An ISO-based Software Process Ontology Pattern Language and its Application for Harmonizing Standards

Fabiano B. Ruy, Ricardo A. Falbo, Monalessa P. Barcellos,
Giancarlo Guizzardi and Glaice K. S. Quirino

[1] Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo, Vitória, Brazil
+55-27-4009-2167
{fabianoruy, falbo, monalessa, gguizzardi, gksquirino}@inf.ufes.br

ABSTRACT

Many efforts have been made for modeling and standardizing software processes. ISO/IEC JTC1/SC7, the ISO sub-committee responsible for software and systems engineering, is one of the most important groups devoted to this task. However, standards developed by this committee are frequently inconsistent and even contradictory. This led to the need for an ISO Study Group to investigate the creation of an ontological infrastructure to establish a common conceptualization for underpinning all SC7 standards. This ISO initiative is a work in progress, which has focused on the software process domain and, in particular, considering the ISO/IEC 24744 standard. In this paper, we advocate in favor of using an Ontology Pattern Language (OPL) as the main component of this ontological infrastructure. We present ISP-OPL (ISO-based Software Process OPL), an OPL that can be applied as a basis for harmonizing software process-related ISO standards, favoring reuse when building aligned specific software process ontologies for Software Engineering sub-domains. In order to evaluate its applicability, we conducted an experiment involving seven domain ontologies, developed using ISP-OPL.[1]

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – *software process models*.
D.2.13 [**Software Engineering**]: Reusable Software – *reuse models*.
K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software process*.

General Terms

Design, Experimentation, Standardization, Languages.

Keywords

Ontology development, ontology patterns, ontology pattern language, standards harmonization, software process.

1. INTRODUCTION

A permanent challenge in Software Engineering (SE) is to deal with quality aspects, improving the resulting products with higher productivity and lower costs. Since the quality of a software product depends heavily on the quality of the software process used to develop it, software organizations are investing more and more in improving their software processes. In this context, several process-related quality standards and maturity models, such as ISO/IEC 12207 [11], ISO/IEC 15504 [14], and CMMI [28], are used to guide software organizations efforts towards quality software processes. These initiatives attempt software process improvement by means of disseminating best practices in an organized and standardized way. However, most of the models and standards are created independently, without necessarily sharing the same semantics. This frequently gives rise to inconsistencies between them. This problem is amplified when different standards are used together, causing semantic interoperability problems [24]. Even standards proposed by the same standardization organization present this problem. This is due to the fact that each standard defines its own scope, structure of process entities, terms and definitions, amongst other things [22].

The International Organization for Standardization (ISO) recognizes this problem. SE standards developed by ISO/IEC JTC1's SC7 (the ISO sub-committee responsible for software and systems engineering standards) frequently employ terms whose definitions vary significantly across standards. In order to treat this problem, ISO created, in 2012, a study group to develop an ontological infrastructure aiming to be a single coherent underpinning for all the SC7 standards [9]. The goal is to establish a basic set of definitional ontologies, which can be used to derive more specific ontologies. These specific ontologies are meant to address different SE sub-domains (e.g., Software Testing), which in turn are the subject of specific SC7 standards (e.g., ISO/IEC/IEEE 29119 [18]) [9]. The ISO initiative is a work in progress, which has focused on the definitional ontologies, taking mainly ISO/IEC 24744 [16] into account. The goal is to develop a Definitional Elements Ontology (DEO) and an aligned Configured Definitional Ontology (CDO) based on ISO/IEC 24744, which could be extended for building Standard Domain Ontologies (SDOs).

We argue that this basic set of definitional ontologies (DEO and CDO) should be represented as core ontologies on software processes, from which the more specific SDOs could be derived. According to [27], a core ontology provides a precise definition of structural knowledge in a specific field that spans across different application domains in this field. Moreover, we argue that, by following a pattern-oriented approach, a core ontology can systematically become more modular and extensible [5].

A core ontology for the ISO harmonization initiative should be: (i) flexible enough for allowing ontology engineers to explore alternative models in the design of specific ontologies for the

[1]Copyright is held by the authors. This work is based on an earlier work: SAC'15 Proceedings of the 2015 ACM Symposium on Applied Computing, Copyright 2015 ACM 978-1-4503-3196-8. http://dx.doi.org/10.1145/2695664.2695796

various software process sub-domains; (ii) modular, in order to allow the ontology engineer to select the ontology fragments relevant to the problem at hands and then reuse it; and (iii) broad enough to cover the general concepts in the software process universe of discourse. For achieving these characteristics, we argue that this core ontology should be organized as an Ontology Pattern Language (OPL). An OPL is a network of interconnected domain-related ontology patterns that provides support for solving a class of ontology development problems for a specific domain. An OPL offers a set of interrelated domain patterns, and a process with explicit guidance on what problems can arise in that domain, informing the order to address these problems, and suggesting one or more patterns to solve each specific problem [5].

A core ontology should also be precise. This is achieved by basing the core ontology on a foundational ontology [27]. Thus, as the starting point for this work, we performed an ontological analysis of the ISO/IEC 24744 metamodel [24] in the light of the Unified Foundational Ontology (UFO) [7]. Based on the results obtained from this ontological analysis, and inspired on a Software Process OPL presented in [5], we have defined the ISO-based Software Process OPL (ISP-OPL).

The main purpose of ISP-OPL is to provide a sound solution for building ontologies in the software process domain, taking ISO standards as the main source of knowledge. This version of ISP-OPL focuses on the project (or endeavor) level, and addresses three main aspects dealt by ISO software process standards: Work Units, including patterns to define the decomposition, dependence, and scheduling of work units; Work Products, considering the nature of software process work products and how they are handled; and Human Resources, dealing with how people are organized in teams, allocated to tasks and perform work units. In order to evaluate ISP-OPL and its applicability, we performed an experiment encompassing the development of seven domain ontologies for SE sub-domains considering mainly ISO/IEC 12207, but also other related standards, such as CMMI [28].

This paper is organized as follows. Section 2 discusses the ISO standard harmonization initiative and the notion of Ontology Pattern Language. Section 3 presents ISP-OPL. Section 4 depicts how the experiment was conducted and the obtained results, including an ISP-OPL application example for the Requirements Engineering process. Section 5 discusses related work. Finally, Section 6 presents our final considerations.

2. ISO STANDARD HARMONIZATION AND ONTOLOGY PATTERN LANGUAGES

Standard harmonization is very important for organizations that seek to solve multiple needs at their different hierarchical levels by using multiple standards [22]. In these cases, standards are frequently used in combination. For instance, organizations use general standards for system development, along with standards that expand on specific processes such as software testing or risk management [9]. Moreover, frequently, organizations also want to combine standards from different sources [8, 22].

Harmonious combination of standards is aided when the standards use consistent concepts. At the beginning of 2012, in the ISO SC7 plenary meeting, a set of problems was raised, among them the following [9]: (i) there is no guidance on how to build a new standard ensuring that it is compatible with other SC7 standards; (ii) clashes in the terminology and in the semantics are observed in the current standards. Resulting from the discussion in this meeting, a study group was created, charged with the goal of investigating the potential utility of ontologies for rationalizing SC7's suite of SE International Standards [9].

This study group has proposed a layered framework comprising an ontology network [9]. In the top of the proposed framework, there is the Definitional Elements Ontology (DEO), which provides definitions for concepts, and constraints that dictate how they must be related. From DEO, a Configured Definitional Ontology (CDO) can be defined. The only CDO being worked to date is a CDO for the ISO/IEC 24744 metamodel (Software Engineering Metamodel for Development Methodologies – SEMDM) [16]. From a CDO, ontologies specific to particular standards, called Standard Domain Ontologies (SDOs), can be derived. The framework also considers in the future, to extend DEO by considering ontological distinctions put forward by foundational ontologies [7]. This extension is called Advanced Foundational Ontology for Standards (AFOS) [9].

SEMDM is the main basis for the entire framework, providing semantics for all ISO/SC7 standards. However, for the success of such initiative, the consistency of this ontological basis is crucial. Thus, in [24], we performed an ontological analysis of SEMDM in the light of the Unified Foundational Ontology (UFO) [7]. With this approach, we aim at providing a truly ontological foundation to the ISO framework. Moreover, we do not need a new foundational ontology (AFOS), but we can rely on an existing foundational ontology, in this case UFO. In [24] we identified several consistency problems in SEMDM fragments, and reengineered these model fragments, based on our ontological analysis.

The CDO based on the SEMDM is meant to be reused and extended in the development of several SDOs for specific software processes, such as Requirements Engineering process (ISO/IEC/IEEE 29148 [19]) and Software Testing process (ISO/IEC/IEEE 29119 [18]). For this reason, ontology patterns (OPs) arise as a promising alternative to organize the ontology framework, maintaining the actual benefits, and improving it to a modular and reusable solution [5]. In such approach, a domain ontology typically results from the composition of several OPs, with appropriate dependencies between them, plus the necessary extensions based on specific needs [2]. However, in order to truly favor reuse, organizing OPs in catalogues is not enough. A pattern language can provide a stronger sense of connection between the patterns, since it expresses several types of relationships among them, such as relations of dependence, temporal precedence of application, or mutual exclusion [5].

An Ontology Pattern Language (OPL) aims to provide holistic support for using domain-related OPs in ontology development. To ensure a stable and sound application of patterns, the patterns are presented in a suggested application order. OPLs encourage the application of one pattern at a time, following the order prescribed by paths chosen throughout the language [5].

In the next section, we present the ISO-based Software Process OPL (ISP-OPL), which has been developed aiming at supporting the ISO Harmonization Initiative.

3. AN OPL FOR ISO SOFTWARE PROCESSES

The aspects addressed by ISP-OPL are: Work Units (WU), Human Resources (HR) and Work Products (WP). The patterns in ISP-OPL were extracted from the reengineered fragments resulting from the ontological analysis of the SEMDM [24], as well as from the Enterprise OPL (E-OPL) proposed in [6].

Figure 1 presents a UML activity diagram showing the language paths of ISP-OPL. As suggested in [5], in this activity diagram, Domain-Related Ontology Patterns (DROPs) are represented by action nodes (the labeled rounded rectangles); initial nodes (solid circles) represent entry points in the OPL, i.e., DROPs in the language that can be used without solving other problems first; control flows (arrowed lines) represent the admissible sequences in which DROPs can be used; merge points (diamond-shaped symbols) represent the merge of paths in the OPL; join/fork nodes (line segments) represent the conjunction of paths (join) or independent and possibly parallel paths (fork); finally, an extension to the original UML notation (dotted lines with arrows) is used to represent variant patterns, i.e., patterns that can be used to solve the same problem in different ways. Moreover, patterns are grouped according to the software process aspect to which they are related: the three big boxes for Work Units, Human Resources and Work Products.

As Figure 1 shows, ISP-OPL has three entry points. The ontology engineer should choose one of them, depending on the scope of the specific software process ontology being developed. The ontology engineer should choose EP1, when the requirements for the new ontology include the definition and planning of work units; she should choose EP2, if the scope of the ontology considers only the execution of work units (performed WUs); EP3 is to be chosen if the ontology engineer aims to model only the structure of work products.

Through entry point EP1, in order to model the structure of WUs, the ontology engineer needs to choose one of (or both) the patterns *WU Composition* (WUC) and *WU Dependence* (WUD). These patterns are used to represent work units defined in an endeavor, without planning a time frame for them. WUC represents the mereological decomposition of work units, specializing Work Unit into Process, Composite Task and Simple Task. WUD deals with the dependence between work units. The *Project Process Definition* (PPD) pattern captures the link between a Process and the Project to which it is defined. The *WU Scheduling* (WUS) pattern is used to represent the time frame of a scheduled WU, defining its planned start and end dates. Next, the ontology engineer can focus on modeling performed work units, i.e., work units already executed. Performed WUs, as past events, have actual start and end dates. The tracking of performed work units against defined work units is treated by the *Performed WU Tracking* (PWUT) pattern, which relates a Scheduled Work Unit to a Performed Work Unit caused by the former. The group encompassing the patterns *Performed WU Composition* (PWUC) and *Performed WU Dependence* (PWUD) uses a similar structure to the group containing WUC and WUD. Additionally, the Project in which a Process is performed can be modeled with the *Project Process Performing* (PPP) pattern.

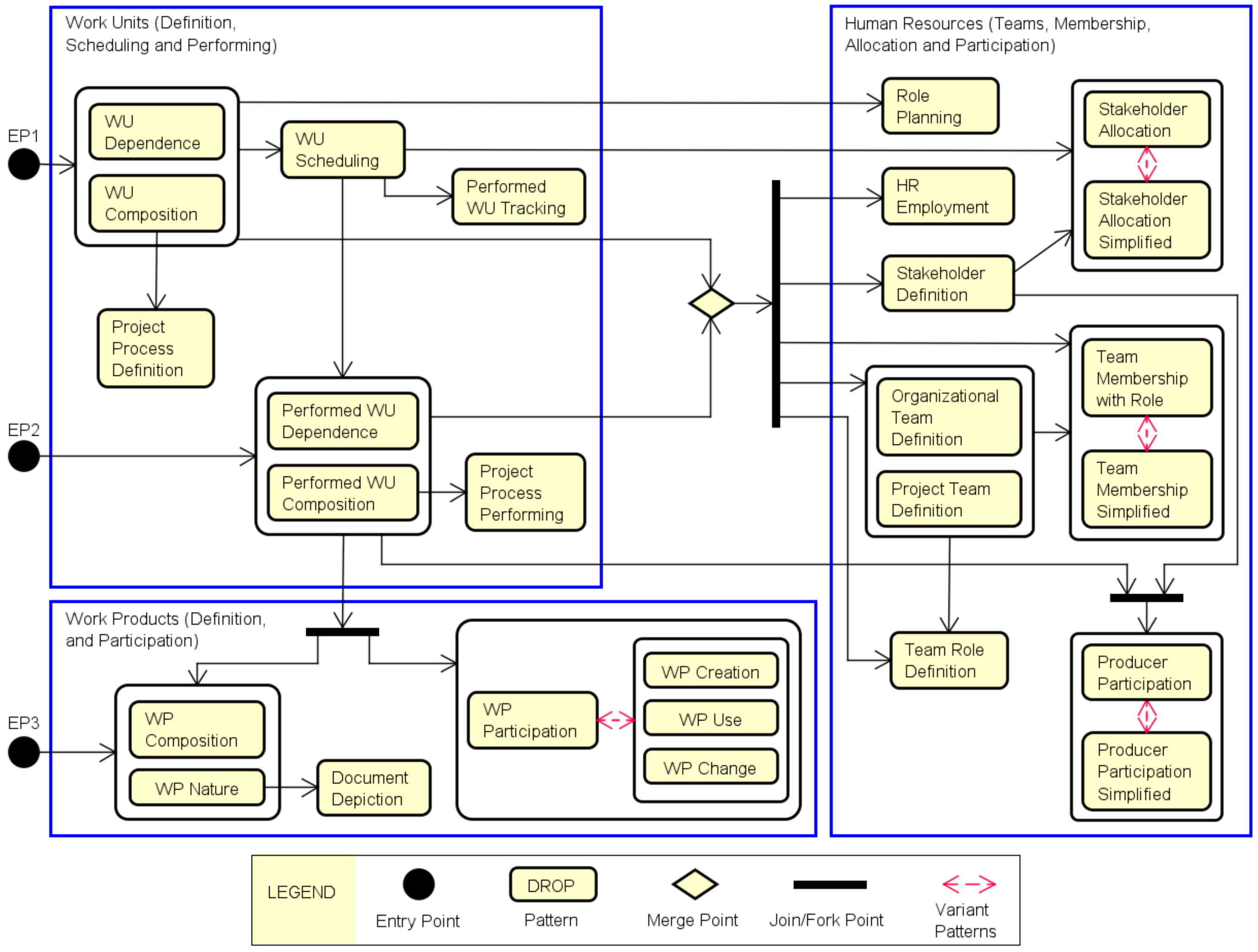
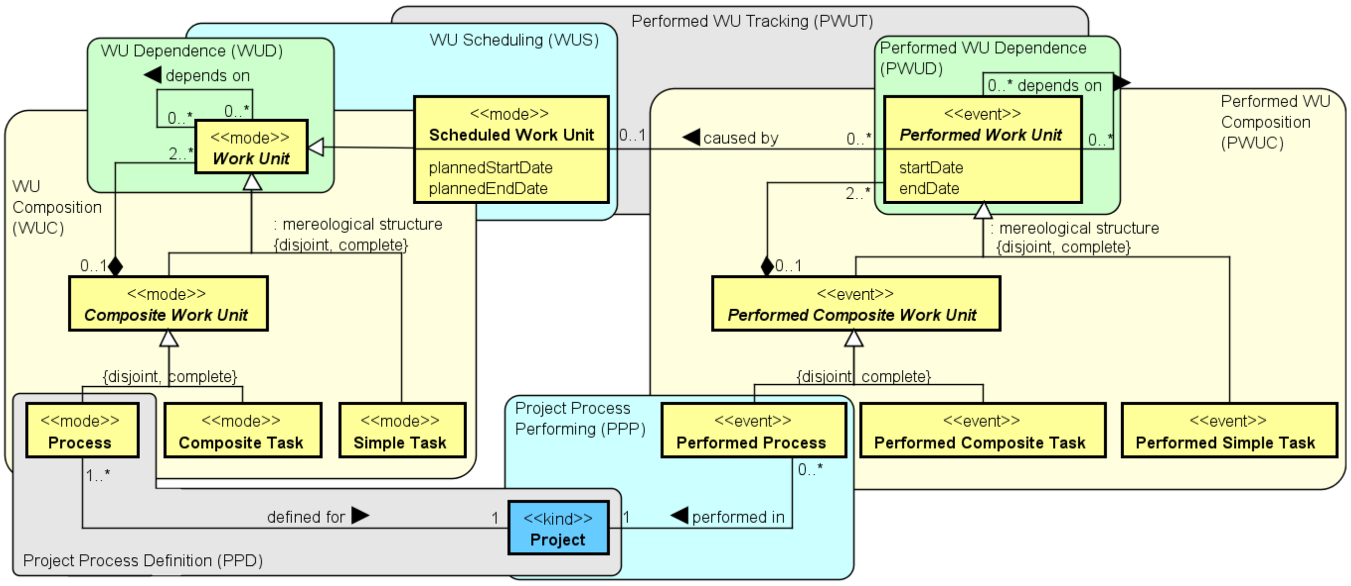


Figure 1. ISO-based Software Process OPL – Process



Performed WU Tracking (PWUT)
WU Scheduling (WUS)
WU Dependence (WUD)
depends on
0..* 0..*
<<mode>> Work Unit
2..*
<<mode>> Scheduled Work Unit
plannedStartDate
plannedEndDate
0..1
caused by
0..*
Performed WU Dependence (PWUD)
0..* depends on
<<event>> Performed Work Unit
startDate
endDate
0..*
2..*
Performed WU Composition (PWUC)
WU Composition (WUC)
: mereological structure {disjoint, complete}
0..1
<<mode>> Composite Work Unit
{disjoint, complete}
<<mode>> Process
<<mode>> Composite Task
<<mode>> Simple Task
0..1
<<event>> Performed Composite Work Unit
{disjoint, complete}
Project Process Performing (PPP)
<<event>> Performed Process
<<event>> Performed Composite Task
<<event>> Performed Simple Task
1..*
0..*
defined for
1
<<kind>> Project
1
performed in
Project Process Definition (PPD)

Figure 2. Patterns of the Work Unit Group

If the requirements for the ontology involve only performed work units, the entry point is EP2, allowing using only the patterns PWUC, PWUD and PPP. Figure 2 shows the complete model of the Work Unit group of patterns, detaching each pattern. Every pattern (of Figure 2 and the following) is represented using OntoUML. OntoUML is a UML profile that enables making finer-grained modeling distinctions between different types of classes and relations according to the ontological distinctions put forth by the Unified Foundational Ontology (UFO) [7].

After modeling Work Units related aspects, the ontology engineer can address human resource related problems by applying the patterns of the Human Resource group (shown in the right side of Figure 1). The *Human Resource Employment* (HRE) pattern establishes the employment relation between an Organization and a Person, which assumes the Human Resource role. This pattern was adapted from the *Employment* pattern of the Enterprise OPL (E-OPL) [6]. The *Stakeholder Definition* (StD) pattern defines the concept of Stakeholder (someone involved in a Project), and distinguishes between two types of stakeholders: Person Stakeholder and Team Stakeholder. Figures 3 and 4 show the patterns HRE and StD, respectively.

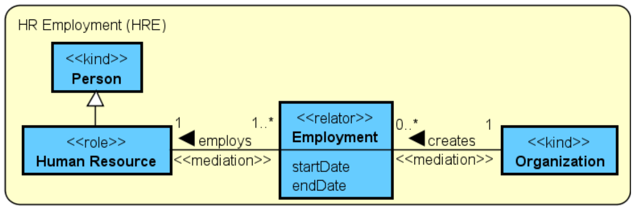


Figure 3. The Human Resource Employment (HRE) Pattern

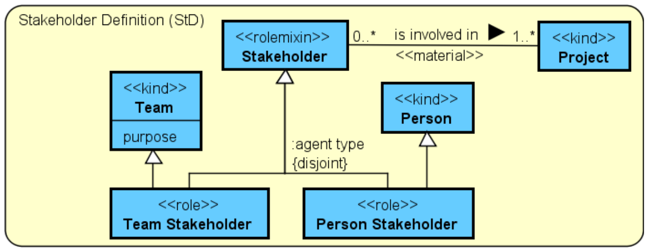


Figure 4. The Stakeholder Definition (StD) Pattern

The *Organizational Team Definition* (OTD) and *Project Team Definition* (PTD) patterns are used to define organizational and project teams, respectively. Both are also adapted from homonymous patterns from the E-OPL. Figure 5 shows these two patterns.

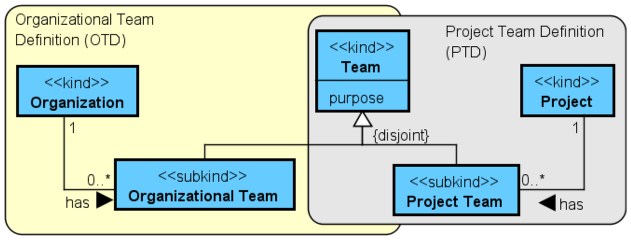


Figure 5. The Patterns Organizational Team Definition (OTD) and Project Team Definition (PTD)

The *Role Planning* (RPl) pattern can be used to model the roles responsible for performing a defined work unit, while the pattern *Team Role Definition* (TRD) can be applied to represent the roles a team can play. Figure 6 shows these two patterns.

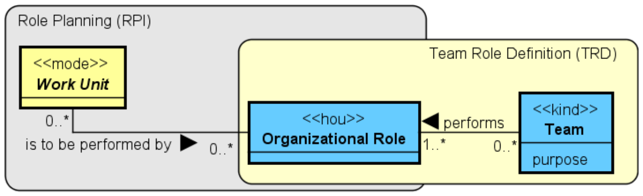
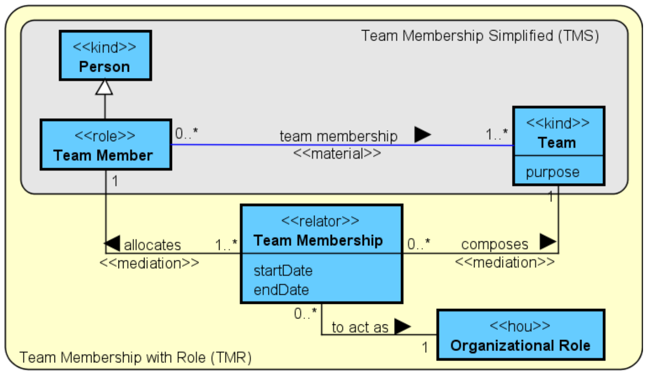


Figure 6. The Patterns Role Planning (RPl) and Team Role Definition (TRD)

In order to represent the membership relation between a team and its members (persons), the ontology engineer can choose one of the alternative patterns *Team Membership Simplified* (TMS) and *Team Membership with Role* (TMR), which are shown in Figure 7.

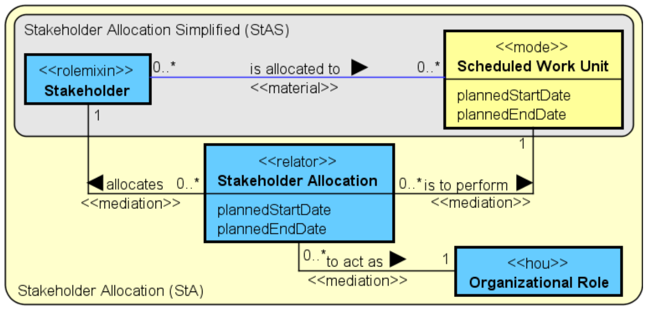
Two alternative patterns can be used to represent the allocation of stakeholders to a scheduled work unit: *Stakeholder Allocation* (StA) and *Stakeholder Allocation Simplified* (StAS). As Figure 8 shows, StA models the relational property (relator) Stakeholder

Allocation that glues the stakeholder to the scheduled work unit and to the organizational role the stakeholder plays. Moreover, the planned start and end dates for the stakeholder allocation are captured. StAS is a simplified version that omits the relator, capturing only the material relation linking stakeholders to scheduled work units.



Team Membership Simplified (TMS)
<<kind>> Person
<<role>> Team Member
team membership <<material>>
<<kind>> Team
purpose
<<relator>> Team Membership
startDate
endDate
allocates <<mediation>>
composes <<mediation>>
to act as
<<hou>> Organizational Role
Team Membership with Role (TMR)

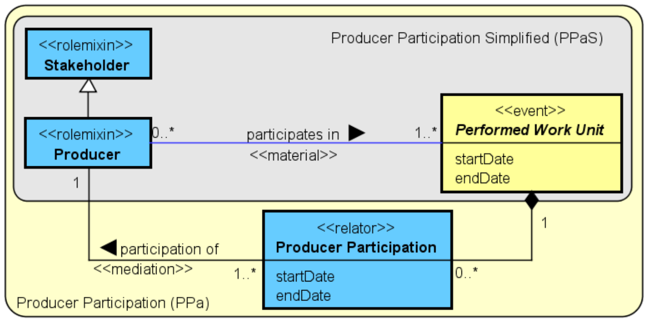
Figure 7. The Alternative Patterns Team Membership with Role (TMR) and Team Membership Simplified (TMS)



Stakeholder Allocation Simplified (StAS)
<<rolemixin>> Stakeholder
is allocated to <<material>>
<<mode>> Scheduled Work Unit
plannedStartDate
plannedEndDate
<<relator>> Stakeholder Allocation
plannedStartDate
plannedEndDate
allocates <<mediation>>
is to perform <<mediation>>
to act as <<mediation>>
<<hou>> Organizational Role
Stakeholder Allocation (StA)

Figure 8. The Alternative Patterns Stakeholder Allocation (StA) and Stakeholder Allocation Simplified (StAS)

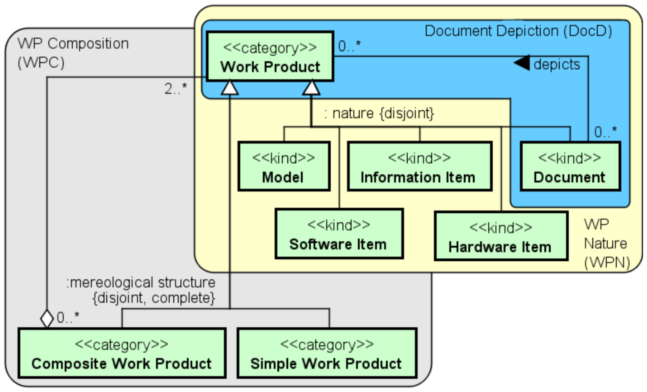
Finally, for dealing with the participation of stakeholders in performed work units, the ontology engineer can choose between the alternative patterns *Producer Participation* (PPa) and *Producer Participation Simplified* (PPaS). The difference between these patterns refers to whether the relator Producer Participation is explicitly represented or not (respectively), as Figure 9 shows.



<<rolemixin>> Stakeholder
Producer Participation Simplified (PPaS)
<<rolemixin>> Producer
participates in <<material>>
<<event>> Performed Work Unit
startDate
endDate
participation of <<mediation>>
<<relator>> Producer Participation
startDate
endDate
Producer Participation (PPa)

Figure 9. The Alternative Patterns Producer Participation (PPa) and Producer Participation Simplified (PPaS)

The last group of patterns constituting ISP-OPL is the group related to Work Products (WP). This group can be achieved from the patterns related to Performed WU, but also through the entry point EP3, which is to be chosen when the ontology engineer wants to represent only the structure of work products. The *WP Composition* (WPC) pattern allows modeling work product mereological decomposition. *WP Nature* (WPN) is related to types of work products (such as Document, Model and Information Item). Once applied WPN, *Document Depiction* (DocD) pattern can be used to model the fact that documents depict other work products. Figure 10 shows these three patterns.



WP Composition (WPC)
Document Depiction (DocD)
<<category>> Work Product
depicts
nature {disjoint}
<<kind>> Model
<<kind>> Information Item
<<kind>> Document
<<kind>> Software Item
<<kind>> Hardware Item
WP Nature (WPN)
:mereological structure {disjoint, complete}
<<category>> Composite Work Product
<<category>> Simple Work Product

Figure 10. The Patterns WP Composition (WPC), WP Nature (WPN) and Document Depiction (DocD)

When the patterns for work unit execution are already applied (through EP1 or EP2), beyond the work product structure, the ontology engineer can also model work products handling. In this case, *WP Participation* (WPPa) pattern sets the participation of work products in performed work units. The relator Work Product Participation is modeled with its specializations for creation, change and usage participation. Alternatively, these three types of participations can be modeled only by means of the corresponding material relations using the patterns *WP Creation* (WPCrea), *WP Change* (WPChan) and *WP Use* (WPUse), as Figure 11 shows.

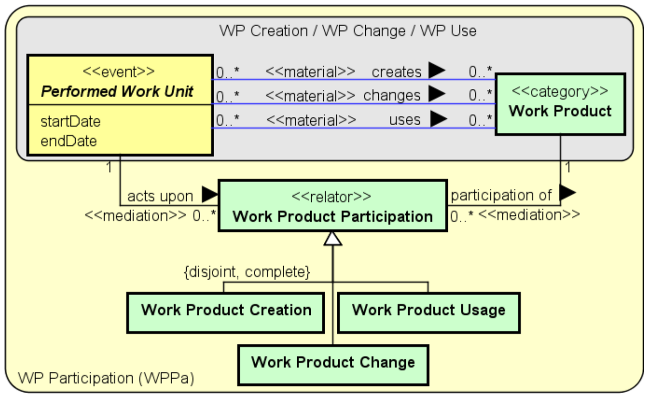


Figure 11. The Work Product Participation Patterns

It is important to highlight that, since the patterns constituting ISP-OPL are described in OntoUML, they carry out the ontological and formal semantics of its modeling constructs such as *kind*, *category*, *role mixin*, *relator*, *mode*, *mixin*, *material relation*, etc. OntoUML is itself a pattern-based language (albeit a domain-independent one), whose modeling primitives are patterns that embody the micro-theories comprising the foundational ontology UFO [8]. As a consequence, the patterns of ISP-OPL are systematically constructed via the manifestation of the ontology-based patterns of OntoUML and UFO. For instance, in the patterns WUC and PWUC (Figure 2) and WPC (Figure 10), we have the direct manifestation of the UFO pattern (micro-theory) of

Mereological Relations [7]. Moreover, in the patterns PPa (Figure 9) and WPPa (Figure 11), we have the direct manifestation of the OntoUML *Relator* pattern [7]. Finally, in the pattern StD (Figure 4), we have the manifestation of *Roles with Multiple Disjoint Allowed Types* pattern, or simply, the *Role Mixin* pattern [7]. As one will be able to observe in the next section, the structures constituting these patterns are carried out and presented in the ontologies created using ISP-OPL.

In order to fully document ISP-OPL for users, we developed the ISP-OPL Specification, version 1.0 (available at http://nemo.inf.ufes.br/OPL). This specification presents ISP-OPL Process and describes each DROP in detail, considering: the pattern name, intent, rationale, competency questions, conceptual model, and axiomatization. Table 1 shows the description of the *Performed WU Composition* (PWUC) pattern.

In the next section, we discuss an experiment applying ISP-OPL for developing seven domain ontologies for Software Engineering sub-domains, taking standards into account.

4. APPLYING ISP-OPL

Software processes encompass a wide number of sub-domains, such as Requirements Engineering, Architectural Design, Detailed Design, Project Management, Quality Assurance, Measurement, Risk Management, etc. For several of them there are standards covering their definitions, activities and related assets. In the context of the ISO Harmonization Initiative, beyond the core knowledge about software process (aimed to be represented by the definitional ontologies), it is necessary to represent each one of these sub-domains. Moreover, it is important that the sub-domain models may be derived from CDOs, originating each of the required SDOs. This section presents an empirical study performed in order to demonstrate how this derivation process can be supported by the application of ISP-OPL for building domain ontologies.

The experiment involved the development of seven domain ontologies representing different sub-domains described by Software Engineering standards. Section 4.1 discusses the experiment design, including the experiment plan and the subjects' profile. Section 4.2 presents, as an example of ISP-OPL application, the Requirements Engineering Process ontology. Finally, Section 4.3 discusses the experiment analysis and main results.

4.1 Experiment Design

The main goal of this experiment was to evaluate ISP-OPL, collecting indicators and other relevant information about its application. Some important questions to answer are related to how the guidance provided by ISP-OPL affects the productivity of ontology engineers when developing domain ontologies for Software Engineering sub-domains; and how the use of ISP-OPL can improve the quality of the resulting ontologies.

The empirical study was conducted following the guidelines presented in [20]. The experiment took place during the second semester of 2014, as part of the course "Ontologies for Software Engineering", an advanced course for graduate students in the Graduate Program in Informatics at Federal University of Espírito Santo, in Brazil.

The subjects of the experiment were 19 graduate students with at least basic knowledge in conceptual modeling. A questionnaire was applied to capture the participants' profile, analyzing their level of education, experience in conceptual modeling, experience in ontology development, and experience with OPLs.

Table 1. The PWUC Pattern Specification

PWUC – Performed WU Composition

Name: Performed WU Composition (PWUC)

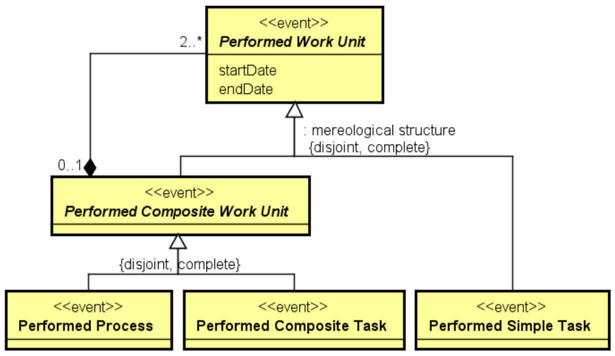
Intent: To represent the composition of performed work units in terms of other performed work units.

Rationale: *Performed Work Units* can be composed of other *performed work units*. Mereologically, a *performed work unit* is simple, or composed of two or more parts. At the basic level, there are *Performed Simple Tasks* that can compose other *performed work units*, but are not decomposable. *Performed Composite Tasks*, in turn, are composed of other performed tasks (*composite* or *simple performed tasks*). At the higher level, *Performed Processes* are also composed of performed tasks, but do not compose any other *performed work unit*.

Competency Questions:

- Concerning their mereological structure, what are the possible types of performed work units?
- How is a performed work unit composed of other performed work units?

Conceptual Model



Axiomatization

A1: $\forall w,c\ partOf(w,c) \rightarrow (w\ != c)$
A **Performed Work Unit** cannot be part of itself.

A2: $\forall p:\ PerformedProcess(p) \rightarrow \neg \exists w\ PerformedWorkUnit(w) \wedge partOf(p,w)$
A **Performed Process** cannot be part of any **Performed Work Unit**.

A3: $\forall w1,w2:\ partOf(w2,w1) \rightarrow (w2.startDate >= w1.startDate) \wedge (w2.endDate <= w1.endDate)$
A **Performed Work Unit** that is part of another should occur within the time interval of its whole.

Regarding the profile, all participants were students in the Computer Science area, being around 90% of master degree students, and 10% of PhD students. Concerning the **experience in conceptual modeling**, 32% informed low experience (less than one year), 47% declared medium experience (from one to three years), and 21% have high experience (more than three years). Regarding the **experience in ontologies development**, we had 47% having their first experience developing an ontology in this experiment, 37% with low experience (less than one year), 16% with medium experience (from one to three years), and no one declared high experience (more than three years). Finally, all participants had the first **experience with OPLs** in this experiment. Therefore, we can say that the group of subjects has, mostly, medium experience in conceptual modeling, low experience in ontologies development and no experience with OPLs.

The course in question covers the following topics: Ontologies - Types and Definitions; Ontologies and Software Engineering; Ontology Development with Ontology Patterns; Ontologies for the Software Engineering Domain; and Applications of Ontologies in Software Engineering. The course instructor (the second author) taught the entire course, except for the ISP-OPL tutorial, which was taught by the first author. The object of study, ISP-OPL, was presented for the class and its specification was made available. The Requirements Engineering domain was taken as example, and the ISP-OPL authors developed this domain ontology, and made it available for the experiment participants. This ontology was also presented for the class.

The participants were divided into seven groups. Each group received one topic covered by the following ISO/IEC 12207 Software Processes: Human Resource Management, Measurement, Risk Management, Software Architectural Design, Software Configuration Management, Software Documentation Management, and Software Maintenance. They had a period of two months between the presentation of the ISP-OPL and the delivery of the domain ontology and related documentation. We had to conduct the study as a homework assignment, because ontology development took a long time to complete.

The experiment had four phases: (i) study of the domain, (ii) development of the domain ontology, (iii) evaluation of the resulting domain ontologies, and (iv) application of questionnaires and interviews.

First, the participants had to study software process standards and models for understanding the domain knowledge. The groups took as basis ISO/IEC 12207 [11], and ISO specific standards for each domain (e.g., ISO/IEC 15939 [15] for Measurement, ISO/IEC/IEEE 15289 [17] for Documentation, and ISO/IEC 14764 [12] for Maintenance). Since ISO standards are usually neutral concerning some process aspects such as work products and human resources roles, additionally, the participants used also the following models: CMMI [28], MR-MPS-SW [26] and SWEBOK [3].

The second phase has started with the presentation of ISP-OPL in class, with the application example for the Requirements Engineering Process domain (see Section 4.2). All the groups had access to the ISP-OPL Specification and to the ontology documentation for the example. Each group had to develop an ontology for the specific sub-domain chosen. The scope for the ontologies included only dealing with performed work units, work products handled by them, and stakeholder participations. As result, each group had to deliver a Reference Ontology Specification, containing the following information: domain ontology purpose, a brief description of the sub-domain being addressed, the sequence of the patterns application, competency questions, domain ontology models (OntoUML models), a glossary of the concepts in the ontology, and a mapping between the concepts in the ontology and the concepts present in the standards used.

In the third phase, the first and the second authors of this paper evaluated the resulting Ontology Specifications. This evaluation allowed us to identify several findings about ISP-OPL and its use, as discussed in Section 4.3.

The last phase involved the application of a questionnaire, the conduction of interviews, and the analysis of the collected data. After delivering the Ontology Specification, the participants answered, individually, a questionnaire with 14 questions divided into two parts. The first part focused on the understandability of the OPL process and its elements; the second part regarded the use of ISP-OPL for creating a domain ontology. In this second part, two questions used a Likert Scale, namely:. (i) "*Do you consider that the OPL application contributed for the quality of the resulting ontology?*"; and (ii) "*If you already have a previous experience in developing ontologies, do you consider that the OPL application contributed for the productivity in the ontology development process?*", wherein the responses could be: *completely disagree*, *disagree*, *indifferent*, *agree*, and *totally agree*. Another question used a scale of difficulty, namely: "*How hard was to use ISP-OPL for creating a domain ontology?*", wherein the responses could be *very hard*, *hard*, *neutral*, *easy* and *very easy*. In the three cases, the participants were asked to justify their responses. Finally, there were two open questions: (i) "*What were the difficulties you found for integrating the selected patterns?*"; and (ii) "*Describe (briefly) the process you followed for developing the ontology by applying ISP-OPL*".

Finally, after analyzing the questionnaire responses, an interview was conducted with each group. A structured interview was prepared. The questions analyzed the perception of the interviewees about the OPL, the process of using the OPL, the patterns, and the group experience using ISP-OPL for creating a domain ontology. Section 4.3 presents the analysis of the main findings collected in this phase.

4.2 The Requirements Engineering Process Ontology

In this section, we present part of an ontology we have developed for the Requirements Engineering (RE) process. In the context of the experiment, the RE Process ontology served as the first application of ISP-OPL, as well as an example for the development of the other domain ontologies by the participants of the experiment. We chose this process, due to its importance as a basis for software development, with concepts appearing in several standards, and because it is a well-known domain for the experiment subjects. The RE Process Ontology was derived from ISP-OPL according to the information extracted from selected ISO SC7 standards, namely: ISO/IEC 15288:2008 – System life cycle processes [13], ISO/IEC 12207:2008 – Software life cycle processes [11], and ISO/IEC/IEEE 29148:2011 – Requirements Engineering [19]. These are the main ISO standards dealing with requirements processes, from which our competency questions were defined. Together, the standards define three requirements-related processes: *Stakeholder Requirements Definition*, *System Requirements Analysis*, and *Software Requirements Analysis*. We present here only the sub-ontology addressing the first process: *Stakeholder Requirements Definition* (Section 6.4.1 in ISO 12207 and ISO 15288, and Section 6.2 in ISO 29148).

Figure 12 shows the chosen patterns and paths of the ISP-OPL process that we followed for developing this ontology. Figure 13 presents the *Stakeholder Requirements Definition Process* sub-ontology. On the top, the concepts with colored background are the ones defined as part of the ISP-OPL patterns. On the bottom, the concepts with blank background are the specific ones from the RE Process Ontology. Relations in the RE Process Ontology are specializations of the homonymous relations in the OPL. Cardinalities are omitted for the sake of legibility.

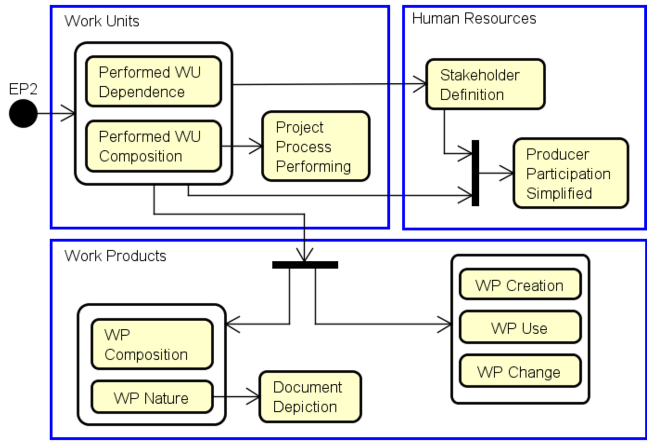


Figure 12. ISP-OPL Patterns and Paths followed

We are interested in describing the execution of requirements processes, including the participations of human resources and work products, as it is the case of organizations adopting these standards in their projects. Thus, we start using ISP-OPL through the entry point EP2. As defined in the aforementioned standards, the *Stakeholder Requirements Definition* process is decomposed in activities, which, in turn, are decomposed in tasks. Thus, we start with the *Performed WU Composition* pattern, modeling the decomposition of performed work units. The ***Stakeholder Requirements Definition Process*** is a subtype of **Performed Process**. This specialized process is composed of five work units: ***Stakeholder Identification***, ***Requirements Identification***, ***Requirements Evaluation***, ***Requirements Agreement*** and ***Requirements Recording***. The first and fourth work units are **Performed Simple Tasks**, and the others are **Performed Composite Tasks**, decomposed into simple tasks as shown in Figure 13.

Another pattern considered useful here is PWUD, which defines dependencies between work units. Although the selected standards do not explicitly set dependencies between tasks, some of them can be easily inferred from the nature of work units and work products handled, as well as by considering the RE literature. Thus, we applied the *Performed WU Dependence* pattern and established dependencies between the work units, as shown in Figure 13. Still regarding work units, the last pattern applied is *Project Process Performing*, establishing the connection between the **Performed Process** and the **Project** wherein it is performed.

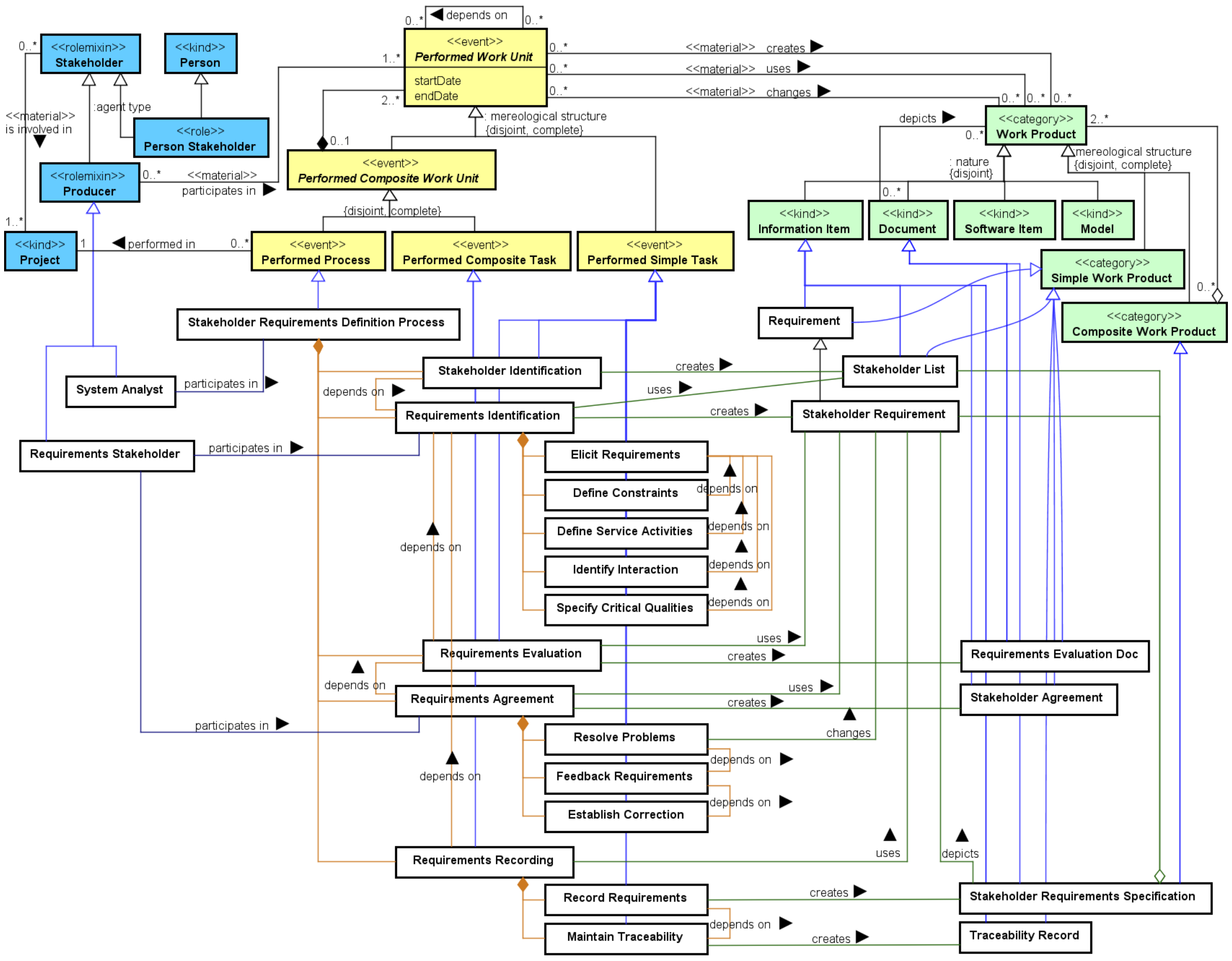

<<rolemixin>> Stakeholder
<<kind>> Person
<<role>> Person Stakeholder
<<rolemixin>> Producer
<<kind>> Project
<<event>> Performed Work Unit
startDate
endDate
depends on
<<material>> creates
<<material>> uses
<<material>> changes
mereological structure {disjoint, complete}
<<event>> Performed Composite Work Unit
<<event>> Performed Process
<<event>> Performed Composite Task
<<event>> Performed Simple Task
{disjoint, complete}
:agent type
<<material>> is involved in
<<material>> participates in
performed in
<<category>> Work Product
depends
: nature {disjoint}
mereological structure {disjoint, complete}
<<kind>> Information Item
<<kind>> Document
<<kind>> Software Item
<<kind>> Model
<<category>> Simple Work Product
<<category>> Composite Work Product
Stakeholder Requirements Definition Process
System Analyst
Requirements Stakeholder
Stakeholder Identification
Requirements Identification
Elicit Requirements
Define Constraints
Define Service Activities
Identify Interaction
Specify Critical Qualities
Requirements Evaluation
Requirements Agreement
Resolve Problems
Feedback Requirements
Establish Correction
Requirements Recording
Record Requirements
Maintain Traceability
Requirement
Stakeholder List
Stakeholder Requirement
Requirements Evaluation Doc
Stakeholder Agreement
Stakeholder Requirements Specification
Traceability Record
participates in
depends on
creates
uses
changes
depicts

Figure 13. The Requirements Process Ontology (Stakeholder Requirements Definition Process sub-ontology)

Once work units are addressed, we can represent human resources. Due to the general nature of the standards, few information is given about human resources participating in work units. Thus, we have modeled only the stakeholder definition and its relation with work units. The first pattern applied is *Stakeholder Definition*, to establish the stakeholder structure to be adopted. We consider only two types of stakeholders: ***System Analyst*** (suggested, but not explicitly named in the standards), and ***Requirements Stakeholder***. Both are **Person Stakeholders** involved in the **Project**. Aiming to represent the participation of stakeholders in work units, the *Producer Participation Simplified* pattern is used, specializing stakeholders as **Producers**, in order to participate in **Performed Work Units**.

The other path of ISP-OPL we followed is through the use of work products patterns. Once we have different types of work products, it is useful to distinguish between them by applying the *WP Nature* pattern. Two subtypes of **Work Product** are considered: **Information Item** and **Document**. In the context of the *Stakeholder Requirements Definition Process*, we identified the following subtypes of **Information Item**: ***Requirement*** (in turn, specialized into ***Stakeholder Requirement***), ***Stakeholder List***, ***Stakeholder Agreement***, and ***Traceability Record***. Moreover, two sub-types of **Document** are considered: ***Requirements Evaluation Doc***, and ***Stakeholder Requirements Specification*** (referred as ***StRS*** in ISO 29148). The ***StRS*** is the main result of this process and aggregates the ***Stakeholder List*** and the set of ***Stakeholder Requirements***. Thus, using the *WP Composition* pattern, we establish ***StRS*** as a **Composite Work Product** (the only one in the ontology), composed of ***Stakeholder Requirements*** and ***Stakeholder List*** (**Simple Work Products**). Additionally, by applying the *Document Depiction* pattern, ***StRS***, as a document, also depicts the ***Stakeholder Requirements***.

Finally, by using the patterns *WP Creation*, *WP Use* and *WP Change*, we established the relationships of *creation*, *usage* and *change* between the work units of the ***Stakeholder Requirements Definition Process*** and the corresponding work products.

The RE Process ontology, created from the application of ISP-OPL, is able to precisely define the concepts and relations for the requirements domain according to the ISO standards. These definitions, aligned to the core process definitions, serve as a common semantic basis for the related standards, contributing to their harmonization.

4.3 Experiment Analysis

From the resulting ontologies developed by the experiment participants, as well as by the application of questionnaires and interviews, we could collect relevant information about the usefulness of OPLs in general, and of ISP-OPL in particular.

Regarding the resulting ontologies, the seven ontology specifications were evaluated by the first two authors of this paper. The evaluation criteria include the correct use of the OPL, proper application of the patterns, and a sound documentation of the ontology, comprising the competency questions, OntoUML models, standards mapping, and other related information. The evaluation served to analyze each resulting ontology, but also to observe the OPL usefulness at all. In the following, we discuss some of the main findings, addressing the strengths, drawbacks and participants perceptions regarding ISP-OPL.

4.3.1 ISP-OPL Strengths

The main strengths perceived during the domain ontologies evaluation are discussed in the following.

Structural Similarity of the Conceptual Models: In a broad view, the first finding we could observe from the resulting domain ontologies regards the structural similarity between the OntoUML models. Since all the ontologies were created based on the OPL patterns, with the same general scope (addressing performed work units, work products handled by them, and stakeholder participations), it is clear the resemblance between the structure of each ontology, showing work units, work products and human resources in a similar way (all similar to Figure 13). This similarity also manifests itself in the ontology concepts and relations. We observed a similar granularity of the work units and work products, and compatible decisions for naming concepts, defining dependencies and setting work product participations. All these findings serve as evidences of compatibility between the domain ontologies, a desired result for the harmonization efforts.

Reuse of Competency Questions: Another interesting finding is the reuse of competency questions (CQs) from the OPL patterns. Once a DROP is chosen, its concepts and relations become part of the domain ontology, where they can be extended. For CQs, a very similar approach holds: once a DROP is chosen, its CQs can be extended for the domain ontology. For example, the *Performed WU Composition* pattern (see Table 1) has the following CQs: (i) *Concerning their mereological structure, what are the possible types of performed work units?*; and (ii) *How is a performed work unit composed of other performed work units?* When this pattern was applied to the Software Configuration Management (SCM) process, the following (extended) specific CQs were created: (i) *What are the possible types of performed work units in the SCM process?*; and (ii) *How is the SCM process decomposed?* This reuse helps the CQ definition, improving the productivity of the ontology engineering process.

Extraction of Concepts and Relations from the Standards: In order to develop the domain ontologies for the specific processes, each group had to analyze the related standards to elicit relevant concepts and relations. Once the domain ontology scope was established and the OPL was available, data extraction started by reading the standards and selecting the relevant concepts and relations. By using the OPL, the modelers could make a more productive extraction, looking only for those concepts and relations they need. For instance, some groups used different text markup to identify the sub-types of previously defined types (such as work units, work products and human resources) in the standards, speeding up the information extraction.

Enrichment of the Models: Standards are general by their nature and, sometimes, information is implicit or even absent. In ISO/IEC 12207, for example, although work units are well organized, there is little information about who (producers) performs them and which work products are handled by them. The supporting models used in the experiment (such as CMMI, MPS.BR and specific standards) allowed the participants to extract more information mainly about producers and work products. However, these concepts should be linked to the concepts and relations already identified in the ontology. Since the OPL patterns define the general organization of these elements, the patterns were used for linking the concepts extracted from different standards. For instance, in the ISO/IEC 12207 Risk Management process, nothing is said about who performs the Risk Management Planning activity, or about the main result it produces. The patterns *Producer Participation Simplified* and *WP Creation* require setting a producer performing this activity and a work product being created, respectively. Thus, looking for these information, the ontology engineers identified in the IEEE Std 1540: 2001 [10] two important concepts for the Risk Management Process: the producer Manager and the composed work product Risk Management Plan. During the OPL application, for all processes, several producers and work products were identified and related to the proper work units. This shows the guidance provided by ISP-OPL, leading to more consistent models.

Foundational Support: As discussed above, ISP-OPL drives the definition of concepts and relations in the ontologies. However, it does not limit the ontology engineer. New domain-specific concepts, relations and axioms not considered in the OPL can be included in the ontology. Since ISP-OPL is aligned with the theories of the Unified Foundational Ontology (UFO), new concepts can be more easily integrated to the concepts extracted from the patterns by applying the same foundational theories underlying OntoUML. These notions help to create a more consistent model, going beyond the OPL support. For instance, some domain ontologies represented *phases* of Work Products, or *roles* assumed by an Information Item, or even the application of the *role mixin* foundational pattern [7]. This is the case, for instance, of the concept Configuration Item, which is not a regular Work Product, but generalizes the notion of different types of work products that have their configuration managed. Regarding this, it is important that the ontology engineers using ISP-OPL also knows the foundational theories underlying OntoUML, in order to correctly extend the patterns. For a discussion regarding the combined use of Foundational Ontology Patterns and Domain-Related Ontology Patterns, see [25].

4.3.2 ISP-OPL Drawbacks

During ontology evaluation, we have also identified the main mistakes made by the groups while developing ontologies using ISP-OPL. It is important to observe and analyze these problems in order to improve the OPL and the way to apply it. Following, we discuss the main mistakes and improvement points.

Concepts Identification/Conciliation: One of the most difficult tasks performed during the development of the ontologies about the specific processes was to extract information of the same process from different standards and to put them together. It is more a harmonization task than an ontology engineering one. ISP-OPL helps the identification of concepts by offering the general concepts and relations in its patterns. However, in different standards, some elements are not structured exactly in the same way. For example, similar work units in different standards can have different composing parts, and conciliating them was a common source of mistakes.

Process Interaction: The processes described in the standards generally refer to other processes. When these processes are represented as conceptual models, those interactions need to be made explicit. In this way, it is possible to identify more associated information, such as precedence, collaboration, use of work products, etc. Since ISP-OPL specializes work units according to their mereological decomposition (WUC / PWUC), and enables the representation of dependencies between them (WUD / PWUD), it is possible to represent the cases where a task “calls” a process, or tasks of another processes. This is a common situation, for example, in the processes Software Configuration Management, Software Maintenance, and Software Documentation Management. These processes are close related, being performed together in most of the cases. Making explicit process interactions helps to describe better the compatibility between processes. However, since each group addressed only one process, none of the groups had a comprehensive view of the process interactions. Thus, some integration work is still needed. Although there are similarities between the ontologies, other tasks, such as aligning the terms used and representing the identified interactions, are necessary. These tasks are important for building a complete Software Engineering ontology network [29], a useful artifact concerning the standards harmonization initiative.

Wrong/Missing Classification: Most of the concepts in the domain ontologies are extended from concepts of the applied patterns (except in few cases of domain specific concepts). Thus, each concept is classified as an extension of one or more previously defined concepts. Some of the domain ontologies presented mistakes in this classification. For instance, some *simple tasks* were classified as *composite tasks* (and vice-versa); some *information items* were classified as *documents* (and vice-versa); and some *producers* were classified as *stakeholders*. Another common mistake regards multiple classification. For example, when modeling a *document* composed of other *work products*, the new concept was classified only as a document, missing the *composite work product* classification. Even with the available OPL specification detailing the patterns, a more precise definition of each concept and relation is needed.

Wrong/Missing Relation: Most of the relations are also specializations of the ones in the pattern models. The main mistakes here regard the lack or wrong definition of certain relations. For instance, the *dependence* relation between *work units* was sometimes missed, and sometimes defined for *tasks* that are not really dependent. Another example of confusion occurred with the relations *depicts* and *componentOf* between work products. Finally, the relations *create* and *use* between work products and work units were sometimes confused and applied swapped.

Insufficient Patterns in ISP-OPL: Although in general ISP-OPL provided a good coverage to the software process domain, we identified at least two situations in which ISP-OPL did not attend the modeling needs of the participants, showing improvement opportunities:

(1) As the *WP Participation* pattern defines (see Figure 11), a *work product participation* can be a *creation*, a *change* or a *usage*. It covers most of the situations, mainly for sequential processes. However, when iteration is considered, a work unit may create a work product in the first cycle, but can only use or change it in the next cycles, since it already exists. For example, in the RE Process ontology (see Figure 13), if the Requirement Recording task occurs in cycles, this task creates the Stakeholder Requirements Specification document only in the first iteration, and then, changes it by including / modifying / excluding its contents. Thus, ISP-OPL needs to consider other types of participation to allow modeling such situation. Thus, for the next version of ISP-OPL, we intend to add a new type of participation: *produces*, meaning that a work unit creates a work product if it does not exists, and then use or change it.

(2) The second case involves the dependencies between *work units*. There is only one type of dependence, and it is not representative enough for all situations. For the next version of ISP-OPL we intend to include patterns for dealing with some of the *Allen Relations* [1], for example *before* and *meet*,

in order to gain more expressivity representing work unit dependencies.

4.3.3 *Analysis of the Participants Perceptions*

In the last phase of the experiment, questionnaires were applied and interviews were made with the participants. Our intention was to get the subjects' perceptions about the use of ISP-OPL to build the domains ontologies. We have selected three main questions to discuss here:

(i) Ease of Reuse: *How hard was to use ISP-OPL for creating a domain ontology?*

(ii) Productivity: *If you have a previous experience developing ontologies, do you consider that the OPL application contributed for the productivity in the ontology development process?*

(iii) Quality: *Do you consider that the OPL application contributed for the quality of the resulting ontology?*

Regarding Ease of Reuse, as Figure 14 shows, 63% of the subjects considered it *easy* or *very easy* to use ISP-OPL for creating the domain ontology; 26% considered it *neutral*; and only 11% said that it was *hard* to use ISP-OPL for creating a domain ontology. Matching with the subjects' profiles, 50% of the low experienced in conceptual modeling considered *easy* to use the OPL, against around 70% of the medium and high experienced. These numbers show that the use of ISP-OPL was considered easy in general, even for the subjects with less experience in conceptual modeling.

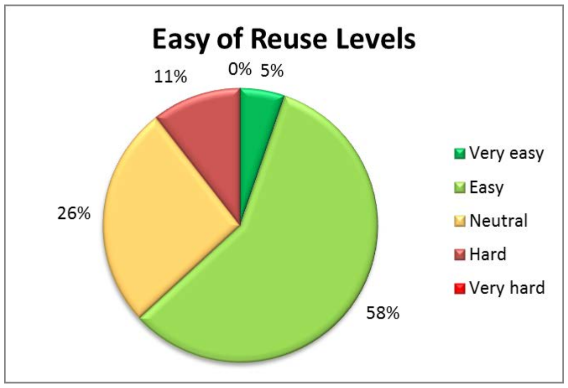


Figure 14. Ease of Reuse Levels

For analyzing Productivity, we had to consider only the participants who had previous experience in ontology development. 10 participants (53%) reported having previous experience in developing ontologies. All of them considered that the use of ISP-OPL speeded up the ontology development. 5 of them (50%) *agree* with this premise, and the other 5 (50%) *totally agree* (50%). Some participants emphasized that the guidance provided by ISP-OPL helped them to define the ontology scope and made the development process more intuitive.

Finally, concerning Quality, when asked if ISP-OPL application contributed for the quality of the resulting ontology, all the 19 subjects *agree*d (37%) or *totally agree*d (63%). Some subjects pointed out that ISP-OPL helped them to capture the main concepts of the domain, to reduce errors, and to create a well-founded ontology.

4.3.4 *Threats to the Validity of the Experiment*

We have identified some limitations and validity threats to our experiment. Firstly, although the participants had similar formation (in Computer Science area), they were students with different experience levels in conceptual modeling and ontology development. Along the course, the participants studied the content necessary to execute the proposed activity. However, the results might be affected by the different experience levels. Secondly, the participants had also different background regarding the sub-domains chosen for building the ontologies. Some of the groups knew well the process being modeled, while others had a first contact with the process being modeled during the activity. Thirdly, the activity was done as a homework, and, although the deadline was the same for all groups, some groups might have spent much more time in the activity than others. Fourthly, the number of participants was small, and thus we had not a representative sample. Because of that, we could not apply statistical hypotheses tests.

5. RELATED WORK

Regarding works on software process standards harmonization, Pardo and colleagues [22, 21] have developed a framework for harmonizing multiple-models using ontologies. Their concerns are the same of ours, about standards interoperability. However, whilst our work focuses on the establishment of ontologies for the domains dealt by the standards, the ontology proposed by Pardo et al. (H2mO – Ontology for the Harmonization of multiple-models) focuses on the harmonization domain itself. The main goal of H2mO is the assignment of a formal and clear definition of the most widely used techniques, methods and related terms in harmonization of multiple models [22]. It copes with concepts such as *Harmonization*, *Integration* and *Comparison* to represent the mappings between models. Although H2mO contemplates more specific concepts such as *Process*, *Activity* and *Resource*, they are used only to map the information acquired from the models. Another important difference is about the application focus. H2mO is used for harmonizing different models applied by an organization. The ontology is used to perform comparison operations (intersection, union, difference and complement) between models, resulting in information about the related models, which helps their integrated adoption by organizations. The focus of ISP-OPL is to promote harmonization on the standards level. The main idea is to represent the knowledge about the software process domain in a reusable way to create standard domain ontologies (SDOs), establishing a semantic base of harmonized concepts to guide standards creation and revision.

Concerning ontology patterns, OPL is a new concept, established in [5], and there are few works published. The first one was the Software Process OPL [5], built from a mature Software Process core ontology grounded in UFO [4]. ISP-OPL was built from the ontological analysis of the ISO/IEC 24744 metamodel [24] in the light of UFO, packaging the resulting ontology fragments into patterns to compose the OPL. This process of patterns definition was inspired by the patterns arrangement of SP-OPL, given that both these languages address the same underlying domain. SP-OPL is for general use of software processes and has patterns regarding organizational standard process, software and hardware resources and procedures. In one hand, ISP-OPL has been designed to meet the ISO harmonization initiative needs. Thus, due to the initial priorities of the ISO initiative, these aspects were not included yet in ISP-OPL. On the other hand, ISP-OPL has established finer-grained patterns, and has more specialized human resource patterns. In particular, it details the composition and nature of work products, as well as its participations in work units, and applies a terminology and structure aligned to ISO SC7 standards.

Another related OPL is the one for the Enterprise domain (E-OPL) [6]. Although constructed in a domain that is different from that of ISP-OPL, the ontology reuse intents motivating E-OPL are the same. Moreover, there is an intersection between the software process and enterprise domains regarding human resources. Once we have some analogous requirements, certain E-OPL pattern solutions motivated ISP-OPL patterns. Thus, the E-OPL patterns concerning employment, team definition and human resource membership have inspired the ISP-OPL corresponding patterns, namely *HR Employment*, *Organizational Team Definition*, *Project Team Definition*, *Team Role Definition*, *Team Membership with Role*, and *Team Membership Simplified*, which used a similar solution adapted to the new needs and terminology.

Finally, the ISO ontological framework [9] is also related to this work. The framework does not consider ontology patterns, but provides two mechanisms for ontology derivation. The first one is based on discarding ontology parts. The idea is that the elements in the definitional ontologies are interconnected and the relations between two concepts may have a minimal cardinality of zero. This means that, for any occurrence of the concept on one side of the relation, it may have no occurrence on the other side. In this case, the concept in the opposite side (and the relation) could be discarded in a derived ontology [9]. We think discarding concepts and relations is not a matter of cardinalities, but it is related to the ontology scope and the domain being modeled. Thus, for example, in ISP-OPL, the pattern PPP associates a *Performed Process* to exactly one (1..1) *Project*. If the resulting ontology does not need this relation, or even the concept, the ontology engineer can choose not to use PPP. In this case, independently of the cardinality values, the relation is not established.

A second mechanism used in the ISO ontological framework is the specialization of concepts in the resulting ontology. This mechanism is used there basically in the same way that it is used in OPLs [5], except for its applicability. The difference is that the framework derivation mechanisms are dealing with a whole model, and the OPL solution treats it in a modular way, reusing each of the patterns needed, following the guidance provided by the language.

6. FINAL CONSIDERATIONS

The ISO harmonization efforts have focused on the development of a layered ontological framework, wherein the semantics described by the higher levels (DEO and CDOs) can be propagated to the other levels (SDOs) [9]. In this context, ontology patterns are a promising approach, since they favor reuse of encoded experiences and good practices [23]. Additionally, Ontology Pattern Languages (OPLs) have the potential to amplify the benefits of ontology patterns, by providing guidance through the ontology derivation process [5].

Our main goal is to provide to the ISO framework features of OPLs, guaranteeing an ontologically consistent and standard-adherent basis that can be used to derive interoperable ontologies for ISO standards in a rich reuse process. In order to pursue this goal, we have developed ISP-OPL, the ISO-based Software Process OPL. This OPL is based on ISO recognized software process standards, such as ISO/IEC 24744 and ISO/IEC 12207, and is grounded in the Unified Foundational Ontology (UFO).

We expect that ISP-OPL can be applied for modeling the several software process domains related to ISO SC7. For evaluating ISP-OPL usefulness and applicability, we have conducted an experiment developing seven software process related domain ontologies, using information of selected ISO and other SE standards. Amongst the several findings and improvement points, we could collect evidences to confirm the previously reported practical benefits of the use of OPLs [5, 6]. We have experienced that the guidance provided by the patterns language in the process of developing domain ontologies resulted in an increased productivity in the development process, and a reduction of inconsistence problems in the produced models. Moreover, the resulting models showed to be more compatible with each other. The findings collected in the experiment are fundamental for advancing ISP-OPL in different perspectives. Technically, the pattern models and descriptions can be improved. In terms of usability, many improvements are planned to make easier the application of ISP-OPL. Regarding the harmonization efforts, the domain ontologies developed consist in a rich material to be integrated as an ontology network [29], and used as base for the development of ontologies for other sub-domains and standards. As the main difficulties faced in the experiment, we can quote the matching of the information from different standards, and the effort for integrating the voluminous number of concepts and relations. Some of these issues are helping us to improve ISP-OPL; others can be used to improve the standards themselves, as inputs for the harmonization efforts.

As an ongoing future work, we are improving ISP-OPL with the experiment feedbacks, and we intend to enlarge ISP-OPL by adding new patterns. Our next steps include working on patterns to deal with techniques, software and hardware resources, and the planning of work products. We also plan to apply ISP-OPL for other relevant software standardized domains, increasing the domain representation coverage. With the new patterns and larger coverage, we expect that ISP-OPL can be accepted as an effective solution for the ISO Harmonization Initiative.

7. ACKNOWLEDGMENTS

This research is funded by the Brazilian Research Agencies CAPES, CNPq (Process Numbers 485368/2013-7 and 461777/2014-2) and FAPES (Process Number 64618358/2013).

8. REFERENCES

[1] Allen, J.F. Maintaining Knowledge About Temporal Intervals, *Communications of the ACM*, Vol.26, no. 11, 1983.

[2] Blomqvist, E., Gangemi, A., Presutti, V. Experiments on pattern-based ontology design. *Proc. Fifth International Conference on Knowledge Capture*, ACM, 2009, 41-48.

[3] Bourque, P., and Fairley, R.E. *Guide to the Software Engineering Body of Knowledge* (SWEBOK (R)): Version 3.0. IEEE Computer Society Press, 2014.

[4] Bringuente, A.C., Falbo,R.A., Guizzardi,G. Using a foundational ontology for reengineering a software process ontology. *Journal of Information and Data Management, 2*, 2011, 511-526.

[5] Falbo, R.A., Barcellos, M.P., Nardi, J.C., Guizzardi, G. Organizing ontology design patterns as ontology pattern languages. *Proc. 10th Extended Semantic Web Conference (ESWC'13)*, Montpellier, France, 2013.

[6] Falbo, R.A., Ruy, F.B., Guizzardi, G., Barcellos, M.P., Almeida, J. P. A. Towards an enterprise ontology pattern language. *Proc. 29th Annual ACM Symposium on Applied Computing*, Gyeongju, Korea, 2014, 323-330.

[7] Guizzardi, G. *Ontological Foundations for Structural Conceptual Models*, Universal Press, The Netherlands, 2005.

[8] Guizzardi, G. Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. *33rd International Conference on Conceptual Modeling (ER 2014)*, Atlanta, USA, 2014.

[9] Henderson-Sellers, B., Gonzalez-Perez, C., McBride, T., Low, G. An ontology for ISO software engineering standards: 1) Creating the infrastructure. *Computer Standards & Interfaces, 36*, 3, 2014, 563-576.

[10] IEEE, IEEE Standard for Software Life Cycle Processes – Risk Management, 2001.

[11] ISO/IEC, ISO/IEC 12207. Systems and Software Engineering – Software Life Cycle Processes , 2008.

[12] ISO/IEC, ISO/IEC 14764. Software Engineering – Software Life Cycle Processes - Maintenance, 2006.

[13] ISO/IEC, ISO/IEC 15288. Systems and Software Engineering – System Life Cycle Processes, 2008.

[14] ISO/IEC, ISO/IEC 15504. Information Technology – Process Assessment. Part 1: Concepts and Vocabulary, 2004.

[15] ISO/IEC, ISO/IEC 15939. Systems and Software Engineering – Measurement Process, 2007.

[16] ISO/IEC, ISO/IEC 24744. Software Engineering – Metamodel for Development Methodologies, 2007.

[17] ISO/IEC/IEEE, ISO/IEC/IEEE 15289. Systems and Software Engineering – Content of life-cycle information products (documentation), 2011.

[18] ISO/IEC/IEEE, ISO/IEC/IEEE 29119. Systems and Software Engineering – Software Testing, 2013.

[19] ISO/IEC/IEEE, ISO/IEC/IEEE 29148. Systems and software engineering – Life cycle processes - Requirements engineering, 2011.

[20] Oates, B. J. *Researching Information Systems and Computing*. Sage, 2011.

[21] Pardo, C., Pino, F.J., García, F., Baldassarre, M.T., Piattini, M. From chaos to the systematic harmonization of multiple reference models: A harmonization framework applied in two case studies. *Journal of Systems and Software, 86*, 1, 2013, 125-143.

[22] Pardo, C., Pino, F.J., García, F., Piattini, M., Baldassarre, M.T. An ontology for harmonization of multiple standards and models. *Computer Standards & Interfaces, 34*, 2012, 48-59.

[23] Presutti, V., Daga, E., Gangemi, A., Blomqvist, E. eXtreme design with content ontology design patterns. *Proc. Workshop on Ontology Patterns*, Washington, EUA, 2009.

[24] Ruy, F.B., Falbo, R.A., Barcellos, M.P., Guizzardi, G. An Ontological Analysis of the ISO/IEC 24744 Metamodel. *Proc. 8th International Conference on Formal Ontology in Information Systems (FOIS'14)*, Rio de Janeiro, Brazil, 2014.

[25] Ruy, F.R., Reginato, C.C., Santos, V.A., Falbo, R.A., Guizzardi, G. Ontology Engineering by Combining Ontology Patterns. *Proc. 34th International Conference on Conceptual Modeling (ER 2015)*, Stockholm, Sweden, 2015.

[26] Santos, G., Kalinowski, M., Rocha, A.R., Travassos, G.H., Weber, K.C., Antonioni, J.A. "MPS.BR program and MPS model: main results, benefits and beneficiaries of software process improvement in Brazil", *In 8th Int. Conf. on the Quality in Information and Communications Technology* (QUATIC), Lisbon, Portugal, 2012.

[27] Scherp, A., Saathoff, C., Franz, T., Staab, S. Designing core ontologies. *Applied Ontology, 6*, 3, IOS Press, 2011, 177-221.

[28] SEI, *CMMI for Development,* Version 1.3. CMU/SEI-2010-TR-033. Software Engineering Institute, Carnegie Mellon University, 2010.

[29] Suárez-Figueroa, M.C. *NeOn Methodology for building ontology networks: specification, scheduling and reuse*. PhD thesis, Universidad Politécnica de Madrid, Spain, 2010.

ABOUT THE AUTHORS:

Fabiano Borges Ruy is PhD student at Federal University of Espírito Santo in Brazil, researching Standards Harmonization and Ontology Engineering at the Ontology and Conceptual Modeling Research Group (NEMO). He holds a Master Degree in Informatics from the same university, and is an Assistant Professor of Software Engineering at the Federal Institute of Espírito Santo. In the last twelve years, he accumulated experience teaching, consulting and researching Software Engineering related themes, especially Software Processes, Development Methodologies, Object Technologies, and Ontologies.

Ricardo de Almeida Falbo received his PhD degree in System Engineering and Computer Science from Federal University of Rio de Janeiro, Brazil, in 1998. Now he is a Full Professor at Federal University of Espírito Santo, Brazil, and a senior member of the Ontology and Conceptual Modeling Research Group (NEMO). He has been doing research in the areas of Ontology and Software Engineering for the past two decades and has published over 130 publications in these areas (including 11 award-wining publications).

Monalessa Perini Barcellos holds a doctorate in Systems Engineering and Computer from the Federal University of Rio de Janeiro, Brazil. She is an associate professor at the Computer Science Department of the Federal University of Espírito Santo, in Brazil. She is member of the Ontology and Conceptual Modeling Research Group (NEMO). Her research focuses on Ontologies and Software Engineering, particularly on Software Engineering Environments, Software Quality, Software Measurement, Software Project Management and Semantic Interoperability. She is member of the Technical Team of the MR MPS.BR (Reference Model for Brazilian Software Process Improvement).

Giancarlo Guizzardi holds a PhD in Computer Science from the University of Twente, in The Netherlands and is a senior Associate Professor of Computer Science at the Federal University of Espírito Santo in Brazil. He has been doing research in the areas of ontology and conceptual modeling and has published over 170 publications in these areas (including 9 award-wining publications). He has contributed to these communities in roles such as keynote speaker, general chair, and PC Chair in conferences such as CAISE, ER, FOIS, SLE and IEEE EDOC. He is an associate editor of the Applied Ontology journal, and a member of the advisory board of the International Association for Ontology and its Applications (IAOA).

Glaice Kelly Quirino has received a B.Sc. degree in Computer Engineering from the Federal University of Espírito Santo, Brazil. She is currently a M.Sc. student in Informatics at the Department of Computer Science at the same university. She is member of the Ontology and Conceptual Modeling Research Group (NEMO). Her major research interests focus on Ontology Pattern Languages, ontology and software engineering, and conceptual modeling.