# Using Formal Concept Analysis for Discovering Knowledge Patterns

Amine Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, Petko Valtchev

# Using Formal Concept Analysis for discovering knowledge patterns

Mohamed Rouane-Hacene[1], Marianne Huchard[2], Amedeo Napoli[3], and Petko Valtchev[1]

[1] Dépt. informatique, UQAM, C.P. 8888, Succ. CV, Montréal, Canada H3C 3P8
[2] LIRMM, 161, rue Ada, F-34095 Montpellier
[3] LORIA, BP 70239, F-54506 Vandœuvre-lès-Nancy

**Abstract.** Design patterns are used in software engineering for guiding code design: they play the role of models to be followed for producing code of better quality. In the same way, knowledge patterns are introduced in knowledge engineering as ontology components that can be used as models and reused as ontology design patterns (ODPs) in ontology engineering. Accordingly, we present in this paper the use of both Formal Concept Analysis (FCA) and Relational Concept Analysis (RCA) for designing compact concept lattices that can be re-engineered as ODPs. Starting with a simple example, it is shown how to derive conceptual and relational abstractions that can be re-engineered as knowledge patterns, following and adapting the meta-modeling activity in software engineering. This paper aims at showing that FCA and RCA are efficient and reliable tools for guiding knowledge engineering. Moreover, this is one of the few attempts to generate effective knowledge patterns from data available as sets of objects, attributes, and relations between objects.

## 1 Introduction

An ontology is supposed to capture and organize domain knowledge within a set of concept and relation definitions encoded in a knowledge representation language [15]. While ontologies are modules of first importance in the framework of Semantic Web, there does not exist yet a universal approach to ontology engineering [12, 9] contrasting the widespread use of UML for software engineering [11, 14]. Moreover, there is a growing interest in considering ontology engineering as a composition of "knowledge patterns" in the same way as code is built using and assembling design patterns in an application software [3, 6]. Such an approach to software engineering favors *model-driven engineering* where the specification of a software is given by a "model " which is further refined. This model is made of classes, relations between classes, and operations associated to the classes and modeling the basis of the software. Let us consider for example the "composite pattern" which is used to represent simple objects and their composition in a composite object with a common interface, i.e. their operations are the same and they understand the same messages. It can be used to represent

files and directories in a Unix system, where the common interface includes operations for renaming, changing access rights, moving, removing, etc. The design solution uses a class "Component" containing the common operations (mostly in abstract form), which is specialized into a class "Leaf" to represent simple objects and a class "Composite" which represents the composite objects [5, 10].

In this paper, we transpose ideas from code design in software engineering for guiding ontology engineering. Actually, we are interested in the refactoring of models such as UML models, as ontological and reusable components. We start with an informal and small-sized problem statement represented within sets of objects, attributes, and relations between objects, or UML diagrams. The later are the inputs of a re-engineering process based on Formal Concept Analysis (FCA [7]) and Relational Concept Analysis (RCA [13]) whose output are concept lattices. These lattices provide a number of potential constructions that can be then applied by a knowledge engineer (a designer) for building a final knowledge pattern. The resulting concepts and relations, and their hierarchical organization, can then be represented into Description Logic (DL [1]) as a set of concepts and roles to be reused as a pattern in ontology engineering. This approach can be considered as a semi-automatic method for designing "ontology design patterns" (ODPs) or "knowledge patterns', i.e. small-sized and compact components that can be reused as building blocks for ontology engineering [3, 6].

A parallel can be drawn between DL-based ontology engineering and concept lattice design [2, 16]. The notion of a concept covering a set of objects or instances (extent) sharing a set of attributes (intent) is common to both FCA and to DL. However, concept and relation descriptions are not obtained in the same way. In the DL framework, the design can be considered as *top-down* as in object-oriented design [14]. The representation begins with abstract atomic concepts and roles, that are then combined with operators for building more complex definitions. A classifier can be used to organize the concepts and roles into a subsumption hierarchy. This top-down approach is intensional and partial w.r.t. domain knowledge: concepts and roles are described as intents, i.e. sets of complex attributes, and only a part of the domain knowledge is represented. By contrast, FCA can be viewed as a learning process working *bottom-up*. FCA builds a concept lattice from a binary table $\texttt{Objects} \times \texttt{Attributes}$ where a concept has an extent and an intent, i.e. a maximal set of objects sharing a maximal set of attributes respectively [7]. This bottom-up approach is extensional as it directly works on sets of objects and attributes, and complete w.r.t. the initial data in the table. Moreover, relations between objects can be taken into account in the Relational Concept Analysis framework (RCA [13]). RCA builds a concept lattice from binary tables $\texttt{Objects} \times \texttt{Attributes}$ and $\texttt{Objects} \times \texttt{Objects}$, where the intent of concepts is made of classical binary attributes and relational attributes as well, reifying links between objects.

As an original contribution, this paper details a process based on FCA and RCA providing a designer a lattice structure supporting a knowledge pattern. This knowledge pattern can be reused for ontology engineering and can be used

either *per se* or for designing and completing an ontology. The process is semi-automated as it requires external interaction. Finally, this is one of the few attempts to automatize the discovery of lattice structures supporting knowledge patterns from data.

The paper is organized as follows. Firstly, a working example is presented and modeling problems are discussed. Then the FCA and RCA frameworks are detailed. The evolution from modeling to concept lattices is discussed in each case. A synthesis and a discussion conclude the paper.

## 2 An example of domain modeling

The design of an ontology for a domain of interest is a key task in knowledge engineering, in particular for Semantic Web applications [9, 15]. The task is rarely fully automated: important steps are usually performed manually by the designer. In fact, a parallel can be drawn between knowledge and software engineering. For example, use case-driven object modeling consists in deriving artifacts of a software, including the class model, from use cases [11, 14]. A use case model captures high-level system requirements through a set of system usage "scripts", whereas the class model describes an infrastructure supporting those requirements, i.e., classes realizing the necessary functionalities. The class model emphasizes the data manipulated by the system and reflects domain entities, *associations* composed of two *roles*, one for each extremity of the association. The problem is the same from a knowledge engineering perspective: how to build a satisfactory, concise, and complete model w.r.t. the domain, and then to represent this model within a knowledge representation formalism.

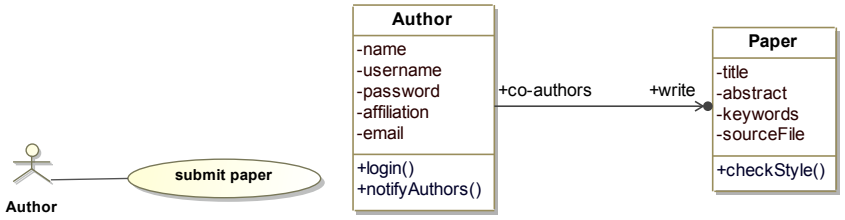


**Fig. 1.** The use case "submit a paper" and its class model in UML-like notation.

We introduce now the running example on which relies this paper, related to the submission and the evaluation of a paper for a conference. The diagrams on the right-hand side of Figures 1 and 2 capture the main classes of the use cases "submit a paper" and "review a paper", respectively, both modeling an operation within a conference management system (CMS). These two models can be considered as the result of a first step in modeling a CMS. They are both partial and reveal various shortcomings w.r.t. quality criteria such as completeness

and redundancy. Indeed, classes modeling conference contributors share many properties, including personal data, that should be centralized in a common superclass, say `Participant`. Moreover, a superclass representing the generic concept of conference artifacts, i.e., papers or reviews, is missing. Such a class, say `Submission`, should be linked to the `Participant` class by an association abstracting the relationships between, on one hand, the different types of contributors, and, on the other hand, the corresponding types of submissions.

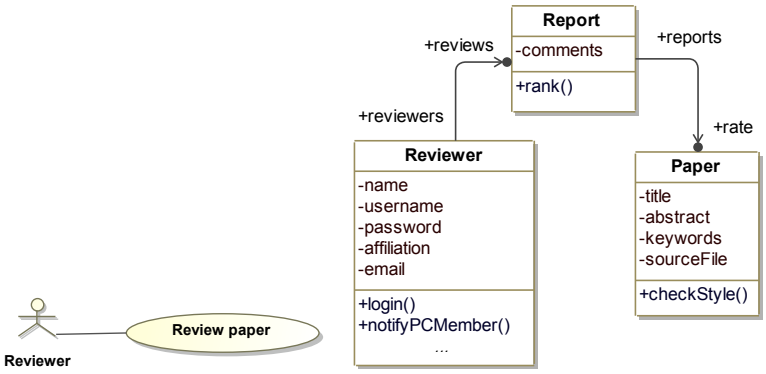


**Fig. 2.** The use case "review a paper" and its class model.

Accordingly, Figure 3 shows a possible factorization of the two partial models. Due to its compactness and its maximal factorization, such a model could support a knowledge pattern to be reused in ontology engineering for CMS and –after adaptation– for CMS-like applications, e.g. participation to an event, an examination, a competition, a course, etc.

In the following, we show how the use of FCA and RCA leads to the discovery of series of construction operations that can help the knowledge engineer in building a global "class model" for CMS. In this resulting class model, the relations between objects are of first importance and should be taken into account. Yet, albeit suitable for abstracting class identification in software and knowledge engineering tasks [8], FCA ignores these links inherent to any realistic datasets and models. At this point RCA comes into play which allows the analysis of objects and relations between objects. Below, a short overview of FCA and RCA precedes the analysis of the CMS problem and the design of a class model supporting and associated knowledge pattern.

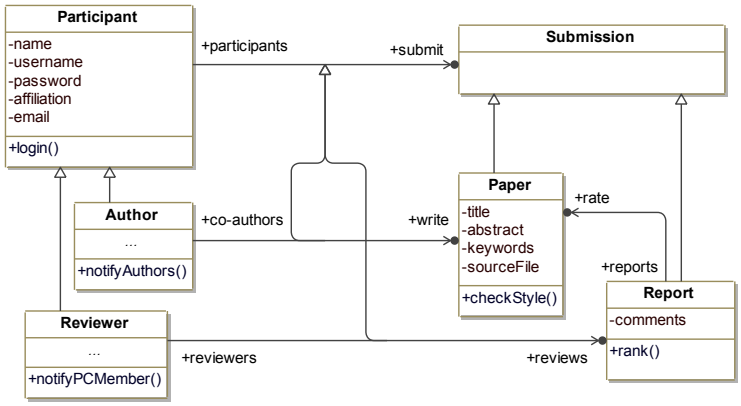


**Fig. 3.** A possible class model for the "Conference Manager System" (CMS).

# 3 FCA and RCA frameworks

## 3.1 The basics of FCA

The basics of FCA are introduced in [7]. In the present case, the formal context given in Table 1 and denoted by $\mathcal{K}_{\texttt{class}}$ binds a set of objects –the four "classes" of the CMS class model– to attributes –the class variables and methods. The corresponding concept lattice $\mathcal{L}_{\texttt{class}}$ in drawn on Figure 4, where, e.g. the concept `Concept_5` represents the `Reviewer` class in the CMS system. The full extent of `Concept_5` is {`Reviewer`} and the full intent is the set {`name`, `username`, `password`, `affiliation`, `email`, `login`(), `notifyPCMember`()}.

| | name | username | password | affiliation | email | title | abstract | keywords | sourceFile | comments | login() | checkStyle() | notifyPCMember() | rank() | notifyAuthors() |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Author | × | × | × | × | × | | | | | | × | | | | × |
| Paper | | | | | | × | × | × | × | | | × | | | |
| Reviewer | × | × | × | × | × | | | | | | × | | × | | |
| Report | | | | | | | | | | × | | | | × | |

**Table 1.** A context encoding the incidence relation between objects, i.e. the "classes" of the CMS model, and their attributes. A cross × means that object $\texttt{o}_\texttt{i}$ has attribute $\texttt{a}_\texttt{j}$.

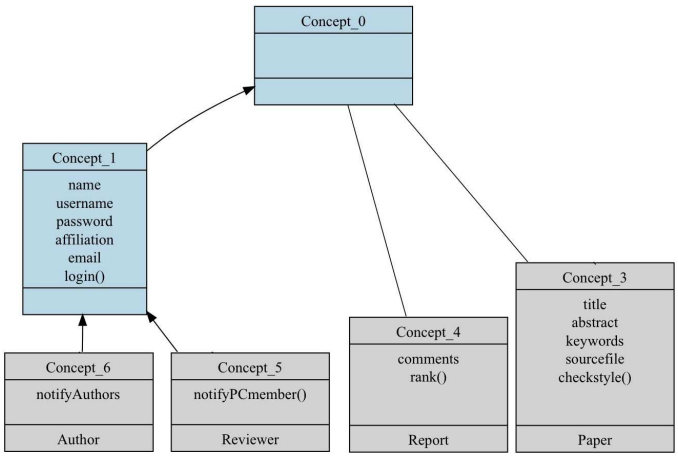


**Fig. 4.** The concept lattice $\mathcal{L}_{\texttt{class}}$ corresponding to the formal context $\mathcal{K}_{\texttt{class}}$ where the bottom concept $\bot$ is omitted.

Beside a class hierarchy, an integrated class model must include associations that are available within partial models, and possibly, abstractions of these associations. The abstraction of associations requires encoding association roles into a formal context together with their attributes. Indeed, the RCA framework presented hereafter addresses these concerns, allowing FCA to take effectively and efficiently into account relational data.

### 3.2 The basics of RCA

RCA was introduced and detailed in [13]. Data is described by a *relational context family* (RCF), composed of a set of contexts $\mathbf{K} = \{\mathcal{K}_{\mathtt{i}}\}$ and a set of binary relations $\mathbf{R} = \{\mathtt{r_k}\}$. A relation $\mathtt{r_k} \subseteq \mathtt{O_j} \times \mathtt{O}_\ell$ connects two object sets, a *domain* $\mathtt{O_j}$ ($\mathtt{dom}(\mathtt{r_k}) = \mathtt{O_j}$) and a *range* $\mathtt{O}_\ell$ ($\mathtt{ran}(\mathtt{r_k}) = \mathtt{O}_\ell$). For example, the RCF corresponding to the CMS example could be composed of two contexts, namely $\mathcal{K}_{\texttt{class}}$ (Table 1) and $\mathcal{K}_{\texttt{assoc}}$ (Table 2), and a set of relations $\mathbf{R} = \{\mathtt{owns}, \mathtt{type}\}$ (see Table 3). $\mathcal{K}_{\texttt{class}}$ encodes classes as objects and class characteristics as attributes, while $\mathcal{K}_{\texttt{assoc}}$ encodes association roles as objects and role names as attributes.

Hereafter, we briefly recall the mechanism of RCA necessary for understanding the following (details are given in [13]). RCA is based on a *relational scaling* mechanism that transforms a relation $\mathtt{r_k}$ into a set of *relational attributes* that are added to the context describing the object set $\mathtt{dom}(\mathtt{r_k})$. To that end, relational scaling follows the DL semantics of role restrictions. For each relation $\mathtt{r_k} \subseteq \mathtt{O_j} \times \mathtt{O}_\ell$, there is an *initial lattice* for each object set, i.e. $\mathcal{L}_{\mathtt{j}}$ for $\mathtt{O_j}$ and $\mathcal{L}_\ell$ for $\mathtt{O}_\ell$. For example, considering $\mathtt{owns} \subseteq \mathtt{O}_{\texttt{class}} \times \mathtt{O}_{\texttt{assoc}}$ these initial lattices are respectively $\mathcal{L}_{\texttt{class}}$ (Figure 4) and $\mathcal{L}_{\texttt{assoc}}$ (Figure 5).

Given a relation $\mathtt{r_k} \subseteq \mathtt{O_j} \times \mathtt{O}_\ell$, a relational attribute $\exists \mathtt{r_k} : \mathtt{c} \text{ –} \mathtt{c}$ being a concept and $\exists$ the existential quantifier– is associated to an object $\mathtt{o} \in \mathtt{O_j}$ whenever

| | 'write' | 'reviews' | 'co-authors' | 'reviewers' | 'rate' | 'reports' |
|---|---|---|---|---|---|---|
| write | × | | | | | |
| reviews | | × | | | | |
| co-authors | | | × | | | |
| reviewers | | | | × | | |
| rate | | | | | × | |
| reports | | | | | | × |

**Table 2.** The context $\mathcal{K}_{\texttt{assoc}}$ describing association roles in Figures 1 and 2.

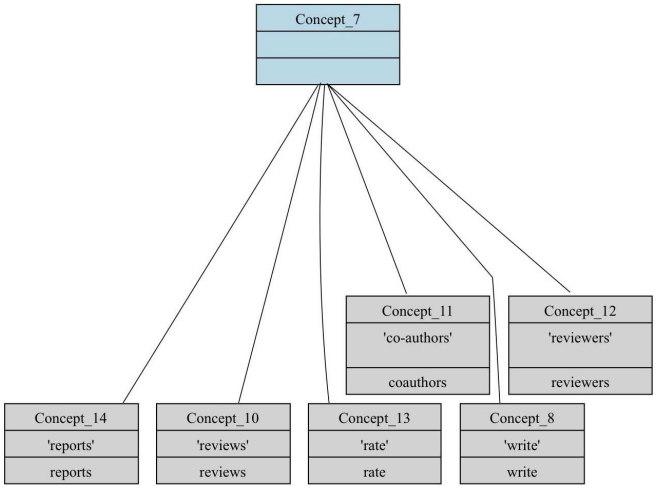


**Fig. 5.** The concept lattice $\mathcal{L}_{\texttt{assoc}}$ corresponding to the formal context $\mathcal{K}_{\texttt{assoc}}$ (Table 2) where the bottom concept $\perp$ is omitted.

$\mathtt{r_k(o)} \cap \mathtt{extent(c)} \neq \emptyset$ (other quantifiers are available, see [13]). For example, let us consider the concept `Concept_10` in the concept lattice $\mathcal{L}_{\texttt{assoc}}$ representing the association role `reviews` (see Figure 5). The attribute "`owns` : `Concept_10`" is associated to the concept `Concept_5` in the lattice $\mathcal{L}_{\texttt{class}}$ that models the class `Reviewer` in the CMS system (see Figure 6).

## 4 The design of the knowledge pattern "CMS"

In this section, we detail the application of the FCA and then RCA processes on the relational context family $\mathbf{K} = \{\mathcal{K}_{\texttt{class}}, \mathcal{K}_{\texttt{assoc}}\}$ and $\mathbf{R} = \{\texttt{owns}, \texttt{type}\}$.

### 4.1 The three main steps of the design

**Design step#0.** In the initial situation –step #0– of the RCA process, we have two lattices built from the class and association role contexts. Actually, these

| owns | write | reviews | co-authors | reviewers | rate | reports |
|---|---|---|---|---|---|---|
| Author | × | | | | | |
| Paper | | | × | | | × |
| Reviewer | | × | | | | |
| Report | | | | × | × | |

| type | Author | Paper | Reviewer | Report |
|---|---|---|---|---|
| write | | × | | |
| reviews | | | | × |
| co-authors | × | | | |
| reviewers | | | × | |
| rate | | × | | |
| reports | | | | × |

**Table 3.** The relations owns and type between classes and association roles. An Author owns the role write, i.e. the domain of write is Author, and write type(s) Paper, i.e. the range of write is Paper.

two initial lattices can be read on Figures 6 and 7 where their concepts have their names ended by (0) (actually, the numbering of concepts in the concept lattices $\mathcal{L}_{\texttt{class}}$ and $\mathcal{L}_{\texttt{assoc}}$ is kept all along the design process). At this step, the relations owns and type are not yet considered. In the class lattice, the concepts of interest are: Concept_0 (Top), Concept_6 (Author), Concept_5 (Reviewer), Concept_4 (Report), and Concept_3 (Paper). In addition, Concept_1 corresponds to a generalization of the concepts Author and Reviewer and factorizes their attributes and methods.

This generalization provides the knowledge engineer a first clue, i.e. a construction for building a concept that can be called Participant generalizing concepts Author and Reviewer.

**Design step#1.** At this step, the relational contexts are considered with relations owns, i.e. domains of association roles, and type, i.e. range of association roles. More precisely, a concept C has an attribute such as "owns : X" where X is a concept whenever C "owns" a role present in the extent of X. For example, the concept Concept_6 in $\mathcal{L}_{\texttt{class}}$ has an attribute "owns : Concept_8", meaning that Author (extent of Concept_6 in $\mathcal{L}_{\texttt{class}}$) owns the role write (extent of Concept_8 in $\mathcal{L}_{\texttt{assoc}}$). Attributes of the form "type : Y" are built in the same way. The complements and generalizations appearing in the lattices at step#1 are the following:

- In the class lattice, the concept Concept_6 (step#0) gets the attribute "owns : Concept_8" indicating that Author owns the role write.
- In the association role lattice, Concept_15 corresponds to a generalization of co-authors and reviewers, whose type is Concept_1 in the class lattice, i.e.

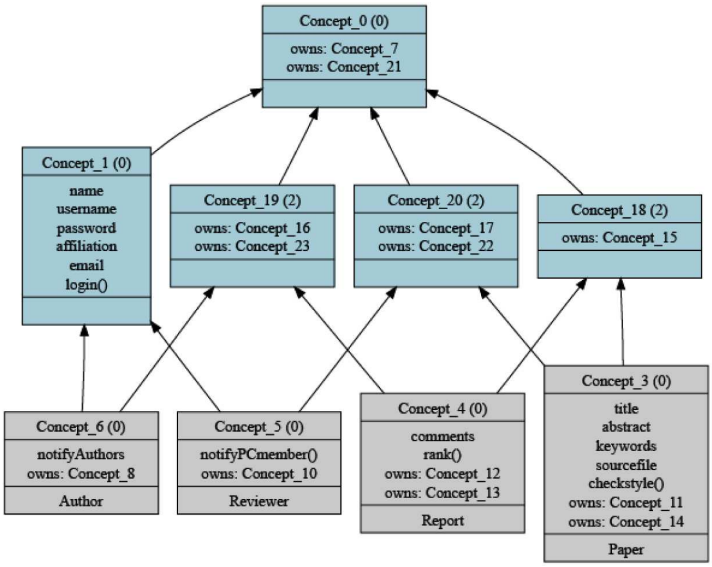


**Fig. 6.** The final "class lattice" where the concept $\perp$ is omitted.

`Participant`. Actually, this provides a second clue and allows the knowledge engineer to discover the role `participants` of the final model.
- In the same way, `Concept_17` generalizes roles related to `Report`, i.e. `reports` and `reviews`, and `Concept_16` generalizes roles related to `Paper`, i.e. `rate` and `write`.

**Design step#2.** At this step, the association role lattice is not significantly modified but the following concepts are created in the class lattice:

- `Concept_18` denotes objects related to `Participant` through the relational attribute "`owns` : `Concept_15`", generalization of the roles `co-authors` and `reviewers`. `Concept_18` is particularly interesting and provides a third clue for the knowledge engineer, i.e. it allows to discover a generalization of concepts `Paper` and `Report` that can be called `Submission`.
- `Concept_19` denotes objects related to `Paper` through "`owns` : `Concept_16`" generalizing `rate` and `write`,
- `Concept_20` denotes objects related to `Report` through "`owns` : `Concept_17`" generalizing `reports` and `reviews`.

**Design step#3.** At step#3, the following generalizations are created in the association role lattice:

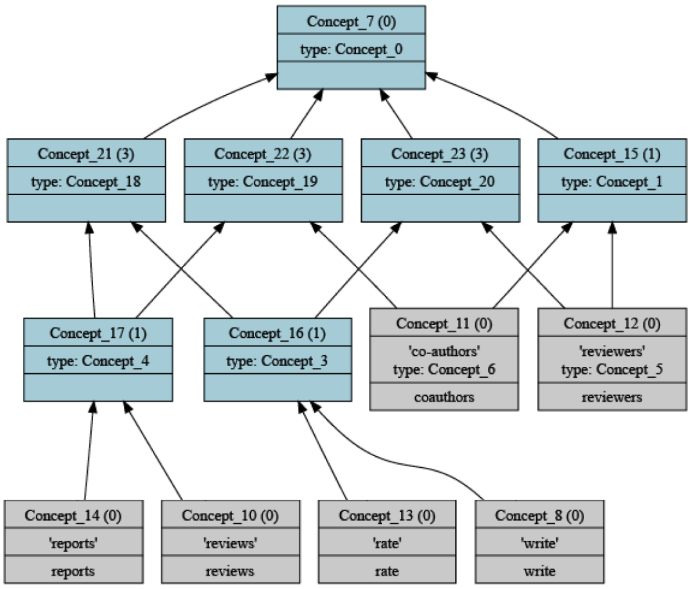


**Fig. 7.** The final "association role lattice" where the concept $\perp$ is omitted.

- `Concept_21` is a role attached to objects related to `Submission` and provides a fourth clue, i.e. it allows to discover the role called `submit` of the knowledge pattern,
- `Concept_22` is a role whose type is composed of objects related to `Paper`, and `Concept_23` is a role whose type is composed of objects related to `Report`.

The following generalizations are created in the class lattice:

- `Concept_19` gets the role "`owns` : `Concept_23`", i.e. `Author` and `Report` are related to objects themselves related to objects related to `Report`,
- `Concept_20` gets the role "`owns` : `Concept_22`", i.e. `Reviewer` and `Paper` are related to objects themselves related to objects related to `Paper`...

### 4.2 Synthesis and discussion

There are four main steps in the design of the class and association role lattice through the FCA and RCA processes. The two lattices will help the knowledge engineer to build an associated knowledge pattern based on the discovery of the different concepts and relations denoted above as "clues":

1. At step#0, `Concept_1` yields the generalization `Participant` (first clue).

2. At step#1, the association roles `co-authors` and `reviewers` are generalized into `Concept_15`, interpreted as the role `participants` attached to the concept `Participant` in the final pattern (second clue).
3. At step#2, `Paper` and `Report` are sharing the role `owns : Concept_15` (`participants`) and are classified in concept `Concept_18`, corresponding to the discovery of class `Submission` in the final pattern (third clue).
4. At step#3, the discovery of `Concept_21` allows to complete `Concept_0` with the role `owns : Concept_21`, i.e. objects related with `Submission` (fourth clue). This last role is inherited by `Participant` and can be attached to this class by the designer.

The final global class model as it can be designed by the knowledge engineer is given on Figure 3. This class model can be then materialized as an effective knowledge pattern, through an implementation say in OWL to be considered in ontology engineering. This example is simple and low-sized w.r.t. the number of involved concepts and relations, but sufficiently generic too to be reused in a number of practical situations, after adaptation (e.g. renaming): e.g. supply and demand, selection committee (see for example an interesting and related discussion in [4]).

However, as also shown in this paper, the design work is not simple even if guided by FCA and RCA, and asks for an active participation of the knowledge engineer (the designer). For example, the discovery and the naming of extracted roles and concepts as discussed above, is neither easy nor immediate. In this way, we still have to make a series of experiments on existing but also on new knowledge patterns for having a better understanding of the design process. For example, we should improve the scaling of the process: what are the patterns that can be easily built and those that are not, depending on the complexity of the initial data, i.e. sets of objects, attributes, and relations, and UML structures. At present, we can only consider small and compact object and relation sets, easy to read and to understand, and not all knowledge patterns have the same dimensions.

## 5 Conclusion

In this paper, we have proposed a semi-automated process based on FCA and RCA for helping a knowledge engineer designing a knowledge pattern. This is an original contribution to what could be called "ontology engineering assisted by computer". Interestingly, this approach is borrowed from software engineering and brings new and powerful ideas for helping ontology engineering. In addition, the approach is supported by working tools such as the Galicia platform[1] and the ERCA (Eclispe Relational Concept Analysis) platform[2].

Apart the consolidation of the preceding work and the need for experiments at a larger scale, there are many research perspectives. For example, there should

[1] `http://sourceforge.net/projects/galicia/`
[2] `http://code.google.com/p/erca/`

be a deep study on the way how meta-modeling in software engineering could be adapted to knowledge engineering with the help of a platform such as Galicia. Meta-modeling aspects of software engineering have to be extended for taking into account the specificity of knowledge engineering requirements. Next, the design process should be abstracted and maximally automated for an optimal use in ontology engineering.

## References


1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. F. Baader, B. Ganter, U. Sattler, and B. Sertkaya. Completing description logic knowledge bases using formal concept analysis. In *IJCAI 2007*, pages 230–235, 2007.
3. P. Clark, J. Thompson, and B.W. Porter. Knowledge patterns. In S. Staab and R. Studer, editors, *Handbook on Ontologies (First Edition)*, pages 191–208. Springer, 2004.
4. M. Ducassé and S. Ferré. Fair(er) and (almost) serene committee meetings with logical and formal concept analysis. In P.W. Eklund and O. Haemmerlé, editors, *Proceedings of ICC 2008*, Lecture Notes in Computer Science 5113, pages 217–230. Springer, 2008.
5. M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, Reading (MA), 2002.
6. A. Gangemi and V. Presutti. Ontology Design Patterns. In S. Staab and R. Studer, editors, *Handbook on Ontologies (Second Edition)*, pages 221–243. Springer, Berlin, 2009.
7. B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, Berlin, 1999.
8. R. Godin and P. Valtchev. Formal concept analysis-based normal forms for class hierarchy design in oo software development. In B. Ganter, G. Stumme, and R. Wille, editors, *Formal Concept Analysis: Foundations and Applications*, LNAI 3626, pages 304–323. Springer Verlag, 2005.
9. A. Gómez-Pérez, M. Fernàndez-López, and O. Corcho. *Ontological Engineering*. Springer-Verlag, Berlin, 2004.
10. G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, Reading (MA), 2004.
11. P. Kruchten. *The Rational Unified Process: An Introduction (2nd Edition)*. Addison-Wesley, 2000.
12. A. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, Boston, 2003.
13. M. Rouane-Hacene, M. Huchard, A. Napoli, and P. Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In *ICFCA 2007*, LNAI 4390, pages 51–65, 2007.
14. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2004.
15. S. Staab and R. Studer, editors. *Handbook on Ontologies (Second Edition)*. Springer, Berlin, 2009.
16. R. Wille. Methods of conceptual knowledge processing. In R. Missaoui and J. Schmid, editors, *International Conference on Formal Concept Analysis, ICFCA 2006, Dresden, Germany*, Lecture Notes in Artificial Intelligence 3874, pages 1–29. Springer, 2006.