Reusing Ontology Design Patterns in a Context Ontology Network

María Poveda-Villalón, Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez
Ontology Engineering Group. Departamento de Inteligencia Artificial.
Facultad de Informática, Universidad Politécnica de Madrid.
Campus de Montegancedo s/n.
28660 Boadilla del Monte. Madrid. Spain
mpoveda@delicias.dia.fi.upm.es, {mcsuarez, asun}@fi.upm.es

Abstract. Reusing knowledge resources, specifically Ontology Design Patterns (ODPs), has became a popular technique within the ontology engineering field. Such a reuse allows speeding up the ontology development process, saving time and money, and promoting the application of good practices. Recently methods and tools to support the reuse of ODPs have emerged. In addition, the existence of detailed examples of real use cases that reuse ODPs favours the adoption and application of such methods. Thus, our objective in this paper is to show an example of how to apply a method for reusing ODPs during the development of a context ontology network to model context-related knowledge that allows adapting applications based on user context. Besides, in this paper we present the main drawbacks found during the application of the reuse method as well as some proposals to overcome them.

Keywords: ontology design pattern, ontology development, context ontology

1 Introduction

With the goal of speeding up the ontology development process, ontology practitioners are starting to reuse [11, 12] as much as possible knowledge resources (ontologies, ontology modules, ontology statements, and ontology design patterns as well as non-ontological resources). This approach also allows ontology developers to save time and money and to promote the application of good practices.

Regarding the ODPs, their reuse has proven different benefits [1] like (a) to make easier the ontology development or (b) to produce ontologies of better quality. The reuse of ODPs should be supported by methods and tools to minimize the reuse effort and to maximize their benefits. In this sense, methods and tools to support the reuse of ODPs have recently emerged.

However, methods and tools are not enough to guarantee a successful reuse of ODPs. In this regard, the existence of detailed examples of how ODPs are reused in real use cases would be of great help. This kind of guided examples would favour the adoption of ODPs and their related methodological and technological support.

Thus, our main objective in this paper is to present a guided example of how we have carried out the reuse of ODPs during the development of an ontology network about user's contextual information, called mIO! ontology network. This ontology

network is a context ontology in the mobile environment that aims to represent contextual knowledge about the user that can influence his interaction with mobile devices. In addition, we also show in this paper some lessons learned during the application of the method for reusing ODPs as well as some proposals for overcoming the drawbacks found in the method.

The remainder of the paper is structured as follows: Section 2 summarizes the state of the art on methodological and technological aspects of the ODPs reuse. Section 3 describes in detail how we have followed the guidelines to reuse ODPs as well as how we have adapted such guidelines during the development of the mIO! ontology network. Section 4 shows the lessons learned during the ODPs reuse. Finally, Section 5 concludes and shows some lines of future work.

2 Ontology Design Patterns and how to reuse them

The idea of applying patterns for modelling ontologies was proposed by [2]. Since then, relevant works on patterns have been the Semantic Web Best Practices and Deployment Working Group[1], the Ontology Design Patterns Public Catalogue[2], and the Ontology Design Patterns Portal[3]. According to [4] Ontology Design Patterns (ODPs) are modeling solutions to solve a recurrent ontology design problem. ODPs are a way of encoding best practices, based on experiences and knowledge of 'good' solutions. In [5], authors distinguish the following six different types of ODPs: reasoning, structural, content, presentation, lexico-syntactic, and correspondence.

Generally patterns are perceived as having three kinds of benefits [5]: (1) reuse benefits, (2) guidance benefits, and (3) communication benefits. The ontology design patterns reuse refers to the activity of using available ontology design patterns in the solution to different modelling problems during the development of new ontologies [15]. The goal of the ODPs reuse is to facilitate the solution of modelling issues and to improve interoperability through using well-proven solutions and best practices, in the form of patterns. In this sense, there are experiments that prove that the reuse of ODPs makes the ontology development process easier and improves the quality of the resulting ontologies [1].

Even when such benefits have been established, there is the need to provide methodological and technological support to the pattern usage with the goal of (a) minimising the reuse effort and (b) maximising the ODPs benefits. In this regard, methods and tools have recently appeared to guide and support the ODPs reuse.

On the one hand, a method for reuse any type of ODP, called XD (for eXtreme Design), is presented in [3, 9]. In this method the ODPs reuse activity is divided into eight tasks (1. Identify requirements to be addressed; 2. Identify available patterns; 3. Divide and transform the problem, select a partial problem; 4. Match selected partial problem to patterns; 5. Select patterns; 6. Apply (reuse) selected patterns and compose them; 7. Evaluate and revise with respect to partial problem; and 8. Integrate partial solutions). The method has as input the Ontology Requirements Specification

[1] http://www.w3.org/2001/sw/BestPractices/OEP/
[2] http://www.gong.manchester.ac.uk/odp/html/index.html
[3] http://ontologydesignpatterns.org

Document (ORSD) and a set of available ODPs, and as output an ontology network including the ODPs reused. In summary, the workflow starts with the identification of the requirements to be addressed by reusing ODPs and the available ODPs registries and libraries. Next, an iterative set of tasks takes place in which (a) each problem is divided into sub-problems, (b) a sub-problem is selected and matched with the ODPs, (c) the ODPs are selected and applied, and (d) the result is evaluated. This cycle is repeated until all the sub-problems are addressed. Finally, the method concludes by integrating all the partial solutions resulting in an ontology network which contains the ODPs selected. The XD method could be specialized to address the reuse of concrete types of ODPs, for example, content patterns [3, 9, 13].

It is well known that the adoption of any emerging methodology or method is facilitated by providing detailed examples of how to apply them to real and complete use cases. However, as far as we know, there are no complete examples of how to apply the XD method to the reuse of any type of ODP. Thus, we present in this paper a detailed example of how we have reused different types of ODPs during the development of a context ontology network.

On the other hand, there are also tools that provide a technological support for the ODPs reuse activity. The plug-in for NeOn Toolkit[4] called "XD Tools"[5] allows the content ODPs reuse accordingly to this variant of the XD method. Whereas, the ontology editor Protégé 3.4.4[6] allows the automatic reuse[7] of the following logical patterns and good practices: "Value Partition", "Enumeration", "N-ary Relation", "OWL List", and "RDF List".

3 ODPs Reuse in the mIO! Ontology Network

As already mentioned, the main purpose of this context ontology, called mIO! ontology network, is to represent knowledge related to the user context, e.g., information on location and time, user information and its current or planned activities, as well as devices located in his surroundings. The ontology aims at solving the challenge of configuring, discovering, executing, and enhancing services based on the user context.

The development of this ontology network about contextual information has been carried out following the NeOn Methodology [12]. During the development of such an ontology network we have reused different knowledge resources (ontologies, non-ontological resources, and ODPs) as explained in [8]. In this paper, we focus exclusively on how we have performed the reuse of ODPs.

To reuse ODPs we have applied the general method described in [13] rather than the one explained in [9], since we are not interested exclusively in content patterns but in any type of ODP. For the same reason we have manually put into practice method instead of using the XD Tools. It is important to add that in some cases during the application of the method we have had to adapt and/or extend the guidelines provided.

[4] http://neon-toolkit.org
[5] http://neon-toolkit.org/wiki/XDTools
[6] http://protege.stanford.edu/download/registered.html#p3
[7] This reuse is performed in a guided way by means of a wizard.

Thus, in this section we describe (a) how we have applied the general XD method[8] for reusing different types of ODPs and (b) how we have adapted and/or extended the guidelines provided by the method.

Task 1. Identify requirements to be addressed

To identify the requirement to be addressed by means of the reuse of ODPs, we have chosen the third question proposed in [13] based on our knowledge about ODPs. The other two questions proposed in [13] could result in selecting requirements that cannot be solved by reusing ODPs and for this reason we have discarded such questions. Therefore, the selected question is *"What requirements can be associated with existing patterns types?"*.

The main drawback at this point is the fact that to be able to answer this question developers should be versed in ODPs types. In this sense, based on our knowledge about ODPs, we have not selected those requirements that could be addressed by existing ODPs types but those that apparently can be solved by reusing existing ODPs.

Table 1 shows both the non-functional and the functional requirements extracted from the ORSD[9] that will be addressed by reusing ODPs. The functional requirements belong to the following subdomains as we can observe in the identifier[10] field: Interface (INT), Device (DSP), Time (TMP), Location (LOC), and User (USR). These functional requirements can be written both as Competency Questions (CQs) and their corresponding responses [6] and as affirmative sentences in natural language (NL) as explained in [8].

Table 1. Requirements to be addressed by reusing ODPs

Non-functional requirements	
• The ontology should be modular.	
Functional requirements	
Requirement identifier	**Requirement**
CQ identifier	**CQ and its response**
DSP_PC7	What devices exist? Display, touchscreen, balise, keyboard, trackball, pulse oximeter, glucose meter, emvironmental temperature sensor, environmental humidity/pressure sensor, Anemometer, CO2 level sensor, printer, camera, GPS Receiver, short distance communication module (NFC. ZigBee, Bluetooth), long distance communication module (GPRS, UMTS, WiFi), processing module, memory module, loudspeaker, microphone and reading/writing module NFC
DSP_PC9	What are the components of a display? A display is composed by: - input interfaces - presentation surface - control interfaces - power system

[8] To simplify the example description, we present a unique process addressing simultaneously all the requirements.

[9] The whole ORSD for the mIO! ontology network is available in [7].

[10] The abbreviations come from the Spanish name of each subdomain: *Intefaz (INT), Dispositivo (DSP), Tiempo (TMP), Localización (LOC),* and *Usuario (USR).*

CQ identifier	CQ and its response
DSP_PC49	What are the components of a CO2 level sensor? A CO2 level sensor is composed by: - source - power system - detector - amplifier - output /download data port (optional)
DSP_PC61	What are the components of a printer? A printer is composed by: - leaf storage receptacle - printhead - ink container - storage device - processing device - communication interface - screen
DSP_PC71	What are the components of a camera? A camera is composed by: - lens - image capture device - image processing device - storage device - communication interface - positioning device - lighting device - screen - microphone
DSP_PC83	What are the components of a GPS receiver? A GPS receiver is composed by: - antenna - signal processor - processing module - communication interface - screen - speaker - microphone
DSP_PC96	What are the components of a speaker? A speaker is composed by: - communication interface - amplifier - decoder - signal processing module - active element - casing
DSP_PC102	What are the components of a microphone? A microphone is composed by: - diaphragm - coil - permanent magnet A condenser microphone is composed by: - diaphragm of lightweight and flexible membrane - rigid backplate - cable to the preamplifier - bias voltage feeder
TMP_PC4	What are the days of the week? Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday
LOC_PC14	When was the last time that the user was in the location X? The day Z from xx: xx: xx to yy: yy: yy
NL sentence identifier	**Affirmative NL Sentence**
INT_CA3	The interfaces types are: - conversational - gestural - graphic - natural language - command line - multi screen - touch - textual - vocal - web
USR_CA7	A user will be in a specific location at any given time. The possible physical movement of the user is associated with this aspect

Task 2. Identify available patterns

To carry out this task, first we have identified different libraries and repositories in which patterns could be found. Next, based on our knowledge about ODPs and on

their descriptions[11] and uses we have sketched preliminary correspondences between the requirements selected in Task 1 and the available ODPs. Note that there are patterns that belong to several libraries or repositories.

The ODPs resources and the ODPs suitable to be reused are:

- **Ontology Design Patterns (ODP) Portal**[12]: this Wiki aims to gather ODPs in a repository. The patterns are reviewed by a quality committee. Besides, users can submit new patterns or modelling issues. In this portal we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - Componency
  - N-Ary Participation
- **W3C Semantic Web Best Practices and Deployment (SWBPD)**[13]**:** this repository contains pattern descriptions and some examples of modelling patterns. The repository also includes other types of best practices as guidelines, repositories of vocabularies and ontologies, learning material, and demos. In this repository we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - N-ary Relations. Pattern 1: Introducing a new class for a relation
  - Specified Values in OWL: "value partitions" and "value sets"
- **NeOn D2.5.1 [10]:** *A Library of Ontology Design Patterns*: *reusable solutions for collaborative design of networked ontologies*. This catalogue contains content patterns classified by domains**.** In this resource we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - Componency (CP-COM-01)
  - N-Ary Participation (CP-NPAR-01)
- **NeOn D5.1.1 [14]**: *NeOn Modelling Components.* This catalogue provides descriptions about logical, architectural, and some content patterns. In this resource we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - Modular Architecture (AP-MD-01)
  - Taxonomy (AP-TX-01)
  - Part-Whole Relation (CP-PW-01)
  - Specified Values: Set of Individuals (LP-SV-01)
  - Specified Values: Subclasses (LP-SV-02)
  - N-ary Relation: New Class (LP-NR-01)

To illustrate how we have mentally sketched a correspondence between requirements and possible ODPs we present here some examples.

Let us take the DSP_PC9 requirement as the starting point. We can observe that in the requirement appear the terms “components”, “composed by” and a list of the parts

---

[11] These descriptions usually include (a) the name of the pattern, (b) a graphical description, (c) an enumeration of its elements, and (d) a set of requirements (in the form of CQs) or use cases (in the form of sentences in natural language) the pattern is intended to address. Sometimes, the code in an ontology implementation language is also provided.

[12] http://ontologydesignpatterns.org/

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

that form a display. These characteristics of the requirement are clues to think about the "Part-Whole Relation" or "Componency" patterns due to the information included in the description and uses cases of these patterns. The same reasoning can be applied to the requirements DSP_PC49, DSP_PC61, DSP_PC71, DSP_PC83, DSP_PC96, and DSP_PC102.

In the case of the requirement TMP_PC4 the days of the week are enumerated. It is well known that this is a unique enumeration and the days of the week are different among them. These features can lead a developer to think in the "Specified Values: Set of Individuals" and the "Specified Values: Subclasses" patterns.

Task 3. Divide and transform the problem, select a partial problem

At this point the method [13] suggests transformations like (a) writing CQs to represent requirements stated only in example scenario sentences if not already present in the ORSD and (b) grouping similar CQs that may be solved together.

In our case, the selected requirements are quite simple so it was not necessary to divide them. In fact, some of them have been grouped during the transformation because of their similarity. Besides, we have only transformed the functional requirements since the non-functional one represents a "manageable piece" itself.

To perform the transformation of the problem we propose the following approach. We have taken into account that the functional requirements selected in Task 1 can be written as CQs and their responses or as affirmative sentences in natural language as Table 1 shown. In addition, we have observed that some suitable patterns to be reused (e.g. the Taxonomy (AP-TX-01) pattern [14]) have as part of their descriptions general use cases in form of sentences in natural language instead of CQs. Bearing in mind all these facts, we have transformed the requirements depending on the descriptions fields of the patterns suitable to be reused. Therefore, some CQs have been transformed into sentences in natural language and vice-versa. Others requirements have been transformed according to the suitable patterns descriptions but maintaining their original format. Table 2 summarizes the transformation types proposed in our approach and shows the specific transformation cases that have occurred during the mIO! use case. Finally, the resulting transformations are shown in Table 3.

Table 2. Requirement transformation cases

		The requirement was transformed into:	
		Sentence in NL	**Competency Question**
The requirement was written as:	**Sentence in NL**	INT_CA3	USR_CA7
	Competency Question	DSP_PC7 TMP_PC4	DSP_PC9 DSP_PC49 DSP_PC61 DSP_PC71 DSP_PC83 DSP_PC96 DSP_PC102 LOC_PC14

It should be noted that there are no guidelines to carry out the requirements transformation. In our case, we have carried out the transformation based on how general are the original requirements in the ORSD. Having in mind this approach, we have abstracted the requirements in order to make them similar to the ODPs descriptions. For example, in the requirements DSP_PC9, DSP_PC49, DSP_PC61, DSP_PC71, DSP_PC83, DSP_PC96, and DSP_PC102 a list of components is shown for each concrete device. During the transformation, we have abstracted these requirements writing them as a unique CQ that could be applied to any of them and that could be match with the CQs addressed by some available patterns.

Table 3. Transformation of subproblems

Requirement identifier	Transformation
INT_CA3	Perform categorization/classification of interface types at different extents of granularity.
DSP_PC7	Perform categorization/classification of devices at different extents of granularity.
DSP_PC9	What are the components of this device? *Response*: A list containing the parts of the device.
DSP_PC49	
DSP_PC61	
DSP_PC71	
DSP_PC83	
DSP_PC96	
DSP_PC102	
TMP_PC4	List the days that make up the week.
LOC_PC14	What is the location of a particular user at a particular time? *Response*: A structure containing the location of the user in a given point of time.
USR_CA7	

It is important to mention here that there are two important drawbacks at this point of the reuse. First, we cannot transform the requirements only into CQs because there are ODPs that only have in their descriptions use cases written as sentences in natural language, as we have already noted. Second, ontology developers should be well versed in ODPs to discern what type of transformation they should carry out. Therefore, we have realized the high importance of making a finer identification of the most suitable ODPs in Task 2.

Task 4. Match selected partial problem to patterns

As it can be observed in Table 3 none of the requirements has been divided into sub-problems, but they have been simply transformed. Therefore, it was not necessary to assign a new numbering for the partial problems since they correspond to the identifier of each requirement.

The matching between the problems to be addressed by reusing ODPs and the ODPs available in libraries and repositories, identified in Task 2, is shown in Table 4.

Table 4. Matching between partial problems and ODPs suitable to be reused

Non-functional requirements		
Requirement		**Suitable pattern(s)**
• The ontology should be modular.		• Modular Architecture (AP-MD-01)
Functional requirements		
Requirement identifier	**Transformation**	**Suitable pattern(s)**
INT_CA3	Perform categorization/classification of interface types at different extents of granularity.	• Taxonomy (AP-TX-01)
DSP_PC7	Perform categorization/classification of devices at different extents of granularity.	• Taxonomy (AP-TX-01)
DSP_PC9	What are the components of this device?	• Componency • Componency (CP-COM-01) • Part-Whole Relation (CP-PW-01)
DSP_PC49		
DSP_PC61		
DSP_PC71		
DSP_PC83		
DSP_PC96		
DSP_PC102		
TMP_PC4	List the days that make up the week.	• Specified Values in OWL: "value partitions" and "value sets" • Specified Values: Set of Individuals (LP-SV-01) • Specified Values: Subclasses (LP-SV-02)
LOC_PC14	What is the location of a particular user at a particular time?	• N-Ary Participation • N-ary Relations. Pattern 1: Introducing a new class for a relation • N-ary Participation (CP-NPAR-01) • N-ary Relation: New Class (LP-NR-01)
USR_CA7		

It is important to mention that at this point the method does not provide detailed guidelines about how to match the requirements with the available patterns. As we have explained in Task 1, we have selected those requirements that could be covered by at least one available ODP. Based on our knowledge about ODPs and on our experience reusing them, we can propose the following heuristics to quickly identify a suitable ODP to be reused while reading the requirements:

- If a requirement mentions something about a list of values, we could infer that the “Specified Values in OWL: "value partitions" and "value sets"”, the “Specified Values: Set of Individuals (LP-SV-01)” or the “Specified Values: Subclasses (LP-SV-02)” patterns could be reused.
- If a requirement is about types and subtypes of a given concept, we could infer that the “Taxonomy (AP-TX-01)” pattern could be reused.
- If a requirement contains more than two concepts related, we could infer that the “N-Ary Participation”, “N-ary Relations. Pattern 1: Introducing a new class for a relation”, “N-ary Participation (CP-NPAR-01)” or the “N-ary Relation: New Class (LP-NR-01)” patterns could be reused.

- If a requirement is about parts of something, we could infer that the "Componency", "Componency (CP-COM-01)" or "Part-Whole Relation (CP-PW-01)" patterns could be reused.
- If a requirement is about the need for modularity within the ontology, we could infer that the "Modular Architecture (AP-MD-01)" pattern could be reused.

Task 5. Select patterns

At this point, we must select the most appropriate pattern in those cases identified in Task 4 in which there are several suitable patterns to cover the requirements. Since there are no detailed guidelines to perform this task, we have taken the following decisions for requirements related to more than one potential ODP:

- For the requirements DSP_PC9, DSP_PC49, DSP_PC61, DSP_PC71, DSP_PC83, DSP_PC96, and DSP_PC102, there is no need to represent transitive relationships. For this reason, we have selected the "Componency (CP-COM-01)" pattern instead of the "Part-Whole Relation (CP-PW-01)" pattern. It is important to mention that we have reused the "Componency" pattern from the ODP Portal instead of the one included in the NeOn D2.5.1 [10], because the ODP Portal provides an OWL file that contains the source code for the pattern.
- For the TMP_PC4 requirement is needed to represent a set of individuals whose enumeration is equivalent to the parent class. Therefore, we have selected the "Specified Values: Set of Individuals (LP-SV-01)" pattern[14] instead of the "Specified Values: Subclasses (LP-SV-02)" pattern.
- For the requirements LOC_PC14 and USR_CA7 there is no need to represent events or situations. For this reason, we have selected the "N-ary Relation: New Class (LP-NR-01)" pattern[15] instead of the "N-ary Participation" pattern or the "N-ary Participation (CP-NPAR-01)" one.

It is worth mentioning that for the non-functional requirement and for the requirements INT_CA3 and DSP_PC7, the ODPs related to those requirements in Table 4 have been directly selected.

Task 6. Apply (reuse) selected patterns and compose them

In this task we have observed that ODPs can take different roles in different stages of an ontology development process depending on their types and the problem that they address. For example, we have distinguished the following situations:

- Reusing ODPs to define the architecture of the ontology network during the conceptualization activity through the "Modular Architecture (AP-MD-01)" pattern.
- Reusing ODPs in the implementation activity to complete the knowledge represented; for example, we can enrich a mereological relationship through the "Componency (CP-COM-01)" pattern.

[14] In this case the use of the "Specified Values: Set of Individuals (LP-SV-01)" pattern and the "Specified Values in OWL: "value partitions" and "value sets"" patterns would be equivalent.

[15] In this case the use of the "N-ary Relation: New Class (LP-NR-01)" pattern and the "N-ary Relations. Pattern 1: Introducing a new class for a relation" patterns would be equivalent.

- Reusing ODPs in the implementation activity to represent logical structures that are not supported by the ontology language, for example, the "N-ary Relation: New Class (LP-NR-01)" pattern.
- Reusing ODPs in the implementation activity to join concepts from different ontologies, for example, thorough the "N-ary Relation: New Class (LP-NR-01)" pattern.

In addition, here we show the result of applying the patterns selected in Task 5 to the mIO! ontology network[16].

The reuse of the "Modular Architecture (AP-MD-01)" pattern has given rise to the modular structure that forms the miO! ontology network. Such a structure is shown in Fig. 1[17].

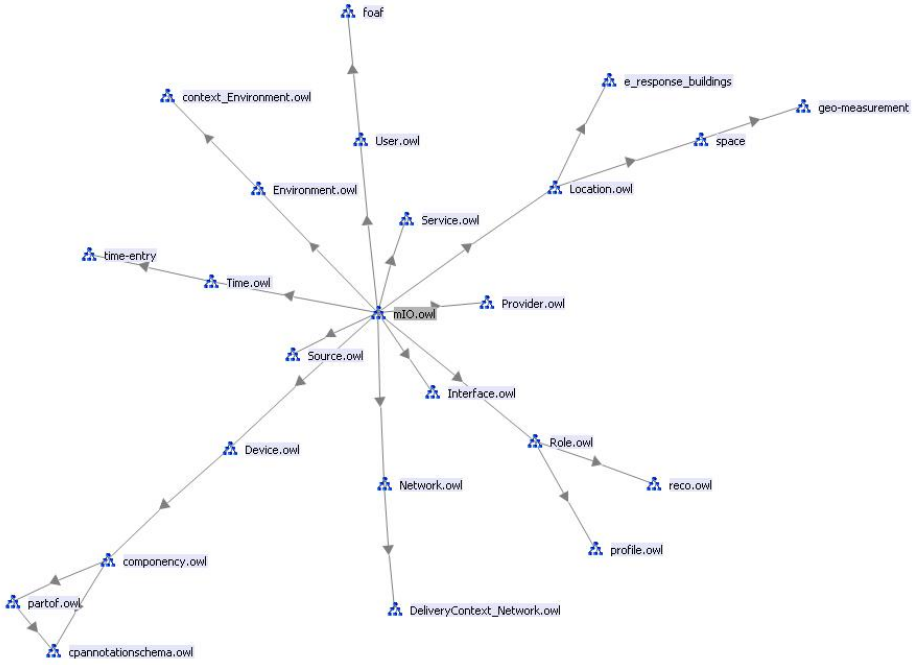


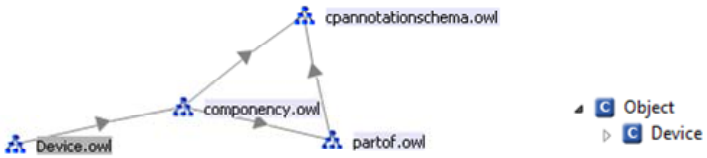
Fig. 1. "Modular Architecture" pattern applied to the mIO! ontology network

The reuse of the "Componency (CP-COM-01)" pattern has been carried out within the Device.owl ontology by importing (See [10] for more information about this operation) (Fig. 2-a) the componency.owl pattern and specializing (See [10] for more information about this operation) (Fig. 2-b) the pattern.

---

[16] The screenshots shown have been taken from NeOn Toolkit's default tabs and the "Relationship Browser" (http://www.neon-toolkit.org/wiki/2.3.1/Relationship_Browser) plug-in.

[17] The blue dotted triangles represent ontologies, whereas the arrows with grey solid triangles represent the "import" relationship between ontologies. These relationships must be understood as follows: the mIO.owl ontology imports the Service.owl one.



(a) Import (b) Specialization

Fig. 2. "Componency" pattern applied to Device subdomain

The result of reusing the "Specified Values: Set of Individuals (LP-SV-01)" pattern during the modelling of the days of the week is shown in Fig. 3[18].

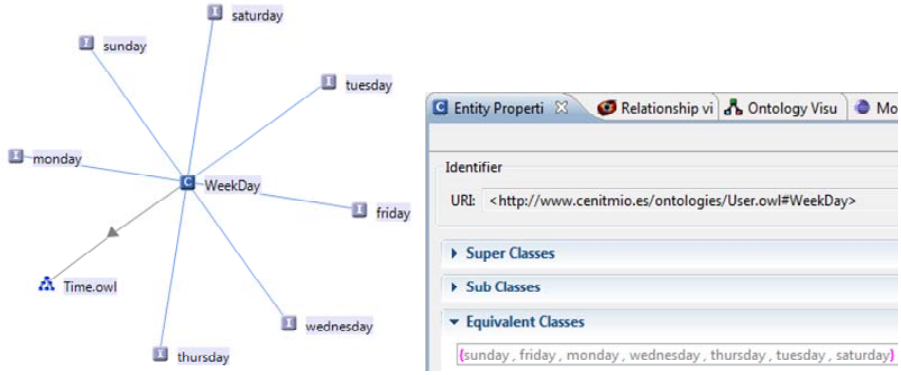


Fig. 3. "Specified Values: Set of Individuals" pattern applied to the days of the week

Fig. 4[19] depicts the result of reusing the "N-ary Relation: New Class (LP-NR-01)" pattern to model the location of a user at given point of time. In the model shown in Fig. 4 has been taken into account both locations in a geo-political entity, such as a country or a city, and locations specified by coordinates.

Finally, it is worth mentioning that Task 7 (Evaluate and revise with respect to partial problem) and Task 8 (Integrate partial solutions) have not been carried out as authors propose in [13] since none of the requirements has been divided into sub-problems. In addition, the reuse of patterns has been carried out by a single development team so there is no need to integrate solutions developed in parallel by different teams. Finally, the evaluation of the obtained model has been carried out during the evaluation of the whole ontology network.

[18] The days of the week are represented by boxes with an "I" to indicate that they are instances, whereas the class "WeekDay" is represented by a box with a "C". In addition the lines that link the class with the instances indicate that they belong to the class.

[19] The ellipses represent classes and the lines with a triangle represent relationships among classes.

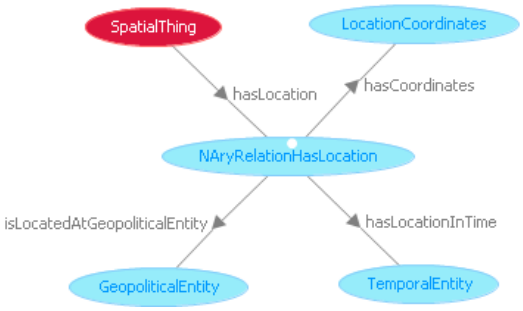


Fig. 4. "N-ary Relation: New Class" pattern applied to locations at given point of time

4 Lessons Learned

After the application of the general XD method to reuse ODPs during the development of the mIO! ontology network, we have realized (a) the usefulness of following a method to guide the ODPs reuse during the ontology building and (b) the advantages of reusing ODPs, ensuring the use of good practices in the ontology.

However, we have also realized the difficulty of applying the abovementioned method because of the lack of detailed guidelines in some of the tasks. For this reason in this section we report (a) some of the lessons we have learned during the application of the method as well as (b) some proposals that could be valuable for any enhancement of the method and/or for any further development that reuses ODPs.

During the execution of *Task 1*, we have realized that a beginner could select requirements that cannot be solved by means of reusing ODPs when he/she answers the questions proposed in [13] for such a task. Thus, ontology developers must have some experience with ODPs to make a more direct identification of the requirements that will be addressed. Such an experience is needed to select only those requirements that can be fulfilled by reusing ODPs.

We can also add that in those cases in which a beginner selects requirements that cannot be solved by means of ODP reuse, such a developer would have at least the following two options: (a) to propose a new pattern to cover such requirements and (b) to cover such requirements with other knowledge resources (such as ontologies or non-ontological resources) as proposed in [12, 16].

In the case of *Task 2*, we also consider that experience with ODPs is required again. In this task some types of patterns, as content patterns, can be identified as suitable to be reused by means of tools (e.g., XD Tools); however, there are other types of patterns that cannot be identified using tools due to (a) such patterns have no CQs in their description to match with the requirements to be addressed or (b) the patterns are not available at on-line libraries. Thus, in these cases, the ontology developer should have large knowledge about both ODPs and repositories to identify manually the patterns.

Besides, in this task it could happen that for a given requirement there are no patterns related or that the developers cannot find patterns suitable to be reused. In that case, we propose the following alternatives that can be included in the method:

(a) to posting a modelling issue[20] within the ODP Portal related to the given requirement; (b) to look for others resources suitable to be reused such as ontologies or non-ontological resources as explained in [12] and [16] respectively; and (c) to manually face the problem and to submit a proposed pattern[21] that covers the given requirement to the ODP Portal.

To carry out *Task 3* the method [13] suggests transforming the requirements (written as example scenario sentences) into CQs. However, in our case we have also had to transform some of the requirements into sentences in natural language. This transformation was needed to match the requirements to be addressed with some ODPs use cases, as already explained in Section 3.

Regarding to *Task 4*, based on our experience applying the method, this task only seems necessary if at least one requirement has been divided in Task 3. In other cases, Task 4 can be considered similar to Task 2.

We can also mention that after performing *Task 5* it is possible that no pattern matches with the problem to be addressed. In this case, we propose to follow the same options already presented for Task 2.

Finally, we have observed that ODPs can be applied at different points of an ontology development. For example, in the mIO! ontology network case the "Modular Architecture (AP-MD-01)" pattern was applied during the conceptualization activity whereas the rest of patterns were applied during the implementation activity.

5 Conclusions and Future Lines of Work

This paper presents an example of how to apply the general XD method to reuse different types of ODPs (logical, architectural, and content patterns). This application has been performed with a real use case within the development of a context ontology network, called mIO! ontology network[22]. This guided example could be used together with the method in further ontology developments, what would favour the adoption of ODPs and of the abovementioned method.

During the process we have taken advantage of following a guided method that sets and orders the tasks to carry out during the reuse of ODPs. The method also provides some examples or criteria to take into account as well as a list of catalogues where to find ODPs. Once the method was applied, we have realized that time was saved in the conceptualization and implementation activities.

However, we have also identified some points during the reuse process where the developers' experience on ODPs seems quite important (tasks 1 and 2). In addition, we have discovered some drawbacks in the general XD method that could be solved by extending and improving the guidelines for tasks 1, 2, 3, and 5, as already explained in Section 4.

As future work we have plan to apply the XD method together with the XD Tools in a collaborative ontology development within a real use case. The idea of this new application of the method is to focus on requirements that have to be divided into

[20] http://ontologydesignpatterns.org/wiki/Community:PostModelingIssue
[21] http://ontologydesignpatterns.org/wiki/Submissions:SubmitAPattern
[22] http://www.oeg-upm.net/index.php/es/ontologies/82-mio-ontologies

different sub-problems. Our final aim is to analyze (a) what tasks the XD Tools make easier and (b) what tasks still need more detailed guidelines.

Acknowledgments. This work has been partially supported by the Spanish project *mIO!* (CENIT-2008-1019).

References

1. Blomqvist, E., Gangemi, A., Presutti, V. *Experiments on Pattern-based Ontology Design*. In Proceedings of K-CAP 2009, pp. 41-48. 2009.
2. Clark, P., Thompson, J., & Porter, B. W. *Knowledge Patterns*. In KR2000: Principles of Knowledge Representation and Reasoning. pp. 591-600. 2000.
3. Daga, E., Blomqvist, E., Gangemi, A., Montiel, E., Nikitina, N., Presutti, V., Villazón-Terrazas, B. *NeOn D2.5.2 Pattern based ontology design: methodology and software support.* NeOn project. http://www.neon-project.org. 2010.
4. Gangemi, A., Gomez-Perez, A., Presutti, V., Suarez-Figueroa, M.C. *Towards a Catalog of OWL-based Ontology Design Patterns*. In proceedings of CAEPIA. 2007.
5. Gangemi, A.; Presutti, V. *Ontology Design Patterns*. Handbook on Ontologies (Second Edition). Springer. International Handbooks on Information Systems. 2009.
6. 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, Trondheim, Norway, 1994.
7. Poveda, M. *Metodología NeOn Aplicada a la Representación del Contexto*. Master Thesis. Spain. *Universidad Politécnica de Madrid*. September, 2010.
8. Poveda, M., Suárez-Figueroa, M.C., García-Castro, R., Gómez-Pérez, A. *A Context Ontology for Mobile Environments*. Proceedings of CIAO 2010. Lisbon, Portugal. 11 October 2010.
9. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E. *eXtreme Design with Content Ontology Design Patterns*. In Proceedings of WOP 2009. Washington D.C., USA, 25 October, 2009, Vol. 516 CEUR Workshop Proceedings. 2009.
10. Presutti, V., Gangemi, A., David S., Aguado de Cea, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M. *NeOn D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies.* NeOn project. http://www.neon-project.org. 2008.
11. Simperl, E. *Reusing ontologies on the Semantic Web: A feasibility study*. Data Knowledge Engineering. Volume 68. Number 10. Pages: 905-925. 2009.
12. Suárez-Figueroa, M.C. *PhD Thesis: NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. Spain. Universidad Politécnica de Madrid. June 2010.
13. Suárez-Figueroa, M.C., Blomqvist, E., D'Aquin, M., Espinoza, M., Gómez-Pérez, A., Lewen, H., Mozetic, I., Palma, R., Poveda, M., Sini, M., Villazón-Terrazas, B., Zablith, F., Dzbor, M. *NeOn D5.4.2: Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks*. NeOn project. http://www.neon-project.org. February 2009.
14. Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V., Sabou, M.. *NeOn D5.1.1: NeOn Modelling Components*. NeOn project. http://www.neon-project.org. March 2007.
15. Suárez-Figueroa, M.C., Gómez-Pérez, A. *First Attempt towards a Standard Glossary of Ontology Engineering Terminology*. 8th Proceedings of TKE 2008. 18-21 August 2008.
16. Villazón-Terrazas, B. *PhD Thesis: A Method for Reuse and Re-engineering Non-Ontological Resources into Ontologies*. Spain. *Universidad Politécnica de Madrid*. To be appeared.