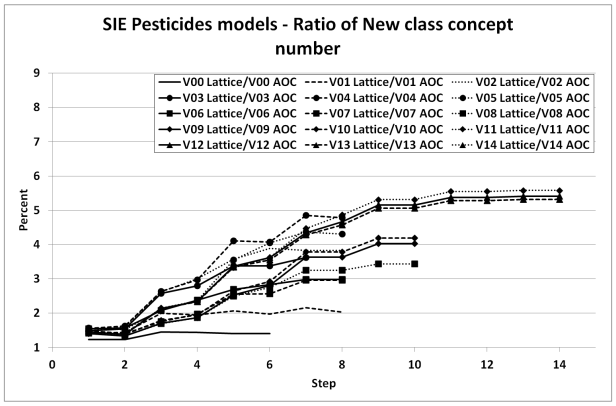

Fig. 9. Ratio $\frac{\#New\ class\ concepts\ in\ lattice}{\#New\ class\ concepts\ in\ AOC-poset}$ for Pesticides models

5 Conclusion

For class model normalization, concept lattices and AOC-posets are two structures giving two different normal forms. UML models that are rebuilt from these structures are interesting in both cases from a thematic point of view. Nevertheless, lattices are often too huge, and AOC-posets offer a good technique to reduce the complexity.

As future work, we plan to go more deeply into an exploratory approach, defining different factorization paths, with model rebuilding at each step with expert validation. This would allow the complexity to be controlled introducing less new concepts at each step. We also plan to use domain ontologies to guide acceptance of a new formal concept, because it corresponds to a thematic concept.

Acknowledgment

This work has been supported by Berger Levrault. The authors also warmly thank X. Dolques for the RCAExplore tool which has been used for experiments.

References

1. Godin, R., Mili, H.: Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices. In: Proceedings of the Eight Annual Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 93), ACM (1993) 394–410
2. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundation. Springer-Verlag Berlin (1999)

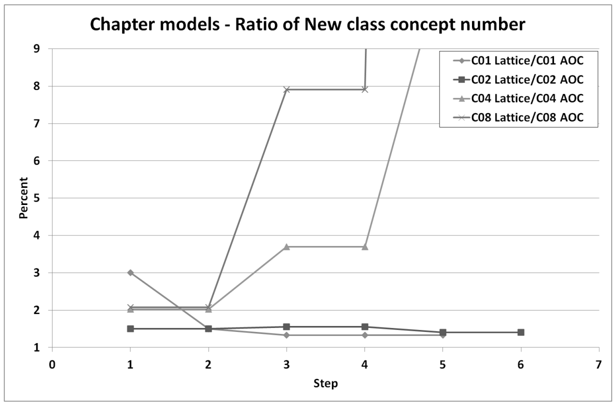

Fig. 10. Ratio $\frac{\#New\ class\ concepts\ in\ lattice}{\#New\ class\ concepts\ in\ AOC-poset}$ for Chapter models

3. Dolques, X., Le Ber, F., Huchard, M.: AOC-Posets: a Scalable Alternative to Concept Lattices for Relational Concept Analysis. In: Proceedings of the Tenth International Conference on Concept Lattices and Their Applications (CLA 2013). Volume 1062 of CEUR Workshop Proceedings., CEUR-WS.org (2013) 129–140
4. Rouane-Hacène, M., Huchard, M., Napoli, A., Valtchev, P.: Relational concept analysis: mining concept lattices from multi-relational data. Ann. Math. Artif. Intell. **67**(1) (2013) 81–108
5. Miralles, A., Huchard, M., Dolques, X., Le Ber, F., Libourel, T., Nebut, C., Guédi, A.O.: Méthode de factorisation progressive pour accroître l'abstraction d'un modèle de classes. Ingénierie des Systèmes d'Information **20**(2) (2015) 9–39
6. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing Iceberg Concept Lattices with TITANIC. Data Knowl. Eng. **42**(2) (2002) 189–222
7. Rouane-Hacène, M.: Relational concept analysis, application to software re-engineering. Thèse de doctorat, Université du Québec à Montréal (2005)
8. Roume, C.: Analyse et restructuration de hiérarchies de classes. Thèse de doctorat, Université Montpellier 2 (2004)
9. Falleri, J.R., Huchard, M., Nebut, C.: A generic approach for class model normalization. In: Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008). (2008) 431–434
10. Osman Guédi, A., Miralles, A., Huchard, M., Nebut, C.: A practical application of relational concept analysis to class model factorization: Lessons learned from a thematic information system. In: Proceedings of the Tenth International Conference on Concept Lattices and Their Applications (CLA 2013). Volume 1062 of CEUR Workshop Proceedings., CEUR-WS.org (2013) 9–20
11. Osman Guédi, A., Huchard, M., Miralles, A., Nebut, C.: Sizing the underlying factorization structure of a class model. In: Proceedings of the 17th IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2013). (2013) 167–172