SEON: A Software Engineering Ontology Network

Fabiano Borges Ruy[1,3], Ricardo de Almeida Falbo[1], Monalessa Perini Barcellos[1], Simone Dornelas Costa[1,2], Giancarlo Guizzardi[1]

[1]Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo, Vitória, Brazil
[2]Computer Department, Federal University of Espírito Santo, Campus Alegre, Alegre, Brazil
[3]Informatics Department, Federal Institute of Espírito Santo, Campus Serra, Serra, Brazil
{fabianoruy, falbo, monalessa, gguizzardi}@inf.ufes.br,
sidornellas@gmail.com

Abstract. Software Engineering (SE) is a wide domain, where ontologies are useful instruments for dealing with Knowledge Management (KM) related problems. When SE ontologies are built and used in isolation, some problems remain, in particular those related to knowledge integration. The goal of this paper is to provide an integrated solution for better dealing with KM-related problems in SE by means of a Software Engineering Ontology Network (SEON). SEON is designed with mechanisms for easing the development and integration of SE domain ontologies. The current version of SEON includes core ontologies for software and software processes, as well as domain ontologies for the main technical software engineering subdomains, namely requirements, design, coding and testing. We discuss the development of SEON and some of its envisioned applications related to KM.

Keywords. Ontology Network · Ontology Engineering · Software Engineering · Ontology Integration · Knowledge Management

1 Introduction

Software process is a knowledge-intensive process involving many people working in different sub-processes and activities. Moreover, knowledge in Software Engineering (SE) is diverse and organizations have problems to capture, retrieve, and reuse it. An improved use of this knowledge is the basic motivation and driver for Knowledge Management (KM) in SE [1].

Ontologies have been widely recognized as a key enabling technology for KM. They are used for establishing a common conceptualization of the domain of interest to support knowledge representation, integration, storage, search and communication [2]. However, some domains that are the target of KM initiatives are often large and complex. This is the case of SE. If we try to represent the whole domain as a single ontology, we will achieve a large and monolithic ontology that is hard to manipulate, use, and maintain [3]. On the other hand, representing each subdomain separately would be too costly, fragmented, and again hard to handle.

SE comprises several interrelated subdomains such as Requirements, Design, Coding, Testing, Project Management, and Configuration Management. In one hand, there are few works in the literature that aims at developing ontologies covering wide portions of the SE domain, such as [4, 5, 6]. On the other hand, it is easy to find many specific ontologies modeling SE subdomains [7, 8, 9, 10, 11]. However, in general, these subdomain ontologies are weakly or even not interrelated, and they are often applied in isolation. In this context, it is important to notice that SE subdomains share concepts, ranging from general (e.g. *Artifact*, *Process*) to ones that are more specific (e.g. *Functional Requirement*, *Test Case*). This striking feature of the SE domain must be considered while representing it. For achieving consistent SE ontologies, concepts and relations should keep the same meaning in any related ontology.

D'Aquin and Gangemi [12] point out a set of characteristics that are presented in "beautiful ontologies", from which we detach the following ones: having a good domain coverage; being modular or embedded in a modular framework; being formally rigorous; capturing also non-taxonomic relations; and reusing foundational ontologies. Most of the existing SE ontologies do not exhibit such characteristics. We believe that an integrated ontological framework, built considering them, can improve ontology-based applications in SE, in particular those related to KM. In such integrated ontological framework, there must be ways for creating, integrating and evolving related ontologies. Thus, we advocate that these ontologies should be built incrementally and in an integrated way, as a network.

An *Ontology Network* (ON) is a collection of ontologies related together through a variety of relationships, such as alignment, modularization, and dependency. A *networked ontology*, in turn, is an ontology included in such a network, sharing concepts and relations with other ontologies [3].

To truly enjoy the benefits of keeping the ontologies in a network, we need to take advantage of the existing resources available in the ON for gradually improving and extending it. Thus, an ON should have a robust base equipped with mechanisms to help its evolution. In our view, an ON should be organized in layers. Briefly, in the background, we need a foundational ontology[1] to provide the general ground knowledge for classifying concepts and relations in the ON. In the center of the ON, core ontologies[2] should be used to represent the general domain knowledge, being the basis for the subdomain networked ontologies. Ideally, these core ontologies should be organized as Ontology Pattern Languages (OPLs) [13] for easing reusing model fragments (ontology patterns) while developing subdomain networked ontologies. Finally, going to the borders, (sub) domain ontologies appear, describing the more specific knowledge.

---

[1] Foundational ontologies span across many fields and model the very basic and general concepts and relations that make up the world, such as object, event, parthood relation etc. [14].

[2] Core ontologies provide a precise definition of structural knowledge in a specific field that spans across different application domains in this field. These ontologies are built based on foundational ontologies and provide a refinement to them by adding detailed concepts and relations in their specific field [15].

In this paper, we present SEON, a Software Engineering Ontology Network. SEON provides a well-grounded network of SE reference ontologies[3], and mechanisms to derive and incorporate new integrated subdomain ontologies into the network. The main goals of this work are: to define a layered architecture for SEON that enables supporting network growing; to introduce SEON and its mechanisms to create and integrate SE subdomain ontologies; and to discuss the use of SEON for supporting KM in SE. Due to space limitations, only small portions of SEON are presented here. The current specification is available at nemo.inf.ufes.br/projects/seon, where a machine processable lightweight version implemented in OWL is also available.

This paper is organized as follows. Section 2 discusses SE ontologies. Section 3 presents SEON and how it builds up from foundational to domain ontologies. Section 4 discusses how SEON can be used to support applications of KM in SE. Section 5 discusses related works. Finally, Section 6 presents our final considerations.

2 Developing Software Engineering Ontologies

A variety of ontologies have been developed modeling the SE domain. According to Calero et al. [7], these ontologies can be classified as: Generic SE Ontologies, having the ambitious goal of modeling the complete SE body of knowledge; or Specific SE Ontologies, attempting to conceptualize only part (a subdomain) of this discipline. Concerning Generic SE Ontologies, Mendes and Abran [4] propose a SE ontology consisting of an almost literal transcription of the SWEBOK [16] text, with over 4000 concepts. Sicilia and colleagues [5] propose an ontology structure to characterize artifacts and activities, also based on SWEBOK. Wongthongtham and colleagues [6] propose an ontology model for representing the SE knowledge, based on SWEBOK [16] and Sommerville's Software Engineering book [17]. Considering the Specific SE Ontologies, a great number of ontologies is available, representing a variety of SE subdomains, such as Software (e.g. [18, 19]), Software Processes (e.g. [9, 10]), Software Requirements (e.g. [20]), Software Testing (e.g. [8]), and Software Configuration Management (e.g. [11]). For others, see [7].

Some of the specific domain ontologies are developed considering their integration with others [7]. Taking this to the extreme, the combination of ontologies of all SE subdomains would result in an ontology of the complete SE domain. Unfortunately, the reality is that this goal is extremely laborious, not only due to its size, but also due to the numerous problems related to ontology integration and merging [7], such as overlapping concepts, diverse foundational theories, and different representation and description levels, among others. In sum, SE comprises a set of highly interconnected subdomains. This interrelated nature affects any possible representation of the SE domain, and the situations in which it can be applied. Despite of the challenges involved, an ontological representation covering a large extension of the SE domain remains a desired solution.

[3] A reference ontology is constructed with the goal of making the best possible description of the domain in reality, representing a model of consensus within a community, regardless of its computational properties [18].

In this context, the notion of Ontology Network (ON) [3] applies. The scenario for ONs is radically different from the relatively narrow contexts in which ontologies have been traditionally developed and applied [3]. For instance, the *NeOn Methodology Framework* [3] provides guidance for engineering networked ontologies, making available detailed processes, guidelines and different scenarios for collaboratively building networked ontologies. For building SEON, we have applied some of the NeOn methodological guidelines, in particular those referring to ontology modularization and evaluation, and to the adoption of a pattern-based design approach. Moreover, we complement these valuable guidelines by defining an architecture for SEON that we believe applies for ONs in general. In the next section, we present the SEON architecture and some of the ontologies that comprise it. Besides, we discuss how to develop or integrate new SE subdomain ontologies to SEON.

3 SEON: The Software Engineering Ontology Network

SEON results from several efforts on building ontologies for the SE field. Although SEON itself is a new proposal, the studies and ontologies that our group has developed along the years were important contributions for defining SEON. Hence, SEON rises with three main premises: (i) being based on a well-founded grounding for ontology development; (ii) offering mechanisms to easy building and integrating new SE subdomain ontologies to the network; and (iii) promoting integration by keeping a consistent semantics for concepts and relations along the whole network. SEON architecture is organized considering three ontology generality levels, as Fig. 1 shows.

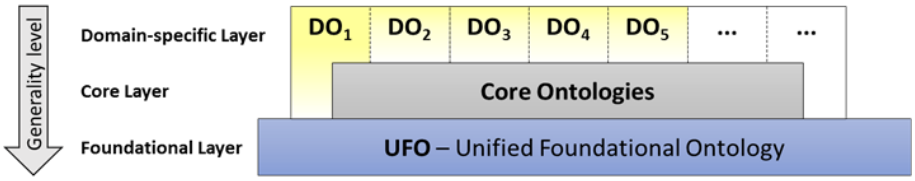

Fig. 1. SEON Architecture.

Foundational layer: at the bottom of SEON, is the Unified Foundational Ontology (UFO), which is developed based on a number of theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology. UFO is divided in three parts: an ontology of endurants (objects) [21], an ontology of perdurants (events) [22], and an ontology of social entities [23]. UFO's ontological distinctions are used for classifying SEON concepts, e.g., as objects, actions, commitments, agents, roles, goals and so on. UFO provides the necessary grounding for the concepts and relations of all networked ontologies.

Core layer: in the center of SEON, there are two core ontologies: the Software Ontology (SwO) and the Software Process Ontology (SPO) [10]. SwO is a core ontology developed based on the work of Wang and colleagues [20], and captures that software products have a complex artifactual nature, being constituted by software artifacts (here called software items) of different nature, namely software systems, programs and code. SPO is a core ontology, also grounded in UFO, aiming at estab-

lishing a common conceptualization on software processes. SPO builds upon SwO, and its current version is organized as an Ontology Pattern Language (OPL)[4] [13]. SPO scope is broader, embracing the following aspects of the software process domain: standard, project and performed processes and their activities, artifacts handled, resources used and procedures adopted by activities, team membership, and stakeholders allocation and participation in activities. For dealing with aspects related to organizations (such as team membership), SPO builds upon a core ontology on enterprises [24], which we consider external to SEON. SPO has been developed for more than two decades, and used as basis to develop several ontologies for many SE subdomains (e.g., [8], [11], [25]).

Domain-specific layer: over the foundational and core layers, SEON places the domain ontologies. Each networked ontology is grounded in SwO/SPO and also in UFO, and encompasses a SE subdomain (e.g., software requirements, design, configuration management, and measurement). Although not represented in Fig. 1, more specific subdomains ontologies can be developed based on other more general subdomain ontologies. For instance, an ontology on requirements at runtime [26] was developed based on the Reference Software Requirements Ontology.

In a nutshell, the foundational ontology offers the ontological distinctions for the core and domain layers, while the core layer offers the SE core knowledge for building the domain networked ontologies. This way of grounding the ontologies in the network is helpful for engineering the networked ontologies, since it provides ontological consistency and makes a number of modeling decisions easier. The SEON building mechanism also takes advantage of ontology patterns by representing its core layer in a pattern-oriented way. SPO is organized as an OPL, in order to become strongly modular, flexible and reusable. Thus, the ontology engineer can explore alternative models in the design of specific ontologies for the various SE subdomains, select the ontology fragments relevant to the problem in hands and reuse them [13]. Fig. 2 shows a fragment of SPO with some of its patterns. This fragment deals with patterns for performed processes and activities; artifacts created, changed or used by those performed activities; and stakeholder participation. Colored blocks are used to delimit each pattern in this figure. During domain ontology development, the ontology engineer selects useful patterns and extends their concepts and relations in the networked ontology (Fig. 4 shows the case for the Requirements Development Process Ontology). In the cases where a domain element is not covered by the core ontologies (SwO/SPO), this domain-specific element should be grounded directly in the foundational ontology (UFO).

The ontology fragments of Fig. 2 and the following are represented in 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) [21].

---

[4] An OPL is a network of interconnected Domain-Related Ontology Patterns (DROPs) that provides holistic support for solving ontology development problems for a specific domain. Besides the DROPs, it contains a process guiding how to use and combine them in a specific order, and suggesting patterns for solving the modeling problems in that domain [14].

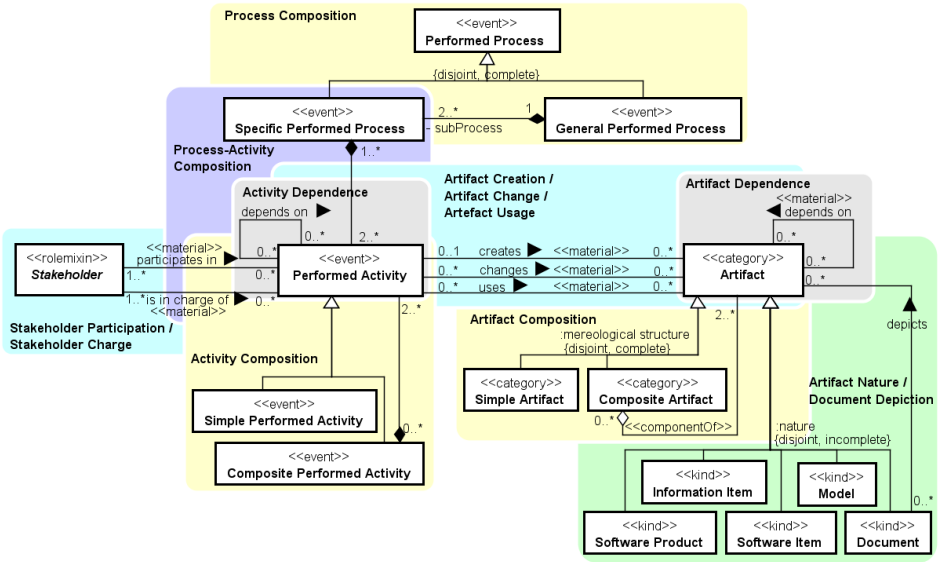

Fig. 2. A portion of SPO, fragmented in patterns.

By reusing the OPL patterns from the core layer, the development of the domain ontologies is faster and the resulting models are more consistent and uniform [27]. The core ontologies, sustained by the foundational ontology, offer a standardized way for describing all the other elements in the network. Thus, since all the domain networked ontologies inherit the same core and foundational grounds, concepts and relations with the same classification have a common and identifiable background. This is a fundamental aspect for ontology integration.

As networked ontologies are developed and added to SEON, we still need to work on integrating them. Although the domain-specific ontologies share the same conceptual basis, given by the foundational and core ontologies, they still need to be aligned with respect to their specific knowledge, making possible to merge networked ontologies in a meaningful way, by representing information in one ontology in terms of the entities in another [3].

The SEON integration mechanism adopts some alignment guidelines for matching and integrating networked domain ontologies. First, ontologies should be compared looking for equivalent concepts. Since the domain ontologies are produced from the same basis (UFO and SwO/SPO), two concepts can only be considered equivalent if they have the same base type, restricting the search field and speeding up the integration process. Thus, for example, artifacts are compared only with artifacts, performed activities with performed activities, and so on. If concepts have a partial matching, this could mean that one concept is a specialization or a part of another.

Two concepts from distinct ontologies can also have a relationship between them. In this case, it is worth analyzing if there is a relationship to be extended from the base ontologies or a new relationship should be included in the subdomain ontology. From this matching, we can determine the correlation level between the ontologies.

Fig. 3 shows the current status of SEON. Each circle represents an ontology. The circles' sizes vary according to the ontologies' sizes in terms of number of concepts, (represented inside the circles in parenthesis). Lines denote links between integrated ontologies, and line thickness represents the coupling level between them in terms of number of relationships between concepts in different ontologies. Blue circles represent the core ontologies; green circles, domain ontologies already integrated to SEON; and gray circles are domain ontologies already developed using UFO and SPO, but not integrated to SEON yet. Due to the wideness and complexity of the SE domain, it requires a continuous and long-term effort. Thus, we expect SEON to continuously evolve, with ontologies being added and integrated incrementally.

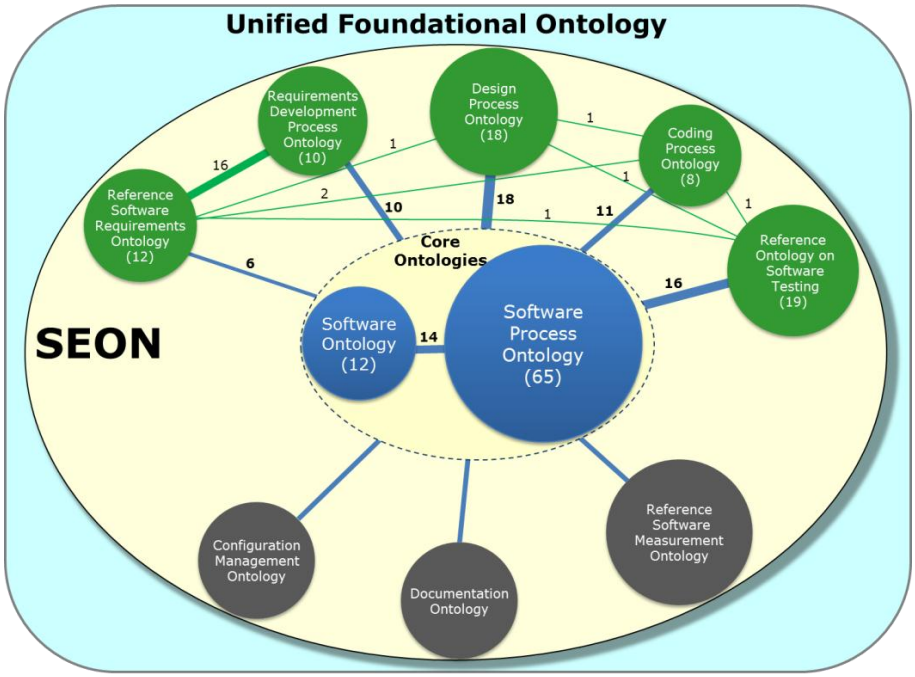

Fig. 3. SEON: The Network view.

It is important to notice that, even adopting a layered architecture (see Fig. 1), SEON is a network. Like so, each new added node contributes for the whole network. When a new ontology is added, it should reuse existing elements. Other ontologies, in turn, may be adapted to keep consistency, in order to share the same semantics along the whole network. Even the core ontologies can evolve to adapt or incorporate new patterns discovered when domain ontologies are created or integrated.

Concerning the ontologies that comprise SEON, we should highlight that they have been developed or reengineered using SPO along the time, some of them before SEON conception. Among them, there are ontologies for the following SE subdomains: Software Requirements, Design, Testing [8], Configuration Management [11], Documentation, and Software Measurement [25]. Currently, as Fig. 3 shows, five of them (those addressing the main technical SE subdomains) are integrated to SEON.

The Reference Software Requirements Ontology (RSRO) is centered in the notion of requirement as a goal to be achieved, and addresses the distinction between functional and non-functional requirements, and how requirements are documented in proper artifacts, among others. It is mainly based on SwO.

The other four domain ontologies focus on describing the technical processes, considering its main assets, and extend SPO. The Requirements Development Process Ontology (RDPO), partially shown in Fig. 4, describes the requirements development process. The top of the figure shows the SPO concepts and relations (as presented in Fig. 2) that are reused from the selection of the suitable ontology patterns from SPO. Thus, RDPO concepts and relations (shown in Fig. 4 below the dotted line) are mostly specialized from SPO. **Requirements Reviewer**, **Requirements Stakeholder** and **Requirements Engineer** extend ***Stakeholder***; **Requirements Development Process** extends ***Specific Performed Process***, and is decomposed into five ***Composite Performed Activities***. These activities are responsible for producing ***Artifacts*** such as the **Requirements Document**, which depicts **Conceptual Models** and is composed of **Documented Requirements**. The concepts in gray below the line are imported from RSRO and integrated into RDPO.

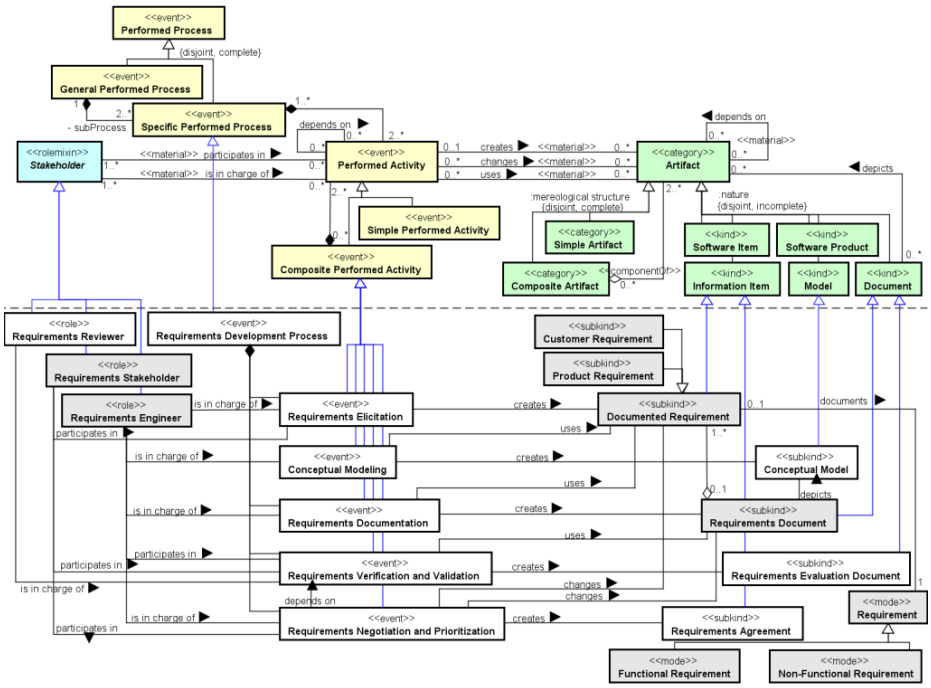

Fig. 1. The Requirements Development Process Ontology (RDPO), with SPO elements.

The Design Process Ontology (DPO) focuses on the architectural and detailed design processes. Concepts and relations in DPO extend the same portion of SPO presented before. The main artifact produced during design is the **Design Document**. This ontology, as the next one, is not shown here due to space limitations.

The Coding Process Ontology (CPO) deals with building the software code based on the requirements and design documents. As the other ontologies, it extends the same portion of SPO. The main CPO product is the **Code** ***Artifact***.

The Reference Ontology on Software Testing (ROoST) [8] addresses the software testing process. Fig. 5 shows a fragment of ROoST, focusing on the testing process and the used and produced artifacts. Although core ontology elements are not presented in Fig. 5, ROoST elements are also specializations from SPO, gotten by reusing patterns [8]. **Test Manager** and **Tester** ***Stakeholders*** participate in ***Performed Activities*** of the **Testing Process**. **Test Planning** creates the **Test Plan**. The four activities **Test Case Design**, **Test Coding**, **Test Execution** and **Test Result Analysis** are performed at the **Unit**, **Integration** and **System Testing** levels, producing testing process artifacts. For example, **Test Case Design** creates **Test Cases**, using **Test Case Design Inputs**, a role that can be played by ***Artifacts*** used as input for the test case design.

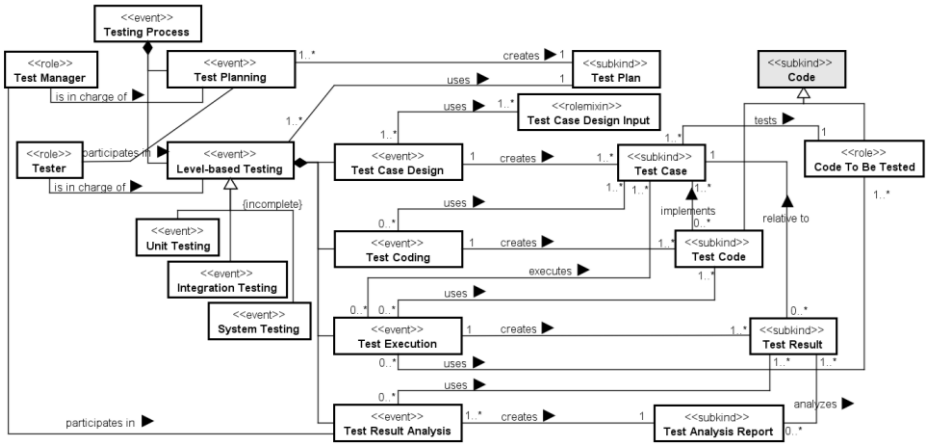

Fig. 5. A fragment of ROoST [8].

Besides the ontologies for the technical SE subdomains already integrated to the network, we expect SEON to continuously grow by adding other SE subdomains ontologies. The SEON integration mechanism has three different ways to incorporate new ontologies into the network, considering the origin of the ontology to be integrated.

In a first situation, consider a new ontology that is created based on UFO and SwO/SPO, and also taking other existing networked ontologies into account. Besides the extensions made from the core ontologies, this ontology tends to use also the related concepts already defined in the other networked ontologies. This situation occurs in RDPO (Fig. 4), which imports concepts from RSRO (the gray ones below the line). This is the best way for increasing SEON, since it reduces modeling and integration efforts, by reusing already defined elements.

The second situation occurs when domain ontologies are developed based on UFO and SwO/SPO, however, independently of the other subdomain networked ontologies. In this situation, although the domain ontology to be integrated to the network shares the same basis of the SEON domain ontologies (UFO/SwO/SPO), some additional integration effort is still required, in order to adapt the common parts focusing on a shared representation. This happened when we integrated ROoST to SEON. ROoST

was developed based on SPO and UFO, but disregarding the other domain ontologies already integrated to SEON. This way, while integrating ROoST, we had to align it with the other existing networked ontologies. Fig. 6 shows a fragment of the integrated model, encompassing elements from four domain networked ontologies: RSRO, DPO, CPO and ROoST. It shows the activities of coding and test case design, and related artifacts. Most of the concepts and relations shown (as the activities for coding and testing) are just imported from their original ontologies. However, some concepts required further decisions. This is the case of the inputs for the **Test Case Design** activity. The **Test Case Design Input** concept is a general role that can be played by different types of ***Artifacts*** able to be used as inputs for that activity. In this case, the suitable artifacts are the ones used for creating the code (**Requirements Document**, **Design Document**) and the **Code** itself, giving rise to three new concepts, specializations of these three artifacts playing the **Test Case Design Input** role.

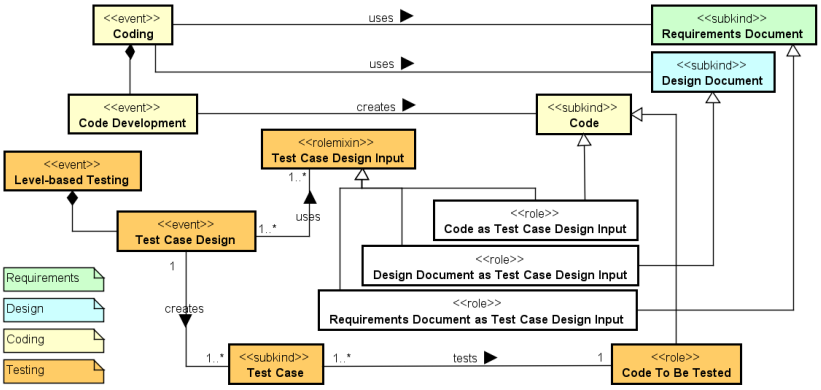

Fig. 2. An Integrated SEON Fragment.

Finally, the third integration situation happens when external ontologies, developed without taking SwO, SPO or UFO as basis, need to be integrated to SEON. In this case, if we have access to modify the ontology, we need to perform an ontological analysis and reengineering before the integration. By this process, the ontology elements are analyzed and adapted to the UFO distinctions and SwO/SPO domain knowledge. The knowledge represented by the ontology is then preserved, but the representation is adjusted for a better integration into SEON. On the other hand, if the ontology cannot be modified, we have to make the necessary links and adaptations only in the SEON side. In this case, techniques for ontology alignment, as discussed in [3], apply. Currently, we do not have any external ontology integrated to SEON.

4 SEON Envisioned Applications

There are several ontology-based initiatives of applying KM in SE. Some of them adopt centralized KM solutions (e.g., KM systems with large centralized knowledge repositories), such as [28]; others focus on distributed KM solutions, such as the ones using Semantic Wikis [29]. In both cases, ontologies are used to support knowledge representation (e.g., by categorizing or annotating knowledge items), integration,

search, and retrieval. However, in KM scenarios spanning different SE subdomains, we need to integrate several ontologies for these subdomains. These are the cases where the benefits of using SEON stand out.

Since the begging of the 2000's, we have been working on ontology-based KM systems to support SE tasks. We started by developing an ontology-based KM infrastructure for a Software Engineering Environment (SEE) [28]. This infrastructure evolved, and more recently, we separate it from the SEE, transforming it in a SE Knowledge Management Portal (SE-KMP). SE-KMP provides general features for managing and assessing knowledge items (including lessons learned, and discussion packages), as well as yellow pages. SE-KMP was extended to manage knowledge related to software testing, in a more specific KM Portal, called TKMP [30]. However, we perceived that, to truly provide benefits for KM in SE, SE-KMP requires integrated ontologies for the several SE subdomains. In fact, this application motivated us to seek for an approach for developing integrated SE domain ontologies, leading to SEON.

In another front of research, we have been working on semantic documentation in SE, by providing an Infrastructure for Managing (SE) Semantic Documents (IMSD) [31]. Semantic documents aim at combining documents and ontologies, and allowing users to access their knowledge in multiple ways. The ultimate goal of semantic documents is not merely to provide metadata for documents, but to integrate documentation and knowledge representation in a way that they use a common structure [32]. We started by annotating requirements documents, and we used a previous versions of the Reference Software Requirements Ontology (RSRO) for this purpose. However, this ontology is not enough to annotate other documents, such as design specifications, source code, and test cases. In particular, for providing information traceability among these artifacts, we need, besides RSRO, other ontologies for design, coding and testing. Moreover, these ontologies need to be integrated. We perceived this clearly when decided to handle in IMSD, besides Requirements Documents, Test Cases. We needed to integrate RSRO to ROoST [8] (the reference ontology on software testing) [31]. With networked ontologies, as in SEON, this effort would not be required. In fact, this scenario of semantic documentation motivated us to start SEON with the set of networked domain ontologies shown in Fig. 3.

Another related scenario for applying SEON is tool integration. Ontologies can be used for semantically integrating heterogeneous tools [11]. Considering different tools working with elements of the same subdomain, an ontology for this subdomain is enough for addressing most of the issues. We have experienced this situation in [11], when we used a Software Configuration Management Ontology for integrating a version control system and a change management tool. However, for more complex contexts, involving several tools, supporting tasks in different SE subdomains, a single ontology is not enough. For instance, to include knowledge items (test cases) in TKMP, we had to import information from different software tools, namely: TestLink, a web-based test management system, and MantisBT, a bug tracking system. To automate this task, we needed to integrate data from both tools.

Addressing ontology integration case by case is an arduous and exhausting task. By integrating these ontologies in SEON, any new initiative that needs to commit to those ontologies could benefit from the efforts already done.

5 Related Works

Regarding the ontologies aiming at covering a large extension of the SE domain [4, 5, 6], in general, they present many concepts usually based on acknowledged references such as SE books or reference models (e.g. [16, 17]). Comparing to SEON, the first notable difference is the source for building the ontologies. These Generic SE ontologies use to be based on a few number of sources, in some cases nearing to transcriptions of the referenced source [4]. Contrariwise, each SEON ontology is built based on a set of references, often considering books and standards of the specific (sub)domain. Besides that, the knowledge from the base layers configures as one more source for building the networked ontologies. A second difference regards modularity, since the networked ontologies, even integrated, can be seen, and used, as separated models. Finally, the most important difference regards the mechanisms provided to build SEON incrementally, supported by the foundational and core layers and their patterns. In sum, SEON design considers important characteristics of "beautiful ontologies", as discussed in [12], such as: having a good domain coverage; considering international standards; being modular; being formally rigorous; capturing also non-taxonomic relations; and reusing foundational ontologies.

Concerning SEON ontologies, due to space limitations, it is not possible to contrast all of them with others already published in the literature. Thus, here we decided to compare only SEON's core ontologies (SwO and SPO). Regarding SwO, related work includes the software ontologies presented in [18] and [19]. The Core Software Ontology (CSO) [18] detaches. Like SwO, CSO is rigorously formalized, grounded in a foundational ontology (DOLCE, while SwO is grounded in UFO), and was built following a pattern-based approach. Moreover, these two ontologies share concerns related to the polysemy of the concept of software, and in this sense they present similar distinctions. We should highlight that CSO has a broader scope than SwO, addressing concepts of object orientation, such as classes, interfaces and methods, as well as representing workflow information. These aspects are not covered by SwO, because we intend to address them in another networked ontology. The Software Ontology presented in [19] is organized in several modules, addressing aspects related to software and relationships to software process, license, and versions, among others. In this sense, the proposed ontology is related not only to SwO, but also to other SEON's ontologies, such as SPO. It is worthwhile to point out that the Software Ontology presented in [19] is not grounded in a foundational ontology.

Regarding SPO, there are some ontologies on software processes published in the literature. Here, due to space limitations, we compare SPO only with the Ontology for Software Development Methodologies and Endeavors (OSDME) [9]. This choice justifies because OSDME is the basis for an international standard (ISO 24744). OSDME addresses aspects related to process, product and producers. SPO has similar coverage. However, SPO is organized as an Ontology Pattern Language to favor reuse, and is grounded in a foundational ontology (UFO). As discussed in [33], the lack of truly ontological foundations leads to some inconsistencies in OSDME, which are solved in SPO, as discussed in [10]. This reinforces the importance of using a foundational ontology as basis for grounding core and domain ontologies in SEON.

Considering Ontology Networks, in [3], three case studies in the fishery and pharmaceutical domains are presented. Three ONs were developed using NeOn methods and technologies. In general, these ONs are composed of ontologies (expressed in OWL) plus non-ontological resources (such as thesauri). Mappings are an important means to relate the networked ontologies. In two of the studies, the network resources were organized according to the ontologies' types and levels, considering general ontologies (e.g., upper level ontologies or ontologies for time and objects) as independent of the focused domain, and ontologies as references for the domain and basis for providing concepts or relating more specific ontologies. Although the similarities regarding the generality levels, SEON states an architecture with well-defined layers, and it is based on ontological foundations and patterns, facilitating the building and integration of new domain ontologies. We should highlight that SEON's architecture is aligned to the one adopted in the ONIONS Project [34] and further with the ontological architecture proposed by Obrst [35].

6 Final Considerations

Ontologies are a key enabling technology for KM in SE. However, knowledge in SE is diverse and interlinked. For dealing with richer KM scenarios, addressing several SE subdomains, we need integrated ontologies. An ontology network can provide such integrated solution. Thus, in this paper, we presented SEON, a Software Engineering Ontology Network. SEON is designed seeking for: (i) taking advantage of well-founded ontologies (all its ontologies are ultimately grounded in UFO); (ii) providing ontology reusability and productivity, supported by core ontologies organized as Ontology Pattern Languages; and (iii) solving ontology integration problems by providing integration mechanisms. Diverse initiatives can benefit from the use of SEON, especially the ones where the focus is semantic interoperability and that involve a number of related SE subdomains. In this paper, we have explored KM-related scenarios, but SEON can also be used to address other scenarios that demands integrated SE ontologies.

In its current version, SEON includes core ontologies for software and software processes, as well as domain ontologies for the main technical software engineering subdomains, namely requirements, design, coding and testing. Other SE domain ontologies should be developed and integrated to SEON to enlarge its coverage. SEON success criteria relate to how easy SEON grows by incorporating new ontologies, and how successful is to apply SEON to solve integration focused problems (such as standard harmonization and tool integration, besides KM). As ongoing work, we are working on incorporating to SEON already developed SE subdomain ontologies for software documentation, measurement, configuration management and project management. Regarding SEON applications, our research agenda includes: a Standard Harmonization Approach supported by SEON; efforts on using SEON-based annotations in semantic documents, allowing integrating information scattered in multiple documents; and using SEON for semantic integration of SE tools, extending [11].

7 Acknowledgments

This research is funded by the Brazilian Research Funding Agency CNPq (Processes 485368/2013-7 and 461777/2014-2) and FAPES (Process 69382549/14).

8 References

1. Rus, I., Lindvall, M.: Knowledge Management in Software Engineering, IEEE Software, pp. 26-38, May/June (2002)
2. O'Leary, D.: Using AI in knowledge management: knowledge bases and ontologies, IEEE Intelligent Systems, vol. 13, pp. 34-39 (1998)
3. Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A.: Ontology Engineering in a Networked World. Springer Science & Business Media (2012)
4. Mendes, O., Abran, A.: Issues in the Development of an Ontology for an Emerging Engineering Discipline. First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Eng. (ONTOSE). Alcalá Henares, Spain (2005)
5. Sicilia, M.A., Cuadrado, J.J., García, E., Rodríguez, D., Hilera, J.R.: The Evaluation of Ontological Representation of the SWEBOK as a Revision Tool. In: 29th Int. Computer Software and Application Conference (COMPSAC), pp. 26-28. Edinburgh, UK, (2005)
6. Wongthongtham, P., Chang, E., Dillon, T., Sommerville, I.: Development of a Software Engineering Ontology for Multisite Software Development. IEEE Transactions on Knowledge and Data Engineering, v. 21, n. 8, pp. 1205-1217 (2009)
7. Calero, C., Ruiz, F., Piattini, M. (editors): Ontologies for Software Engineering and Software Technology. Springer Science & Business Media (2006)
8. Souza, E.F., Falbo, R.A., Vijaykumar, N.L.: Using Ontology Patterns for Building a Reference Software Testing Ontology. In: 17th IEEE Int. Enterprise Distributed Object Computing Conference Workshops (EDOCW), pp. 21-30. Vancouver (2013)
9. González-Pérez, C., Henderson-Sellers, B.: An Ontology for Software Development Methodologies and Endeavours. In: [7] (2006)
10. Bringuente, A.C., Falbo, R.A., Guizzardi, G.: Using a Foundational Ontology for Reengineering a Software Process Ontology. Journal of Information and Data Management, v. 2, n. 3, pp. 511 (2011)
11. Calhau, R.F., Falbo, R.A.: An Ontology-based Approach for Semantic Integration. In: 14th IEEE International Enterprise Distributed Object Computing Conference, Vitória, Brazil. Los Alamitos: IEEE Computer Society, pp. 111-120 (2010)
12. d'Aquin, M., Gangemi, A.: Is there beauty in ontologies? Applied Ontology, vol. 6, n.3, p. 165–175 (2011)
13. Falbo, R.A., Barcellos, M.P., Nardi, J.C., Guizzardi, G.: Organizing Ontology Design Patterns as Ontology Pattern Languages. In: Proceedings of the 10th Extended Semantic Web Conference - ESWC 2013. Montpellier, France (2013)
14. Guarino, N.: Formal Ontology and Information Systems. In: Guarino, N. (ed.) Formal Ontology and Information Systems, pp. 3–15, IOS Press, Amsterdam (1998)
15. Scherp, A., Saathoff, C., Franz, T., Staab, S.: Designing Core Ontologies. Applied Ontology, v. 6, n.3, pp. 177–221 (2011)
16. Bourque, P., Fairley, R.E.: Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press (2014)
17. Sommerville, I.: Software Engineering. International Computer Science Series. Ed: Addison Wesley (2004)

18. Oberle, D., Grimm, S. & Staab, S.: An ontology for software. In Handbook on Ontologies, 2nd ed. Berlin: Springer-Verlag (2009)
19. Malone, J., Brown, A., Lister, A.L., Ison, J., Hull, D., Parkinson, H., & Stevens, R.: The Software Ontology (SWO): a resource for reproducibility in biomedical data analysis, curation and digital preservation. Journal of Biomedical Semantics, 5, 25 (2014)
20. Wang, X., Guarino, N., Guizzardi, G., Mylopoulos, J.: Towards an Ontology of Software: a Requirements Engineering Perspective. Proc. of the 8th Int. Conference on Formal Ontology in Information Systems, Rio de Janeiro, Brazil, volume 267, pp. 317–329 (2014)
21. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Fundamental Research Series. Centre for Telematics and Information Technology. Netherlands (2005)
22. Guizzardi, G., Wagner, G., Falbo, R.A., Guizzardi, R.S.S, Almeida, J.P.A.: Towards Ontological Foundations for the Conceptual Modeling of Events. In: 32th Int. Conference on Conceptual Modeling (ER 2013), pp.327-341. Hong-Kong, China (2013)
23. Guizzardi, G., Falbo, R.A., Guizzardi, R.S.S.: Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The Case of the ODE Software Process Ontology. In: Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments, pp. 244-251. Recife, Brazil (2008)
24. Falbo, R.A., Ruy, F.B., Guizzardi, G., Barcellos, M.P., Almeida, J.P.A.: Towards an Enterprise Ontology Pattern Language. In: Proc. of the 29th Annual ACM Symposium on Applied Computing - SAC '14, pp. 323–330 (2014)
25. Barcellos, M.P., Falbo, R.A., Moro, R.D.: A Well-Founded Software Measurement Ontology. In: 6th International Conference on Formal Ontology in Information Systems, 2010, Toronto. Amsterdam: IOS Press, 2010. v.209. p.213 – 226 (2010)
26. Duarte, B.B., Souza, V.E.S., Leal, A.L.C., Falbo, R.A., Guizzardi, G., Guizzardi, R.S.S.: Towards an Ontology of Requirements at Runtime. Proc. of the 9th International Conference on Formal Ontology in Information Systems, Annecy, France (2016)
27. Ruy, F.B., Reginato, C.C., Santos, V.A., Falbo, R.A., Guizzardi, G.: Ontology Engineering by Combining Ontology Patterns. In: Proc. of the 34th International Conference on Conceptual Modeling (ER 2015), Stockholm, Sweden, pp. 173-186 (2015)
28. Natali, A. C. C., Falbo, R.A., Knowledge management in software engineering environments. In Proc. XVI Brazilian Symposium on Software Engineering, pp. 238-253 (2002)
29. Maalej, W., Panagiotou, D., Happel, H.-J.: Towards Effective Management of Software Knowledge Exploiting the Semantic Wiki Paradigm, In: Herrmann, K. Brugge, B. (eds), Software Engineering, GI, LNI, vol. 121, pp. 183 - 197 (2008)
30. Souza, E.F., Falbo, R.A., Vijaykumar, N.L.: Using lessons learned from mapping study to conduct a research project on knowledge management in software testing. In: 41st Euromicro Conference on Software Engineering and Advanced Applications (2015)
31. Falbo, R.A., Braga, C.E.C., Machado, B.N.: Semantic Documentation in Requirements Engineering. In: 17th Workshop on Requirements Eng. (WER). Pucón, Chile (2014)
32. Eriksson, H.: The Semantic-Document Approach to Combining Documents and Ontologies. Int. Journal of Human-Computer Studies, v. 65, n. 7, pp. 624-639 (2007)
33. 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)
34. Gangemi, A., Pisanelli, D.M. and Steve, G.: An Overview of the ONIONS Project: Applying Ontologies to the Integration of Medical Terminologies, Data Knowledge Engineering, 31, 2, pp. 183–220 (1999)
35. Obrst, L.: Ontological Architectures. In: Poli, R., Healy, M., Kameas, A. (editors), Theory and Applications of Ontology: Computer Applications, Springer (2010)