Experiments on pattern-based ontology design

Conference Paper · January 2009

DOI: 10.1145/1597735.1597743 · Source: DBLP

CITATIONS
48

READS
242

3 authors:

Eva Blomqvist
Linköping University
59 PUBLICATIONS **572** CITATIONS
SEE PROFILE

Aldo Gangemi
University of Bologna
240 PUBLICATIONS **7,230** CITATIONS
SEE PROFILE

Valentina Presutti
Italian National Research Council
126 PUBLICATIONS **1,487** CITATIONS
SEE PROFILE

Experiments on Pattern-based Ontology Design

Eva Blomqvist
STLab, ISTC-CNR
via Nomentana 56
Rome, Italy
eva.blomqvist@istc.cnr.it

Aldo Gangemi
STLab, ISTC-CNR
via Nomentana 56
Rome, Italy
aldo.gangemi@cnr.it

Valentina Presutti
STLab, ISTC-CNR
via Nomentana 56
Rome, Italy
valentina.presutti@cnr.it

ABSTRACT

This paper addresses the evaluation of pattern-based ontology design through experiments. An initial method for reuse of content ontology design patterns (Content ODPs) was used by the participants during the experiments. Hypotheses considered include the usefulness of Content ODPs for ontology developers, and we additionally study in what respects they are useful and what open issues remain. The main conclusions are that Content ODPs are perceived as useful by ontology developers. The ontology quality is improved when Content ODPs are reused, coverage of the task increases, usability is improved, and common modelling mistakes can be avoided.

Categories and Subject Descriptors

I.2.4 [**Artificial intelligence**]: Knowledge Representation Formalisms and Methods

1. INTRODUCTION

Under the assumption that there exist classes of problems in ontology design that can be solved by applying common solutions (as experienced in software engineering), ontology design patterns (ODPs) can support reusability on the design side. ODPs, as described in [10], can be of several types. Content ODPs are small (or cleverly modularized) ontologies with explicit documentation of design rationales, representing modeling good practices, as described further in section 3. In this paper we present evaluations concerning the use of Content ODPs, in order to show their benefits, and to determine what kind of tool and method support is needed for the reuse process. By performing a set of experiments, we identify a number of benefits, e.g. that ontology development is made easier and the ontologies produced are of better quality.

The paper is organized as follows: Section 2 introduces related work on experimentation, while in Section 3 we briefly introduce Content ODPs and the eXtreme Design (XD) methods. The main contribution of this work is presented in Section 4, which describes the experiments performed and the result analysis, and in Section 5 that draws conclusions and outlines future work.

2. RELATED WORK

To the best of our knowledge, there is currently no work presented on experimentally showing benefits of patterns in ontology engineering. We therefore draw inspiration from the software engineering field. Initially, when software patterns first emerged, their benefits were claimed through logical arguments and industry experiences, such as in [3]. However, it soon became evident that scientific proofs, e.g. controlled experiments, were needed. Also in the field of Knowledge Acquisition the need for more rigorous experiments has been noted, e.g. in [18]. In ontology engineering, patterns are considered an important design support, e.g. by the W3C Ontology Engineering and Patterns Task Force[1], but again rigorous experiments are barely needed.

In [14] the authors propose to evaluate reuse in both an engineering and a cognitive science sense. In the cognitive science sense the coverage and the understandability of the reuse model is evaluated. In a computer science sense, initial evaluations should be a proof of concept implementation, then a deeper evaluation should demonstrate utility, e.g. through showing a quality improvement of the artefacts produced, including process improvements as well.

Reports on pattern experimentation include experiments showing the benefits of software patterns for Human Computer Interaction. In [5] two sets of experiments were conducted, using groups of developers evaluating interface design as well as solving design problems. Data was collected from questionnaires and artefacts resulting from solving the tasks. The first set of experiments used 9 pairs of developers, and 7 pairs in the revised version. The groups where divided into those

[1] http://www.w3.org/2001/sw/BestPractices/OEP/

exposed to patterns and those who were not. A similar study was presented in [6] but in this case patterns were introduced at different stages of the experiment.
Experiments focused on the communication aspects of patterns have dealt with software patterns in teaching. The study in [12] focuses on comparing the use of software patterns and anti-patterns, i.e. 'bad examples'. Other experiments, such as [15], have studied the effects of pattern usage on software maintenance processes. Another class of experiments has focused on studying existing systems, trying to identify pattern presence in running systems and connecting this to characteristics of the system as in [1] and [13].
In this paper, we present experiments partly covering both cognitive and engineering aspects of Content ODPs. With respect to cognitive aspects, we focus on the understandability, in terms of how ODPs are perceived and used. With respect to engineering aspects, proof of concept implementations have already been presented elsewhere, whereby we focus on a deeper analysis of Content ODPs by analyzing the ontologies produced. In terms of setup, our experiments are similar to those of [5] and [6], since we are using several groups, and both questionnaires and analysis of artefacts. We are also studying patterns as a teaching aid, and we have conducted some of the experiments in teaching settings. Since ontologies that show explicitly pattern usage are still rare, we are not able to conduct studies similar to [15] and [1] at this stage. The setup of our experiments face similar problems as for general Knowledge Acquisition, see [18], most of which are addressed in section 4.1. In the following section we introduce Content ODPs, and subsequently in section 4 the experiments are described.

3. ONTOLOGY DESIGN PATTERNS

For the reader to understand the experiments performed, we present some background on the ODPs in focus, and the reuse method applied as a general guideline for the participants. In this paper we focus on Content ODPs, but there exist different types of ODPs: for details see [10].

3.1 Content ODPs

Content ODPs are small ontologies with explicit documentation of design rationales, which can be used as building blocks in ontology design, as shown in [8, 16]. Content ODPs are used as modeling components: ideally an ontology results from the composition of a set of Content ODPs, with appropriate dependencies between them, plus the necessary design expansion based on specific needs. A Content ODP is always associated with a General Use Case (GUC) [8], i.e. the general problem that the ODP is a solution for. GUCs are usually expressed using Competency Questions (CQs) [11]. As an example, in Figure 1, we show the UML diagram of the Content ODP called *information realization*. It represents the relations between information

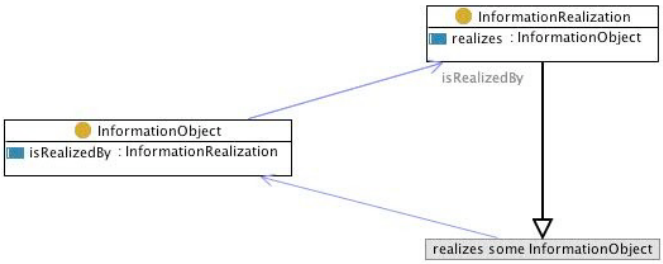

Figure 1: The information realization Content ODP's UML graphical representation.

objects such as poems, songs, formulas, and their physical realizations, such as printed books, registered tracks, physical files. For example, this Content ODP helps modeling the multimedia domain; it is in fact used in the COMM ontology [2]. The GUC of this Content ODP can be expressed by two CQs; '*What are the physical realizations of this information object?*' and '*What information objects are realized by this physical object?*'. Content ODPs are collected and presented on a dedicated semantic wiki, *ontologydesingpatterns.org*[2]. In addition to their diagrammatic representation, Content ODPs are described using a number of catalogue entry fields (c.f. software pattern templates), such as *name*, *intent*, *consequences*, and *building block* (linking to an OWL realization of the pattern). Refer to entry for this pattern from the ODP portal[3] for further details. Reusing Content ODPs is a special case of ontology reuse. Ontology reuse in general is a hard research issue. Currently the support available for finding reusable ontologies consists mainly of search engines, such as Watson[4], SWOOGLE[5], and Sindice[6]. Tools such as the NeOn toolkit in combination with the Watson plug-in let a developer integrate parts of the retrieved ontologies into the one being built. However, the matching is simple keyword-based matching. More elaborate matching schemes have been proposed in OntoCase [4]. Nevertheless, the problems of selecting the right ontology, e.g. Content ODPs, reusing and specializing it, or composing several ontologies, are largely unaddressed.

3.2 Method for Pattern-based Design

To structure the experiment setting and restrict the degrees of freedom when experimenting on ODPs, we decided to introduce a specific ODP reuse method to the experiment participants. Based on previous design experience and principles of ontology modelling, we have developed a set of initial methods for pattern-based ontology design, called eXtreme Design (XD) [17]. XD focuses on two main aspects; the problem space and

[2]http://www.ontologydesignpatterns.org
[3]http://ontologydesignpatterns.org/wiki/Submissions:Information_re
[4]http://watson.kmi.open.ac.uk
[5]http://swoogle.umbc.edu/
[6]http://sindice.com/

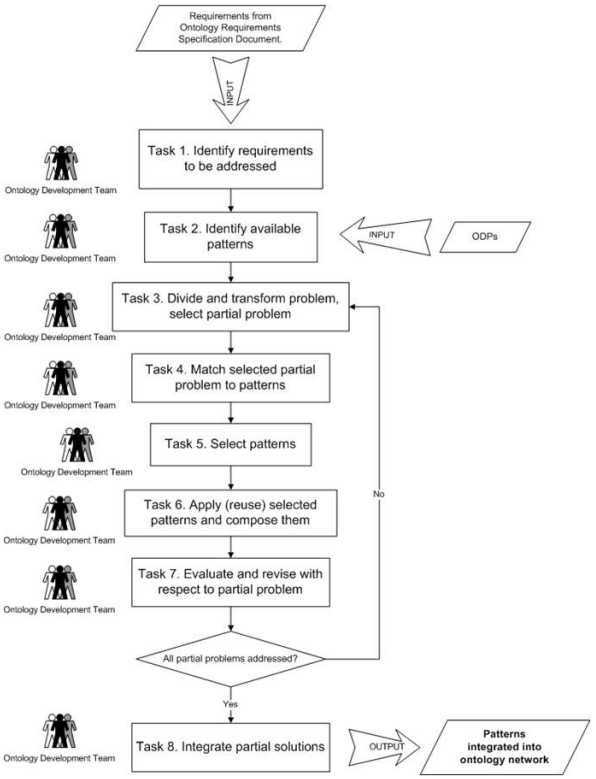

Figure 2: Overview of the XD method.

the solution space. The problem space represents the actual (local) problems, typically expressed as competency questions (CQs). The solution space is composed of ODPs. XD provides means for matching elements of the problem space to elements of the solution space.
The XD method for Content ODP reuse, including a detailed division of the steps to be carried out, was used in the experiments, in order to help participants use Content ODPs in a uniform manner. However, a deeper elaboration and evaluation of XD is still future work. The XD Content ODP reuse method is illustrated in Figure 2. Further details can be found in [19].

4. EXPERIMENTS ON CONTENT ODP REUSE

By analyzing the results and opinions of participants modelling ontologies, with and without Content ODPs, we address the set of questions below. For a detailed report of these experiments see [7].

1. Are Content ODPs perceived as useful by the participants?

2. Are the ontologies constructed using Content ODPs better in some modelling quality sense, than the ontologies constructed without patterns?

3. Are the tasks given to the participants solved faster when using Content ODPs?

4. How do participants use the Content ODPs provided, and what support would be beneficial?

5. What common modelling 'mistakes' can be identified, both when not using patterns and when using the available Content ODPs?

4.1 Experiment Settings

Subjective opinions were collected through questionnaires, while objective measurements were based on studying the ontologies produced by the participants.

4.1.1 Session Setup

We divided the experiment into three sessions. Each session involved a different set of participants, and was held in different time periods (the numbering of the session indicates its chronological order). The experimental variable of all sessions was to let the set of participants construct ontologies "without using patterns" and then "with Content ODPs". In all sessions the participants 1) filled out a questionnaire recording their background and previous knowledge, 2) participated in training on OWL modelling, 3) solved the first experiment task (Task 1) without having any insight into ODPs, 4) filled out a questionnaire recording their experience and opinions of Task 1, 5) participated in training on Content ODPs, 6) solved the second experiment task (Task 2) where a catalogue of Content ODPs were available, and 7) filled out a questionnaire recording their experience and opinions of Task 2.
The first training occasion (see 2 above) intended to leverage the previous knowledge of the participants, and to introduce the XD method, in order to limit the degrees of freedom of the setting. In experiment sessions 1 and 3, practical training was incorporated, while in session 2 only theoretical lectures were provided (for 2 and 5 above). In sessions 1 and 3, the participants also filled out a questionnaire after the practical pattern training. The participants had a limited time to solve Task 1 and 2. For sessions 1 and 3 the time was limited to 2 hours, for session 2 the limit was 1 hour (due to logistic issues). All settings used the same tool for modelling, i.e. TopBraid Composer[7], and all settings used the same two task descriptions (and instructions). Task 1 was set within the music industry domain and Task 2 within the domain of hospitals as work places[8]. The two tasks have been designed in order to pose the same modeling issues in different domains. Hence, they can be addressed by reusing the same set of Content ODPs, all available in the pattern base provided to the participants during the experiment. The pattern base consisted of the 22

[7]http://www.topquadrant.com/products/TB_Composer.html. The reason for choosing this tool is simply that at the time of the experiment it was the "least leaky" ontology editor for managing multiple ontologies with substantial import and workspace management needs, as XD requires.
[8]Detailed description of the tasks can be found at http://ontologydesignpatterns.org/wiki/Training:NeOn_2008_Tutorial_on_ Computational_Ontologies

patterns at this time submitted to the ODP portal[9]. 17 of those were patterns addressing broad competency questions (e.g. modeling parts or situations), while 2 belonged to the biology domain, 2 to the fishery and agriculture domain, and one to the business domain. Out of those 22 patterns, the tasks were constructed to cover problems matching the general requirements of 6 of the patterns (where one pattern was an alternative to the combination of two others). Additionally, four patterns were applicable as alternatives to patterns in the set above, although the intents of these patterns were slightly different from the intent of the task descriptions. Each task also consisted of minor parts that could not be solved using any pattern.

The total number of participants was 45, distributed over the three sessions. In sessions 1 and 3 the participants worked in pairs (the same pairs throughout the session), while in session 2 participants worked alone. Questionnaires were always answered individually. The questionnaires contained several types of questions. The majority of the questions were propositions, where answers were selected from a 5-point Likert-scale, ranging from 'strongly agree' to 'strongly disagree'. Another common type was free text questions, where participants explained their opinions freely.

4.1.2 Ontology Analysis Methods

The focus of the evaluation was on the functional and usability levels (defined in [9]), although some methods cover the structural level to some extent. The ontologies were analysed with respect to four different aspects; 1) coverage of problem, 2) usability, 3) modelling mistakes/incomplete solutions, and 4) pattern usage.

Coverage of problem. To measure the coverage, two different measures were used; terminological coverage and task coverage, i.e. two of the functional measures defined in [9]. Terminological coverage means that the vocabulary of the modelling problem is represented in the solution, whereas task coverage' means that the intended tasks, i.e. CQs, are supported by the solution.

The terminological coverage was assessed through deriving for each task a set of terms that should be covered, by means of (1) direct extraction of terms and (2) abstraction from named entities in the modelling problem description. For example, from the sentence '*During 2005, the band recorded the album Stadium Arcadium*', the term set {*year, band, recording, album*} can be derived. The coverage was evaluated based on how many terms were represented in the solution, either directly or as 'near-synonyms', i.e. terms with a meaning very close to the original term in the context of this task. For example, both 'musical band' and group' are near-synonyms to 'band' in our task context.

The task coverage was assessed based on the amount of queries enabled by the solutions, compared to what queries can be derived from the task descriptions. For each task, a set of CQs were derived from the task description, e.g. from the sentence '*The album was released in May 2006*' the CQ '*When was an album released?*' can be derived. The solutions were evaluated with respect to the set of CQs and for each CQ its solution in the particular ontology was classified either as 'covering the CQ', 'covering the CQ but showing some shortcomings', or 'not covering the CQ'. Shortcomings in this context can be that one has to know the semantics of the property names in order to pose a correct query, or that the solution does not enable the encoding of complete queries.

Usability. The usability, i.e. the clarity and understandability of the ontology, was measured using two sets of characteristics. The first set includes usability profiling measures, as defined in [9], considering the presence of (i) naming conventions, (ii) labels, and (iii) comments. The second set includes structural aspects providing formal semantics to the ontology. Formal semantics provides increasing clarity of the intended meaning of the ontology elements, through the presence of (iv) inverse relations, (v) disjointness axioms, and (vi) level of axiomatization, measured based on the number of complex class denitions.

The presence of a naming convention was assessed by studying the naming of concepts, properties and instances of the ontology. For the remaining characteristics the number of possible occurrences of each characteristic (e.g. labels or inverse relations) was compared to the actual number of occurrences. For example, each concept, instance and property could have a label defined, but in many cases the ontology developer has defined labels for only a fraction of those elements. These fractions where put into four categories; 'none', 'some', 'most', and 'all', where 'some' denotes less than two thirds and 'most' denotes more than two thirds.

Mistakes and patterns Modelling 'mistakes' and the presence of Content ODPs were both identified and analyzed through inspection of the solutions. We define, in this context, modelling 'mistakes' as incomplete solutions that attempt to solve a specific problem but that have shortcomings, see also above. The presence of patterns was identied in the ontologies where no patterns had been used explicitly, i.e. by identifying design choices similar to Content ODPs, as well as in the cases where patterns were actually used.

4.2 Analysis and Results

In this section the results of the three sessions are presented and compared.

4.2.1 Participants' Background

The participants in the three sessions were mostly inexperienced ontology developers. This is an important target group of ODPs, however experiments also targeting expert ontology engineers are part of future work. The subjects of the first session (18 participants) were mainly research assistants and PhD students in com-

[9]http://ontologydesignpatterns.org/wiki/Submissions:Main

puter science and related fields, but without much experience in ontologies. Many had some modelling experience, e.g. UML and ER-modelling. In the second session the subjects (8 participants) were PhD students and senior researchers in computer science and related fields. In this group three persons had more substantial experience in developing ontologies, however only in other logical formalisms than OWL. Finally, the third session consisted of a group of master students (19 participants), participating in the experiment as a part of a course. A majority of the students had previously taken an introductory course in information modelling, where also ontologies (not OWL) were treated.

4.2.2 Result Summary and Analysis

The modeling problems of both the tasks were generally perceived as reasonably easy to understand, and the participants felt familiar with both domains, according to the opinions recorded through questionnaires. The tool created slightly more problems for the group of master students in the third session, but generally the tool was perceived as easy to use.

A majority of the participants had to remodel some parts of the ontology during the development process, i.e. they had to change some modelling decisions and refactor the solution (no difference between the two tasks). The participants were also asked if they felt they had time to solve all problems in a 'good' way, i.e. if they felt satisfied with their solution. Many participants were satisfied, however slightly less so for the second task. This could be due to the task itself, but it could also depend on the introduction of Content ODPs. In this case the second session differs from the two others. They were more dissatisfied, and most likely this can be referred to that they had only half the time, in combination with receiving no practical training (see section 4.1.1) and not working in pairs.

Problems commonly exemplified by the participants were for the first task how to use certain constructs in OWL, e.g. specific modelling primitives, and expressivity problems such as modelling n-ary relations, how to interpret ambiguous requirements, and assessing the consequences of modelling decisions. After the second task, problems were focused on how to match the task to the Content ODPs, how to select patterns, how to reuse the building blocks, and how to compose several patterns. Most participants used the patterns as building blocks, i.e. reused the OWL implementation linked from the catalogue, rather than just as inspiration.

When asked about the Content ODPs, the participants found them hard to understand at first, but already after the brief training there was a considerable improvement. Some of the patterns were found quite 'obvious', i.e. presenting trivial solutions, while others introduce new ideas that the participants did not come up with themselves (in their own opinion) before studying the patterns. The ease of use of Content ODPs was improved by training, however the tool created problems since there are no dedicated functionalities supporting pattern reuse. This is not only a problem of TopBraid Composer, since current pattern support is scarce in ontology engineering tools. One tool, Protégé[10], includes support for logical patterns, but we are not aware of any tool supporting Content ODP reuse.

In all three sessions, a majority of the participants agreed that Content ODPs were in fact useful, i.e. they 'agree to some extent' or 'strongly agree'. In the three sessions 86%, 50%, and 64% of the participants respectively stated that the patterns were useful, and only very few stated that the patterns were not useful (0%, 25% and 7% respectively), the rest neither agreed nor disagreed. The least support is found in the second session (having less time and no practical pattern training). We conclude that Content ODPs are perceived as useful, but only given time to understand the patterns. However, it is not an extensive training that is needed. As a motivation to their responses the participants mentioned that patterns help to decompose the problem and modularize the ontologies, assist the understanding of complex modelling choices, increase the quality of the ontology, and make the modelling easier due to the reuse of implemented building blocks.

Some questions also concerned the participants perception of how the Content ODPs helped them. A clear majority of participants agreed that they constructed 'better' ontologies, and a majority also agreed that it was easier, but in this case many participants were also uncertain. There is however no agreement on solving the task faster when using patterns. A majority of the opinions instead seem to suggest that they were slower, which is also supported by the decrease in coverage of the ontologies (see below).

The increased quality of the solutions, ontology quality as defined by the measures in section 4.1.2, is supported by the analysis of the produced ontologies. The average terminological coverage was lower for the ontologies resulting from the second task than the first one, as seen in Table 1. This could indicate a lack of time to complete the second task, especially since the largest effect can be seen in session 2. There could also be another reason, since participants were in two of the sessions participating in a course, and may have been more focused on studying the patterns, or even evaluating them, rather than on completing the task.

The results of the task coverage assessment, adjusted for the decrease in terminological coverage, can be seen in Table 2. The intuition behind the adjustment is that

Task	Sessions	Coverage
Task 1	1,2,3	83.1%
Task 2	1,2,3	69.1%
Task 2	1,3	74.3%

Table 1: Terminological coverage.

[10] http://protege.stanford.edu/

we want to study the parts of the tasks that were actually included in the solution, then assessing how well these parts solved the intended tasks. The category 'optimally supported CQs' here refer to the amount of ontologies solving the tasks (competency questions) according to best practices, while 'supported CQs' additionally include those solutions covering the CQs but having considerable shortcomings, as defined in section 4.1.2. We can see a significant improvement for the second task. The second session does not show the same increase, most likely due to the reduced time, lack of practical pattern training and the fact that they were not working in pairs, whereby the increase is even more evident if only considering the other two sessions.

A considerable quality improvement for all sessions

Task	Sessions	Measure	Coverage
Task 1	1,2,3	Opt. supported CQs	30.0%
		Supported CQs	65.2%
Task 2	1,2,3	Opt. supported CQs	39.9%
		Supported CQs	73.5%
Task 2	1,3	Opt. supported CQs	48.4%
		Supported CQs	88.3%

Table 2: Functional coverage.

can be seen with respect to usability. In Figures 3 and 4 the results of the two experimental tasks are shown (categories explained in section 4.1.2), i.e. number of ontologies displaying the characteristics on different levels of adoption. Naming conventions were not followed strictly in any of the ontologies, in neither task.
With respect to common modelling 'mistakes', i.e. incomplete solutions as defined previously, the quality of the ontologies improved in all three sessions. In solutions to the first task three common 'mistakes' could be distinguished. The first problem is related to distinguishing between persons and their roles. Many participants modeled roles of people as subclasses of a 'person' concept, without taking into account that roles usually depend on a particular situation. In the music industry task there are roles of band members, such as 'vocalist' and 'guitarist'. If modeled as subclass of 'person' a particular vocalist (instance of the concept) is always a vocalist, in any situation. It makes it impossible to state that a particular band member had the role of 'vocalist' when recording a particular track, while he took the role of 'guitarist' when recording another track.

Another problem where incomplete solutions were found was the distinction between information and its physical realization, in this case the first task contained songs and tracks, i.e. a recording of a specific song by a certain band. Most participants failed to make this distinction, and modeled these as one concept. The third common problem was when modelling n-ary relations, such as the situation when a reviewer comments a specific album release. The participants solved this problem in

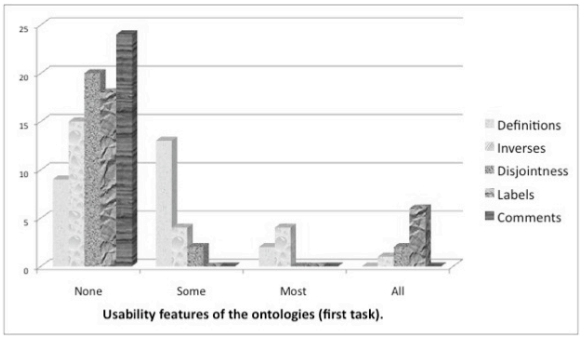

Figure 3: Usability measures (first task).

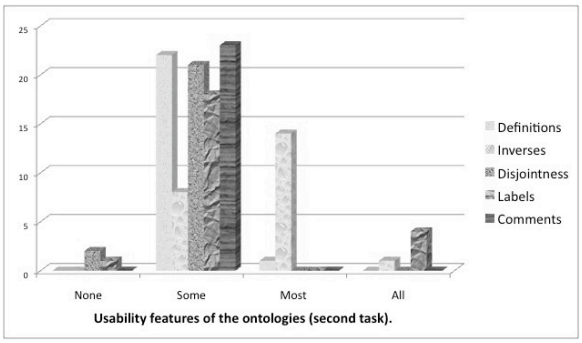

Figure 4: Usability measures (second task).

different ways, but many failed to capture the ternary nature of the relation, i.e. failing to capture either who the reviewer was, what he said, or about what album.
In the ontologies resulting from the second task neither of the first two problem types could be observed, and the number of ontologies displaying problems when modelling n-ary relations had decreased. Most often these modeling problems were in the second task solved using the Content ODPs. However, patterns were occasionally used incorrectly, which indicates that better support for understanding and reusing patterns is needed. Nevertheless, in addition to pointing at an increased quality, these results indicate the suitability for Content ODPs for teaching best practices in modelling.
With respect to pattern presence, we firstly analyzed the ontologies resulting from the first task. Those ontologies that had successful solutions to the problems where common 'mistakes' were noted above, had almost all made design choices similar to those proposed by Content ODPs. For the second task, appropriate patterns were applied in a majority of the cases, but they were not always correctly reused. Especially the **situation** pattern, providing a general vocabulary for n-ary relations, was in several cases misused. In some cases the participants failed to correctly compose patterns, e.g. introducing redundancy. An example is when the roles of people are modeled both as a direct relation between a person and a role, and additionally as an instance of a situation that in turn relates a person, a

role, and a time period.

4.2.3 Research Questions Revisited

Based on the data collected, we are now able to discuss the research questions from the beginning of section 4.

Content ODPs perceived as useful? To answer this we note that on average two thirds of the participants found the patterns useful to some extent (including the session with no practical pattern training), only about 11% did not find them useful . These results let us claim that content patterns are in fact perceived as useful. More in detail, the Content ODPs help by guiding the development and by improving the quality of the solutions. However, developers need proper training to use patterns, hence we believe that patterns will probably be even more valuable to experienced developers.

Increased quality? We found that pattern-based ontologies are improved in functional coverage and usability aspects, exhibit fewer common modelling mistakes', and follow best practices in modelling to a higher extent. On the other hand, the terminological coverage decreases, hence the ontologies are less complete from the terminological viewpoint. This could be connected to the time it takes to understand and use patterns, or using more time for discussing patterns as an argumentation means, or evaluating them (see next paragraph). A decrease in terminological coverage (about 10% in the well-trained sessions), due to a slower development pace, can be easily traded for a substantial improvement in functional coverage (between 35% and 60% in the well-trained sessions). After all, our hypothesis is that Content ODPs help mainly in covering functional rather than terminological requirements. An uncertainty factor is the effect of maturation, which however is probably limited, since the experiments were restricted in time, and since the domains of the two tasks were different.

Faster development? From this study we cannot find any evidence that Content ODPs speed up the development. Nevertheless, some participants stated that the patterns helped in reducing the amount of routine work to be performed by the designer. This could indicate that when developers are used to patterns, the reuse of OWL building blocks can reduce the development time, but this is so far speculations. One important observation made during the sessions was an increased discussion activity among the participants when patterns were introduced. We did not measure this quantitively, and we have no clear explanation for the observation, other than that patterns may have triggered a deeper analysis of the modelling problems. However, this may also be a reason for the participants perceiving to be slower in the second task, since they probably spent more time discussing than in the first task. Other factors may impact the speed. Firstly, the learning curve of the participants: they may be faster in using patterns when they are more familiar with them. Secondly, the particular set of ODPs could have an impact on the time, e.g. the specificity of the patterns or the understandability of their descriptions. In the reported experiments, most of the patterns were general and domain independent, hence more specific patterns could also improve this aspect.

Pattern support? Most participants had problems with finding candidate patterns in the catalogue, matching them to the requirements of the current task, selecting the most suitable patterns, and reusing, e.g. specialising and composing, patterns. Current tools do not give any specific support for pattern reuse. In order to really benefit from patterns, such support is definitely needed. Tools for searching, browsing, and matching requirements to the GUCs applicable to patterns, for comparing and evaluating possible choices and their consequences, as well as for performing operations on Content ODPs when reused, need to be implemented. Some functionalities supporting the XD methods are currently being developed as a plug-in for the NeOn toolkit.

Common problems? Some common problems were discovered, see problems listed in the previous section. That is not a complete list, but it shows commonalities across the specific modelling tasks given to the participants. Also, since the problem types are quite generic, such commonalities constitute an interesting set of anti-patterns'.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have presented experiments in order to assess the usefulness of Content ODPs. Results show that Content ODPs are perceived as useful by ontology developers, and that the quality of the ontologies improves when Content ODPs are reused. The most significant quality improvements include functional coverage, usability aspects, and as fewer modelling mistakes'. The ontology developers additionally perceived that the modelling was made easier, but the hypothesis that patterns can speed up' the development process was not supported. Rather, most participants felt they were slower at the beginning, but that they constructed better' ontologies. The effect of maturation is certainly an uncertainty factor in these results, however the effect was reduced by conducting the study within a very limited time frame and introducing a new domain and task when introducing the patterns in the modeling task.

Future work with respect to pattern experimentation will include experiments using ontology experts; for this target group, we expect that the benefits will be even more prominent. Experiments will also be conducted to evaluate the XD methods, and the tool being developed. Further experiment will also cover other domains and different types of tasks, in accordance with guidelines from [18]. The next set of experiments being planned include the setting where one group of participants solves Task 1 without patterns and Task 2 with Content ODPs, while another group receives the tasks in opposite order, hence solving Task 2 without patterns

and Task 1 using Content ODPs. Already studies are in progress addressing how ontology engineers find and subsequently understand and select patterns, in order to detail the support needed for these tasks.
During the experiments the participants commented on what problems they perceived when using patterns, and several common problems were related to tool and method support for the Content ODP reuse process. We are currently in the process of developing such tool support, as mentioned above. In addition to this, guidelines are needed. Future work includes the refinement and evaluation of the XD family of methods, as well as incorporation and tool support for collaborative XD.

Acknowledgments

Research reported in this paper was supported by the EU under the IST-2006-027595 project NeOn.

6. REFERENCES

[1] A. Ampatzoglou and A. Chatzigeorgiou. Evaluation of object-oriented design patterns in game development. *Information and Software Technology*, 49(5), 2007.

[2] R. Arndt, R. Troncy, S. Staab, L. Hardman, and M. Vacura. COMM: designing a well-founded multimedia ontology for the web. In *The Semantic Web: ISWC 2007 + ASWC 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 30–43, Berlin, Heidelberg, 2008. Springer.

[3] K. Beck, J. O. Coplien, J. Crocker, L. Dominick, G. Meszaros, F. Paulisch, and J. Vlissides. Industrial experience with design patterns. In *ICSE-18, Conference Proceedings*. IEEE Press, Los Alamitos, 1996.

[4] E. Blomqvist. *Semi-automatic Ontology Construction based on Patterns*. PhD thesis, Linköping University, Department of Computer and Information Science at the Institute of Technology, 2009.

[5] E. S. Chung, J. I. Hong, J. Lin, M. K. Prabaker, J. A. Landay, and A. L. Liu. Development and evaluation of emerging design patterns for ubiquitous computing. In *Proceedings of the 2004 Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques*. ACM Press, New York, 2004.

[6] A. Dearden, J. Finlay, L. Allgar, and B. McManus. Evaluating pattern languages in participatory design. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*. ACM Press, New York, 2002.

[7] M. Dzbor et al. D5.6.2 Experimentation with parts of NeOn methodology. Technical report, NeOn Project, 2009.

[8] A. Gangemi. Ontology Design Patterns for Semantic Web Content. In *M. Musen et al. (eds.): Proceedings of the Fourth International Semantic Web Conference*, Galway, Ireland, 2005. Springer.

[9] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling ontology evaluation and validation. In *Proceedings of ESWC 2006*, pages 140–154, 2006.

[10] A. Gangemi and V. Presutti. Ontology design patterns. In *Handbook on Ontologies, 2nd Ed.*, International Handbooks on Information Systems. Springer, 2009.

[11] M. Gruninger and M. S. Fox. The role of competency questions in enterprise engineering. In *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, 1994.

[12] P. Kotze, K. Renaud, A. Dearden, K. Koukouletsos, and B. Khazaei. Patterns, anti-patterns and guidelines - effective aids to teaching HCI principles? In *Inventivity: Teaching theory, design and innovation in HCI - Proceedings of HCIEd2006*, 2006.

[13] G. Masuda, N. Sakamoto, and K. Ushijima. Evaluation and analysis of applying design patterns. In *Proc. of the International Workshop on the Principles of Software Evolution*, 1999.

[14] G. Papamargaritis and A. Sutcliffe. Applying the domain theory to design for reuse. *BT Technology Journal*, 22(2):104–115, 2004.

[15] L. Prechelt, B. Unger-Lamprecht, M. Philippsen, and W. F. Tichy. Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. *IEEE Trans. on Software Engineering*, 26(6), 2002.

[16] V. Presutti and A. Gangemi. Content ontology design patterns as practical building blocks for web ontologies. In *Proceedings of the 27th International Conference on Conceptual Modeling - ER2008*, pages 128–141, 2008.

[17] V. Presutti et al. D2.5.1: A library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. NeOn Deliverable 2.5.1, NeOn Consortium, 2008.

[18] N. R. Shadbolt, K. O'Hara, and L. Crow. The experimental evaluation of knowledge acquisition techniques and methods: History, problems and new directions. *International Journal of Human-Computer Studies*, 51(4):729–755, 1999.

[19] M. C. Suárez-Figueroa et al. D5.4.2. revision and extension of the neon methodology for building contextualized ontology networks. Technical report, The NeOn Project., 2009.