Experimenting with eXtreme Design

Eva Blomqvist, Valentina Presutti, Enrico Daga, and Aldo Gangemi

STLab, ISTC-CNR, via Nomentana 56, 00161 Rome, Italy
eva.blomqvist@istc.cnr.it,
{valentina.presutti,enrico.daga,aldo.gangemi}@cnr.it

Abstract. Ontology Design Patterns (ODPs) support reusability and use of best practices in ontology engineering. Previous studies have shown that Content ODPs, in particular, have some measurable beneficial effects on the produced ontologies. However, another conclusion was that methodology and tool support was needed. Now such support exist, in the form of the XD methodology and the XD Tools. In this paper we present a set of experiments for (i) confirming previous conclusions concerning the usefulness of Content ODPs, (ii) investigating the usefulness of the XD methodology, and (iii) investigating the usefulness of the XD Tools. Main conclusions are that we can confirm most of the previous results concerning the usefulness of Content ODPs, and certain negative effects observed previously are now mitigated by the new tool support. The tool is perceived as quite useful, however it also adds some overhead. The XD methodology is found to be a helpful means to organize the design process, and the main benefit shown through the experiments is the testing focus, resulting in a drastic decrease of certain frequent mistakes.

1 Introduction

Ontology Design Patterns (ODPs) are emerging as an important support for various ontology engineering tasks. 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), ODPs can support reusability on the design side. As described in [1] ODPs can be of several types e.g. focusing on logical language constructs, architectural issues, naming, or efficient provision of reasoning services. In this paper we focus on Content ODPs (CPs). CPs are small (or cleverly modularized) ontologies with explicit documentation of design rationales, representing modeling best practices, and can be used as building blocks in ontology design [2,3].

As an example we describe a CP that is called *AgentRole*. It represents the relation between agents, e.g., people, and the roles they play, e.g., manager, meeting chair, father, and friend, as well as the disjointness of agents and roles. Figure 1 shows the UML diagram (produced through TopBraid Composer[1]) of the OWL[2] building block representing this CP. CPs are collected in different

[1] For notation details, see tool documentation: http://www.topquadrant.com/products/TB_Composer.html

[2] http://www.w3.org/2004/OWL/

P. Cimiano and H.S. Pinto (Eds.): EKAW 2010, LNAI 6317, pp. 120–134, 2010.

catalogues, such as the *ODP portal*[3]. In addition to their diagrammatic representation CPs are described using a number of catalogue entry fields (c.f. software pattern templates), such as *name*, *intent*, *covered requirements*, *consequences*, and *building block* (linking to an OWL realization of the pattern).

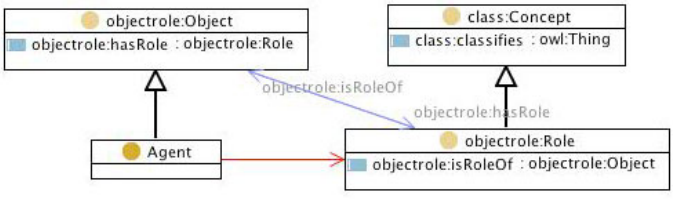


Fig. 1. The AgentRole Content ODP's graphical representation in UML

In [4] we presented the results of initial experiments on CP reuse, showing that CPs are indeed useful for constructing better and more (re)usable ontologies. However, we also concluded that additional method and tool support would be needed in order to truly benefit from CPs; common problems the participants experienced were to find the right CPs for their requirements, correctly specialize and compose them, as well as discover possible mistakes in the solutions. In response to the results of the study in [4] we have developed the XD Tools and a methodology (the eXtreme Design methodology - XD) for CP-based ontology design [5]. In this paper we present a new set of experiments with the following three aims; (i) to confirm the usefulness of CPs, (ii) to investigate the usefulness of the XD methodology, and (iii) to investigate the usefulness of the XD Tools. The novel contributions of this work include (1) a set of integrated tools called "XD Tools" for CP-based ontology design, and their user-based evaluation, (2) the evaluation of CP-based ontology design conducted with XD Tools, and comparison with results presented in [4], (3) the evaluation of the XD methodology.

The rest of the paper is structured as follows: Section 1.1 describes related work and in Section 2 we introduce the experimental setting, and evaluation criteria. Section 3 describes the XD Tools, and we analyze the results of the CP-based ontology design experiment as well as how XD Tools are perceived by the users. In Section 4, after a brief description of the XD methodology, we present its evaluation, and in Section 5 we discuss conclusions and future work.

1.1 Related Work

Early ontology engineering methodologies were [6], [7], and [8], while more recent ones focus on collaboration [9], or transfers popular software engineering processes to ontology engineering, e.g. the Unified Process [10]. The only pattern-based methodologies we are aware of are [11] (not considering collaboration, and patterns are assumed to be a non-evolving set) and [12] (tailored to ambient intelligence applications). When proposed, methodologies are commonly

[3] http://www.ontologydesignpatterns.org

not evaluated, only exemplified through use-cases as in [7] and [12], or analyzed through theoretical comparisons as in [10]. Instead, their later adoption in ontology engineering projects prove their usefulness. Methodology evaluation (for selection), as performed in software engineering (e.g. through NIMSAD [13]), has not been widely used in ontology engineering. Methodologies and tools have different focus and aims, hence, it is impossible to proclaim the 'best' tool or methodology if there is no well-defined frame of reference for the evaluation.

Currently, pattern-based ontology engineering methods and tools are present primarily on the logical level, e.g. for ontology learning and enrichment as in [2], the Ontology Pre-Processor Language (OPPL) and methods for applying it as a means for logical ODP reuse [14], and the proposal for a high-level pattern language in [15]. Use of ODPs have been present in some ontology engineering environments, such as the logical pattern templates in Protégé 3 (as explained in [16]), and the template wizard supporting OPPL pattern definitions in Protégé 4 [14], targeting mainly the use of Logical ODPs, although it supports the introduction of CPs in an ontology with a macro-like mechanism. The combination of Naming and Logical ODPs has also been proposed for supporting ontology refactoring in [17]. Benchmarking and evaluation has been more widely performed on the tool level, compared to the methodological level, e.g. through comparisons such as [18] and more recently in [19], and observational studies, as in [20].

2 Experimental Method

The focus is on evaluating the effectiveness of CP-based ontology design. We have conducted experiments in order to evaluate how CPs improve the quality of the results, i.e., the ontologies, and whether the XD Tools and methodology have an additional impact. Below we describe the experiment setting, the participants, and the criteria we have applied in order to analyze the resulting ontologies.

2.1 Experimental Setting

Similarly to [4] experiments were carried out during master and PhD courses. We divided the experiment into **two sessions**[4], each involving a different group of participants. The experimental variable of both sessions was to make the participants, who worked in pairs, first construct ontologies 'without using CPs', then 'with CPs and the XD Tools', and finally 'with CPs, the XD Tools and also following the XD methodology'. Furthermore, each participant (individually) filled out a questionnaire at the end of each task, as well as a background questionnaire at the beginning of the session[5]. 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 open questions, where participants explained their opinions freely.

[4] The sessions were separated in time and involved different sets of participants.

[5] All the questionnaires used can be downloaded at http://stlab.istc.cnr.it/documents/papers/QuestionnairesExperiment2010.zip

Both sessions were organized into three slots, as summarized in Table 1. Training in Slot 1 aimed at leveraging the previous knowledge of the participants, to limit the degrees of freedom of the setting. Tasks were expressed in terms of *Competency Questions* (CQs) [21], expressing the domain requirements to be addressed by the ontology. The two groups solved the same exercise as Task 1 (theater domain). For Tasks 2 and 3 the exercises were switched between the two sessions in order to reduce the impact of the task domain and participant maturation on the results, i.e., in the first session Task 2 was set in the music domain and Task 3 in the hospital domain, but in session two the exercises were switched. All tasks had the same number of CQs and contained approximately the same set of modelling issues, i.e., some n-ary relations, modelling of roles and time periods, etc.[6], so that the problems could be addressed by reusing a similar set of CPs from the catalogue (available for Task 2-3).

Table 1. Experimental setting

Slot 1	**Slot 2**	**Slot 3**
Background questionnaire		
Training (OWL modeling)	Training (CPs and XD tools)	Training (XD methodology)
Task 1: modeling without any insight into CPs (3 hours)	Task 2: modeling by using a catalogue of CPs, and XD Tools (3 hours)	Task 3: modeling by following the XD methodology (in addition to CPs and XD Tools) (3 hours)
Questionnaire 1	Questionnaire 2	Questionnaire 3

In both sessions the participants used the same ontology editor, i.e., TopBraid Composer[7]. The catalogue of CPs consisted of 56 patterns (i.e. all submissions available in the ODP portal). 32 of those were patterns addressing broad competency questions (e.g. modeling part-whole relations, or situations), while 17 belonged to the fishery domain, 6 to the biology and agriculture domains, and one to the business domain. Out of those 56 patterns, the tasks were constructed to cover problems matching the general requirements of 6 of the patterns. Additionally, 13 other 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 of the patterns included in the catalogue.

2.2 Participants

The total number of participants was 35, distributed over the two sessions (19 in the first and 16 in the second). Participants were mostly inexperienced ontology

[6] Details of the tasks can be found at http://ontologydesignpatterns.org/wiki/Training:PhD_Course_on_Computational_Ontologies_%40_University_ of_Bologna

[7] The reason for choosing this tool was that at the time of Session 1 it was the most stable Eclipse-based ontology editor compatible with the XD Tools plugin.

developers, an important target group of ODPs. The subjects of the first session were mainly master students in computer science and business informatics, without much experience in ontologies (except a course introducing information and data modelling). In the second session[8] the subjects were PhD students and junior researchers in mainly computer engineering, informatics, and law. In this group a few persons had more substantial experience developing ontologies, and already knew the basics of OWL, however, none had previously used ODPs.

2.3 Ontology Analysis Methods

To make the results comparable to [4], we used the same methods and measures to assess the characteristics of the output ontologies. Here we only give a brief summary of the measures, as they are explained in detail in [4]. The focus of the evaluation was on the functional and usability levels, as defined in [22]. The ontologies were analysed with respect to four aspects; 1) coverage of problem, 2) usability, 3) modelling mistakes/incomplete solutions, and 4) pattern usage.

Coverage of problem. Two different measures were used; terminological coverage and task coverage. 'Terminological coverage' measures the amount of the vocabulary of the domain problem, i.e. the terms (allowing for morphological variations or synonyms) used to express the CQs, that are represented in the solution. 'Task coverage' is a measure of the amount of the intended tasks, i.e. CQs, that is supported by the solution, i.e. the amount of CQs that can be executed as SPARQL queries on the model. Each CQ was classified either as 'excellently covered', 'covered with shortcomings', or 'not covered'. Shortcomings in this context can be that one has to know the implicit semantics of the property names in order to pose a correct query.

Usability. The usability, i.e. the clarity and understandability of the ontology, was measured using a set of usability profiling measures, and a set of structural aspects providing formal semantics, i.e. clarity of meaning, to the ontology. They are: the amount of (i) labels, (ii) comments, (iii) inverse relations, and (iv) disjointness axioms, as well as the (v) level of axiomatization, e.g. measured based on the number of complex class definitions.

Mistakes and patterns. Modelling 'mistakes' and the presence of CPs 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.

3 Tool Support for Ontology Design with CPs

The study presented in [4] pointed at the need for tool support for ODP-based ontology design. The main needs identified by users were support for finding

[8] Despite the difference in participant background we choose to present most results in this paper as averages over both groups, only in the cases when significant differences in the results were noted do we separate the presentation of results.

and selecting the right ODPs for their requirements, correctly specializing and composing them, and discovering possible mistakes in the resulting ontology. In order to address these requirements we have developed the eXtreme Design Tools (XD Tools)[9], a set of software components available as an Eclipse plugin, accessible through a *perspective* - eXtreme Design - compatible with Eclipse-based ontology design environments such as TopBraid Composer and the NeOn Toolkit[10].

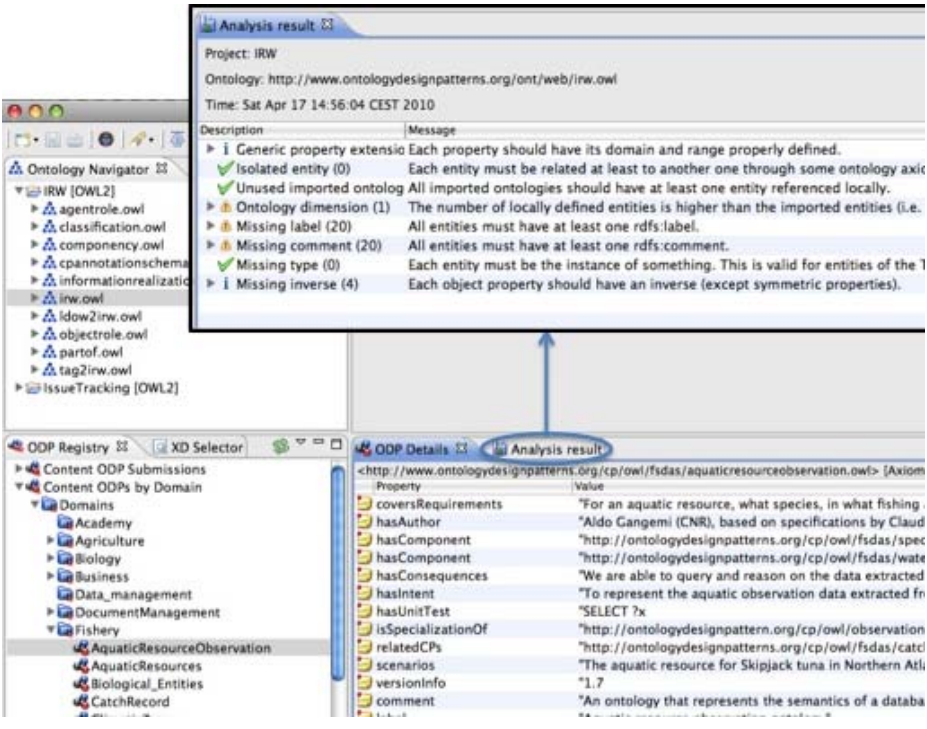


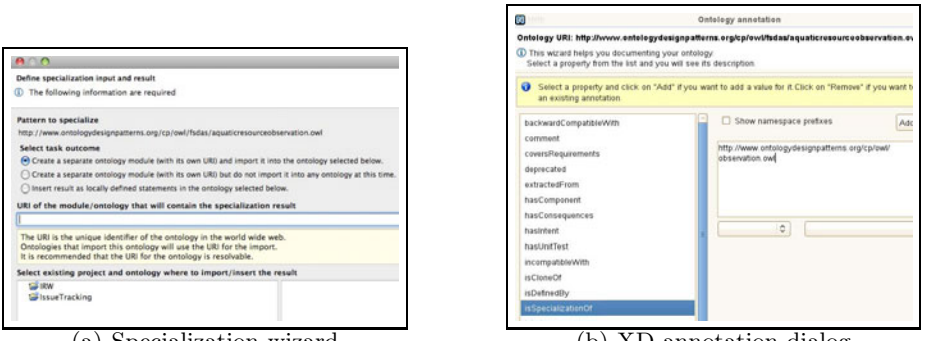
Fig. 2. XD Tools GUI in the NeOn Toolkit

3.1 Main Functionalities of XD Tools

Currently, XD Tools is comprised of five main components supporting pattern-based design. The overall view of XD Tools GUI is depicted in Figure 2. XD Tools' components are the following:

ODP Registry browser: Exposes sets of CPs to the user, in a tree-like view categorized by different aspects such as the domain of the CP (Figure 2 bottom-left). In this way users can access a set of reusable CPs without having them locally stored. The default registry is provided by the ODP portal.

[9] http://stlab.istc.cnr.it/stlab/XDTools
[10] http://neon-toolkit.org



(a) Specialization wizard (b) XD annotation dialog

Fig. 3. XD specialization wizard and annotation dialog

The **ODP Details view** (Figure 2 bottom-right) shows all annotations of a selected CP. In this way CPs can be examined without downloading the OWL building block, or accessing an external website. By right clicking on a CP it can be downloaded through the "Get" command.

XD Selector: Proposes CPs, which can be reused, to the user. The task of matching the intent of a CP to the specific requirements can be challenging, especially if the CP catalogue is large. Since pattern selection is one of the most difficult tasks to automate, we have developed an extensible system that permits to plug in multiple services. Currently two services are available, i.e., search based on keyword indexing and latent semantic indexing, respectively. The suggested CPs can be downloaded through the "Get" command.

Specialization wizard: CP specialization, as the primary step of their reuse, concerns the specialization of ontology elements in the CP, through axioms such as subsumption. This can be challenging for an inexperienced user if it is done one element at a time, without guidance. From a user perspective, CP specialization has the following steps: (i) import the pattern into the working ontology, (ii) declare subClasses/subProperties for each of the (most specific) pattern elements needed, and (iii) add any additional axioms needed. The specialization wizard provided by XD Tools (Figure 3(a)) guides the user through this process, with some steps being optional and some required. The wizard is activated by right clicking on a CP and selecting "Specialize".

XD Annotation dialog: Supports annotation of ontologies, based on customized annotation vocabularies. The annotation properties already provided by OWL/RDF and vocabularies such as OMV [23] and the CP annotation schema[11] are provided by default[12].

XD Analyzer: Provides feedback to the user with respect to how 'best practices' of ontology design have been followed. The XD Analyzer has a

[11] http://ontologydesignpatterns.org/schemas/cpannotationschema.owl
[12] Since CPs are small ontologies the properties can be used for ontologies in general.

pluggable architecture, allowing to easily extend the set of heuristics expressing 'best practices'. Three levels of messages are produced; errors, warnings (identified 'bad practices'), and suggestions (proposals for improvement). An error is, for instance, a missing type, i.e., all instances should have a specified class they are instances of. Examples of warnings are missing labels and comments, and isolated elements that are not referred to by other elements. Proposing to create an inverse for each object property that has no inverse so far is on the level of suggestions. An example view of the Analyzer is shown at the top of Figure 2.

In addition, XD Tools provide several help functions, such as inline info boxes, help sections in the Eclipse help center, and cheat sheets.

3.2 Experimental Results: CP-Based Ontology Design

Four of the research questions posed in [4] are also the basis of the experimental setting described in this paper.

1. Are CPs perceived as useful by the participants?
2. Are the ontologies constructed using CPs 'better', in some modelling quality sense, than the ontologies constructed without patterns?
3. Are the tasks solved faster when using CPs?
4. What common modelling 'mistakes' can be identified, both when not using patterns and when using the available CPs?

Perceived usefulness (1). Table 2 compares the results observed in the two sessions with the ones obtained in the previous setting (see [4]). On average, the fraction of participants who perceive the CPs as useful has increased, with a decrease of those stating they were not useful. However, there is a significant difference between the two sessions. The only major difference (apart from the background of the participants that did not impact the results in [4]) between the two sessions, which in our opinion can explain the difference, is the tool support. While the first session had an initial version of the XD Tools, the second session had XD Tools in a stable version with full functionality.

Table 2. Perceived usefulness (percentage of participants who agreed or disagreed, the rest neither agreed nor disagreed)

Setting	Useful	Not Useful
Setting presented in [4]	67%	11%
Session 1	67%	8%
Session 2 (more stable tool support)	93%	4%

Result quality - Coverage (2). Table 3 compares terminological and task coverage of the ontologies resulting from the execution of Task 1 and Task 2, according to the ontology analysis method described in Section 2. For what concerns task coverage, the results are inconclusive, since the increase is very small.

While, with regard to terminological coverage the results have improved in the new setting, compared to [4]. In fact, in the previous setting, the terminological coverage decreased from Task 1 to Task 2, hence the ontologies were less complete from the terminological viewpoint. In the new setting the coverage keeps stable, which may be attributed to the new tool support.

Table 3. Terminological and Task Coverage. Percentages indicate an average over all ontologies (details on criteria in Section 2 and in [4])

	Terminological coverage	**Task coverage**	
		Covered (excellently or with shortcomings)	**Covered excellently**
Task 1	80%	69%	41%
Task 2	79%	70%	44%

Result quality - Usability (2). Table 4 compares the usability indicators between the two tasks. These results are comparable to [4], and confirm that usability is the aspect showing the clearest improvement when introducing CPs.

Table 4. Usability. Percentages indicate an average over all ontologies, e.g. in Task 1 on average 76% of classes and properties had labels, in Task 2 the average was 86%

	Labels	**Comments**	**Disjoint classes**	**Inverse prop.**	**Complex class def.**
Task 1	76%	0%	2%	6%	5%
Task 2	86%	35%	37%	42%	21%

Solving tasks faster (3). There is still no evidence that CPs support faster development, but as noted above the reduced coverage has now been remedied by introducing additional tool support. Hence, in this setting there is not any objective evidence for being slower either (as opposed to the previous study).

Modelling 'mistakes' (4). We have identified a set of frequent modelling mistakes in the ontologies of both tasks. The by far most frequent mistake (occurring in 93% of the ontologies of the first task and 80% of the ontologies of the second task) was missing n-ary relations, i.e., where the requirements clearly state a dependency between three or more elements but in the ontology these are modeled as binary relations, loosing the n-ary nature. Other frequent mistakes were failure to separate roles from persons, e.g., by stating that vocalist is a subclass of person it becomes difficult to state the time period when that role was held or in what band the person acted as vocalist, and missing or wrong datatype properties, e.g., declaring a class for time intervals but failing to add properties holding the actual start and end dates. The most frequent ones in the ontologies developed in Task 1, i.e., 6 types of mistakes, were all still present in the ontologies developed in Task 2 and we have used them to compare the results.

The fraction of ontologies showing these 6 types of 'mistakes', **decreased on average by 44% in Task 2**. The mistakes were not listed or discussed by the teacher between the sessions in order to reduce the effect of maturation. In addition, no new types of errors were introduced. A few of the mistakes did not show such a drastic decrease, e.g., problems when modeling n-ary relations only decreased with 14% even when having access to CPs dedicated to this issue.

In addition to the previous experiments we also anlayzed the modularity of the produced ontologies. In other words, we added one research question:

5. Do CPs increase the modularity of ontologies?

The results show that in Task 1 none of the ontologies are modularized, i.e. they are comprised of only one OWL-file, while in Task 2 the ontologies contain on average 7.5 modules, and all ontologies are comprised of more than one module. It has to be noted that no specific instructions on producing modular ontologies were given, hence, *the reuse of CPs inherently introduces a modular structure*. Although this may seem obvious, one should note that CPs can not only be used a components, imported and directly reused, but also as 'inspiration' and guidelines for creating your own solutions.

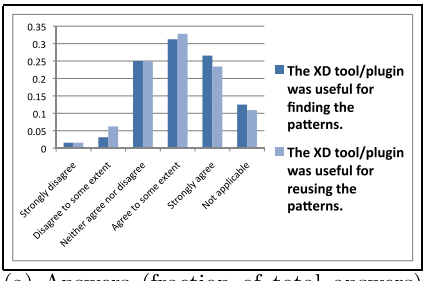
3.3 Experimental Results: Perception of the XD Tools

One major difference between the previous experiments [4] and the current setting was the introduction of the XD Tools. XD Tools was available for Task 2 and 3 in both sessions, however, in the first session XD Tools was still in a testing phase, i.e., contained several bugs and was not entirely stable and user friendly. During this experiment we aimed at answering the following questions:

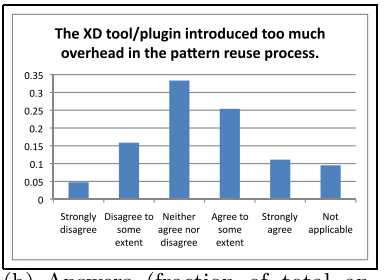
1. How well is XD Tools perceived to support the process of *finding* CPs?
2. How well is XD Tools perceived to support the process of *specializing* CPs?
3. Does the XD Tools introduce too much overhead in the process, i.e., annoying the users?

These questions were addressed mainly by asking the participants to assess the usefulness of the different aspects of XD Tools when filling out the questionnaires, as well as asking directly if they felt that it introduced too much overhead. The responses (fractions of total number of responses) from both sessions can be seen in Figure 4. It should be noted that in Session 1 some pairs experienced technical problems with the tool, hence, all but one of the 'not applicable' responses originate in this session. In Session 2 technical problems were solved.

We note that 58% of the respondents agreed that the XD Tools was useful for finding CPs (only 5% disagreed), and 56% agreed that the tool was useful for reusing CPs (8% disagreed). Clearly, the majority of the participants found the tool useful, as opposed to browsing the ODP portal and specializing patterns without guidance (which was the alternative method presented to them). About 25% of the participants were unsure. A result that supports the conclusion that the tool was useful is the fact that in this new set of experiments terminological coverage did not decrease, which was the case when introducing CPs in the



(a) Answers (fraction of total answers) to the propositions.



(b) Answers (fraction of total answers) to the proposition.

Fig. 4. Evaluation results for XD Tools

previous setting. Additionally, in the second session (using the stable version of XD Tools) we observed a much higher satisfaction with CPs than for the other session. In our opinion this supports the conclusion that the tool was helpful and reduced the effort for users to find and reuse CPs. Finally, more than one third of the participants (36%) agree that the tool does introduce too much overhead (while 33% of the participants are not sure). From informal discussions with the participants we conclude the need of finding a better balance between enforcing best practices on one hand, and providing shortcuts as users get more experienced on the other.

4 eXtreme Design

With the name eXtreme Design (XD) we identify an *agile*[13] approach to ontology engineering, a family of methods and tools, based on the application, exploitation, and definition of ODPs for solving ontology development problems [5]. Below we describe the part of XD targeted in this paper.

4.1 XD Methodology for CP Reuse

We focus on XD for CP reuse in ontology design (hereafter referred to simply as 'XD'), which is currently the most elaborated part of the XD family. In XD a development project is characterized by two sets: (i) the problem space, composed of the actual modeling issues (local problems), e.g., to model roles played by people during certain time periods; (ii) the solution space, made up of reusable modeling solutions, e.g., a piece of an ontology that models time-indexed roles (a CP). Each CP, as well as the local problem, is related to ontology requirements expressed as CQs or sentences, but on different levels of generality. If a local problem can be described, partly or completely, in terms of the CQs of a CP then that CP can be selected and reused for building the solution.

[13] We borrow the term *agile* from Software Engineering because XD is inspired by eXtreme Programming and Software Factories as described in [5] and brings the main principles of agile Software Engineering into Ontology Engineering.

XD is test-driven and task-focused, resulting in highly modular ontologies where each module solves a small set of requirements. Main principles of XD are pair design, the intensive use of CPs, and collaboration, for details see [5]. The iterative workflow of XD contains 12 steps, where the first four steps are concerned with project initiation, scoping, and requirements engineering (i.e. deriving the CQs from user stories), and the three final steps are concerned with the integration of modules into a final solution, hence, it is focused on the collaboration between the pairs. The evaluation of the collaborative part is ongoing work, hence, in these experiments we focus on the iteration by one design pair (creating and testing the modules), whereas the relevant steps include:

5. **Select a coherent set of CQs.** One or more of the CQs, i.e. a coherent set treating one modelling issue, are selected for a first development iteration.
6. **Match the CQs to CPs.** By matching the selected CQs to the requirements covered by CPs, candidate CPs for reuse are identified.
7. **Select CPs to use.** From the set of candidates the CPs that best fit the local problem without unnecessary overhead are selected.
8. **Reuse and integrate selected CPs.** Reusing CPs mean to import them into the ontology module to be built, specialize their classes and properties, and compose them, i.e., add properties or axioms that connect the CP specializations so that the module is able to answer the CQs.
9. **Test and fix.** The CQs are transformed into unit tests, e.g., SPARQL[14] queries, and test instances are added. Tests are run, and any errors discovered are fixed, before selecting a new set of CQs for the next iteration.

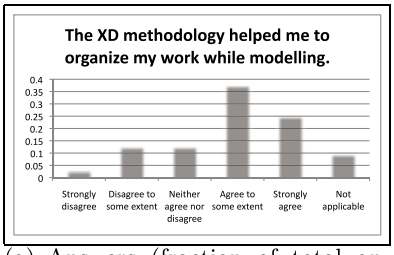
4.2 Experimental Results: The XD Methodology

In Task 3 the XD methodology was introduced. New questions were added to the questionnaires to record the participants' experience, but we performed the same analyses on the ontologies as for Task 1 and 2. Mainly we were trying to answer the following set of questions:

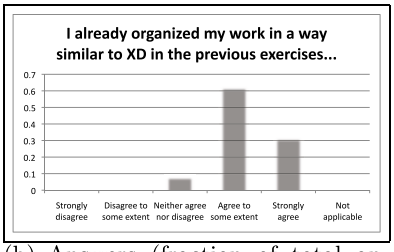
1. Is the XD methodology perceived as useful by the participants?
2. Is the XD methodology a 'natural' way to work with CPs?
3. Are the ontologies constructed using the XD methodology 'better', in some modelling quality sense, than the ontologies constructed 'only' using CPs?
4. Are the tasks solved faster when using the XD methodology, compared to 'only' using CPs?
5. What common modelling 'mistakes' can be identified and are they different from the ones noted when 'only' using CPs?

Usefulness of XD (1). That XD helped them to organize their work was proposed to the participants, and their answers can be seen in Figure 5(a). Only 6% of the participant claimed they did not follow XD closely, hence, we conclude that the XD methodology is perceived as useful for organizing your work.

[14] http://www.w3.org/TR/rdf-sparql-query/



(a) Answers (fraction of total answers) to the proposition.



(b) Answers (fraction of total answers) to the proposition.

Fig. 5. Evaluation results for XD methodology

'Natural' way to work with CPs (2). Methodologies are sometimes perceived as awkward and restrictive by users, however, participants felt comfortable with XD, as can be seen in Figure 5(b). From this we conclude that XD is descriptive and pragmatic. This is not surprising since the methodology has been developed based on our own experience how to approach the problem.

Result quality (3). The terminological coverage of ontologies increased slightly (79% in Task 2 to 83% in Task 3), and the task coverage increased from 69% to 81%. The substantial increase seems to be in the task coverage. On the usability side, levels are similar for Task 2 and 3, only the disjointness axioms show a substantial increase (from 37% to 52%).

Solving tasks faster (4). While still applying the same time limit to solve a problem, task and terminological coverage increased (although the increase in terminological coverage is limited). Our opinion is that this is mainly due to that errors are found more easily (hence faster), although we have to consider some possible effects of participant maturation as well. We believe that XD helps designers to faster problem solving, still, we need to produce stronger evidence for supporting this claim in future studies.

Modelling 'mistakes' (5). The types of mistakes that are frequent are the same as in Task 2, but with a decrease of the occurrence of the top-6 common errors of 15%. Two types of errors decrease significantly more than the others, i.e. the problems in representing n-ary relations (decrease by 64%) and missing datatype properties (decrease by 46%). We believe that the decrease can be attributed to the test-driven nature of XD. By requiring ontology engineers to test their model in a structured fashion, errors that can easily be discovered through unit tests, e.g., missing properties, are indeed discovered.

5 Conclusions

In this paper we presented experiments on CP-related methods and tools. These experiments follow up on, and confirm the results of experiments presented in [4], as well as extend the scope to include experiments on the XD methodology and XD Tools. The aim of the new experiments was threefold; (i) confirming

conclusions on the usefulness of CPs, (ii) investigating the usefulness of the XD methodology, and (iii) investigating the usefulness of the XD Tools.

We can confirm almost all of the results in [4]. However, terminological coverage kept stable in this setting while in the previous one it decreased. This can be easily explained by the new tool support, facilitating the reuse of CPs. The XD Tools was perceived as useful for finding and reusing CPs, however it needs a better balance between enforcing best practices and allowing for shortcuts in the workflow, since many participants felt that it added some overhead. The effects of the XD methodology can be seen mainly in the ontology quality, i.e., increased task coverage and particular previously frequent mistakes that drastically decreased. The frequent mistakes were all connected to missing parts, hence, we conclude that one main benefit of XD is its test-driven nature that forces the user to check every new module against the requirements. Additionally, we conclude that XD is perceived by users as a natural way of working with CPs, still, they felt the methodology was useful for guiding their modelling.

Future work contains further experiments on CPs and their relation to ontology quality, e.g. including other aspects and different groups of participants. For the XD methodology we already started investigating collaboration and integration aspect in some initial observations, but proper experiments are needed. More focused user testing of the XD Tools is also on the agenda, e.g., testing the different components separately, to get clearer indications on what parts of the user interaction can be improved. XD Tools will also be extended with more elaborate CP selection; we are currently working on methods for CQ-based CP selection rather than simple keyword search.

Acknowledgements. This research was partly funded by the European Commission through projects NeOn (FP6 IST-2005-027595) and IKS (FP7 ICT-2007-3/No. 231527).

References

1. Gangemi, A., Presutti, V.: Ontology Design Patterns. In: Handbook on Ontologies, 2nd edn. International Handbooks on Information Systems. Springer, Heidelberg (2009)
2. Blomqvist, E.: OntoCase-Automatic Ontology Enrichment Based on Ontology Design Patterns. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 65–80. Springer, Heidelberg (2009)
3. Presutti, V., Gangemi, A.: Content Ontology Design Patterns as practical building blocks for web ontologies. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 128–141. Springer, Heidelberg (2008)
4. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on Pattern-Based Ontology Design. In: K-CAP 2009. ACM, New York (2009)
5. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: eXtreme Design with Content Ontology Design Patterns. In: Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with ISWC 2009. CEUR Workshop Proceedings, vol. 516 (November 2009)

6. Grüninger, M., Fox, M.: Methodology for the Design and Evaluation of Ontologies. In: Proceedings of IJCAI 1995, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13 (1995)
7. Uschold, M.: Building Ontologies: Towards a Unified Methodology. In: Proceedings of Expert Systems 1996, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK (December 1996)
8. Fernández, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: from Ontological Art towards Ontological Engineering. In: Proceedings of the AAAI 1997 Spring Symposium Series on Ontological Engineering (1997)
9. Pinto, H.S., Staab, S., Tempich, C.: DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain (2004)
10. Nicola, A.D., Missikoff, M., Navigli, R.: A software engineering approach to ontology building. Inf. Syst. 34(2), 258–275 (2009)
11. Clark, P., Porter, B.: Building concept representations from reusable components. In: Proceedings of AAAI 1997, pp. 369–376. AAAI Press, Menlo Park (1997)
12. Maass, W., Janzen, S.: A Pattern-based Ontology Building Method for Ambient Environments. In: Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with ISWC 2009, Washington D.C., USA, October 25. CEUR Workshop Proceedings, vol. 516 (2009)
13. Jayaratna, N.: Understanding and Evaluating Methodologies: NIMSAD, a Systematic Framework. McGraw-Hill, Inc., New York (1994)
14. Iannone, L., Rector, A., Stevens, R.: Embedding Knowledge Patterns into OWL. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 218–232. Springer, Heidelberg (2009)
15. Noppens, O., Liebig, T.: Ontology Patterns and Beyond - Towards a Universal Pattern Language. In: Proceedings of WOP2009 collocated with ISWC 2009, vol. 516 (November 2009), CEUR-WS.org
16. Stevens, R., Aranguren, M.E., Wolstencroft, K., Sattler, U., Drummond, N., Horridge, M., Rector, A.L.: Using OWL to model biological knowledge. International Journal of Man-Machine Studies 65(7), 583–594 (2007)
17. Sváb-Zamazal, O., Svátek, V., Scharffe, F.: Pattern-based Ontology Transformation Service. In: Dietz, J.L.G. (ed.) KEOD, pp. 42–47. INSTICC Press (2009)
18. Corcho, O., Fernández-López, M., Gómez-Pérez, A.: Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering 46(1), 41–64 (2003)
19. Mizoguchi, R., Kozaki, K.: Ontology Engineering Environments. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, 2nd edn. Springer, Heidelberg (2009)
20. Dzbor, M., Motta, E., Buil, C., Gomez, J.M., Görlitz, O., Lewen, H.: Developing Ontologies in OWL: an Observational Study. In: Proc. of the OWLED 2006 Workshop on OWL: Experiences and Directions, vol. 216 (2006), CEUR-WS.org
21. Gruninger, M., Fox, M.S.: The role of competency questions in enterprise engineering. In: Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice (1994)
22. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Modelling Ontology Evaluation and Validation. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 140–154. Springer, Heidelberg (2006)
23. Hartmann, J., Sure, Y., Haase, P., Palma, R., del Carmen Suárez-Figueroa, M.: OMV – Ontology Metadata Vocabulary. In: Welty, C. (ed.) Ontology Patterns for the Semantic Web Workshop, Galway, Ireland (2005)