Une étude sur la mise en forme de patrons de conception pour les ontologies avec l'analyse formelle de concepts

Béatrice Fuchs[1], Marianne Huchard[2] et Amedeo Napoli[3]

[1] *LIRIS, Universités de Lyon 1, Nautibus, 8 bd Niels Bohr, F-69100 Villeurbanne (bfuchs@liris.univ-lyon1.fr)*
[2*] *LIRMM, UMR 5506, 161 rue Ada, 34392 Montpellier Cedex 5 (huchard@lirmm.fr)*
[3] *LORIA, BP 239 F-54506 Vandœuvre-lès-Nancy (Amedeo.Napoli@loria.fr)*

RÉSUMÉ. *Dans cet article, nous proposons un première réflexion sur les liens existant entre les patrons de conception en génie logiciel et patrons de conception pour ontologies. À l'image de la conception de code, la conception d'une ontologie est une tâche complexe, et les patrons de conception pour ontologies guident la conception d'une ontologie en jouant le rôle de construction de référence. Il est donc important de comprendre et instrumenter les façons de détecter ce qu'est un patron d'ontologie (pour ensuite l'adapter et le réutiliser). Cet article montre comment des techniques d'extraction de connaissances comme l'analyse formelle de concepts (AFC) et l'analyse relationnelle de concepts (ARC) sont utilisées pour concevoir un patron d'ontologie. Un exemple concret et générique est complètement traité et montre l'intérêt de l'approche.*

ABSTRACT. *This paper proposes a first study on the links existing between design patterns in software engineering and the so-called ontology design patterns (ODPs). ODPs are used for guiding the design of an ontology in the field of Semantic Web, which remains a primary but hard and complex task. Thus, it is important to be able to detect and to understand what can be a good and reusable ODP. In this paper, we show how knowledge discovery methods such as Formal Concept Analysis (FCA) and Relational Concept Analysis (RCA) can be used for extracting and completing an ODP from a representation problem statement.*

MOTS-CLÉS : *patrons d'ontologies, extraction de connaissances, représentation de connaissances, web sémantique.*

KEYWORDS: *ontology design pattern, knowledge discovery, knowledge representation, semantic web.*

2e soumission à *LMO*, le 31 décembre 2009.

1. Introduction

Qu'est-ce qu'un *patron de conception* (*design pattern*) en génie logiciel et à quoi cela sert-il ? Un patron de conception est une "solution à un problème de conception dans un contexte" [JéZ 06], où la notion de contexte se rapporte à un ensemble de situations récurrentes dans lesquelles le patron peut s'appliquer et où la solution fait référence à une règle de conception canonique qui peut être appliquée pour résoudre le problème. Parmi les nombreuses caractéristiques des patrons introduites dans [JéZ 06], les patrons de conception se distinguent par leur aptitude à guider et documenter la conception de logiciels à objets en fournissant plusieurs éléments :

– un vocabulaire partagé pour communiquer à propos de conception logicielle (une normalisation des éléments du domaine),

– une description abstraite de constructions complexes,

– des briques de base ou des cas de conception à partir desquels des constructions logicielles plus complexes peuvent être créées, avec la constitution possible de catalogues de patrons de conception.

Tous ces points, s'ils s'avèrent très utiles en génie logiciel, ne le sont pas moins en représentation des connaissances tout en y apparaissant sous des formes différentes : la notion d'ontologie et de vocabulaire partagé, les différents niveaux d'abstraction dans une ontologie, avec des concepts et expressions conceptuelles de bas niveau (concepts atomiques et primitifs) à partir desquels se construisent les définitions et autres constructions conceptuelles plus complexes (concepts définis). L'idée de patron de conception se transpose donc *mutatis mutandis* à la conception d'ontologie : une ontologie est l'équivalent d'un programme et doit être codée de la même façon ou presque. Ainsi, les patrons de conception doivent jouer un rôle dans ce type de conception et servir de guide et de support pour le développement d'une ontologie.

Du point de vue de la représentation des connaissances, une ontologie matérialise un "modèle" d'un domaine particulier du monde réel. Pratiquement, un tel modèle se représente comme un ensemble de définitions de concepts munis d'attributs et de relations entre ces concepts. L'implantation effective d'une ontologie nécessite un langage de représentation des ontologies — de la famille des langages de représentation des connaissances — et des modules associés de raisonnement [BAA 03, ANT 04]. S'il est important de disposer d'ontologies dans le cadre du Web sémantique par exemple [FEN 03], il reste néanmoins difficile de les construire [STA 09].

L'idée suivie dans cet article est de montrer comment se servir du modèle des patrons en génie logiciel [GAM 94] pour analyser et extraire des patrons équivalents pour la conception d'ontologies [GAN 09]. Les questions auxquelles nous essayons de répondre sont : qu'est-ce qu'un patron de conception d'ontologie, comment l'obtenir, l'utiliser et le réutiliser ? Puis, comment construire une ontologie en étant guidé par les connaissances du domaine mais aussi par une méthode de conception, comment mettre en place le schéma d'une ontologie — au sens d'un ensemble de concepts munis d'attributs et de relations et organisés en hiérarchie — de façon semi-automatique

en s'appuyant sur une méthode d'apprentissage. Pour cela, l'analyse formelle de concepts (AFC) et l'analyse relationnelle de concepts (ARC) permettent de partir d'un ensemble d'objets décrits par leurs attributs et de relations entre objets pour construire un treillis de concepts formels dont la description se compose d'attributs binaires et relationnels.

En outre, en représentation des connaissances et raisonnement — raisonnement à partir de cas en particulier — on se sert volontiers d'un processus d'adaptation pour justement adapter un élément de connaissance en cas de changement de contexte : ce type de technique s'avère bien adapté à la recherche de patrons sous contraintes et à leur adaptation en fonction du contexte d'utilisation. C'est là une autre idée de l'article d'esquisser comment des techniques issues du monde de la représentation des connaissances et de l'apprentissage peuvent être réutilisées avec profit dans le contexte du génie logiciel [FAL 07, ROU 07a] (qui est d'ailleurs une tendance actuelle forte [AKE 06, STA 09]).

L'article est organisé comme suit. Un exemple de patron d'ontologie est donné dans le paragraphe suivant qui va nous servir à illustrer les propos. Ensuite, des éléments sur les ontologies, les patrons d'ontologies et la conception d'ontologies sont précisés. Puis, l'exemple introduit au paragraphe 2 est étudié en détail et il est montré comment enrichir et réifier le schéma UML de l'exemple courant en utilisant l'analyse formelle et l'analyse relationnelle de concepts. Une brève synthèse conclut l'article.

2. Un exemple introductif

Dans cet article, nous partons d'un énoncé de problème de représentation général et simple. Ensuite, nous montrons comment obtenir par analyse formelle un patron, qui, après avoir été codé avec un langage de représentation des connaissances peut être considéré comme un patron d'ontologie. L'énoncé choisi ici est celui de "la soumission et de l'évaluation d'un article à une conférence". Un participant est auteur (`Author`) ou évaluateur (`Reviewer`). Selon son rôle, il écrit (`write`) un papier (`Paper`) ou rédige (`compose`) un rapport d'évaluation (`Report`). À son tour, un papier a un ou plusieurs co-auteurs (`co-authors`) tandis qu'un rapport est fourni par des évaluateurs (`reviewers`).

Cet énoncé s'appuie sur quatre concepts (`Participant`, `Author`, `Reviewer` et `Paper`) et deux relations, d'une part `write + co-authors` entre auteurs et articles, et d'autre part `compose + reviewers` entre évaluateurs et rapport d'évaluation. À partir de là, et sous contrôle de l'analyse formelle et de l'analyse relationnelle de concepts, nous allons construire par classification le schéma final donné à la figure 1. En particulier, les deux abstractions `Participant` et `Submission` et les relations qui existent entre elles sont découvertes et explicitées par l'analyse formelle et l'analyse relationnelle de concepts. L'AFC et l'ARC permettent d'extraire des concepts qui sont implicites dans les données, tout comme un raisonnement cherche à rendre explicite l'implicite !

Dans ce qui suit, nous présentons la problématique et les enjeux des ontologies dans le cadre du Web sémantique ainsi que la notion de patron d'ontologie dans sa généralité.

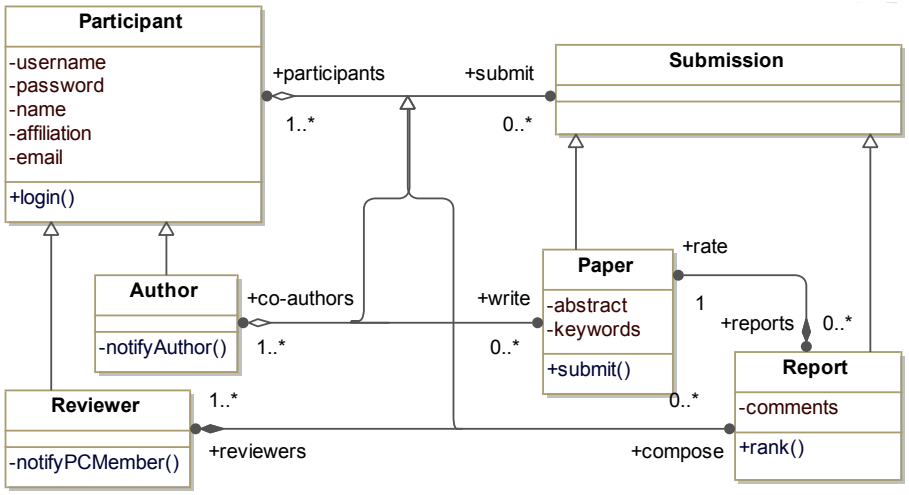

Figure 1. *Un schéma UML du patron "Soumission et évaluation d'un article à une conférence" obtenu après analyse de l'énoncé du problème original*

3. Quelques mots sur les ontologies et les patrons d'ontologies

Le Web sémantique peut se concevoir comme un Web où les "agents logiciels" communiquent avec d'autres agents logiciels pour rendre des services et résoudre des problèmes pour les "agents humains" [FEN 03]. Ainsi, le Web sémantique se veut un espace de communication et d'interactions où les agents logiciels tirent parti des ressources disponibles pour traiter les problèmes. Parmi les ressources disponibles, il faut distinguer les documents de toutes sortes et les "ontologies". Les ontologies établissent une terminologie commune entre les agents logiciels et humains en leur permettant de partager un même sens sur les concepts et relations manipulés. C'est pourquoi les ontologies prennent une grande place dans de nombreux domaines liés plus ou moins directement au Web sémantique, comme la gestion des connaissances, l'intégration de données, la recherche d'information, les systèmes collaboratifs et le commerce électronique.

3.1. *Sur les ontologies*

Pratiquement une ontologie se représente comme un ensemble de définitions de concepts munis d'attributs et de relations entre concepts. Une ontologie fournit un

modèle explicite, non ambigu et opérationnel d'un certain domaine, codé dans un formalisme de représentation des connaissances comme une logique de descriptions ou OWL [BAA 03, ANT 04]. Le *schéma d'une ontologie* quant à lui est constitué de l'ensemble des concepts et des relations qui existent entre concepts, à la façon d'un réseau sémantique. Trois grandes stratégies de conception sont habituellement considérées :

– la conception manuelle,

– la réutilisation d'ontologies existantes,

– la conception semi-automatique d'ontologies à partir de ressources, par exemple à l'aide de méthodes de fouille de données ou de fouille de textes [BEN 08a, BEN 08b].

Quelques développements récents en ingénierie des ontologies mettent en jeu des processus de découverte de connaissances qui peuvent avoir un intérêt en matière de construction d'ontologies, en particulier à partir de documents textuels. Ces processus mettent en jeu des méthodes de fouille de données et de textes comme l'analyse formelle de concepts [CIM 05]. Par exemple, dans [BEN 08a, BEN 08b], l'analyse formelle de concepts sert à extraire et formaliser des concepts à partir de tables de données binaires `objets` $\times$ `attributs` tandis que l'analyse relationnelle de concepts permet d'extraire des concepts munis d'attributs relationnels — codant des relations entre objets — à partir de données relationnelles cette fois (dans des tables `objets` $\times$ `objets`). Les treillis résultants peuvent servir de schémas d'ontologies où un concept est muni d'attributs "atomiques" et relationnels. Ces approches ne peuvent pas être entièrement automatiques pour plusieurs raisons. Un des problèmes est relatif à la couverture du corpus de textes en entrée : il est difficile de prétendre avoir toutes les connaissances explicitement disponibles pour interpréter effectivement l'ensemble des textes et souvent les connaissances sont disséminées temporellement et géographiquement. Un autre problème est qu'il est difficile d'extraire un savoir-faire et une bonne pratique à partir de documents textuels et que l'expertise est là directement nécessaire (ou encore que les bonnes pratiques sont rarement explicites mais plutôt "enfouies").

3.2. *Sur les patrons d'ontologies*

Quelle que soit la stratégie de conception d'ontologies suivie, il s'avère utile et nécessaire d'être guidé à l'image de ce qui se pratique en génie logiciel, où la production de code s'appuie sur l'utilisation et la réutilisation de "patrons de conception" [GAM 94]. Les principes appliqués au logiciel sont ici réutilisés et adaptés pour donner naissance à des *patrons de conception d'ontologies* ou ODP pour *Ontology Design Patterns* [BLO 08, GAN 09]. Comme en ingénierie du logiciel, un patron propose une solution générique et réutilisable pour un problème de conception, un modèle — implanté en OWL — pour résoudre un problème local de conception d'ontologie.

Plusieurs types d'ODP se distinguent. Les "ODP logiques" recouvrent des structures élémentaires comme la "partition" qui est à la base du formalisme FOAF (pour représenter des réseaux de personnes [BRI 05]). Les "ODP de réingénierie" cherchent à factoriser et contrôler l'architecture de la hiérarchie des concepts. Et en particulier les "ODP de contenu" cherchent à fournir des schémas de représentation pour des problèmes, certains génériques comme la composition et les objets composites, les relations n-aires, les séquences et les collections, et d'autres plus spécifiques et propres à un problème de conception particulier (comme celui qui sert d'exemple dans cet article).

Un patron de conception d'ontologie peut s'appréhender, à l'image de l'acquisition de connaissances à partir d'expertise, comme un bloc de conception générique, l'expression d'un savoir-faire ou un guide de bonne pratique. Un tel patron intervient dans la mise en forme, la concision et la qualité d'une ontologie. Un parallèle peut être fait avec les "gabarits" de l'industrie manufacturière par exemple.

Dans le même ordre d'idée et dans une optique de fouille de données, un patron assimilé à un bloc de conception élémentaire qui matérialise une bonne pratique devrait pouvoir se comparer à un motif présentant une certaine régularité ou fréquence dans un ensemble de données. Quelles sont les données dont il est question ici et comment caractériser, extraire et ensuite interpréter un tel motif ? Les données recouvrent des collections d'ontologies (supposées "bien faites") et les motifs sont des constructions récurrentes dans ces données qui vont avoir un support et une fréquence. Un motif devient un patron de conception s'il fait preuve d'intérêt et s'il est effectivement réutilisé.

Un parallèle peut être fait encore avec les travaux qui cherchent à déterminer et extraire des patrons de conception en génie logiciel à partir de modèle de code ou de conception (voir par exemple [TON 99, ARÉ 04, GUE 08]).

3.3. *Sur la conception d'ontologies*

Dans la pratique, le concepteur d'une ontologie n'a finalement que peu d'assistance disponible : des ontologies qu'il a pu trouver mais qu'il peut quelquefois très difficilement réutiliser, des constructions d'un langage de représentation de connaissances pas toujours adaptées aux besoins courants, et, dans le meilleur des cas, une brassée de bonnes pratiques glanées ça et là : par exemple dans [ALL 08] ou sur un site de référence comme `ontologydesignpatterns.org`, qui fournit un catalogue de patrons disponibles.

Un scénario habituel consiste à rechercher, quand cela est possible, des ontologies en rapport avec le problème, les étudier et en sélectionner des éléments réutilisables pour les adapter. Parmi ces ontologies, certaines ontologies sont supposées être réutilisables par spécialisation comme les ontologies de haut niveau que sont DOLCE ou SUMO [OBE 07]. Là encore la tâche de compréhension est difficile et les nombreuses propriétés des concepts — quelquefois d'ordre métaphysique — ne sont pas toujours

adaptées aux besoins courants tout comme la couverture de telles ontologies est bien trop grande dans la plupart des cas pratiques.

À l'opposé, il existe de petites ontologies, bien faites, concises, simples à comprendre et à utiliser (voir `ProtegeOntologiesLibrary` par exemple), des "ontologies simplifiées" (voir DOLCE+DnS Ultralite sur `ontologydesignpatterns.org`), mais aussi la possibilité de réutiliser des ontologies par morceaux [D'A 08]. Dans le même ordre d'idées, il faut encore mentionner des formalismes comme (i) SKOS [MIL 05], pour *Simple Knowledge Organization System*, qui permet de représenter des vocabulaires, des thésaurus, des partitions et des "folksonomies", et (ii) FOAF [BRI 05] qui permet de représenter des réseaux de personnes en relation.

Ainsi, il est souhaitable que la conception d'ontologies s'appuie sur la réutilisation de "petites ontologies modulaires" qui vont jouer le rôle de patrons de conception d'ontologies. Ces patrons de conception, qui peuvent être vus comme des composants d'un genre particulier, doivent offrir des fonctionnalités simples et d'intérêt général et doivent être faciles d'accès et de manipulation, comme cela est discuté ci-dessous.

4. L'extraction d'un patron d'ontologie guidée par AFC/ARC

4.1. *Quelques éléments sur l'AFC et l'ARC*

L'AFC consiste à construire un treillis de concepts à partir d'un tableau binaire `Objets` $\times$ `Attributs` [GAN 99, BAR 70]. Formellement, un *contexte* $\mathcal{K}$ est la donnée d'un triplet (O, A, I) où O est un ensemble d'objets, A est un ensemble d'attributs, et $I \subseteq O \times A$ une relation entre O et A. Un concept regroupe un ensemble (maximal) d'objets partageant un ensemble (maximal) d'attributs, ce qui se formalise par la définition de la connexion de Galois suivante (dénotée en AFC par l'apostrophe $'$) :

$$\begin{array}{l} ' : \mathcal{P}(O) \rightarrow \mathcal{P}(A);\ \ X' = \{a \in A \mid \forall o \in X, oIa\}. \\ ' : \mathcal{P}(A) \rightarrow \mathcal{P}(O);\ \ Y' = \{o \in O \mid \forall a \in Y, oIa\}. \end{array}$$

Les ensembles maximaux d'objets — appelés *extensions* (*extents*) — sont en correspondance bijective avec les ensembles maximaux d'attributs correspondants — attributs que tous les objets possèdent, appelés *intensions* (*intents*). Les couples $(X, Y) \in \mathcal{P}(O) \times \mathcal{P}(A)$ d'ensembles en correspondance vérifiant $X = Y'$ et $Y = X'$ sont appelés des *concepts formels* et forment un *treillis* complet par rapport à l'ordre de l'inclusion des extensions : $(X_1, Y_1) \sqsubseteq (X_2, Y_2)$ dès que $X_1 \subseteq X_2$. Ce treillis est appelé treillis des concepts et sa construction est une tâche primordiale en AFC.

L'AFC est bien adaptée au traitement de données se présentant sous forme de tableaux binaires ou pouvant s'y rapporter. Cependant, la machinerie de l'AFC est limitée lorsqu'il s'agit de traiter des tableaux de données non binaires ou des données où figurent des relations binaires entre les objets. Ainsi, l'échelonnage conceptuel permet de traduire des données multi-valuées en données binaires par l'intermédiaire de

prédicats qui "binarisent" les attributs : un domaine de valeurs $\{1, 2, 3\}$ pour l'attribut a sera transformé en trois attributs binaires $a = 1$, $a = 2$ et $a = 3$. C'est pour extraire et représenter les relations inter-objets qu'intervient l'analyse relationnelle de concepts [ROU 07b]. Le modèle de données de l'ARC est la *famille de contextes relationnels* (FCR) notée $\mathcal{R} = (\mathbf{K}, \mathbf{R})$, où $\mathbf{K}$ est un ensemble de contextes $\mathcal{K}_i = (O_i, A_i, I_i)$, $\mathbf{R}$ un ensemble de relations $r_k \subseteq O_i \times O_j$, où O_i et O_j sont les ensembles d'objets des contextes formels $\mathcal{K}_i$ et $\mathcal{K}_j$.

Une relation r, par exemple "les chercheurs qui sont auteurs de papiers", met en correspondance des objets d'un contexte $\mathcal{K}_i$, le *domaine* de r, avec des objets d'un contexte $\mathcal{K}_j$, le *co-domaine* de r. L'idée ici est de reprendre le processus de l'AFC et de l'adapter au traitement de données relationnelles par l'intermédiaire d'un échelonnage particulier, où les prédicats sont associés à des concepts dans un treillis et traduisent un ordre partiel. La prise en compte d'une relation r entre des objets de O_i dans $\mathcal{K}_i$ et des objets de O_j dans $\mathcal{K}_j$ consiste à considérer les objets o_j tels que $r(o_i) = o_j$ et à chercher à quelles extensions de concepts appartient o_j : selon que $o_j = r(o_i)$ est ou non dans l'extension du concept C_j (dans le treillis associé à $\mathcal{K}_j$ préalablement construit), l'attribut dénoté $r : C_j$ est assigné à l'objet o_i.

4.2. *Le traitement d'un exemple*

L'obtention d'un patron de conception ayant de bonnes qualités de réutilisation s'appuie sur l'expérience renouvelée, dans différents contextes, d'une organisation des concepts répondant de manière satisfaisante à une demande récurrente en termes de modélisation. Dans notre exemple (figure 1), l'abstraction en un concept `Participant` des notions d'auteurs et d'évaluateurs peut avoir plusieurs effets bénéfiques. La mise en œuvre au sein d'une base de données peut se faire par une seule table plutôt que par deux : une personne qui serait à la fois auteur et évaluateur n'apparaîtra pas dans deux tables, avec tous les problèmes bien connus que cela entraîne sur la cohérence des informations entre les deux représentations de la personne. Une personne suivra une même procédure de login qu'elle veuille déposer un article ou une fiche de lecture. En revanche, dès qu'elle aura choisi le rôle avec lequel elle veut agir, elle sera notifiée en tant qu'auteur ou en tant qu'évaluateur.

De même, l'abstraction `Submission` permettra quant à elle de décrire des informations complémentaires telles que les dates de soumission ou de modification (de l'article ou de la fiche de relecture). L'association entre `Participant` et `Submission` notée `participants-submit` sera le support de procédures générales, comme celle de rappeler à un participant toutes ses soumissions.

Dans la réalité, la conception du patron suit plusieurs étapes. Dans le cas de la sélection des papiers à une conférence, nous approchons le problème en partant des *deux fonctionnalités à rendre*, c'est-à-dire la soumission des papiers par les auteurs et le dépôt des fiches de lecture par les évaluateurs, ce qui nous fait aboutir aux deux diagrammes présentés respectivement figure 2 et figure 3. Le passage de ces deux

visions séparées, reposant sur les fonctionnalités à rendre, au patron de conception, nécessite la découverte de concepts implicites et la classification de tous les concepts mis en jeu. C'est pour effectuer cette extraction et cette classification de concepts que sont utilisées d'abord l'analyse formelle de concepts puis l'analyse relationnelle de concepts.

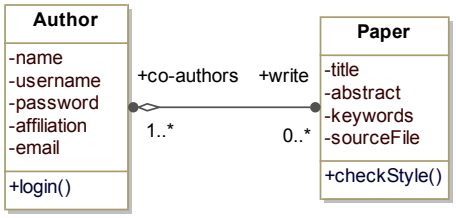

Figure 2. *Partie du patron concernant le processus de soumission d'un article*

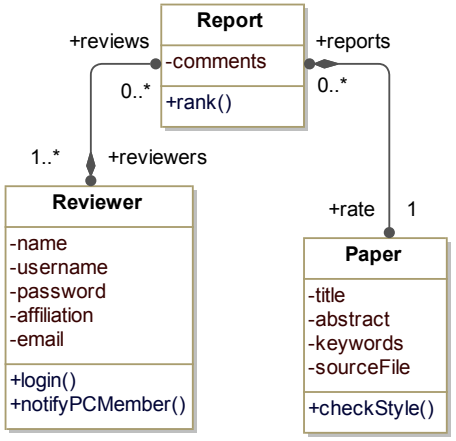

Figure 3. *Partie du patron concernant le processus de relecture d'un article soumis*

Ainsi, le tableau 1 présente le contexte formel des classes associées aux modèles "soumission" et "évaluation" et décrits par des propriétés (attributs et méthodes). De ce tableau il est possible d'extraire un concept d'extension `{Author, Reviewer}` et d'intension `{name, username, password, affiliation, email, login()}`, qui représente la notion plus générale de `Participant`. À la figure 4, le sous-treillis restreint aux concepts qui portent la mention `(0)` derrière leur identificateur est le treillis issu du tableau 1. La notation est "réduite" au sens où les extensions complètes se retrouvent en héritant les extensions "vers le haut", tandis que les intensions complètes se retrouvent en héritant les intensions "vers le bas".

Le travail d'abstraction effectué sur les classes s'étend aux rôles grâce à l'analyse relationnelle de concepts. La mise en œuvre de l'analyse relationnelle de con-

Classes	name	username	password	affiliation	email	title	abstract	keywords	sourcefile	comments	login()	checkstyle()	notifyPCmember()	rank()	notifyAuthors
Author	×	×	×	×	×						×				×
Paper						×	×	×	×			×			
Reviewer	×	×	×	×	×						×		×		
Report										×				×	

Tableau 1. *Contexte des classes décrites par leurs propriétés*

cepts (ARC) s'appuie sur une famille de relations binaires, dont certaines décrivent des objets par des attributs et d'autres décrivent des relations entre objets. Pour cela, plusieurs contextes sont ajoutés au contexte des classes : un contexte décrivant les rôles (tableau 2) qui se limite ici à leur attribuer des noms ; un contexte *relationnel* indiquant quels rôles possèdent les classes (tableau 3) ; un autre contexte *relationnel* associant les rôles et leurs types (tableau 4). L'ARC fait émerger les abstractions par raffinements successifs des treillis de classes et de rôles. Les différentes étapes de la construction des deux treillis, treillis des classes et treillis des rôles, sont détaillées ci-dessous. Ce sont en réalité ces deux treillis qui donnent sa réalité au schéma UML donné à la figure 1.

La construction du patron : étape 0.

Dans la situation initiale (étape 0), deux treillis sont construits à partir des contextes des classes et des rôles (tableaux 1 et 2) dont les concepts sont ceux présents dans les figures 4 et 5 et dont le nom est suivi de la mention `(0)`. Dans le treillis des classes sont créés : `Concept_0` (`Top`), `Concept_6` (`Author`), `Concept_5` (`Reviewer`), `Concept_4` (`Report`), `Concept_3` (`Paper`), et `Concept_1`, qui correspond à une "superclasse" de `Author` et `Reviewer` et qui factorise leurs propriétés communes et qui représente l'abstraction que nous nommerons `Participant`.

À cette étape les contextes relationnels ne sont pas considérés et ce qui s'y rapporte dans les intensions (propriétés `type` ou `owns`) n'existe pas encore.

La construction du patron : étape 1.

Ici, les contextes relationnels sont adjoints aux contextes initiaux auxquels ils se rapportent : `owns` est ajouté au contexte des classes, tandis que `type` est ajouté au contexte des rôles. Ainsi, plusieurs treillis sont construits sur la base de plusieurs contextes associés. Le contexte des classes va être complété par le contexte relationnel `owns` modifié pour prendre en compte les concepts de rôles créés à l'étape précédente (tableau 3). Dans ce nouveau contexte, une classe possède (`owns`) un concept

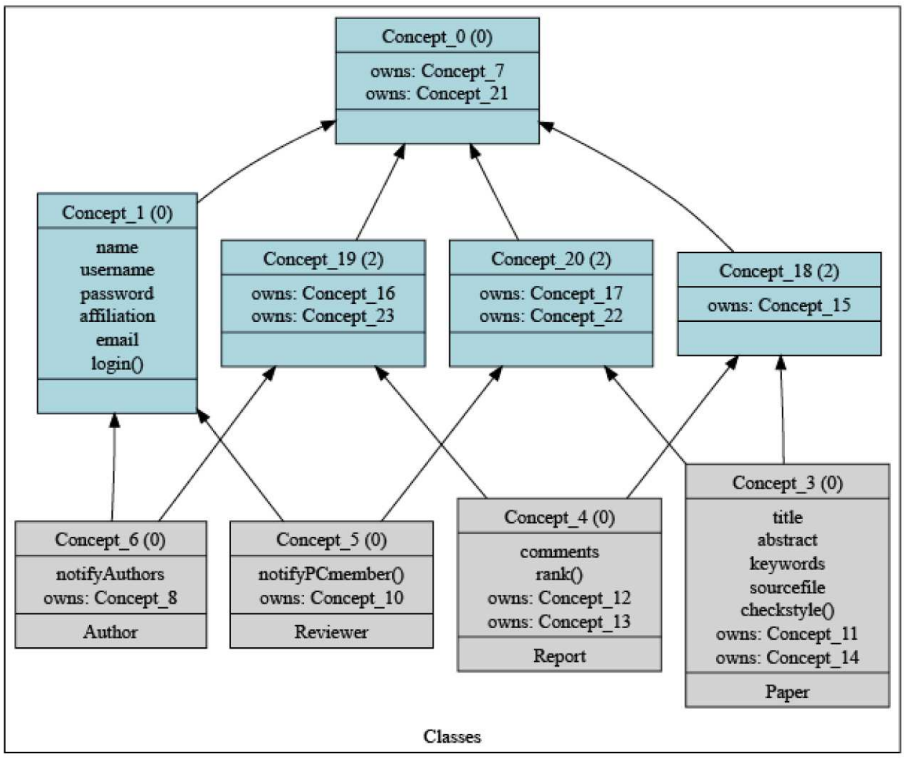

Figure 4. *Le treillis des classes*

si elle possède (`owns`) un rôle de l'extension de ce concept. Le contexte des types de rôles est construit de manière identique (tableau 4) et ajouté au contexte des rôles. Lorsque des treillis sont construits avec ces deux paires de contextes associés (classes + `owns`) et (rôles + `type`), on voit apparaître des compléments d'informations sur les concepts existants ainsi que de nouvelles abstractions. Par exemple, un concept créé à l'étape 0 comme `Concept_6` va disposer d'une propriété dans son intension nommée "`owns` : `Concept_8`" qui indique qu'`Author` possède le rôle `write`. Un nouveau concept apparaît dans le treillis des rôles, le concept `Concept_15`, qui correspond à un nouveau rôle, super-rôle de `co-authors` et `reviewers`, de type `Concept_1`, correspondant à la notion de `Participant`.

Pour terminer, sont créés à l'étape 1 dans le treillis des rôles : `Concept_17` qui généralise les rôles en relation avec `Report`, `Concept_16` qui généralise les rôles en relation avec `Paper`, `Concept_15` qui généralise les rôles en relation avec `Concept_1` (`Author` ou `Reviewer`, c'est-à-dire `Participant`) et permet de découvrir le rôle `participants` du patron final.

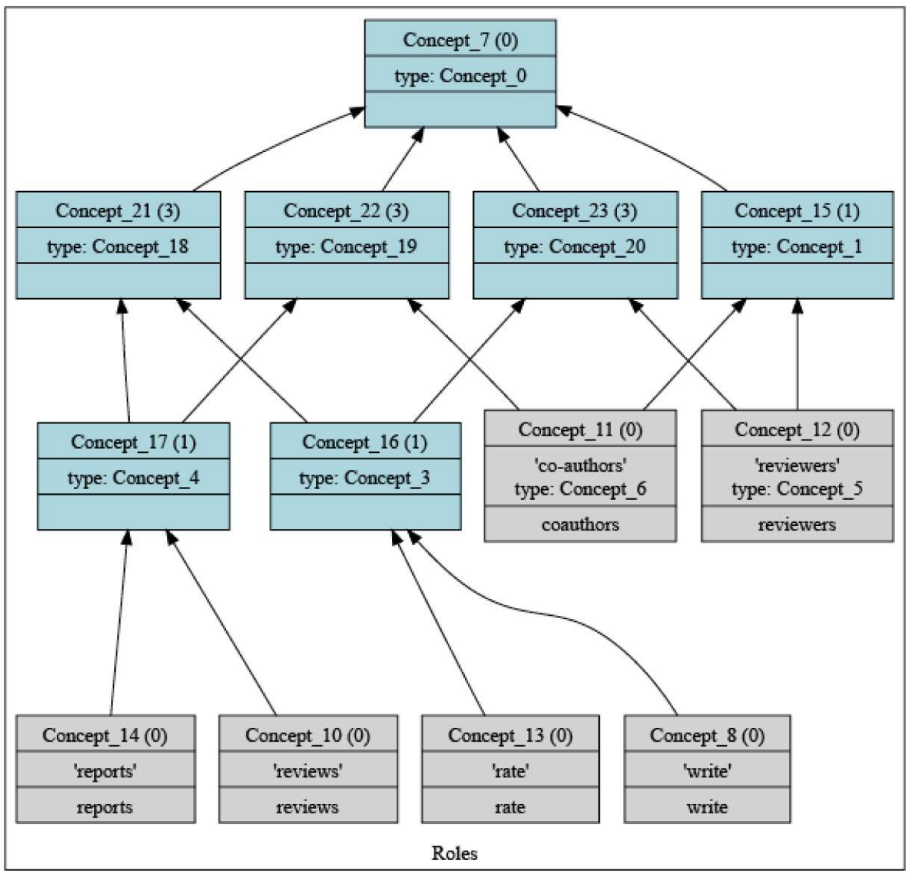

Figure 5. *Le treillis des rôles*

La construction du patron : étape 2.

Les concepts suivants sont créés dans le treillis des classes : `Concept_19` qui dénote les objets en relation avec `Paper` par le rôle "`owns : Concept_16`", `Concept_20` qui dénote les objets en relation avec `Report` par le rôle "`owns : Concept_17`", `Concept_18` qui dénote les objets en relation avec des `Participant` par le rôle "`owns : Concept_15`". Ce dernier concept est particulièrement intéressant et donne naissance à une abstraction que nous appelons `Submission`.

Le treillis des rôles n'est pas modifié de manière intéressante à cette étape.

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

Tableau 2. *Le contexte des rôles décrits par leurs noms*

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

Tableau 3. *Le contexte* `owns` *des classes décrites par les rôles qu'elles possèdent*

La construction du patron : étape 3.

Les généralisations dans le treillis des rôles fournissent au cours de l'étape 3 : `Concept_21` qui est un rôle détenu par des objets en relation avec `Submission` (découverte du rôle `submit` du patron final), `Concept_22` qui est un rôle détenu par des objets en relation avec `Paper`, et enfin `Concept_23` qui est un rôle détenu par des objets en relation avec `Reports`.

Et dans le treillis des classes, ce sont principalement les intensions qui sont complétées : `Concept_19` avec le rôle "`owns : Concept_23`" qui signifie qu'`Author` et `Report` sont bien en relation avec des objets en relation avec des objets en relation avec `Report`, `Concept_20` avec le rôle "`owns : Concept_22`" qui signifie que `Reviewer` et `Paper` sont bien en relation avec des objets en relation avec des objets en relation avec `Paper` ...

4.3. *Synthèse et discussion*

Pour faire la bilan de cette construction en quatre étapes principales, l'AFC et l'ARC nous ont permis d'extraire pas à pas le patron de conception qui nous intéresse :

– À l'étape 0, `Concept_1` est utilisé pour créer l'abstraction `Participant`.

– À l'étape 1, l'existence de la classe `Participant`, partagée par les rôles `co-authors` et `reviewers` (type commun), provoque un regroupement dans

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

Tableau 4. *Le contexte* `type` *des rôles décrits par les classes qui les typent*

le `Concept_15` des rôles `reviewers co-authors`, interprété comme le rôle `participants` du patron final.

– À l'étape 2, `Paper` et `Report` se retrouvent tous les deux à partager le rôle `Concept_15` (`participants`) et sont regroupés dans le concept `Concept_18`, ce qui correspond à la découverte de la classe `Submission` du patron final.

– À l'étape 3, la découverte de `Concept_21` permet de compléter `Concept_0` par la connaissance du partage du rôle nommé `submit` vers les soumissions (`Submission`). Ce rôle est hérité par `Participant` et peut être porté par cette classe sur décision du concepteur.

La construction de ce patron d'ontologie est matérialisée par le diagramme de la figure 1 et se termine effectivement lorsque les concepts mis en jeu sont représentés en OWL. Ici l'exemple est simple au niveau du nombre de concepts impliqués et extraits, mais suffisamment générique pour mettre en avant le côté réutilisable de ce patron : soumettre un papier, faire une demande, faire une commande, etc.

Cependant, il reste encore beaucoup de choses à étudier en détail notamment la modalité de la construction semi-automatique d'un patron, qui peut partir d'un prototype unique comme ici ou qui peut s'appliquer plus systématiquement à une collection d'ontologies, en mettant éventuellement en jeu un processus de fouille d'arbres ou de graphes, mais aussi d'alignement d'ontologies par morceaux (avec réutilisation de certaines parties seulement des ontologies considérées). Ces aspects du processus d'extraction de patron d'ontologie et les données sur lesquelles il peut être appliqué nécessite de s'intéresser au passage à l'échelle du processus.

Dans tous les cas, la conception de patrons d'ontologies telle qu'elle est envisagée ici s'intéresse à de "petites structures ontologiques", faciles à modifier et appréhender par le concepteur. Un dernier mot concerne l'intelligibilité des structures ontologiques extraites : s'il est facile de nommer les concepts et rôles issus de l'exemple, cela n'en sera pas toujours de même dans la réalité vraisemblablement.

5. Conclusion

Cet article est une première ébauche sur la conception de patrons de conception d'ontologies à partir d'un schéma UML, avec une étude des liens qui existent avec les patrons de conception en génie logiciel. Il existe un bon nombre de liens entre les deux disciplines dont il faut pouvoir profiter pour mettre en œuvre et guider la conception d'une ontologie par l'utilisation et la réutilisation de patrons d'ontologies. Cette réutilisation de patrons doit être étudiée de près tout comme la possibilité de découverte de patrons à partir d'ontologies existantes, par l'intermédiaire de processus de même type que la fouille de textes, d'arbres ou de graphes, appliqués cette fois à des collections choisies d'ontologies.

Remerciements

Les auteurs tiennent à remercier Mohamed Rouane-Hacène et Petko Valtchev de l'UQAM Montréal pour leur avoir fourni les bases de l'exemple présenté ici, ainsi que pour toutes les discussions scientifiques partagées sur le sujet.

6. Bibliographie

[AKE 06] AKERMAN A., TYREE J., « Using ontology to support development of software architectures », *IBM Systems Journal*, vol. 45, n° 4, 2006, p. 813–826.

[ALL 08] ALLEMANG D., HENDLER J., *Semantic Web for the Working Ontologist*, Morgan Kaufmann, 2008.

[ANT 04] ANTONIOU G., VAN HARMELEN F., *A Semantic Web Primer*, The MIT Press, Cambridge, Massachusetts, 2004.

[ARÉ 04] ARÉVALO G., BUCHLI F., NIERSTRASZ O., « Detecting Implicit Collaboration Patterns », *11th Working Conference on Reverse Engineering (WCRE 2004)*, IEEE Computer Society, 2004, p. 122–131.

[BAA 03] BAADER F., CALVANESE D., MCGUINNESS D., NARDI D., PATEL-SCHNEIDER P., Eds., *The Description Logic Handbook*, Cambridge University Press, Cambridge, UK, 2003.

[BAR 70] BARBUT M., MONJARDET B., *Ordre et classification – Algèbre et combinatoire (2 tomes)*, Hachette, Paris, 1970.

[BEN 08a] BENDAOUD R., NAPOLI A., TOUSSAINT Y., « Formal Concept Analysis : A unified framwork for building and refining ontologies », GANGEMI A., EUZENAT J., Eds., *Knowledge Engineering : Practice and Patterns – Proceedings of the 16th International Conference EKAW*, Lecture Notes in Computer Science 5268, 2008, p. 156–171.

[BEN 08b] BENDAOUD R., TOUSSAINT Y., NAPOLI A., « PACTOLE : A methodology and a system for semi-automatically enriching an ontology from a collection of texts », EKLUND P., HAEMMERLÉ O., Eds., *Proceedings of the 16th International Conference on Conceptual Structures, ICCS 2008, Toulouse, France*, Lecture Notes in Computer Science 5113, 2008, p. 203–216.

[BLO 08] BLOMQVIST E., « Pattern Ranking for Semi-automatic Ontology Construction », WAINWRIGHT R., HADDAD H., Eds., *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008)*, ACM, 2008, p. 2248–2255.

[BRI 05] BRICKLEY D., MILLER L., « FOAF vocabulary specification », Working draft, 2005, http ://www.foaf-project.org.

[CIM 05] CIMIANO P., HOTHO A., STAAB S., « Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis », *Journal of Artificial Intelligence Research*, vol. 24, 2005, p. 305–339.

[D'A 08] D'AQUIN M., MOTTA E., SABOU M., ANGELETOU S., GRIDINOC L., LOPEZ V., GUIDI D., « Toward a New Generation of Semantic Web Applications », *IEEE Intelligent Systems*, vol. 23, n° 3, 2008, p. 20–28.

[FAL 07] FALLERI J.-R., ARÉVALO G., HUCHARD M., NEBUT C., « Use of Model Driven Engineering in Building Generic FCA/RCA Tools », EKLUND P., DIATTA J., LIQUIÈRE M., Eds., *CLA*, CEUR Workshop Proceedings 331, 2007.

[FEN 03] FENSEL D., HENDLER J., LIEBERMAN H., WAHLSTER W., Eds., *Spinning the Semantic Web*, The MIT Press, Cambridge, Massachusetts, 2003.

[GAM 94] GAMMA E., HELM R., JOHNSON R., VLISSIDES J., *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading (MA), USA, 1994.

[GAN 99] GANTER B., WILLE R., *Formal Concept Analysis*, Springer, Berlin, 1999.

[GAN 09] GANGEMI A., PRESUTTI V., « Ontology Design Patterns », STAAB S., STUDER R., Eds., *Handbook on Ontologies (Second Edition)*, Springer, Berlin, 2009.

[GUE 08] GUEHENEUC Y., ANTONIOL G., « Approach for Design Pattern Identification », *IEEE Transactions on Software Engineering*, vol. 34, n° 5, 2008, p. 667–684.

[JéZ 06] JÉZÉQUEL J., « Patrons de conception », AKOKA J., COMYN-WATTIAU I., Eds., *Encyclopédie de l'informatique et des systèmes d'information*, p. 1126–1135, Vuibert, Paris, 2006.

[MIL 05] MILES A., BRICKLEY D., « SKOS Core Vocabulary Specification », Technical report, 2005, World Wide Web Consortium (W3C).

[OBE 07] OBERLE D., ANKOLEKAR A., HITZLER P., CIMIANO P., SINTEK M., KIESEL M., MOUGOUIE B., BAUMANN S., VEMBU S., ROMANELLI M., « DOLCE ergo SUMO : On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO) », *Journal of Web Semantics*, vol. 5, 2007, p. 156–174.

[ROU 07a] ROUANE-HACENE M., DAO M., HUCHARD M., VALTCHEV P., « Aspects de la réingénierie des modèles UML par analyse de données relationnelles », *Ingénierie des Systèmes d'Information*, vol. 12, n° 5, 2007, p. 39–68.

[ROU 07b] ROUANE-HACENE M., HUCHARD M., NAPOLI A., VALTCHEV P., « A proposal for combining Formal Concept Analysis and description Logics for mining relational data », KUZNETSOV S., SCHMIDT S., Eds., *Proceedings of the 5th International Conference on Formal Concept Analysis (ICFCA 2007), Clermont-Ferrand*, LNAI 4390, Springer, Berlin, 2007, p. 51–65.

[STA 09] STAAB S., STUDER R., Eds., *Handbook on Ontologies (Second Edition)*, Springer, Berlin, 2009.

[TON 99] TONELLA P., ANTONIOL G., « Object Oriented Design Pattern Inference », *Proceedings ICSM'99 Conference*, 1999, p. 230–238.