Analyzing inheritance hierarchies through Formal Concept Analysis: A 22-years walk in a landscape of conceptual structures

Marianne Huchard

Analyzing inheritance hierarchies through Formal Concept Analysis

A 22-years walk in a landscape of conceptual structures

Marianne Huchard

LIRMM (CNRS et Université de Montpellier),
161 rue Ada, 34095 Montpellier Cedex 5,
France
huchard@lirmm.fr

Abstract

Designing or renovating inheritance hierarchies in the domain of programming or in the domain of modeling still remains a tricky task. It involves integrating domain concepts sometimes with no clear frontier, finding the good abstractions and avoiding duplicated information. In this paper, we review research work that addressed this topic with the use of Formal Concept Analysis (concept lattices) since the seminal paper of R. Godin and H. Mili at OOPSLA'93. We overview the different attempts, the explored limits, and the current issues.

Keywords Inheritance hierarchy design, specialization/generalization hierarchy design, Formal Concept Analysis

1. Introduction

Since the beginnings of the object-oriented approaches, the design of inheritance hierarchies (more commonly called specialization/generalization hierarchies in the modeling domain) has been a major concern. Many controversies and issues lay on single versus multiple inheritance (Cargill 1991; Waldo 1991) or between natural modeling versus sub-typing (Castagna 1997; Ducournau 2002). Each approach has its own vision of what a correct inheritance hierarchy should be: main approaches for supporting automated design (and analysis) have considered eliminating duplications, improving abstraction by introducing more general artifacts, in particular super-types or super-classes, and establishing well-founded inheritance links. In the following sections, we will outline the landscape of these works since the 1990s. The objective is not to provide an exhaustive survey, but rather outlining the main landscape components. We will show how lattices, and more particularly Galois/concept lattices played a central role. Two main periods can be observed. In the first period (Section 2), inheritance hierarchies from object-oriented source code and object-oriented database schemas were the focus of attention. Several ad-hoc algorithms have been developed during this period, in parallel with proposals explicitly based on lattices. In the second period (Section 3), the need to take into account elaborate overloading and to generalize to modeling languages (like UML), have resulted in extending the initial single-lattice-based framework to take into account several categories of entities that can be the subject of specialization. This results in a multiple-lattice framework, which is the most advanced theoretical approach as far as we know. We conclude and give some perspectives in Section 4.

2. The single-lattice period: languages, databases, specifications

In the first period, several directions have been studied for eliminating duplication and fostering the appearance of new abstractions. Roughly speaking, the main objective is to eliminate (or to avoid introducing) multiple declarations of a property (attribute or operation). An effective manner of removing such a multiple declaration involves, in simple cases, introducing a copy of this property in a super-class and removing the other copies. In more complex cases, part of the description of the property (like an *abstract* signature) is put in a super-class, and specializations (like concrete methods) remain in the sub-classes. This may require the addition of a new super-class, often addressing a lack of abstraction.

Some existing work deals with local addition of a class (Casais 1992; Godin et al. 1995; Dicky et al. 1995, 1996). While this addition is done, some local pre-existing multiple declarations may be detected and removed and above all, none is introduced. The second strategy flattens an existing hierarchy to rebuild it entirely. Authors have considered then which characteristics the classes own (Casais 1991; Cook 1992; Godin and Mili 1993) or have limited the analysis to characteristics the classes use (Snelting and Tip 1998, 2000). Some work focused on extracting interfaces (Cook 1992; Godin and Mili 1993), while others focused on restructuring a set of classes (Moore 1996; Casais 1992) or a database schema (Missikoff and Scholl 1989; Rundensteiner 1992; Yahia et al. 1996, 1997; Lammari et al. 1998; Cherfi and Lammari 2002). These different points of view explain on which description the methods rely: attribute names only (Astudillo 1997; Yahia et al. 1996, 1997), method signatures only (Cook 1992), attribute names, attribute types, as well as method signatures (Chen and Lee 1996; Moore 1995, 1996; Huchard and Leblanc 2000), constraints (Lammari et al. 1998; Cherfi and Lammari 2002), attribute or parameter types (Missikoff and Scholl 1989; Rundensteiner 1992). Information on the method body is used by Moore (1996); Casais (1992) to find common expressions, while descriptions of method specialization are used by Godin and Mili (1993); Godin et al. (1995, 1998); Dicky et al. (1995, 1996); Casais (1992)

What is really remarkable about almost all approaches, is the similarity in the intended structure. As all the approaches attempt to eliminate duplication, they are guided by the need to factorize characteristics (1), to establish links that correspond to inclusion or refinement of characteristics (2), and to ensure a fairly compact structure (3). For example, a structure where each characteristic is introduced in a specific class would not be compact, because many classes would be necessary. Theory tells us that there is a unique structure that satisfies the three needs. This structure, firstly introduced in the object-oriented domain by Godin and Mili (1993), is the AOC-poset, a particular sub-order of the concept lattice that classifies the classes depending on their characteristics. Formal Concept Analysis (FCA) gives the basics for these conceptual structures (Ganter and Wille 1999). It considers data composed of formal objects (here classes) described by formal attributes (here characteristics like class attributes, class methods, etc.). The input data is a formal context indicating which formal object (class) owns which formal attribute (characteristic). The concept lattice organizes by inclusion the concepts that correspond to all maximal class groups that share a maximal characteristic set. In the concept lattice (also called Galois lattice), a characteristic (resp. a class) is introduced by a unique concept and inherited top-down (resp. bottom-up). The AOC-poset is the concept lattice restricted to the concepts introducing at least one characteristic or at least one class. It drastically reduces the complexity, because for n_{cl} classes and n_{ch} characteristics, the concept lattice may have $2^{min(n_{cl},n_{ch})}$ concepts, while the AOC-poset has less than $n_{cl}+n_{ch}$ concepts. The AOC-poset is also known under the names of pruned lattice, or Galois sub-hierarchy. In our context, concepts of these conceptual structures are interpreted as classes. Concepts that introduce only characteristics are new super-classes. The specialization links in the concept lattice are interpreted as inheritance. The mathematical construction ensures that all the inheritance links are present and consistent. As several descriptions need comparing characteristics and representing specialization between characteristics, this is embedded in formal contexts with taxonomic characteristics. Taxonomic characteristics are sets of characteristics provided with a specialization order. When a class owns a characteristic of one taxonomy, it owns automatically its more general characteristics.

Most of the existing approaches can be compared using this theoretical framework. Some of them build the entire lattice (Missikoff and Scholl 1989; Rundensteiner 1992; Yahia et al. 1996; Snelting and Tip 2000). The majority builds the AOC-poset (Godin and Mili 1993; Godin et al. 1995, 1998; Dicky et al. 1995, 1996; Huchard et al. 2000) or an approximate structure (Cook 1992; Moore and Clement 1996; Chen and Lee 1996; Yahia et al. 1996; Cherfi and Lammari 2002). In some references, including Cook (1992); Moore and Clement (1996); Chen and Lee (1996), the underlying conceptual structure is not identified by the authors, but it has been characterized later in Huchard et al. (2000). Robert Godin proposed to consider the use of conceptual structures as the equivalent of building a normal form in the domain of databases (Godin and Valtchev 2005). Metrics that compare an inheritance hierarchy to such FCA normal form have been designed in this spirit (Dao et al. 2002).

We outline the FCA approach with a simple example (without taxonomic attributes). Figure 1 shows an initial class model with classes (e.g., $FishConsumer$), attributes (e.g., $caloryCount$) and roles (e.g., $prefers$). This class model can be encoded for an FCA analysis by establishing a formal context where classes are described by the names of the attributes and roles that they own. From commonalities between classes that appear in the class lattice, a new class model, shown in Figure 2, can be derived. New superclasses can be seen (names are given afterwards with human intervention): $People$ (factorizes attributes $name$ and age), $Consumer$ (factorizes role $prefer$), $OrgProducer$ (factorizes attribute $labelType$), $Dish$ (factorizes attribute $caloryCount$ and role $prefer$), $Food$ (factorizes attributes $allergen$ and $nutriValue$), and $OrgFood$ (factorizes role $isProducedBy$).

3. The multi-lattice period: class models

When the problem switched from programming languages to modeling languages, it became more clear that using only FCA, even with taxonomic characteristics, was not sufficient to deal with the complexity of extracting hidden abstractions. This issue already existed, for example if classes C_1 and C_2 had attributes whose type was classes C_3 and C_4 respectively, and if C_3 and C_4 themselves had attributes with

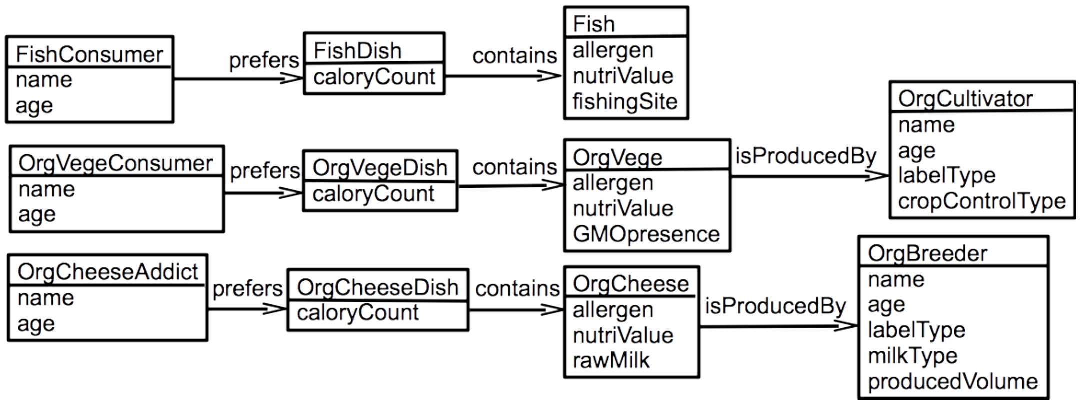

Figure 1. Initial class model on consumers, producers, dishes and food

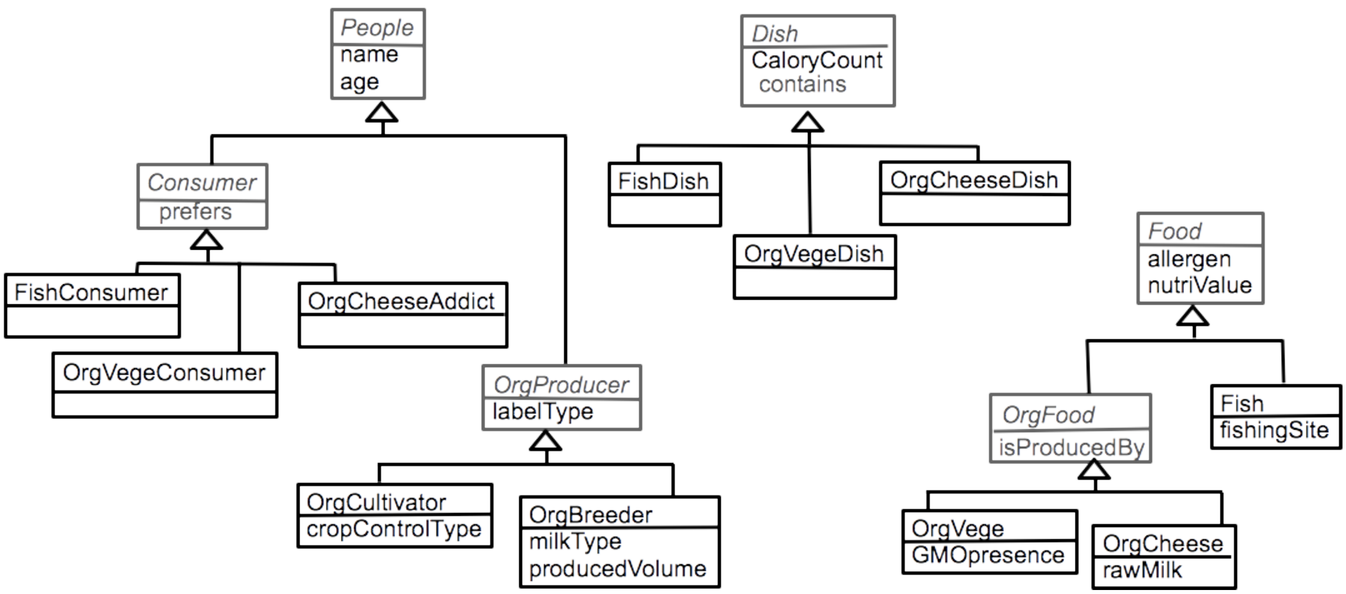

Figure 2. Final class model built with the help of FCA. New abstractions of classes are shown in grey. Roles are integrated as attributes

same type, information could not be taken into account to make a superclass of C_1 and C_2, because it has to be built by iterating on abstraction extraction: firstly, a super-class of C_3 and C_4 has to be built, and only afterwards, this allows to recognize that C_1 and C_2 share attributes whose type is this new super-class.

Relational Concept Analysis (RCA), that extends FCA, has been designed to deal with that iterative abstraction extraction (Dao et al. 2004). It has been later extended to make data mining with any sort of relational data (Hacène et al. 2013). The data model is no more restricted to classes (formal objects) and their characteristics (formal attributes, being attributes, methods, etc. with possible taxonomies). It may now be composed of several categories of formal objects (classes, associations, attributes, roles, operations, parameters, etc.). These formal objects are described by formal attributes (their names, technical properties like being abstract, static, etc.) in object-attribute relations. Besides, they are described in object-object relations like class-owns-attribute, class-owns-role, association-owns-role, operation-owns-parameter. Let us remark that this data model is similar to the meta-model of a programming language (like Java, C++, Smalltalk), or of a modeling language (like OMT, Entity-Relationship, or UML). The formal objects are the instances of this meta-model. The object-object relations come from the meta-relations. The object-attribute relations correspond to the meta-attributes. The chosen data model encodes the entities the data analysis focuses on. The more complete it is, the deeper the analysis is, at the cost of a possible complexity in the results.

Then RCA iteratively builds concept lattices. Each formal object category has its own lattice. Classes are classified in the class lattice, attributes in the attribute lattice, roles in the role lattice, associations in the association lattice, and so on. The concepts from the various lattices are connected by relational attributes that abstract the object-object relations. In class lattices, the concepts are interpreted as classes (possibly new super-classes of existing classes); In attribute lattices, the concepts are interpreted as attributes (possibly new attributes abstractions are built, like "attributes with same name "age" and same type "int"); etc. At the initial step, initial lattices are built using only the object-attribute relations. During the next steps, iteration allows us to propagate the discovered abstractions. Concepts built at step $n-1$ are integrated in the object-object relations by a scaling process, causing the emergence of new concepts at step n. This is done until a fix-point is reached. Let us look at a very simple example. When two attributes a_{C_1} and a_{C_2} with same name a are found in classes C_1 and C_2, at Step 0, an attribute concept C_a is built to group them. At Step 1, C_1 and C_2 can be grouped because they own an attribute from the C_a group. In RCA, several types of scaling process have been defined. For the purpose of class model analysis, the existential scaling is used.

A simple encoding of the class model of Figure 1 with RCA can consider the following object categories: classes, attributes, roles and the object-object relations: $hasAttribute$, $hasRole$ and $hasTypeEnd$. From the class lattice, the attribute lattice and the role lattice, the class model of Figure 3 can be built. Thanks to the reification of all model elements, their connections and to the iterative approach, new super-classes have been discovered: $OrgDish$ (factorizes role $contains$ towards $OrgFood$) $OrgConsumer$ (factorizes role $prefers$ towards $OrgDish$). Besides more detailed information is given on role ends: $isProducedBy$ ends to $OrgProducer$, $prefers$ (of $Consumer$) ends to $Dish$, $prefers$ (of $OrgConsumer$) ends to $OrgDish$, $contains$ (of $Dish$) ends to $Food$, $contains$ (of $OrgDish$) ends to $OrgFood$. It was not possible to find that with the basic FCA approach, because some relation ends cannot be known at the beginning of the process.

As reported in Guédi et al. (2013b), first experiments showed serious limitations because of the complexity of the result (Roume 2004; Hacène 2005; Falleri et al. 2008). The analysis of these experiments and a more systematic experiment done on 15 versions of a class model (Guédi et al. 2013a) allowed us to identify a feasible approach: (1) by avoiding encoding technical aspects that generate useless abstractions, (2) by limiting the scope of analyzed relations (to navigable roles only, or by stopping the process at one step). We are currently focusing on AOC-poset use with a possible risk of non-convergence and on exploratory use of RCA. To give a simple example of the importance of making a good tuning, in experiments reported in Guédi et al. (2013a); Miralles et al. (2015) on the same class model containing $\simeq$170 classes, using lattice structure and a bad configuration may lead at Step 6 to $\simeq$10000 concepts in the class lattice, while using AOC-poset and good configuration leads to $\simeq$200 concepts in the class lattice (at the same Step 6).

In parallel to studies about feasibility, some attempts have been made to tackle issues linked to the names of the model elements: how to recognize hyponyms/hyperonyms, name conflict, synonyms, etc. Initial work was done in Rouane et al. (2007) and focuses on synonyms and name conflicts. It uses similarity measures based on WORDNET (Fellbaum 1998) and LUCENE[1]. The second approach analyzes identifiers with different techniques including Part of Speech Tagging and dependence analysis to organize terms in lexical views (Falleri et al. 2009, 2010). Despite these positive developments, taking into account information conveyed by terms is still under study. Another interest of using the natural language processing techniques would be to propose names for the new discovered abstractions. This is still an open question.

4. Conclusion and perspectives

Even if we overviewed work that spanned over two decades, we think that the design of inheritance (or specialization/generalization) hierarchies is still a complex issue that should benefit from tool assistance with the approaches that we presented above. Among the current work, we are designing ways of efficiently presenting the results to the designers. The progress of ontology engineering and the possibility to combine knowledge from ontologies with knowledge included in source code, UML models or information systems models bring new perspectives for improving class hierarchy design. The multiplication of models on same topics and domains and the necessity to communicate information between various software systems also prompted the need for aligning class models, an issue which could be investigated with the help of RCA.

Acknowledgments

Most of the work reported here would not have been possible without the support of many students and colleagues, who are co-authors of the cited publications and to whom I am sincerely grateful. Tools Galicia[2], erca[3], RCAexplore[4], and AOC-poset builder[5] as well as the Pesticides model designed by André Miralles (Pinet et al. 2010) and its 15 versions have played a key role.

[1] http://lucene.apache.org/java/docs/index.html

[2] http://www.iro.umontreal.ca/∼galicia/

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

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

[5] http://www.lirmm.fr/AOC-poset-Builder/

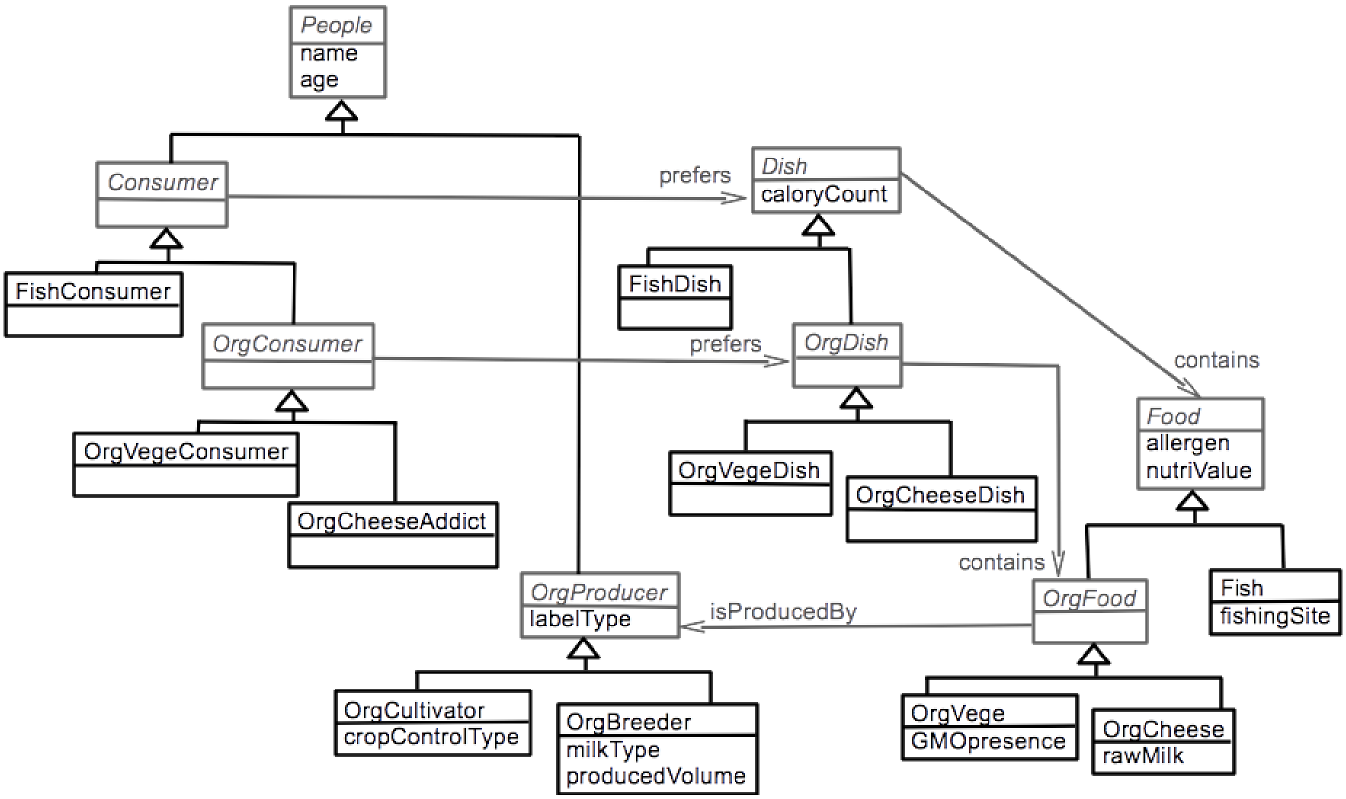

Figure 3. Final class model built with the help of RCA. New abstractions of classes and roles are shown in grey. The most specialized roles are not presented for the sake of readability (for example the role $prefers$ which connects $FishConsumer$ and $FishDish$ is not drawn here, although it is still present). They can be introduced in UML, with "subsets" modifier, for example the most specialized role $OrgCheese$-$isProducedBy$-$OrgBreeder$ must be declared "subsets $isProducedBy$" to indicate it must be a specialization of $OrgFood$-$isProducedBy$-$OrgProducer$. Data of the illustrative example can be found at: http://www.lirmm.fr/recherche/equipes/marel/datasets/dataset-consumers-producers-dishes-food

References

H. Astudillo. Maximizing object reuse with biological metaphor. *Theory and Practice of Object Systems*, 3(4):235–251, 1997.

T. A. Cargill. Controversy: The case against multiple inheritance in C++. *Computing Systems*, 4(1):69–82, 1991.

E. Casais. *Managing Evolution in Object Oriented Environments : An Algorithmic Approach*. Thèse de doctorat, Université de Genève, 1991.

E. Casais. An incremental class reorganization approach. In O. Lehrmann Madsen, editor, *Proceedings of ECOOP'92, Utrecht, The Netherlands*, volume 615 of *Lecture Notes in Computer Science*, pages 133–152. Springer-Verlag, Berlin, 1992.

G. Castagna. *Object-Oriented Programming, a Unified Foundation*. Birkhauser, 1997.

J.-B. Chen and S. C. Lee. Generation and reorganization of subtype hierarchies. *Journal of Object Oriented Programming*, 8(8), 1996.

S. S.-S. Cherfi and N. Lammari. Towards and Assisted Reorganization of Is-A Hierarchies. In *Object-Oriented Inf. Systems*, volume 2425 of *LNCS*, pages 536–548. Springer-Verlag, 2002.

W. Cook. Interfaces and Specifications for the Smalltalk-80 Collection Classes. In *Proceedings of OOPSLA'92, Vancouver, Canada*, special issue of ACM SIGPLAN Notices, 27(10), pages 1–15, 1992.

M. Dao, M. Huchard, T. Libourel, C. Roume, and H. Leblanc. A new approach to factorization - introducing metrics. In *8th IEEE International Software Metrics Symposium (METRICS 2002), 4-7 June 2002, Ottawa, Canada*, pages 227–236. IEEE Computer Society, 2002.

M. Dao, M. Huchard, M. R. Hacène, C. Roume, and P. Valtchev. Improving Generalization Level in UML Models Iterative Cross Generalization in Practice. In *ICCS 2004*, pages 346–360, 2004.

H. Dicky, C. Dony, M. Huchard, and T. Libourel. ARES, Adding a class and REStructuring Inheritance Hierarchies. In *11 ièmes journées Bases de Données Avancées, Nancy*, pages 25–42, 1995.

H. Dicky, C. Dony, M. Huchard, and T. Libourel. On automatic class insertion with overloading. In *Proceedings of OOPSLA'96, San Jose (CA), USA*, special issue of ACM SIGPLAN Notices, 31(10), pages 251–267, 1996.

R. Ducournau. "real world" as an argument for covariant specialization in programming and modeling. In J. Bruel and Z. Bellahsene, editors, *Advances in Object-Oriented Information Systems, OOIS 2002 Workshops, Montpellier, France, September 2, 2002, Proceedings*, volume 2426 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2002. .

J. Falleri, V. Prince, M. Lafourcade, M. Dao, M. Huchard, and C. Nebut. Using natural language to improve the generation of model transformation in software design. In *Proceedings of the International Multiconference on Computer Science and Information Technology, IMCSIT 2009, Mragowo, Poland, 12-14 October 2009*, pages 199–206. IEEE, 2009.

J. Falleri, M. Huchard, M. Lafourcade, C. Nebut, V. Prince, and M. Dao. Automatic extraction of a wordnet-like identifier network from software. In *18th IEEE ICPC 2010, Braga, Minho, Portugal, June 30-July 2, 2010*, pages 4–13. IEEE Computer Society, 2010. ISBN 978-0-7695-4113-6.

J.-R. Falleri, M. Huchard, and C. Nebut. A generic approach for class model normalization. In *ASE 2008*, pages 431–434, 2008.

C. Fellbaum. *Wordnet: An Electronic Lexical Database*. MIT Press, 1998. ISBN 0-262-06197-X.

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

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

R. Godin and P. Valtchev. Formal concept analysis-based class hierarchy design in object-oriented software development. In *Formal Concept Analysis, LNCS 3626*, pages 304–323, 2005.

R. Godin, G. Mineau, and R. Missaoui. Incremental structuring of knowledge bases. In G. Ellis, R. Levinson, A. Fall, and V. Dahl, editors, *1st Int. Symp. on Knowledge Retrieval, Use, and Storage for Efficiency (KRUSE'95)*, pages 179–193. Dept. of Computer Science, University of California at Santa Cruz, 1995.

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*, 4(2), 1998.

A. O. Guédi, M. Huchard, A. Miralles, and C. Nebut. Sizing the underlying factorization structure of a class model. In *17th IEEE EDOC 2013, Vancouver, BC, Canada, September 9-13, 2013*, pages 167–172, 2013a. .

A. O. Guédi, A. Miralles, M. Huchard, and C. Nebut. A practical application of relational concept analysis to class model factorization: Lessons learned from a thematic information system. In *10th Int. Conf. on Concept Lattices and Their Applications, 2013.*, pages 9–20, 2013b. URL `http://ceur-ws.org/Vol-1062/paper1.pdf`.

M. R. Hacène. *Relational concept analysis, application to software re-engineering*. Thèse de doctorat, Université du Québec À Montreal, 2005.

M. R. Hacène, M. Huchard, A. Napoli, and P. Valtchev. Relational concept analysis: mining concept lattices from multi-relational data. *Ann. Math. Artif. Intell.*, 67(1):81–108, 2013.

M. Huchard and H. Leblanc. Computing Interfaces in Java. In *Proc. IEE International conference on Automated Software Engineering (ASE'2000)*, pages 317–320, 11-15 September, Grenoble, France., 2000.

M. Huchard, H. Dicky, and H. Leblanc. Galois lattice as a framework to specify algorithms building class hierarchies. *Theoretical Informatics and Applications*, 34:521–548, January 2000.

N. Lammari, R. Laleau, and M. Jouve. Processus d'optimisation conceptuelle d'un schéma orienté-objet. *L'objet*, 1998.

A. Miralles, M. Huchard, X. Dolques, F. L. Ber, T. Libourel, C. Nebut, and A. Osman-Guédi. Méthode de factorisation progressive pour accroître l'abstraction d'un modèle de classes. *Ingénierie des systèmes d'information*, 2:9–39, 2015.

M. Missikoff and M. Scholl. An Algorithm for Insertion into a Lattice: Application to Type Classification. *Proceedings of the 3rd Int. Conf. FODD'89*, pages 64–82, 1989.

I. Moore. Guru - A Tool for Automatic Restructuring of Self Inheritance Hierarchies. In *Proceedings of TOOLS-Europe'95, Versailles, France*, 1995.

I. Moore. Automatic Inheritance Hierarchy Restructuring and Method Refactoring. In *Proceedings of OOPSLA'96, San Jose (CA), USA*, special issue of ACM SIGPLAN Notices, 31(10), pages 235–250, 1996.

I. Moore and T. Clement. A Simple and Efficient Algorithm for Inferring Inheritance Hierarchies. In *Proceedings of TOOLS-Europe'96, Paris, France*, 1996.

F. Pinet, A. Miralles, S. Bimonte, F. Vernier, N. Carluer, V. Gouy, and S. Bernard. The use of UML to design agricultural data warehouses. In *International Conference on Agricultural Engineering (AgEng 2010)*, pages 1–10, 2010.

M. H. Rouane, M. Dao, M. Huchard, and P. Valtchev. Aspects de la réingénierie des modèles UML par analyse de données relationnelles. *Ingénierie des Systèmes d'information (RSTI série)*, 12:39–68, 2007.

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

E. A. Rundensteiner. A Class Classification Algorithm For Supporting Consistent Object Views. Technical report, University of Michigan, 1992.

G. Snelting and F. Tip. Reegineering class hierarchies using conept analysis. In *SIGSOFT '98*, pages 99–110, Novembre 1998.

G. Snelting and F. Tip. Understanding class hierarchies using concept analysis. *ACM Transactions on Programming Languages and Systems*, 22(3):540–582, May 2000.

J. Waldo. Controversy: The case for multiple inheritance in C++. *Computing Systems*, 4(2):157–171, 1991.

A. Yahia, L. Lakhal, R. Cicchetti, and J. Bordat. iO2, An Algorithmic Method for Building Inheritance Graphs in Object Database Design. In *Proceedings of the 15th International Conference on Conceptual Modeling ER'96*, 1996.

A. Yahia, L. Lakhal, and J. Bordat. Designing Class Hierarchies of Object Database Schemas. In *13 ièmes journées Bases de Données Avancées*, pages 371–390, 1997.