Engineering Ontologies & Ontologies for Engineering

Celebrating Ricardo Falbo's Career

João Paulo A. Almeida
Giancarlo Guizzardi (Eds.)

nemo

Engineering Ontologies and Ontologies for Engineering

Engineering Ontologies and Ontologies for Engineering

Celebrating Ricardo Falbo's Career

Editors
João Paulo A. Almeida
Giancarlo Guizzardi

Contributors
Ana Regina Rocha
Archimedes A. Detoni
Crediné Silva de Menezes
Danielli dos Reis Costa
Érica Ferreira de Souza
Fabiano B. Ruy
Fernanda Araujo Baião
Giancarlo Guizzardi
Gleison Santos
Guilherme Horta Travassos
João Paulo A. Almeida
Julio C. Nardi
Karina Villela
Káthia Marçal de Oliveira
Maria das Graças da Silva Teixeira
Maria Luiza M. Campos
Monalessa Perini Barcellos
Nicola Guarino
Renata Guizzardi
Silvia das Dores Rissino
Thayza Sacconi Guarnier
Victorio A. Carvalho
Vítor E. Silva Souza

Editors

João Paulo A. Almeida
Federal University of Espírito Santo
Vitória, Brazil

Giancarlo Guizzardi
Free University of Bozen-Bolzano
Bolzano, Italy
Federal University of Espírito Santo
Vitória, Brazil

Cover design by João Paulo A. Almeida

Cover photo by William Souza

ISBN 978-1393963035 (EPUB version)

Contents

Preface

This collection of essays celebrates the career of Prof. Ricardo de Almeida Falbo on the occasion of his formal retirement. The volume includes reflections from collaborators and former students, casting light on his academic work and contributions. A few personal notes are also offered, and some of the technical essays also include personal notes commemorating his captivating qualities. Authors were free to contribute in English and Portuguese (Falbo's native language).

A first chapter (by the organizers of this volume) argues that Falbo's career consists of two streams that run in complementary directions: from Ontologies to Software Engineering, and from Software Engineering to Ontology Engineering. The chapter surveys his seminal contributions in these two streams and discuss how they interplay.

As an ontologist, Falbo developed reference ontologies for a number of complex software engineering domains and leveraged on these ontologies to improve the methods and tools of software engineering. This first direction of his research is further explored in Chapter 2 by former colleagues of his Ph.D. trajectory in the Federal University of Rio de Janeiro (Ana Regina Rocha, Guilherme H. Travassos, Káthia Marçal de Oliveira, Karina Villela, Gleison Santos and Crediné Silva de Menezes). Some historical background to this stream is provided in Chapter 3 (by Crediné Silva de Menezes) revealing their early studies in the area of ontologies. This stream led to many fruitful works and was followed up by his former Ph.D. student turned colleague Monalessa Perini Barcellos. She offers in Chapter 4 her personal tour through their joint and more recent developments in the interplay between ontologies and software engineering.

As an insightful and practically minded engineer, Falbo tackled the challenge of supporting others in developing ontologies. His contributions in this area include pattern-based solutions for ontology design; a systematic method for ontology engineering (SABiO); and a number of computational tools. This second stream is also present in this volume. Chapter 5 is greatly inspired by Falbo's work on ontology patterns. In that chapter, Renata Guizzardi, Maria Luiza M. Campos and Fernanda A. Baião focus on collaboration, a constant element in Falbo's approach. They propose a Collaboration Core Ontology and derive from it a catalogue of patterns and ultimately a pattern language. A further chapter in this volume (by Danielli Costa, Maria das Graças da Silva Teixeira, Silvia Rissino and Thayza Guarnier) shows an application of SABiO in the healthcare domain, demonstrating the usefulness of the approach and its guiding nature. Nicola Guarino adds to this part a personal note, where he highlights Falbo's key role to enabling the application of formal ontology in information systems.

A final part of this volume focuses on his influence on former students. A former M.Sc. student turned colleague (Vítor Silva Souza) reviews the development of a line of work initiated in his M.Sc. thesis under Falbo's supervision, and which is still bearing fruits. Érica Ferreira de Souza adds a personal letter on his impact to her Ph.D., emphasizing his generosity. Finally, four former Ph.D. students discuss the influence of Falbo's supervision to their teaching careers in three different campi of the Federal Institute of Espírito Santo.

Engineering Ontologies and Ontologies for Engineering

Giancarlo Guizzardi[1,2] and João Paulo A. Almeida[1]

[1] Ontology and Conceptual Modeling Research Group (NEMO),
Federal University of Espírito Santo, Vitória, Brazil
{gguizzardi,jpalmeida}@inf.ufes.br
[2] Conceptual and Cognitive Modelling Research Group (CORE),
Free University of Bozen-Bolzano, Bolzano, Italy
giancarlo.guizzardi@unibz.it

Abstract. This paper is written in honour of Ricardo de Almeida Falbo, on the occasion of his formal retirement. Some aspects of his career are discussed from two complementary directions. The first direction concerns his contributions from the area of ontologies to software engineering – helping to shape what is now known as Ontology-Driven Software Engineering. The second direction regards his contributions employing insights from Software Engineering to derive solutions to the area of Ontology Engineering.

Keywords: Pattern-Based Ontology Engineering · Ontology-Based Software Engineering

1 Introduction

Ricardo Falbo studied Mechanical Engineering, with graduate studies in Information Systems (System Analysis) and Industrial Engineering, as well as a master's degree in Environmental Engineering. Despite his formal training in engineering, his professional interests soon led him to the world of software. After working as systems developer, he initiated a career as a lecturer in Computer Science. He then went to pursue a PhD in Computer Science, in which he wanted to address the problem of knowledge integration in software engineering tools.

The engineering background has had a clear impact during his entire research and teaching career. Despite of his interest in theoretical work, he has always been obsessed with the development of practical engineering tools that practitioners could use for problem-solving. Simplicity and usability have always driven his involvement in development efforts.

During his PhD studies, he studied logics and knowledge representation, and was introduced by one of his supervisors (Crediné Menezes) to the work of the philosopher Mario Bunge [30]. At that time, he also came in contact with an emerging community of ontologies in computer science in the mid-1990s [45, 84]. From that point on, the backbone of his research career would develop in the interplay between ontologies and software engineering.

As an early ontologist, Falbo made seminal contributions to the area of what came later to be known as Ontology-Driven Software Engineering. These contributions, dis-

cussed in Section 2, include the creation of a network of ontologies capturing knowledge in a multitude on software engineering sub-domains. Moreover, by leveraging on these software engineering ontologies, he made contributions to developing methodological approaches for performing a number of software engineering activities, including a pioneering work in the area of Domain Engineering. Finally, still with the support of these ontologies, he led the development of a number of computational tools for supporting software engineering activities designed to support both automated reasoning as well as knowledge integration.

Falbo made several important contributions adapting mature techniques from Software Engineering to the then incipient area of Ontology Engineering. These contributions discussed in Section 3 include the SABiO method for ontology engineering, as well as computational tools developed for supporting the development of ontologies following that method. Furthermore, by adapting the notion of generic process models, he proposed a practical approach that revived the (by then, forgotten) notion of Task Ontologies. Finally, once more drawing on the software engineering tradition, he contributed to conceptual clarification work in the area of Ontology Design Patterns and proposed a seminal approach for developing Pattern Languages in this area.

After discussing these two streams of contributions, we present some final considerations in Section 4. Finally, Section 5 concludes the paper with some personal notes from the authors.

2 Ontology-Driven Software Engineering

2.1 Software Engineering as a Domain

Falbo was already a software engineer when he discovered the area of ontologies. For this reason, it was natural for him to treat Software Engineering as his first complex and vast domain of interest, and one whose knowledge should be explicitly captured as domain ontologies. Throughout his research career, he and his collaborators produced a multitude of domain and task ontologies addressing many sub-domains in Software Engineering. These include *Software Process* (appearing as early as [15], but later refined in [10], [53], and [69]); *Software Development Methodologies* [75]); *Software Requirements* [28,39,64,66], including *Run-Time Requirements* [38]; *Software Organizations* [23]; *Software Measurement* (including domain [25] and task ontologies [24]); *Software Configuration Management* (including domain [22] and task ontologies [32]); *Software Quality* [40], *Software Code and Design* [2,3], *Software Testing* [60,82], *Software Errors* [37], and *Software Project Management* [26,27].

This line of work later culminated on the *SEON (Software Engineering Ontology Network)* project (http://dev.nemo.inf.ufes.br/seon/). As indicated by the name, the idea of SEON is the creation of an integrated network of formal domain ontologies for supporting Knowledge Management in Software Engineering [74]. As depicted in Figure 1, the network is organized in layered structure in which a number of domain ontologies are created by extending Core Ontologies, which in turn are grounded in the UFO foundational ontology [49,55].

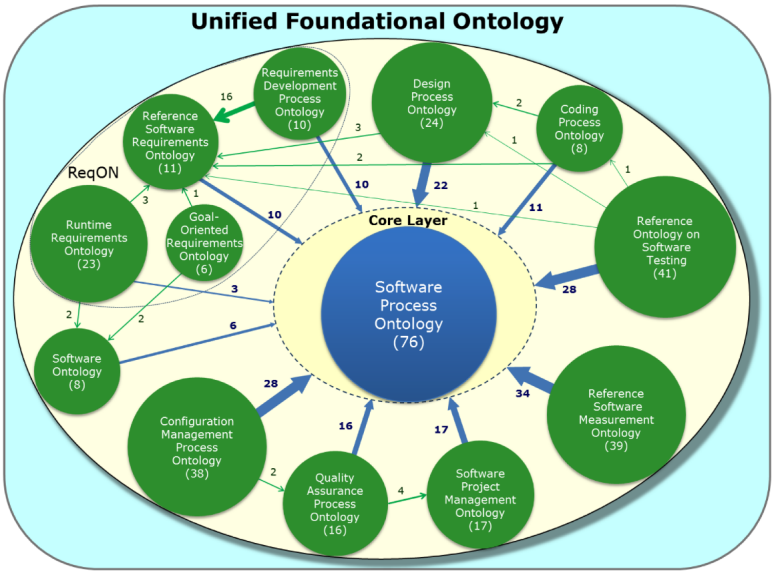

Fig. 1: The SEON Ontology Network [74].

2.2 Ontology-Based Methodological Support for Software Engineering Activities

In addition to his contribution developing knowledge artifacts for software engineering (see Section 2.1) and frequently building on these artifacts, Falbo made several methodological contributions to the area. These include ontology-based approaches for Requirements Engineering [14], for software documentation [26, 27], as well as for the semantic integration of Software Tools, including COTS (Components Off-the-Shelf) tools [31]. We focus here, however, on the contributions of Falbo to the topic of Domain Engineering.

In general, a domain engineering process is composed of the following subactivities: *domain analysis* and *domain design*, the latter being further decomposed in *infrastructure specification* and infrastructure *implementation*. Intuitively, domain engineering can be considered analogous to software application engineering, however, operating at a meta-level, i.e., instead of eliciting requirements, designing and implementing a specific application, the target is on a family of applications in a given domain [47, 67].

The product of a domain analysis phase is a *domain model*. A domain model defines objects, events and relations that capture similarities and regularities in a given domain of discourse. Moreover, it serves the purposes of a unified reference model to be used when ambiguities arise in discussions about the domain (communication), and a source of knowledge that can be used in a learning process about that domain. In summary, the specification produced by the domain modelling activity is *"a shared representation of entities that domain experts deem relevant in a universe of discourse, which can be used to promote problem-solving, communication, learning and reuse in a higher level of abstraction"* [21]).

As discussed in [49], the challenge in domain modeling is finding the best concepts that can be used to create representations of phenomena in a universe of discourse that are both as reusable as possible and still truthful to reality. The field, however, at the time was severely debilitated by a lack of concrete and consistent formal bases for making modeling decisions. Given the clear consonance between domain models in this context and a domain ontology, Falbo, Guizzardi and colleagues [11, 52] propose a domain engineering approach that could profit from the existing formal and theoretical maturity of the area of formal ontologies [48]. In this approach, they advocate the role of formal ontologies for software reuse and demonstrate how ontologies can support several tasks of a reuse-based software process. In order to support that process, they propose a new formalization for the LINGO language (see Section 3.1), and a constructive approach for deriving object-oriented frameworks from domain models (domain ontologies) represented in that language, but one which preserves the intended semantics of the original models. The framework derivation methodology proposed is supported by a spectrum of techniques, namely, mapping directives, design patterns, formal translation rules, and a framework using computational reflection mechanisms implementing the semantics of LINGO. In particular, in [52] they introduce a design pattern to preserve some ontological properties of part-whole relations (non-reflexivity, asymmetry, transitivity and shareability) in object-oriented implementations.

This approach was pioneer in introducing ontologies to domain engineering. It has been then employed for the creation of a number of object-oriented frameworks in many of the software engineering domains discussed in Section 2.1 (e.g., [11, 13, 52]). In fact, it has been employed for the construction and integration of basically all tools constituting the ODE Semantic Software Engineering Environment (see Section 2.3). However, besides being a method for Ontology-Based Domain Engineering, this was also an early method for Ontology Engineering, allowing the codification of ontologies in terms of object-oriented code. This Ontology Engineering aspect of the approach (including the aforementioned object-oriented code generation capabilities) was automated by the construction of the Ontology Development tool discussed in Section 3.2.

2.3 Ontology-Based Tools for Software Engineering

Another clear contribution of Ricardo Falbo to the area of Ontology-Based Software Engineering was formulated as as goal of *Semantic-Aware Model-Based Tools for Software Engineering* [13]. Falbo's idea was to approach software engineering as a domain (see Section 2.1), which can then be explicitly represented as a set of interconnected ontologies. By leveraging on these ontologies, one can, on one hand, (i) develop Software Engineering tools that can benefit from automated reasoning over these models; on the other hand, (ii) promote seamless integration of different software tools that are employed in software engineering processes.

In fact, since his PhD thesis [41], Falbo emphasized the idea that there can be no unique Software Engineering process suitable for developing all kinds of software in all kinds of contexts. To address that, he contributed to a meta-tool (called TABA, see Figure 2) that instantiates different software engineering tools each of which is suitable for different software engineering settings. This is done by leveraging on a set of ontologies for the software engineering domain [15].

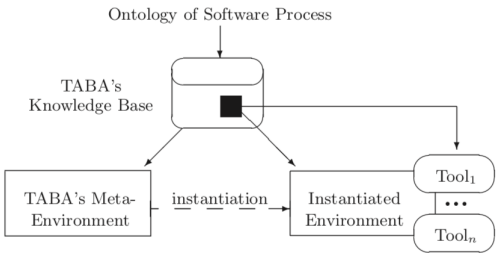

Fig. 2: The TABA meta-tool (from [15]).

After his PhD defense, Falbo founded in the Federal University of Espírito Santo a research group named LabES (Laboratory for Software Engineering). In that context, he took this project of a Semantic Software Engineering Environment (SSEE) to another level. In a long term project entitled ODE (Ontology-Based Software Development Environment) [20, 42, 77] (see Figure 3), Falbo and his group aimed at addressing the aforementioned challenges (i) and (ii), and produced an integrated suite of tools for software engineering to support ontology-based automated reasoning (in Prolog) and domain knowledge integration. ODE was designed as an ontology-based Knowledge Management environment for Software Engineering [7], supporting domain ontology creation and management, as well as ontology-based software engineering activities such as software process definition as well as quality control and documentation [40], software estimation [68], resource allocation, risk analysis [18, 33], human resource management [77], agent-based proactive knowledge dissemination [16], etc.

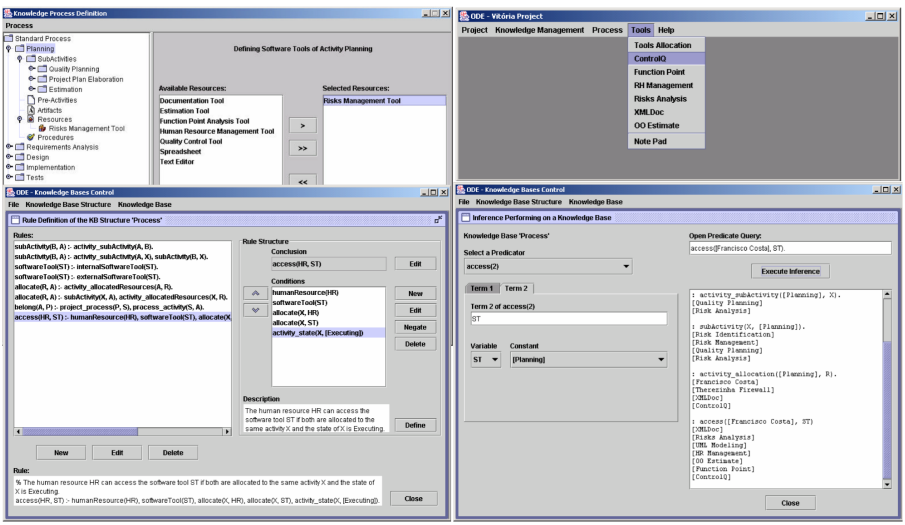

Fig. 3: The ODE Suite of Tools (from [77]).

3 An Engineering Approach to Ontology Development

3.1 A Systematic Method for Ontology Engineering

In the second half of the 1990s, we started to see the emergence of the first ontology engineering methodologies. These proposals were extrapolations from the experience of developing ontologies in specific domains. Examples included the TOVE methodology [44] and the METHONTOLOGY [43]. In that context, and drawing from their experience in building ontologies for the TABA Software Engineering Meta-Environment (see Section 2.3), Falbo and colleagues proposed an initial approach for systematizing the landscape of the state of the art in ontology engineering at the time [15]. This approach is illustrated in Figure 4.

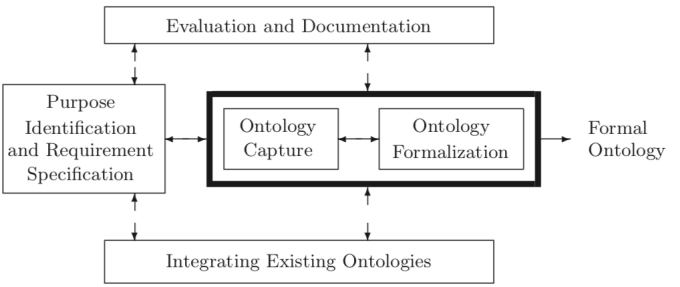

Fig. 4: An initial proposal of a method for Ontology Engineering (from [15]).

A distinctive feature of this approach, which differentiated it from other existing methods in the literature back then was the proposal of a visual language for ontology representation. Taking inspiration from the conceptual modeling literature, Falbo and colleagues propose LINGO [41][3], an *epistemological level* [50] diagrammatic language containing modeling primitives such as subtyping and parthood. Moreover, this work also proposes first-order logics rendering of these primitives such that the language can be used as a device for theory inclusion, i.e., whenever a language fragment is used, an equivalent first-order rendering of that fragment would be generated. The semantics of this language was advanced in [52] and [48]. The proposed set-theoretical semantics was formulated as to facilitate the translation from LINGO models to the Object-Oriented paradigm (see Section 2.2). Furthermore, by incorporating additional ontological distinctions in the set of modeling primitives of this language (e.g., those distinguishing types, roles and attributions), this approach later inspired the creation of a full-blown conceptual modeling language grounded in a foundational ontology, namely, OntoUML [49].

In fact, following his continuous interest in conceptual modeling and knowledge representation, Falbo contributed to the evolution of OntoUML [5, 78] as well its underlying foundational ontology UFO [56]. In particular, to the proposal of a version of

[3] Despite the meaning of “slang or jargon shared by a specific community", LINGO was also used as an abbreviation for *LINGuagem de Ontologias*, i.e., Language for Ontologies, in Portuguese. In [15], the authors use the English translation for LINGO (GLEO - Graphical Language for Expressing Ontologies).

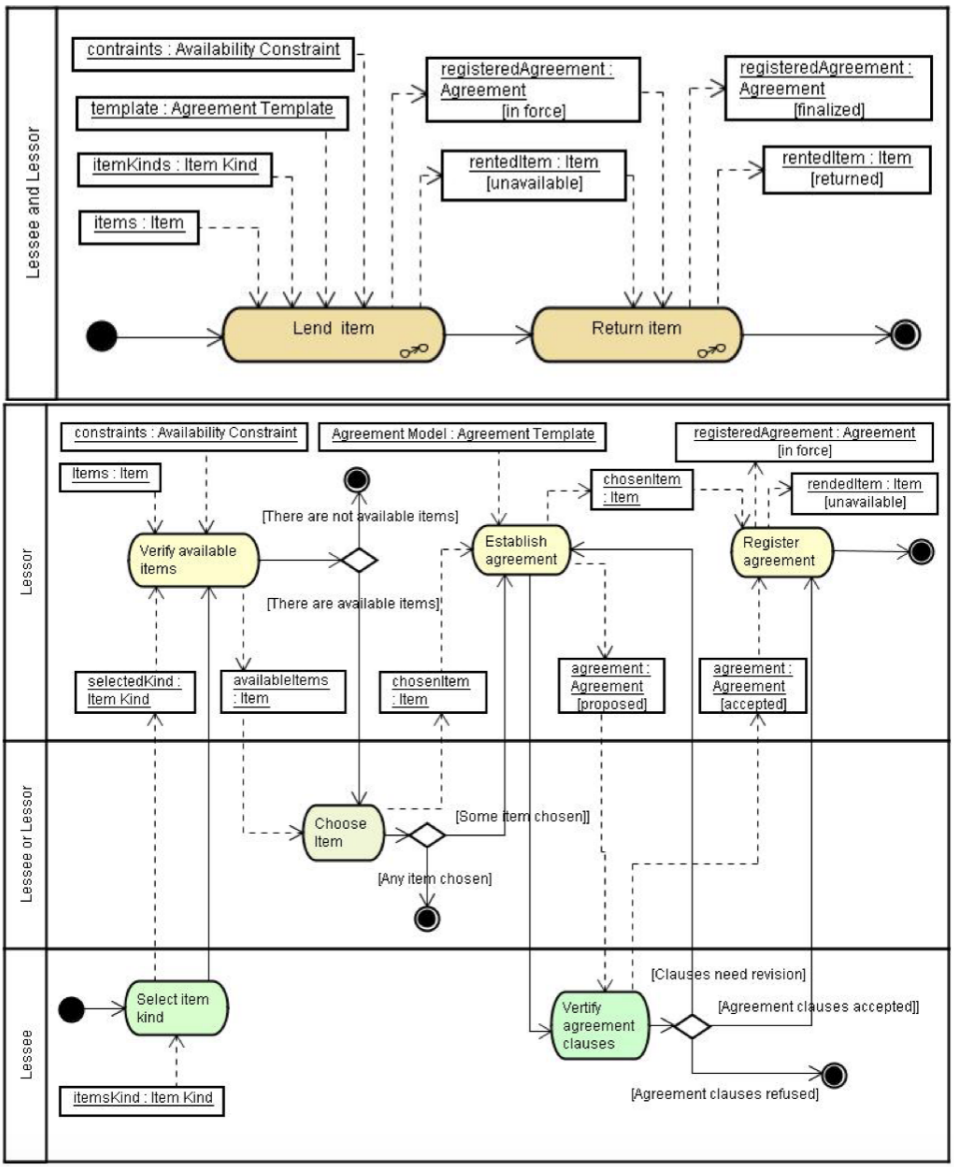

Fig. 5: Examples of a Task Ontology (from [62]).

OntoUML (dubbed E-OntoUML [61]) that was designed for the representation of the so-called Task Ontologies [46]. Task Ontologies were on a par with Domain Ontologies in early engineering methodologies. Besides, analogous to the way domain ontologies were intended to represent domain regularities independent of application, Task Ontologies were meant to represent a "generic task or activity (like diagnosing or selling), by specializing the terms introduced in the top-level ontology". Like in the case for domain ontologies, most ontology engineering approaches used logical languages for representing Task Ontologies. In fact, in earlier ontology engineering methods, Task Ontologies were associated with PSMs (Problem-Solving Methods) in Knowledge-Based Systems Engineering. However, once ontologies moved beyond that community, Task Ontologies were more and more neglected in the literature. By building on the literature of generic process models and even *workflow patterns* [36], Falbo contributes to proposing a version of UML Activity Diagrams, combined with OntoUML structural models for capturing generic and reusable task models [62]. Figure 5 depicts an example of a Task Ontology following Falbo's approach. At the top of the figure, we have the representation of a generic "Lending Task", with the refinement of the "Lend Item" sub-task in the lower part. Types such as *(chosen, rented) Item* or *Agreement* represent roles in these tasks that can be played by entities of multiple *kinds*, i.e., the so-called *role mixins* [49], and should be represented in a complementary OntoUML model.

Building on the experience acquired developing in a multitude of domains (e.g., see Section 2.1), Falbo then evolved his initially proposed method culminating in the Systematic Approach for Building Ontologies: SABiO[4] [6]. (See Figure 6 for an overview.)

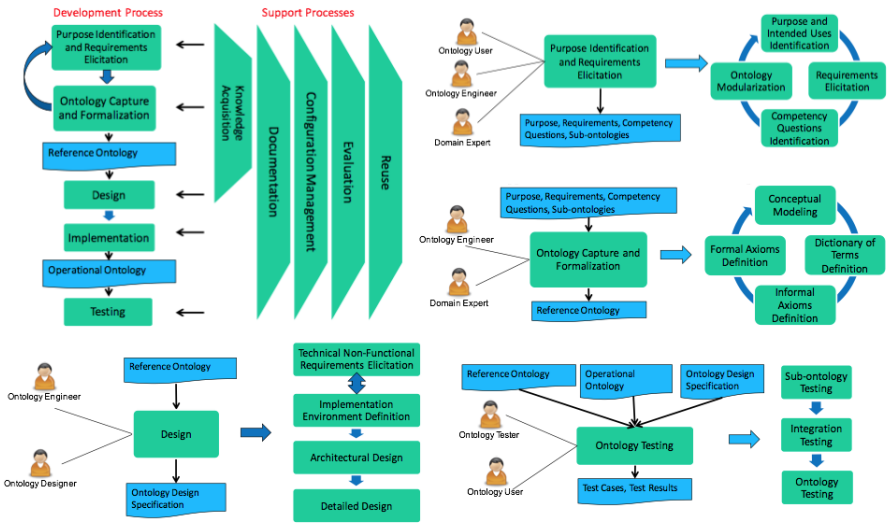

Fig. 6: An overall view of the SABiO methodology [6]).

[4] The term 'sábio' in portuguese means 'wise'.

Drawing on the methodological approach defended in [50] and later in [12], SABiO incorporates an explicit distinction between *Reference Ontologies* and *Operational Ontologies*: while the former are models created with the purpose of maximizing *domain appropriateness* (expressivity and truthfulness to the domain being represented) and *comprehensibility appropriateness* [49], the latter are codified versions of these designed with the purpose of addressing specific sets of non-functional requirements, architectural choices, and modeling languages. Another aspect of Falbo's work that influenced SABiO is his approach on Ontology Patterns as discussed in Section 3.3. Besides the development of several ontologies in software engineering (see Section 2.1), over the years, this methodology has been adopted in a number of initiatives for building ontologies in domains as diverse as aspect-orientation [58, 70], clinical reasoning [73], and knowledge explanability [29], among many others.

3.2 Software Engineering-inspired Tools for Ontology Engineering

Due to his engineering background, since his first attempts to propose an ontology engineering method (see Section 3.1), Falbo was aware of the importance of supporting such methods with computer-based tools. In [63, 83], he proposes a tool termed ODEd (Ontology Development Editor) (see Figure 7). ODEd was designed (i) to support the definition of concepts and relations, using graphical representations (based both on LINGO and on an initial UML profile for ontology representation); and (ii) to promote automatic generation of some classes of axioms from models created in these languages. Moreover, ODEd also supports the derivation of object-oriented frameworks from ontologies, following the method discussed in Section 2.2. In addition, it supports the creation of a hypertextual documentation for the ontology. Finally, it includes a software agent (OntoBoy) that assists the user in the ontology creation process.

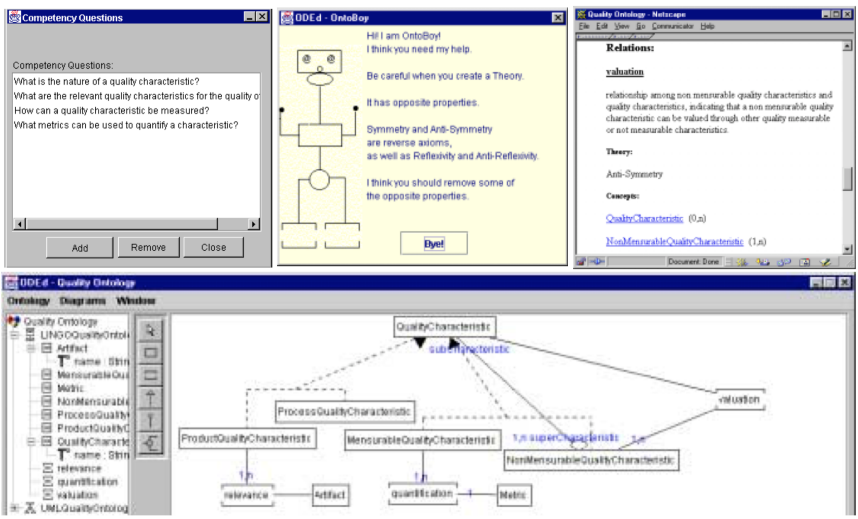

Fig. 7: The ODEd Ontology Development Tool (from [63]).

3.3 Pattern-Based Ontology Engineering

The connection between ontologies and design patterns appears in the work of Falbo and colleagues since the early 2000's [48, 51, 52]. In [12], building on the separation of reference and operational ontologies (see Section 3.1), and drawing on his software engineering background, Falbo collaborated to the terminological clarification regarding the notions of patterns in ontology engineering. Moreover, in this paper, Falbo and colleagues make the case that the confusion related to the term ontology patterns was connected to the lack of a clear separation of conceptual and operational concerns in areas such as Knowledge Engineering and the Semantic Web. In that work, they contribute to organizing the space of ontology patterns, showing that the ontology engineering literature at times abused the semantics of these terms borrowed from the Software Engineering literature. For example, the term Ontology Design Pattern is used not as a *standard solution to a recurrent design problem* but as a very general category, including patterns as different as what in software engineering is termed *Analysis Pattern* as well as idiom (or ontology coding pattern), the latter concerning solutions to circumvent language specific problems. In the proposed space, this work distinguishes the categories of: (a) *Conceptual Patterns* comprising *Foundational Ontology Patterns* (conceptual solutions extracted from Foundational ontologies), and DROPs or *Domain-Related Ontology Patterns* (conceptual solutions extracted from Reference Domain ontologies); (b) *Architectural Patterns* addressing architectural problems (e.g., ontology complexity management); (c) *Design Patterns* addressing design problems, such as guaranteeing some reasoning patterns, addressing ontology non-functional requirements (e.g., tractability), or circumventing limitations of a class of languages (e.g., how to model n-ary relations in languages that are limited to binary relations); (d) *idioms*. In that paper, they also investigate the different mechanisms through which patterns are reused, namely, *reuse by extension*, typically associated with DROPs, or *reuse by analogy*, which is the main mechanism associated with foundational patterns (but not only).

In [8], Falbo and colleagues show that, while in the ontology engineering community patterns were mostly (re)used as stand alone entities, a whole new range of benefits could be achieved if there were complete guidelines guiding the process of interrelating patterns forming solutions to larger problems. Borrowing the term from software engineering, they them propose the notion of Ontology Pattern Language (OPLs) (later refined in [9]) as a *network of interrelated ontology patterns along with a procedure for systematically using them in tandem*. A pattern language defines not only a temporal ordering for the application of patterns but also relations between these patterns, such as those showing that they require the presence of each other, or that they are variants as solutions to the same problem. In that paper, they also demonstrate the role of Core Ontologies for extracting DROPs that can them be easily combined to form pattern languages: *"as patterns move closer to a Domain ontology, they agglutinate to form a stable model, i.e., the constraints on how they can be inter-related become so strong that the very domain model is practically the only way they can appear together, thus, lacking the potential for recurrence which is part of the very definition of what a pattern is. That is why we advocate that DROPs occurring at the level of Core Ontologies are the best candidates for being organized as ontology pattern languages."*.

As initially demonstrated in [54], in a modeling language such as OntoUML that explicitly commits to a foundational ontology, its modeling primitives are not low granularity ones such as class, attribute or relationships, but conceptual patterns reflecting micro-theories of the underlying foundational ontology. Hence, such a language is a pattern language in the strong sense of language[5]. In fact, as shown in [85], the grammar of OntoUML can be expressed as a graph-rewriting pattern grammar.

In [78, 79], Falbo and colleagues demonstrate how an iterated combination of "pattern representation systems" at different levels can contribute to maximize the benefits of reuse of ontological structures. This idea (partially illustrated in Figure 8) amounts (in a nutshell) to the following strategy: (i) a domain-independent pattern grammar such as OntoUML (constituted by foundational patterns) can be used to directly build domain ontologies; (ii) by using the approach proposed in [8, 9], one can then build a Domain-Specific OPL by extracting DROPs from these core ontologies; (iii) these OPLs can be used to effectively create domain ontologies in the respective domains.

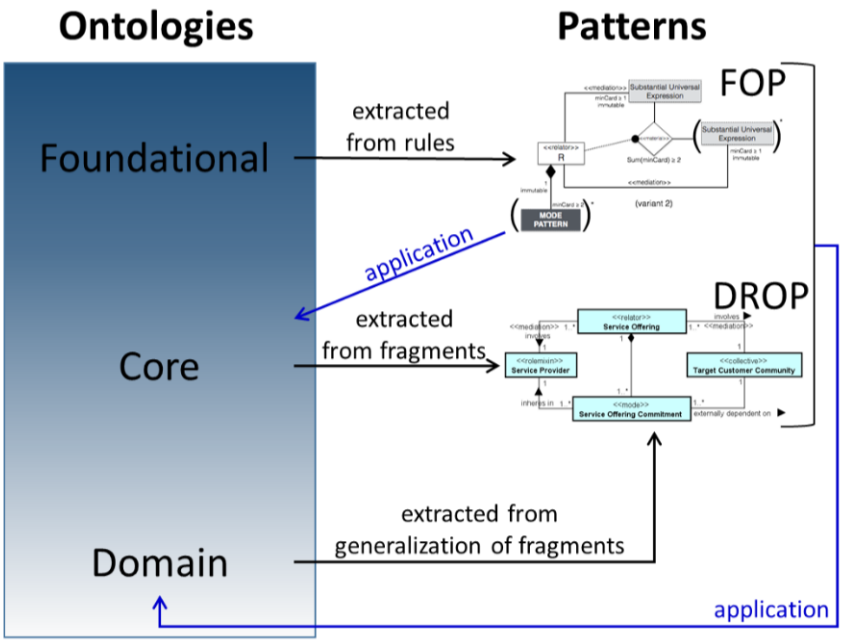

Fig. 8: Combining the benefits of Foundational Ontology Patterns and DROPs [6]).

[5] In [8], when discussing OPLs as systems of DROPs, Falbo and his co-authors write "the use of the term 'language' is, in fact, a misnomer, given that a pattern language does not typically define per se, a grammar with an explicit associated mapping to a semantic domain. However, if we focus on a more general concept of a representation system, we may consider the constituent patterns as an alphabet of higher-granularity primitives. Moreover, in this case, we can consider the procedural rules prescribing how these primitives can be lawfully combined as defining a set of valid possible instantiations for that representation system. Perhaps, a more appropriate name would be 'Pattern System'."

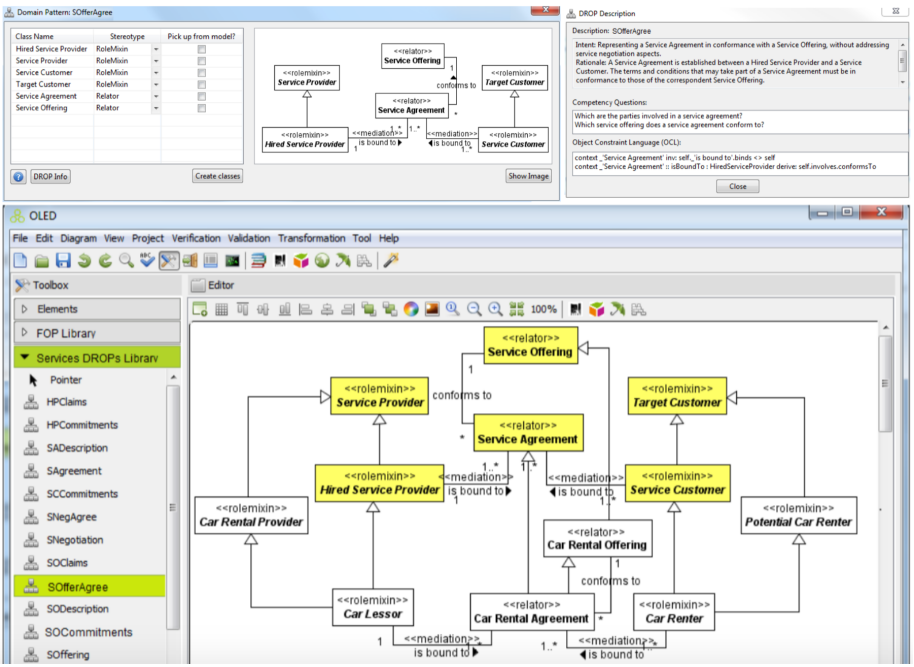

Fig. 9: Supporting for pattern definition and reuse in OLED (from [78]).

In line with their concern in developing computationally supporting tools for their approach (see Section 3.2, in [78, 79], this strategy is implemented in the OntoUML editor (OLED) (see Figure 9).

An example of the application of this strategy to create an ontology of Car Rentals is shown in Figure 10. As depicted there, we have the application, by analogy, of a the *Role* and *RoleMixin* OntoUML patterns, as well as the application, by extension, of the *Service Offering and Agreement* DROP (extracted from the UFO-S ontology, which is itself represented in OntoUML [65]).

Over the years, a number of OPLs have been developed following this approach addressing several domains, including *Service Modeling* [17, 72], *Software Process Harmonization* [76], *Enterprise Modeling* [19], *Software Testing* [81] *Measurement* [25], *Configuration Management* [4], among many others. As always, conscious of the need for providing engineering tools for operationalizing the use theoretical results, in [71, 80], Falbo and colleagues treat OPL design as a domain in itself and employ a systematic language engineering method to propose a domain-specific modeling language for representing OPLs (called OPL-ML). Figure 11, summarizes the visual notion of this language, whose application is then illustrated in Figure 12 where it is employed in the design of S-OPL (an OPL for service modeling based in UFO-S [17, 72]).

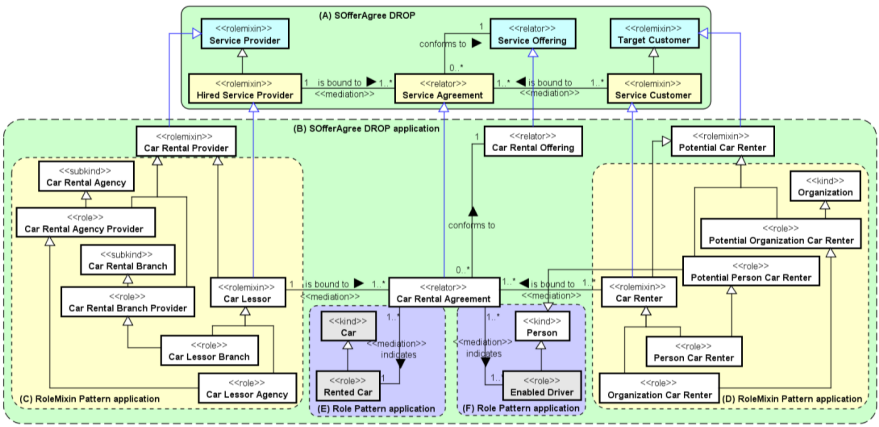

Fig. 10: Fragment of a Car Rental Agreement ontology built from patterns [78].

Structural Model	
Element	**Symbol**
Pattern	<<name>>
Pattern Group (expanded format)	<<name>>
Pattern Group (black box format)	<<name>>
Variant Pattern Group (expanded format)	<<name>>
Variant Pattern Group (black box format)	<<name>>
Relation “requires”	
Relation “requires a pattern of”	

Fig. 11: Summary of the OPL-ML Visual Notation (from [71]).

4 Final Considerations

In this paper, we review the long term research program carried out by Ricardo Falbo throughout his career. On one hand, as an engineer, Falbo has always been driven by concrete solutions and for the need of developing practical engineering tools. In particular, as a software engineer, he continuously adopted insights from software engineering to propose new and to improve existing solutions in Ontology Engineering. As shown in this paper, these include contributions to developing: a systematic method for ontology engineering; a set of pattern-based solutions for ontology design; and a number of computational tools for the discipline. On the other hand, as an ontologist, he brought the theoretically grounded methods of that discipline to continuously tackle conceptual problems in software engineering. This includes developing reference ontologies for a number of complex software engineering domains, as well as leveraging on these

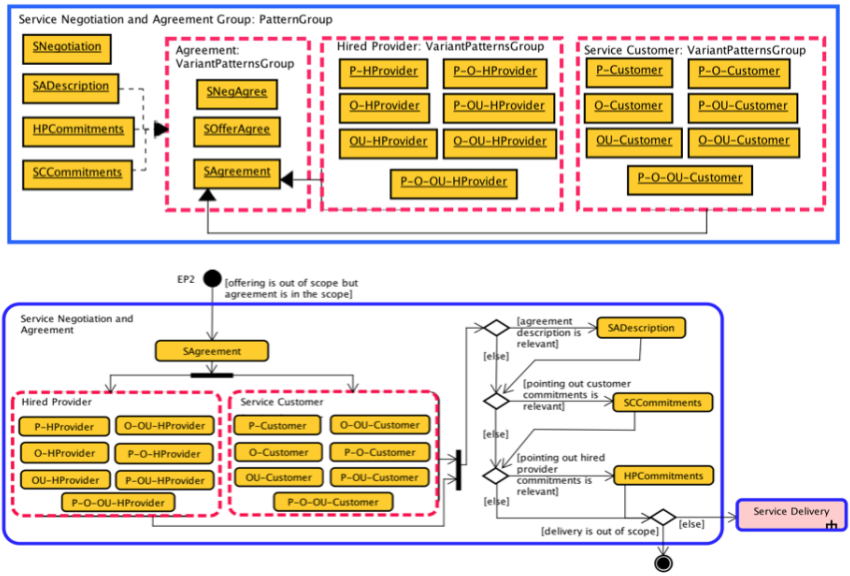

Fig. 12: Using OPL-ML to design S-OPL (from [80]).

ontologies to: improve the methods of that discipline regarding, for instance, requirements engineering, application integration, and domain engineering; develop semantic software engineering tools based on these ontologies for supporting effective knowledge management in Software Engineering.

Unswayed by buzzwords and fashion, Falbo consistently followed this research program, leading for many years a Laboratory for Software Engineering Research (LabES), and later co-founding the Ontology and Conceptual Modeling Research Group (NEMO). In these initiatives, he has always been a natural leader, combining intellectual sharpness and rigour, creativity, enthusiasm, kindness and humility, in a manner that is uncommon in science (as well as generally in life). In his research career, Falbo inspired many colleagues and students, and established a network of collaborators in different countries. Together they produced results that are constantly used by scholars internationally. However, despite these remarkable qualities as a researcher, Falbo used to say that if he had to choose between being a scientist, a practitioner, or a professor, he would chose the latter. This is because, he judged, in that way, he could perhaps have the chance of having a deeper and long lasting influence on people. That certainly worked on us.

5 Personal Notes

Giancarlo: I met Ricardo when I was still a bachelor student and he was working towards finishing his PhD thesis. In the mid 1990's, I was drawn by the problem of producing precise representations that could capture application-independent domain

regularities. After attending one of his presentations, I decided to send him my very first paper on that topic, to which he replied “This is very interesting! What you are doing here is an ontology. Have you heard about ontologies?", then giving me a copy of Nicola Guarino’s ‘The Ontological Level’ [45] (see Nicola Guarino’s personal letter in this volume). This is how I was first introduced to the research topic that would be the cornerstone of my own research program for more than two decades. Since then, we have developed a pleasant, fruitful and long-lasting research collaboration that has so far resulted in 44 joint publications, in topics ranging from Domain and Ontology Engineering, Web Engineering, Ontologies in Software Engineering and Enterprise Modeling, Domain-Specific Visual Language Design, as well as in the foundations of the UFO ontology and the OntoUML language. But, first and foremost, I consider him a mentor, one of my dearest friends and, simply, one of the best people I know.

João Paulo: Towards the end of my undergraduate studies (around 1998), Ricardo and Crediné Menezes taught together a course on object-oriented software engineering. The course was naturally informed by their investigation into ontology-based approaches, which resulted in a very interesting experience to me as a student. It was an excellent first impression of him! Years later, when I was hired at UFES, I soon realized that he was not just a great teacher, but also a wonderful friend and an extraordinary colleague. He is generous, sincere, and one of the most sensible voices around. He thinks and speaks calmly, always focusing on the essence, with no room for distractions or pretension! I have the greatest honor and pleasure to work with him. Together, we supervised a number of Ph.D. students and worked on advances in UFO (particularly UFO-B and UFO-S) and OntoUML. Ricardo has a simple and practical attitude to work that gets things done with zero stress. This attitude is a key part of what we now call warmly “Falbo’s approach”. I am very grateful he has shared his approach with us. It is not only an approach to ontology but also an inspiring approach to life!

References

1. Proceedings of the 14th international conference on Software engineering and knowledge engineering, SEKE 2002, Ischia, Italy, July 15-19, 2002. ACM (2002). https://doi.org/10.1145/568760, https://doi.org/10.1145/568760
2. de Aguiar, C.Z., de Almeida Falbo, R., Souza, V.E.S.: Ontological representation of relational databases. In: Carbonera, J.L., Guizzardi, G. (eds.) Proceedings of the XI Seminar on Ontology Research in Brazil and II Doctoral and Masters Consortium on Ontologies, São Paulo, Brazil, October 1st-3rd, 2018. CEUR Workshop Proceedings, vol. 2228, pp. 140–151. CEUR-WS.org (2018), http://ceur-ws.org/Vol-2228/paper9.pdf
3. de Aguiar, C.Z., de Almeida Falbo, R., Souza, V.E.S.: OOC-O: A reference ontology on object-oriented code. In: Laender et al. [59], pp. 13–27. https://doi.org/10.1007/978-3-030-33223-5_3, https://doi.org/10.1007/978-3-030-33223-5_3
4. Almeida, A.C., Schwabe, D., Lifschitz, S., Campos, M.L.M.: Cm-opl: An ontology pattern language for configuration management task. In: ONTOBRAS. pp. 233–238 (2018)
5. Almeida, J.P.A., de Almeida Falbo, R., Guizzardi, G.: Events as entities in ontology-driven conceptual modeling. In: Laender et al. [59], pp. 469–483. https://doi.org/10.1007/978-3-030-33223-5_39, https://doi.org/10.1007/978-3-030-33223-5_39

6. de Almeida Falbo, R.: Sabio: Systematic approach for building ontologies. In: Guizzardi, G., Pastor, O., Wand, Y., de Cesare, S., Gailly, F., Lycett, M., Partridge, C. (eds.) Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering co-located with 8th International Conference on Formal Ontology in Information Systems, ONTO.COM/ODISE@FOIS 2014, Rio de Janeiro, Brazil, September 21, 2014. CEUR Workshop Proceedings, vol. 1301. CEUR-WS.org (2014), http://ceur-ws.org/Vol-1301/ontocomodise2014_2.pdf
7. de Almeida Falbo, R., Arantes, D.O., Natali, A.C.C.: Integrating knowledge management and groupware in a software development environment. In: Karagiannis, D., Reimer, U. (eds.) Practical Aspects of Knowledge Management, 5th International Conference, PAKM 2004, Vienna, Austria, December 2-3, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3336, pp. 94–105. Springer (2004). https://doi.org/10.1007/978-3-540-30545-3_9, https://doi.org/10.1007/978-3-540-30545-3_9
8. de Almeida Falbo, R., Barcellos, M.P., Nardi, J.C., Guizzardi, G.: Organizing ontology design patterns as ontology pattern languages. In: Cimiano, P., Corcho, Ó., Presutti, V., Hollink, L., Rudolph, S. (eds.) The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7882, pp. 61–75. Springer (2013). https://doi.org/10.1007/978-3-642-38288-8_5, https://doi.org/10.1007/978-3-642-38288-8_5
9. de Almeida Falbo, R., Barcellos, M.P., Ruy, F.B., Guizzardi, G., Guizzardi, R.S.S.: Ontology pattern languages. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns - Foundations and Applications, Studies on the Semantic Web, vol. 25, pp. 133–159. IOS Press (2016). https://doi.org/10.3233/978-1-61499-676-7-133, https://doi.org/10.3233/978-1-61499-676-7-133
10. de Almeida Falbo, R., Bertollo, G.: A software process ontology as a common vocabulary about software processes. IJBPIM **4**(4), 239–250 (2009). https://doi.org/10.1504/IJBPIM.2009.032281, https://doi.org/10.1504/IJBPIM.2009.032281
11. de Almeida Falbo, R., Guizzardi, G., Duarte, K.C.: An ontological approach to domain engineering. In: Proceedings of the 14th international conference on Software engineering and knowledge engineering, SEKE 2002, Ischia, Italy, July 15-19, 2002 [1], pp. 351–358. https://doi.org/10.1145/568760.568822, https://doi.org/10.1145/568760.568822
12. de Almeida Falbo, R., Guizzardi, G., Gangemi, A., Presutti, V.: Ontology patterns: Clarifying concepts and terminology. In: Gangemi, A., Gruninger, M., Hammar, K., Lefort, L., Presutti, V., Scherp, A. (eds.) Proceedings of the 4th Workshop on Ontology and Semantic Web Patterns co-located with 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 21, 2013. CEUR Workshop Proceedings, vol. 1188. CEUR-WS.org (2013), http://ceur-ws.org/Vol-1188/paper_11.pdf
13. de Almeida Falbo, R., Guizzardi, G., Natali, A.C.C., Bertollo, G., Ruy, F.F., Mian, P.G.: Towards semantic software engineering environments. In: Proceedings of the 14th international conference on Software engineering and knowledge engineering, SEKE 2002, Ischia, Italy, July 15-19, 2002 [1], pp. 477–478. https://doi.org/10.1145/568760.568843, https://doi.org/10.1145/568760.568843
14. de Almeida Falbo, R., Martins, A.F., Segrini, B.M., Baiôco, G., Moro, R.D., Nardi, J.C.: Um processo de engenharia de requisitos baseado em reutilização de ontologias e padrões de análise. In: Kong, M., Pow-Sang, J.A., Anticona, M.T., García, L.A.F. (eds.) VI Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento-JIISIC'07, 31 de Enero al 2 de Febrero del 2007, Lima, Perú. pp. 59–68. Facultad de Ciencias e Ingeniería and Departamento de Ingeniería, Pontificia Universidad Católica del Perú (2007)

15. de Almeida Falbo, R., de Menezes, C.S., Rocha, A.R.: A systematic approach for building ontologies. In: Coelho, H. (ed.) Progress in Artificial Intelligence - IBERAMIA 98, 6th Ibero-American Conference on AI, Lisbon, Portugal, October 5-9, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1484, pp. 349–360. Springer (1998). https://doi.org/10.1007/3-540-49795-1_31, https://doi.org/10.1007/3-540-49795-1_31
16. de Almeida Falbo, R., Pezzin, J., Schwambach, M.M.: A multi-agent system for knowledge delivery in a software engineering environment. In: Chu et al. [35], pp. 253–258, http://ksiresearchorg.ipage.com/seke/Proceedings/seke/SEKE2005_Proceedings.pdf
17. de Almeida Falbo, R., Quirino, G.K., Nardi, J.C., Barcellos, M.P., Guizzardi, G., Guarino, N., Longo, A., Livieri, B.: An ontology pattern language for service modeling. In: Ossowski, S. (ed.) Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016. pp. 321–326. ACM (2016). https://doi.org/10.1145/2851613.2851840, https://doi.org/10.1145/2851613.2851840
18. de Almeida Falbo, R., Ruy, F.B., Bertollo, G., Togneri, D.F.: Learning how to manage risks using organizational knowledge. In: Holz, H., Melnik, G. (eds.) Advances in Learning Software Organizations, 6th International Workshop, LSO 2004, Banff, Canada, June 20-21, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3096, pp. 7–18. Springer (2004). https://doi.org/10.1007/978-3-540-25983-1_2, https://doi.org/10.1007/978-3-540-25983-1_2
19. de Almeida Falbo, R., Ruy, F.B., Guizzardi, G., Barcellos, M.P., Almeida, J.P.A.: Towards an enterprise ontology pattern language. In: Cho, Y., Shin, S.Y., Kim, S., Hung, C., Hong, J. (eds.) Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014. pp. 323–330. ACM (2014). https://doi.org/10.1145/2554850.2554983, https://doi.org/10.1145/2554850.2554983
20. de Almeida Falbo, R., Ruy, F.B., Moro, R.D.: Using ontologies to add semantics to a software engineering environment. In: Chu et al. [35], pp. 151–156, http://ksiresearchorg.ipage.com/seke/Proceedings/seke/SEKE2005_Proceedings.pdf
21. Arango, G.: Domain analysis methods. Software Reusability pp. 17–49 (1994)
22. Arantes, L.D.O., Falbo, R.D.A., Guizzardi, G.: Evolving a software configuration management ontology (2007)
23. Barcellos, M.P., de Almeida Falbo, R.: Using a foundational ontology for reengineering a software enterprise ontology. In: Heuser, C.A., Pernul, G. (eds.) Advances in Conceptual Modeling - Challenging Perspectives, ER 2009 Workshops CoMoL, EThe-CoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS, Gramado, Brazil, November 9-12, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5833, pp. 179–188. Springer (2009). https://doi.org/10.1007/978-3-642-04947-7_22, https://doi.org/10.1007/978-3-642-04947-7_22
24. Barcellos, M.P., de Almeida Falbo, R.: A software measurement task ontology. In: Shin, S.Y., Maldonado, J.C. (eds.) Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013. pp. 311–318. ACM (2013). https://doi.org/10.1145/2480362.2480428, https://doi.org/10.1145/2480362.2480428
25. Barcellos, M.P., de Almeida Falbo, R., Moro, R.D.: A well-founded software measurement ontology. In: Galton, A., Mizoguchi, R. (eds.) Formal Ontology in Information Systems, Proceedings of the Sixth International Conference, FOIS 2010, Toronto, Canada, May 11-14, 2010. Frontiers in Artificial Intelligence and Applications, vol. 209, pp. 213–226. IOS Press (2010). https://doi.org/10.3233/978-1-60750-535-8-213, https://doi.org/10.3233/978-1-60750-535-8-213

26. Bastos, E., Barcellos, M.P., de Almeida Falbo, R.: Exploring ontologies for semantic documentation in project management. In: Freitas, F., Baião, F.A. (eds.) Proceedings of the Brazilian Seminar on Ontologies (ONTOBRAS 2015), São Paulo, Brazil, September 8-11, 2015. CEUR Workshop Proceedings, vol. 1442. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1442/paper_9.pdf
27. Bastos, E., Barcellos, M.P., de Almeida Falbo, R.: Using semantic documentation to support software project management. J. Data Semantics **7**(2), 107–132 (2018). https://doi.org/10.1007/s13740-018-0089-z, https://doi.org/10.1007/s13740-018-0089-z
28. Bernabé, C.H., Souza, V.E.S., de Almeida Falbo, R., Guizzardi, R.S.S., Silva, C.: GORO 2.0: Evolving an ontology for goal-oriented requirements engineering. In: Guizzardi, G., Gailly, F., Maciel, R.S.P. (eds.) Advances in Conceptual Modeling - ER 2019 Workshops FAIR, MREBA, EmpER, MoBiD, OntoCom, and ER Doctoral Symposium Papers, Salvador, Brazil, November 4-7, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11787, pp. 169–179. Springer (2019). https://doi.org/10.1007/978-3-030-34146-6_15, https://doi.org/10.1007/978-3-030-34146-6_15
29. Bouter, C.: Constructing an Explanation Ontology for the Communication and Combination of Partial Explanations in a Federated Knowledge Environment. Master's thesis (2019)
30. Bunge, M.: Treatise on basic philosophy: Ontology I: the furniture of the world, vol. 3. Springer Science & Business Media (1977)
31. Calhau, R.F., de Almeida Falbo, R.: An ontology-based approach for semantic integration. In: Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2010, Vitória, Brazil, 25-29 October 2010. pp. 111–120. IEEE Computer Society (2010). https://doi.org/10.1109/EDOC.2010.32, https://doi.org/10.1109/EDOC.2010.32
32. Calhau, R.F., de Almeida Falbo, R.: A configuration management task ontology for semantic integration. In: Ossowski, S., Lecca, P. (eds.) Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012. pp. 348–353. ACM (2012). https://doi.org/10.1145/2245276.2245344, https://doi.org/10.1145/2245276.2245344
33. de Carvalho, V.A., Coelho, A., de Almeida Falbo, R.: Apoio automatizado à gerência de riscos cooperativa. In: Losavio, F., Travassos, G.H., Pelechano, V., Díaz, I., Matteo, A. (eds.) Memorias de la X Conferencia Iberoamericana de Software Engineering (CIbSE 2007), Isla de Marguerita, Venezuela, Mayo 7-11, 2007. pp. 297–310 (2007)
34. Castro, J., Cernuzzi, L., Gordillo, S.E. (eds.): Memorias de la IX Conferenci a Iberoamericana de Software Engineering (CIbSE 2006), La Plata, Argentina, Abril 24-28, 2006 (2006)
35. Chu, W.C., Juzgado, N.J., Wong, W.E. (eds.): Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'2005), Taipei, Taiwan, Republic of China, July 14-16, 2005 (2005), http://ksiresearchorg.ipage.com/seke/Proceedings/seke/SEKE2005_Proceedings.pdf
36. van Der Aalst, W.M.: Workflow patterns. Encyclopedia of Database Systems pp. 3557–3558 (2009)
37. Duarte, B.B., de Almeida Falbo, R., Guizzardi, G., Guizzardi, R.S.S., Souza, V.E.S.: Towards an ontology of software defects, errors and failures. In: Trujillo, J., Davis, K.C., Du, X., Li, Z., Ling, T.W., Li, G., Lee, M. (eds.) Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11157, pp. 349–362. Springer (2018). https://doi.org/10.1007/978-3-030-00847-5_25, https://doi.org/10.1007/978-3-030-00847-5_25
38. Duarte, B.B., de Castro Leal, A.L., de Almeida Falbo, R., Guizzardi, G., Guizzardi, R.S.S., Souza, V.E.S.: Ontological foundations for software requirements with a focus on require-

ments at runtime. Applied Ontology **13**(2), 73–105 (2018). https://doi.org/10.3233/AO-180197, https://doi.org/10.3233/AO-180197
39. Duarte, B.B., Souza, V.E.S., de Castro Leal, A.L., de Almeida Falbo, R., Guizzardi, G., Guizzardi, R.S.S.: Towards an ontology of requirements at runtime. In: Ferrario, R., Kuhn, W. (eds.) Formal Ontology in Information Systems - Proceedings of the 9th International Conference, FOIS 2016, Annecy, France, July 6-9, 2016. Frontiers in Artificial Intelligence and Applications, vol. 283, pp. 255–268. IOS Press (2016). https://doi.org/10.3233/978-1-61499-660-6-255, https://doi.org/10.3233/978-1-61499-660-6-255
40. Duarte, K.C., Falbo, R.A.: Uma ontologia de qualidade de software. In: Workshop de Qualidade de Software, João Pessoa. pp. 275–285 (2000)
41. Falbo, R.: Integração de Conhecimento em Ambientes de Desenvolvimento de Software. Ph.D. thesis, Federal University of Rio de Janeiro (COPPE/UFRJ) (1998)
42. Falbo, R.D.A., Natali, A.C.C., Mian, P.G., Bertollo, G., Ruy, F.B.: Ode: Ontology-based software development environment. In: IX Congreso Argentino de Ciencias de la Computación (2003)
43. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: Methontology: from ontological art towards ontological engineering (1997)
44. Grüninger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies (1995)
45. Guarino, N.: The ontological level. Philosophy and the cognitive sciences (1994)
46. Guarino, N.: Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy, vol. 46. IOS press (1998)
47. Guillermo, A., Ruben, P.D.: Domain analysis concepts and research directions (1991)
48. Guizzardi, G.: Software Development for and with Reuse: A Case Study with a Video on Demand System (also published as the technical report entitled: *A Methodological Approach for Development with and for reuse based on Formal Domain Ontologies*). Master's thesis (2000)
49. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Telematica Instituut / CTIT (2005)
50. Guizzardi, G.: On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In: Vasilecas, O., Eder, J., Caplinskas, A. (eds.) Databases and Information Systems IV - Selected Papers from the Seventh International Baltic Conference, DB&IS 2006, July 3-6, 2006, Vilnius, Lithuania. Frontiers in Artificial Intelligence and Applications, vol. 155, pp. 18–39. IOS Press (2006), http://www.booksonline.iospress.nl/Content/View.aspx?piid=5421
51. Guizzardi, G., de Almeida Falbo, R., Filho, J.G.P.: From domain ontologies to object-oriented frameworks. In: Workshop on Ontologies (ONTO-2001), Joint German-Austrian Conference on Artificial Intelligence, Vienna, Austria (2001)
52. Guizzardi, G., de Almeida Falbo, R., Filho, J.G.P.: Using objects and patterns to implement domain ontologies. J. Braz. Comp. Soc. **8**(1), 43–56 (2002)
53. Guizzardi, G., de Almeida Falbo, R., Guizzardi, R.S.S.: Grounding software domain ontologies in the unified foundational ontology (UFO): the case of the ODE software process ontology. In: Lencastre, M., e Cunha, J.F., Valecillo, A. (eds.) Memorias de la XI Conferencia Iberoamericana de Software Engineering (CIbSE 2008), Recife, Pernambuco, Brasil, February 13-17, 2008. pp. 127–140 (2008)
54. Guizzardi, G., das Graças, A.P., Guizzardi, R.S.S.: Design patterns and inductive modeling rules to support the construction of ontologically well-founded conceptual models in ontouml. In: Salinesi, C., Pastor, O. (eds.) Advanced Information Systems Engineering Workshops - CAiSE 2011 International Workshops, London, UK, June 20-24, 2011. Proceedings. Lecture Notes in Business Information Processing, vol. 83, pp. 402–413. Springer (2011). https://doi.org/10.1007/978-3-642-22056-2_44, https://doi.org/10.1007/978-3-642-22056-2_44

55. Guizzardi, G., Wagner, G., Almeida, J.P.A., Guizzardi, R.S.S.: Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. Applied Ontology **10**(3-4), 259–271 (2015). https://doi.org/10.3233/AO-150157, https://doi.org/10.3233/AO-150157
56. Guizzardi, G., Wagner, G., de Almeida Falbo, R., Guizzardi, R.S.S., Almeida, J.P.A.: Towards ontological foundations for the conceptual modeling of events. In: Ng, W., Storey, V.C., Trujillo, J. (eds.) Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8217, pp. 327–341. Springer (2013). https://doi.org/10.1007/978-3-642-41924-9_27, https://doi.org/10.1007/978-3-642-41924-9_27
57. Johannesson, P., Lee, M., Liddle, S.W., Opdahl, A.L., López, O.P. (eds.): Conceptual Modeling - 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings, Lecture Notes in Computer Science, vol. 9381. Springer (2015). https://doi.org/10.1007/978-3-319-25264-3, https://doi.org/10.1007/978-3-319-25264-3
58. Júnior, P.A.P., Penteado, R.A.D.: Obascid (-tool): an ontologically based approach for concern identification and classification and its computational support. Journal of the Brazilian Computer Society **24**(1), 3 (2018)
59. Laender, A.H.F., Pernici, B., Lim, E., Palazzo M. de Oliveira, J. (eds.): Conceptual Modeling - 38th International Conference, ER 2019, Salvador, Brazil, November 4-7, 2019, Proceedings, Lecture Notes in Computer Science, vol. 11788. Springer (2019). https://doi.org/10.1007/978-3-030-33223-5, https://doi.org/10.1007/978-3-030-33223-5
60. Maciel, C.P.C., de Souza, É.F., Vijaykumar, N.L., de Almeida Falbo, R., Meinerz, G.V., Felizardo, K.R.: An empirical study on the knowledge management practice in software testing. In: Genero, M., Kalinowski, M., Molina, J.G., Pino, F., Conte, T., Marín, B., Brito, I., Giachetti, G. (eds.) Proceedings of the XXI Iberoamerican Conference on Software Engineering, Bogota, Colombia, April 23-27, 2018. pp. 29–42. Curran Associates (2018)
61. MARTINS, A., Falbo, R.A., Guizzardi, G., Almeida, J.P.A.: Uso de uma ontologia de fundamentação para dirimir ambiguidades na modelagem de processos de negócio. VII Simpósio Brasileiro de Sistemas de Informação (SBSI 2011) (2011)
62. Martins, A.F., de Almeida Falbo, R.: Models for representing task ontologies. In: de Freitas, F.L.G., Stuckenschmidt, H., Pinto, H.S., Malucelli, A., Corcho, Ó. (eds.) Proceedings of the 3rd Workshop on Ontologies and their Applications, Salvador, Bahia, Brazil, October 26, 2008. CEUR Workshop Proceedings, vol. 427. CEUR-WS.org (2008), http://ceur-ws.org/Vol-427/paper4.pdf
63. Mian, P.G., de Almeida Falbo, R.: Supporting ontology development with oded. In: de Leon Ferreira de Carvalho, A.C.P., Maldonado, J.C., Augusto, J.C., de Mendonça Neto, M.G., Acuña, S.T. (eds.) Proceedings of 2nd Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC'02), Salvador, Brasil, October 2002. pp. 1–11 (2002)
64. Nardi, J.C., de Almeida Falbo, R.: Uma ontologia de requisitos de software. In: Castro et al. [34], pp. 111–124
65. Nardi, J.C., de Almeida Falbo, R., Almeida, J.P.A., Guizzardi, G., Pires, L.F., van Sinderen, M., Guarino, N., Fonseca, C.M.: A commitment-based reference ontology for services. Inf. Syst. **54**, 263–288 (2015). https://doi.org/10.1016/j.is.2015.01.012, https://doi.org/10.1016/j.is.2015.01.012
66. Negri, P.P., Souza, V.E.S., de Castro Leal, A.L., de Almeida Falbo, R., Guizzardi, G.: Towards an ontology of goal-oriented requirements. In: Pastor, O., Guizzardi, R.S.S., Kaplan, G.N., Molina, J.G., Moreira, A., Brito, I.S., de la Vara, J.L., Genero, M., da Silva, F.Q.B., Cernuzzi, L. (eds.) Proceedings of the XX Iberoamerican Conference on Software Engineer-

ing, Buenos Aires, Argentina, May 22-23, 2017. pp. 469–482. Curran Associates (2017), http://cibse2017.inf.ufes.br
67. Neighbors, J.M.: Software construction using components. Ph.D. thesis, University of California, Irvine (1980)
68. de Oliveira Arantes, L., de Carvalho, V.A., de Almeida Falbo, R.: Uma ferramenta integrada de apoio a estimativas de tamanho e esforço em um ambiente de desenvolvimento de software. In: Castro et al. [34], pp. 493–496
69. de Oliveira Bringuente, A.C., de Almeida Falbo, R., Guizzardi, G.: Using a foundational ontology for reengineering a software process ontology. JIDM **2**(3), 511–526 (2011), http://seer.lcc.ufmg.br/index.php/jidm/article/view/164
70. Parreira, P.A., Penteado, R.A.D.: Obascid: an ontologically-based approach for concern identification and classification. In: 2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS). pp. 141–150. IEEE (2016)
71. Quirino, G.K.S., Barcellos, M.P., de Almeida Falbo, R.: OPL-ML: A modeling language for representing ontology pattern languages. In: de Cesare, S., Frank, U. (eds.) Advances in Conceptual Modeling - ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, Valencia, Spain, November 6-9, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10651, pp. 187–201. Springer (2017). https://doi.org/10.1007/978-3-319-70625-2_18, https://doi.org/10.1007/978-3-319-70625-2_18
72. Quirino, G.K., Nardi, J.C., Barcellos, M.P., de Almeida Falbo, R., Guizzardi, G., Guarino, N., Bochicchio, M.A., Longo, A., Zappatore, M.S., Livieri, B.: Towards a service ontology pattern language. In: Johannesson et al. [57], pp. 187–195. https://doi.org/10.1007/978-3-319-25264-3_14, https://doi.org/10.1007/978-3-319-25264-3_14
73. Radović, M., Tošić, M., Milošević, D., Milošević, M., Milošević, M.: Semantic approach to script concordance test
74. Ruy, F.B., de Almeida Falbo, R., Barcellos, M.P., Costa, S.D., Guizzardi, G.: SEON: A software engineering ontology network. In: Blomqvist, E., Ciancarini, P., Poggi, F., Vitali, F. (eds.) Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10024, pp. 527–542 (2016). https://doi.org/10.1007/978-3-319-49004-5_34, https://doi.org/10.1007/978-3-319-49004-5_34
75. Ruy, F.B., de Almeida Falbo, R., Barcellos, M.P., Guizzardi, G.: An ontological analysis of the ISO/IEC 24744 metamodel. In: Garbacz, P., Kutz, O. (eds.) Formal Ontology in Information Systems - Proceedings of the Eighth International Conference, FOIS 2014, September, 22-25, 2014, Rio de Janeiro, Brazil. Frontiers in Artificial Intelligence and Applications, vol. 267, pp. 330–343. IOS Press (2014). https://doi.org/10.3233/978-1-61499-438-1-330, https://doi.org/10.3233/978-1-61499-438-1-330
76. Ruy, F.B., de Almeida Falbo, R., Barcellos, M.P., Guizzardi, G.: Towards an ontology pattern language for harmonizing software process related ISO standards. In: Wainwright, R.L., Corchado, J.M., Bechini, A., Hong, J. (eds.) Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015. pp. 388–395. ACM (2015). https://doi.org/10.1145/2695664.2695796, https://doi.org/10.1145/2695664.2695796
77. Ruy, F.B., Bertollo, G., de Almeida Falbo, R.: Knowledge-based support to process integration in ODE. CLEI Electron. J. **7**(1) (2004). https://doi.org/10.19153/cleiej.7.1.3, https://doi.org/10.19153/cleiej.7.1.3
78. Ruy, F.B., Guizzardi, G., de Almeida Falbo, R., Reginato, C.C., Santos, V.A.: From reference ontologies to ontology patterns and back. Data Knowl. Eng. **109**, 41–69 (2017). https://doi.org/10.1016/j.datak.2017.03.004, https://doi.org/10.1016/j.datak.2017.03.004

79. Ruy, F.B., Reginato, C.C., dos Santos, V.A., de Almeida Falbo, R., Guizzardi, G.: Ontology engineering by combining ontology patterns. In: Johannesson et al. [57], pp. 173–186. https://doi.org/10.1007/978-3-319-25264-3_13, https://doi.org/10.1007/978-3-319-25264-3_13
80. das Graças da Silva Teixeira, M., Quirino, G.K., Gailly, F., de Almeida Falbo, R., Guizzardi, G., Barcellos, M.P.: Pon-s: A systematic approach for applying the physics of notation (pon). In: Schmidt, R., Guédria, W., Bider, I., Guerreiro, S. (eds.) Enterprise, Business-Process and Information Systems Modeling - 17th International Conference, BPMDS 2016, 21st International Conference, EMMSAD 2016, Held at CAiSE 2016, Ljubljana, Slovenia, June 13-14, 2016, Proceedings. Lecture Notes in Business Information Processing, vol. 248, pp. 432–447. Springer (2016). https://doi.org/10.1007/978-3-319-39429-9_27, https://doi.org/10.1007/978-3-319-39429-9_27
81. de Souza, É.F., de Almeida Falbo, R., Vijaykumar, N.L.: Knowledge management applied to software testing: A systematic mapping. In: The 25th International Conference on Software Engineering and Knowledge Engineering, Boston, MA, USA, June 27-29, 2013. pp. 562–567. Knowledge Systems Institute Graduate School (2013), http://ksiresearchorg.ipage.com/seke/Proceedings/seke/SEKE2013_Proceedings.pdf
82. de Souza, É.F., de Almeida Falbo, R., Vijaykumar, N.L.: Roost: Reference ontology on software testing. Applied Ontology **12**(1), 59–90 (2017). https://doi.org/10.3233/AO-170177, https://doi.org/10.3233/AO-170177
83. Souza, V.E.S., de Almeida Falbo, R.: Supporting ontology axiomatization and evaluation in oded. In: Astudillo, H., Taramasco, C. (eds.) Memorias de la VIII Conferencia Iberoamericana de Software Engineering (CIbSE 2005), Valparaiso, Chile, Mayo 2-6, 2005. pp. 59–70 (2005)
84. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. The knowledge engineering review **11**(2), 93–136 (1996)
85. Zambon, E., Guizzardi, G.: Formal definition of a general ontology pattern language using a graph grammar. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (eds.) Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic, September 3-6, 2017. Annals of Computer Science and Information Systems, vol. 11, pp. 1–10 (2017). https://doi.org/10.15439/2017F001, https://doi.org/10.15439/2017F001

Ontologies in Software Development Environments

Ana Regina Rocha[1], Guilherme Horta Travassos[1], Káthia Marçal de Oliveira[2], Karina Villela[3], Gleison Santos[4], Crediné Silva de Menezes[5]

[1] Federal University of Rio de Janeiro (COPPE/UFRJ), Brazil
[2] Polytechnic University of Hauts-de-France (UPHF), France
[3] Fraunhofer Institute for Experimental Software Engineering (Fh IESE), Germany
[4] Federal University of the State of Rio de Janeiro (UNIRIO), Brazil
[5] Federal University of Rio Grande do Sul (UFRGS), Brazil

Abstract. Software development is a knowledge-intensive business. Different kinds of knowledge are essential for software practitioners to support their activities in software organizations. This paper presents knowledge integration in software engineering environments (SDEs) using the framework provided by the meta-environment TABA. To that end, an ambitious and pioneering work was to introduce knowledge in SDEs through ontologies. Thanks to this approach, three other works have used ontologies to define Domain Oriented Software Development Environments, Enterprise Oriented Software Development Environments and Corporation Oriented Software Development Environments. This paper summarizes the history and scientific contributions of these different works based on the use of ontologies in SDEs.

Keywords: Ontology, Software Development Environment, Knowledge Management

1 Introduction

Different kinds of knowledge are essential for software practitioners to support their activities in software organizations, such as knowledge about the domain for which software is being developed, new technologies, local practices and policies, who knows what in the organization, guidelines, best practices, and previous experiences with techniques, methods and the software process.

A Software Development Environment (SDE) is a computational system that provides support for the construction, management, and maintenance of a software product. An SDE consists of a repository that stores all the information related to the software project throughout its life cycle, and tools (Computer-Aided Software Engineering tools) that support the involved technical and managerial activities. SDEs differ from one another depending on their database nature, the scope of provided tools, or adopted technology.

This paper describes our experience of building software development environments supported by the use of ontologies using the framework provided by the meta-environment TABA [1]. TABA aims at creating SDEs in different application domains

according to the specific requirements of each application domain and the technology chose. Creating an SDE involves defining a software development process and selecting CASE tools to be provided in the SDE. Software developers may then use the resulting SDE in the development of specific software products. TABA has several components addressing the various aspects of building SDEs and CASE tools.

In the following sections, we present the first work on the use ontologies on TABA and three evolutions of this pioneering work: the Domain Oriented Software Development Environments (DOSDE), the Enterprise Oriented Software Development Environments (EOSDE) and the Corporation Oriented Software Development Environments (COSDE).

2 Knowledge Integration in Software Development Environments

Software Development Environments (SDEs) are built to support complex software development processes, providing a framework to integrate tools along three main dimensions: control, data, and presentation. With the increase of knowledge-based support in SDEs, Falbo et al. [2] identified the necessity to consider also a knowledge integration dimension.

Knowledge-based tools have to communicate with each other in an SDE. So, they must share a common vocabulary and the same interpretation for the terms used. It is the primary goal of knowledge integration: to establish the semantics of the information exchanged between the various tools in an SDE.

Even in those SDEs that embody some knowledge-based support, the approach presents the following problems:

a) the knowledge can hardly be shared or reused;
b) each tool is built based on a specific conceptualization and uses its vocabulary, making the communication between tools complicated;
c) there is usually a large portion of knowledge that is common to several tools, leading to redundancy and inconsistency;
d) knowledge acquisition is made from scratch for each new tool to be built, increasing the costs, and lowering the productivity of the development of knowledge-based tools.

To share knowledge, however, it was necessary to change the way tools are built with the construction of a knowledge model for the environment that could be used by each of the tools.

The merits of an analysis of knowledge on a more abstract level than that of representation languages have been recognized since the publication of [3]. According to Newell, it is necessary to consider a level of discourse (the knowledge level) where the knowledge and the problem solving could be dealt with the independence of their possible implementations (the symbolic level). In the knowledge level, problem-solving agents are characterized in terms of the action they can perform, their knowledge and their goals. A formalism is only a symbolic system that encodes a knowledge body.

The first effort of Falbo et al. approach [2] was placed on the symbolic level. For a

successful knowledge integration, however, it must happen at the knowledge level too. The critical point in this issue is the knowledge modeling. Falbo et al. shift the focus on the development of knowledge-based tools to an approach focusing on the sharing of common knowledge. This approach emphasizes the *development for reuse*, more specifically, knowledge reuse.

The main impediment for the knowledge sharing originates from the lack of a primary domain conceptualization over which the various tools can be built. Therefore, the use of ontologies was chosen as the critical point to deal with knowledge integration [4, 5].

One of the main benefits of the use of ontologies in Knowledge-Based System (KBS) development is the opportunity to adopt a more productive strategy for the Knowledge Acquisition (KA). In traditional Knowledge Engineering, for each new application to be built, a new conceptualization is developed. In an ontology-based approach, the KA can be accomplished in two stages. First, the general domain knowledge relevant to several applications should be elicited and specified as ontologies. These, in turn, are used to guide the second stage of the KA when the particularities of a specific software application are considered. In this way, the same ontology can be used to guide the development of several software applications, minimizing the costs of KA and allowing knowledge sharing and reuse.

To capture a knowledge model and its corresponding domain theory, infrastructure to make knowledge components available to be reused and shared among tools was built. This infrastructure is a *Knowledge Server*.

One crucial aspect to be considered in TABA Workstation is the software development process. The meta-environment and its instantiated SDEs need to handle some knowledge about software processes. This knowledge has to be shared along the Workstation. Due to this, a Knowledge Server for the software process domain was built [6].

The Software Process Knowledge Server (SPKS) was developed in the following manner. First, an ontology of the software development process was built. Given the complexity of this domain, a leveled approach for building ontologies was adopted. The ontology of the software process was developed on the top of domain ontologies for activity, procedure and resource. Each one of these ontologies originates from a knowledge module and has a number of instantiations that also originates knowledge modules. The knowledge modules were implemented in Prolog, which inference engine is already integrated into the environment as a result of the first initiative to promote knowledge integration in TABA Workstation. Figure 1 shows part of the knowledge module for activity ontology. Figure 2 shows part of a knowledge module derived from an ontology instantiation for the Planning Activity. To handle task knowledge, task models for planning and assignment, which are the main problem types that occur in the definition of a software process, were developed.

This infrastructure was used to develop Assist-Pro, an intelligent assistant, to support the definition of software processes. Process definition using *Assist-Pro* involves the following steps: (i) project feature definition, (ii) selection of a life cycle model, (iii) process decomposition, (iv) method and technique assignment, and (v) definition of inputs and outputs for each activity.

```
subactivity(A,C) :- subactivity(A,B), subactivity(B,C).
superactivity(B,A) :- subactivity(A,B).
preactivity(A,B) :- output(S,A), input(S,B).
preactivity(A,C) :- preactivity(A,B), preactivity(B,C).
posactivity(A,B) :- preactivity(B,A).
```

Fig. 1: Activity Ontology Knowledge Module.

```
activity (Planning, ManagingActivity).
activity (SoftwareScopeDefinition, ManagingActivity).
activity (ResourceEstimating, ManagingActivity).
activity (EffortEstimating, ManagingActivity).
activity (Scheduling, ManagingActivity).
input (ScopeStatement, ResourceEstimating).
input (ScopeStatement, EffortEstimating).
input (EffortEstimation, Scheduling).
output (ProjectPlan, Planning).
output (ScopeStatement, SoftwareScopeDefinition).
output (ResourceEstimation, ResourceEstimating).
output (EffortEstimation, EffortEstimating).
output (TimelineChart, Scheduling).
subactivity (SoftwareScopeDefinition, Planning).
subactivity (ResourceEstimation, Planning).
subactivity (EffortEstimation, Planning).
subactivity (Scheduling, Planning).
```

Fig. 2: Planning Activity Knowledge Module.

Knowledge Servers address the four problems enumerated in the following manner:

a) **Difficulty to share and reuse knowledge** - Since the Knowledge Server makes knowledge modules (that operationalize ontologies and their instantiations) and reasoner templates (that operationalize task models) available, it is possible to re-use and share both domain and task knowledge.

b) **Communication problems due to different conceptualizations adopted by each tool** - Since the knowledge modules are derived from ontologies, there is a fundamental domain conceptualization and a well-established vocabulary underlying the development of tools, minimizing the communication problems.

c) **Knowledge redundancy and inconsistency** - Typically, the knowledge common to various tools is the domain knowledge, captured as ontologies and implemented in the Knowledge Server knowledge modules. Consequently, since the Knowledge

Server makes these knowledge modules available, the knowledge redundancy and inconsistency in the SEE can be minimized.

d) **High costs and low productivity in the development of knowledge-based tools for the SEE** - Because the Knowledge Server offers reusable components for both domain and task, it is possible to adopt a more productive strategy in the development of knowledge-based tools for the SDE. In this approach, knowledge acquisition can be accomplished in two stages. The Knowledge Server strongly supports the first one. The second one, although more firmly supported by the experts, is still guided by the first stage. In this way, the productivity can be increased and costs reduced.

3 Domain-Oriented Software Development Environment

It is well known that one of the most critical activities of software development is probably the correct identification and description of what the software system must accomplish (requirements). It is particularly hard when the software team does not have enough knowledge about the problem domain and has no software development expertise in that domain [7]. Considering this fact and the previous work of the integration of knowledge in SDEs previously presented, Oliveira had worked on how to integrate the knowledge of the domain, using ontologies, to support the activities of the software development in SDEs. The result of this research was the definition of a new class of SDE named Domain Oriented Software Development Environment (DOSDE).

To define what knowledge should be introduced in an SDE to make a DOSDE, we must consider the domain in general, as it could be used in several applications and not for some specific application. To organize this knowledge, we use two kind of ontologies to describe and organize this knowledge, according to Guarino's [8] classification: **domain ontologies,** that describe the vocabulary related to a generic domain (such as medicine, or automobiles); and **task**, ontologies, that describe generic tasks or activities (like diagnosis or selling). To use this knowledge, we took into account the well-known activities in a software development process.

For example, the domain ontology defined for a DOSDE in cardiology [7] was composed of five sub ontologies (partially showed in Figure 3): (i) heart anatomy (concepts about the heart structure and the physiology), (ii) findings (concepts that are used in the physician's investigation process), (iii) therapy (general kinds of therapies and their feature), (iv) diagnosis (concepts and characteristics that identify syndrome and etiology diagnoses); and (v) pathologies (representing different situations of heart components). The domain ontology for cardiology contains 70 concepts with 80 properties and a set of axioms.

Task ontologies were defined based on Mizoguchi et al. [9, 10] considering three different levels. The first is the lexical level, which lists the syntactic aspects of the problem-solving description, using generic nouns (i.e., concepts), generic verbs (i.e., activities) and generic adjectives that modify the objects. The second level is called conceptual and it contains activities, objects, statuses that correspond to the generic verbs, nouns and adjectives respectively of the lexical level. Finally, the third level,

called symbolic, represents concepts and constraints in a formal language. A task ontology is thus limited to describe the task conceptually. It does not detail how it can be solved; it does not describe the task's controls. The detailed resolution of a task is described by a Problem Solving Method (PSM). A PSM determines how a task can be decomposed into subtasks, how to control the order of execution of these subtasks and what requirements are necessary from the domain knowledge [11].

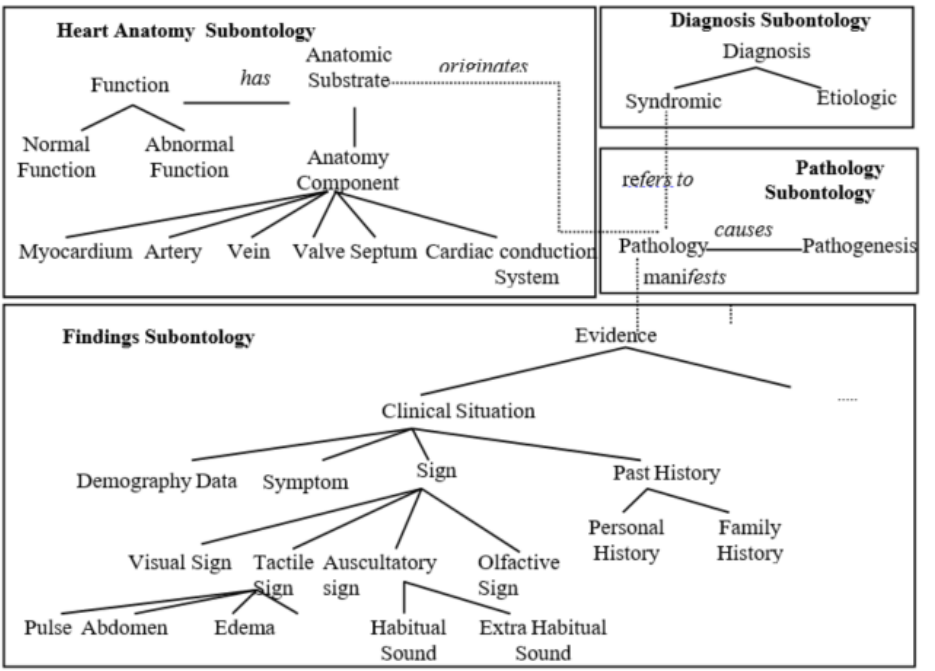

Fig. 3: Concepts and sub ontologies for cardiology.

To describe a task in a DOSDE, we combine task ontology and PSMs in a single model that we called Problem Solving Theory (PST) The PST follows the structure of the task ontology description in the three levels proposed by Mizoguchi et al. [9] as described in the following sections. An example of a description of a task ontology in DOSDE is presented in Figure 4 for the Configuration task solved by the Propose & Revise PSM (P&R). A complete description of how to describe tasks using ontologies can be found in [7].

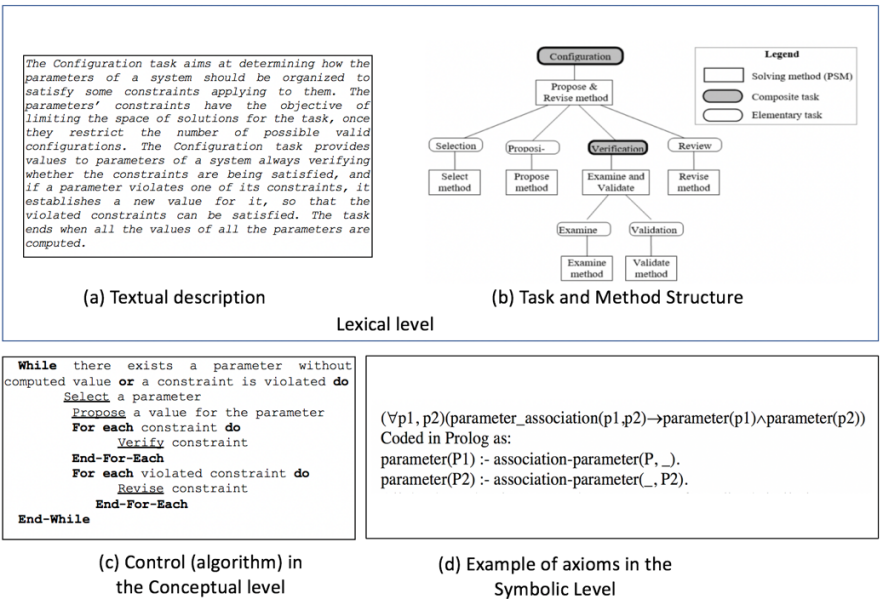

Fig. 4: Problem Solving Theory for Configuration task

The task ontology and the PSMs describe the tasks using independent domain concepts that may be filled by concepts from the application domain (concepts that are listed in the domain ontology). We decided to map the domain ontology and the PSTs at a more abstract level than the concepts. We map the sub ontologies of the domain ontology to the tasks of the PSTs. The mapping indicates the concepts more closely related to the task and to which the software developers should pay particular attention. We called the final result (i.e., the domain ontology mapped to the PSTs) a Domain Theory. It can be challenging to identify all possible tasks of a domain, but we consider it essential to identify, at least, the most typical ones. Each identified task is mapped with the sub ontologies that contain essential concepts related to that task. In our example for cardiology, we identified the following task: diagnosis, planning (for example, therapeutic planning), simulation and monitoring. We mapped Diagnosis task with the Findings, Pathology and Diagnosis sub ontologies since to make a diagnosis. One needs to know about findings, kinds of pathologies and diagnosis. Similarly, for Therapeutic Planning, the relevant sub ontologies are Therapy and Pathology; for the Monitoring task, the sub ontologies are Heart Anatomy and Findings; and for the Simulation task, only the Heart anatomy subontology. If we needed to develop a system for diagnosis in cardiology, we would know that it is essential to study first the kinds of pathology we can diagnose, their diagnosis and the associated findings. In the same way, when studying signs (a concept from Findings), we can see in which tasks or activities they are used (in this case, they are used for Diagnosis and Monitoring).

In one way or another, all the software development activities depend on domain knowledge. However, we observe a stronger influence of domain knowledge for those

activities closer to the application domain, such as software requirements analysis and design of the software system, rather than those closer to the solution domain, such as coding or even testing. Consequently, we develop a set of tools that use the domain and task ontologies defined to support several activities as follows:

- **Assisting Domain Understanding in Requirements Elicitation** - In the software requirements analysis, domain orientation is essential. The main sub-activity in the software requirements analysis is requirements elicitation (or knowledge acquisition for knowledge-based systems). During this sub-activity, the domain theory works as a starting point for the software developers. They can identify what knowledge is relevant to future applications. The domain theory by itself represents knowledge for any possible software application in this domain. By exploring the mapping between the tasks and sub ontologies or just identifying some data used for those tasks (e.g., collecting documents used for a task in the client organization), developers can find what concepts of the domain theory they need to study and understand. These concepts represent a source of information (features and descriptions) supporting the understanding of the entire domain of the future application, the elicitation of requirements.
- **Assisting Domain Understanding in Requirements Elicitation -** When the software developers understand the domain and the requirement elicitation with the users is complete, they should describe those requirements. It can be done, for example, with use cases [12]. We proposed to use the ontologies in DOSDE to support the definition of the use cases (see, for instance, [13] for an example of using a task description to help to define a use case with a DOSDE for the acoustic propagation.
- **Assisting in Design -** DOSDEs may also assist during the design activity, particularly in data modeling. Software developers can use the organization of the domain ontology as a suggestion for the first draft of the design. Considering the works from literature (see, for instance [14]) that highlight the links between ontological structures and data entity relationship and object-oriented design, we investigated how we could generate the first draft of a conceptual data model based on the domain ontology. To do this, we defined a mapping between ontological constructs and entity-relationship constructs. With this mapping and all the information presented in the domain ontology, we can support the software developers in the conceptual data modeling.

4 Enterprise Oriented Software Development Environments

Developing DOSDE for different domains, we could see the importance of knowledge in an organization supporting their activities. Further, we could verify that as well as domain knowledge, other kinds of knowledge are of interest to increase software productivity and quality. It includes knowledge about the organization itself, specialized knowledge about software development and maintenance, obtained on previous software projects within the organization, and knowledge about its clients. To deal with these different kinds of knowledge Villela et al. [15] evolved DOSDE to consider

enterprise knowledge organized in ontologies in the definition of Enterprise-Oriented Software Development Environment (EOSDE).

EOSDE has the following goals: (a) to provide software developers with all relevant knowledge for software development held by the company, and (b) to support organizational learning about software development. EOSDEs are firmly based on ontologies.

Figure 5 gives an overview of the components of an EOSDE. The Knowledge Management Infrastructure is composed of Organizational Memory and the Knowledge Management Services/Tools. Knowledge Management Services/Tools support the storage of data, knowledge and experiences in the Organizational Memory, promoting the dissemination and evolution of its contents. Software Engineering Services/Tools support the activities of software development and maintenance as well as the management of these activities. These services/tools must be able to provide software engineers with all the knowledge held by the organization that is relevant for the activity being carried out using the Knowledge Management Infrastructure. A project repository stores all data related to the software project. The organizational memory is composed of the following components:

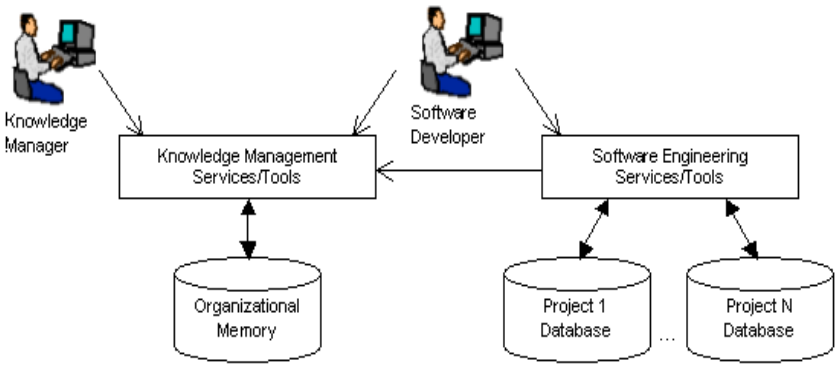

Fig. 5: Overview of the EOSDE Components

- The Domain Theory and Description of Tasks components that organize, respectively, the domain theory and PST knowledge as defined in the previous section.
- The Software Engineering (SE) Theory component guides the registration and dissemination of organizational knowledge about software engineering, such as historical data, lessons learned and best practices.
- The Enterprise Description component contains a description of the organization, identifying the generic tasks that are performed, and the software engineering knowledge necessary in the context of the organizational structure and processes. If the organization develops and maintains software for its own use (Non-Software Company), this component also sets which domain knowledge is required throughout the organizational structure and processes. The organizational process models allow the specification of the context in which a knowledge item was created and the application context for it. The

organization's knowledge map is also part of this component and defines the competencies each employee has and to which degree these competencies are held.

- The Description of Clients component that is specific of EOSDE created for Software Companies, which develops and/or maintains software for clients. It is similar to the Enterprise Description component, but it describes the client organizations. In this context, an Enterprise Ontology is fundamental to define a common vocabulary to guide the description of any organization.
- The Knowledge/Data Bases component stores knowledge and data relevant to the organization acquired and updated throughout many software projects. Each knowledge item stored in the environment is associated with one or more concepts and instances of these concepts obtained from the EOSDE ontologies. It enables subsequent retrieval of different types of knowledge items based on the selection of concepts and instances, regardless of the specific tools used to record and read the knowledge items.

From this description of the EOSDE components, it can be seen that the use of ontologies is critical to make the retrieval of knowledge stored in the environment as well as communication among multiple users and tools more straightforward. When retrieving knowledge items, the purpose of ontologies is to supply vocabulary whose terms are used as indexes to access the knowledge items and as links among multiple knowledge/database contents. Furthermore, when defining synonyms and acronyms for concepts, ontologies provide linguistic equivalents that may occur in text documents and can be used to access knowledge. As regards communication, the defined ontologies have the purpose of reducing terminological and conceptual mismatches by providing shared vocabularies. A typical class model can be created based on an ontology and used by various tools as well as matches among classes from different models can be made through their association to the ontology terms.

As mentioned previously, the Enterprise Ontology aims to supply a common vocabulary that can be used to represent useful knowledge on the organizations involved in a software project for the software developers. It can be useful for:

- supplying a structure to organize knowledge and guide knowledge acquisition in one or more organizations,
- allowing the development of generic tools based on its structure,
- promoting the integration among tools that manipulate knowledge related to the ontology,
- facilitating the development of systems that manipulate knowledge on the organization (for example, a system that supports an organizational process),
- assisting the identification of professionals with the appropriate competencies for discussing ideas about a subject, for guiding the execution of a task, or for putting together a team to suit the characteristics of the project.

Figure 6 shows the sub ontologies of the enterprise ontology, which were defined to answer the questions on how the organization is perceived in its environment; how the organization is structured; and how the distribution of authority and responsibility is accomplished, who works in the organization and how the desired and possessed

competencies have been distributed within it, how the organization behaves and the objectives it has.

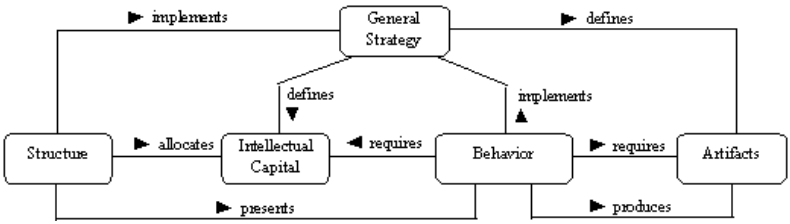

Fig. 6: Subontologies of the Enterprise Ontology

The Enterprise Ontology was developed combining new concepts with others defined by Fox et al. [16] and the TOVE project (TOronto Virtual Enterprise) [17].

The *Intellectual Capital* subontology deals with aspects such as taxonomy of competence, the interaction between experience and knowledge, availability of competencies and breakdown of the knowledge domain.

The *Structure* subontology deals with the organization of organizations, distribution of authority and responsibilities among organizational units, how they are broken down into organizational units, distribution of authority and responsibilities among positions, specification of functions and positions, staff allocation, the definition of teams and definition of objectives.

The *Artifacts* subontology groups the concepts and relationships that define artifacts in terms of their nature and composition. An Artifact is anything produced by humans and not by natural causes that are able to exert different roles in an organization, such as the product of an activity.

The aspects covered by the *Behavior* sub ontology include activity as an act of transformation, taxonomy of activity, process and activity breakdown, adoption of procedures, a taxonomy of procedures, method as a systematic procedure, automation of procedures, organizational processes and related norms as well as organizational projects. An Activity is an action of transforming raw material and/or input artifacts into output artifacts, which may require competencies and the use of goods for production. An activity can be classified in operational activity, managerial activity or quality control activity according to its nature and into the primary activity or a support activity according to its role in the fulfillment of the organization's mission. An activity can also be made up of a set of other activities. A Process is a set of structured activities that produce artifacts or services of value to the organization, for a client or a business market. Procedures are instructions for executing activities and are classified into methods, techniques and guidelines. Methods, as well as Techniques, can be classified according to the type of activities they can support. Guidelines are further classified into templates and norms. Software tools may support a procedure. An organization has its behavior defined by the set of processes executed within it and they may comply with norms. Projects are undertakings initiated by the organization that entails processes to guide their activities and have project teams allocated to them.

The *General Strategy* subontology establishes the vocabulary to describe how the organization interacts with its environment, that is, its domain of performance, the artifacts/services it offers and the relationships with client organizations.

5 Corporation Oriented Software Development Environments

Corporations are large organizations that have subordinated software development organizations and need to manage the diversity of processes in this context. Those software development organizations typically have different needs and characteristics that influence their software process. For instance, they might have to adhere to specific maturity models and software development models or deal with different business needs and markets. To increase competitiveness and control over the software development practices, the corporation must have specific mechanisms to support its software development organizations to reach higher maturity levels.

Santos et al. [18] evolved EOSDE to a broader context by creating the Corporation-Oriented Software Engineering Environments (COSEE). COSEE aims at supporting corporations to manage their organizations' software processes diversity and maturity levels. Moreover, the computational support allows corporations and organizations to manage and control their several software processes in a coordinated and semi-independent way along with the organizational knowledge needed to define, improve and execute them.

In order to do so, COSEE needs a common vocabulary to support describing any organization (or corporation) structure and that can be used to represent useful knowledge to software engineers during the execution of software projects. Hence, we evolved the Enterprise Ontology to (i) describe Corporation; (ii) extend the definition of Process concept to Standard Process, Specialized Process, and Project Process; and (iii) define Process Asset concept. The knowledge provided by the revised Enterprise Ontology is used to support data modeling and to guide the use and construction of case tools associated with the definition and execution of software processes and descriptions of both corporations and organizations structure.

6 Conclusion

Software engineering is a knowledge-intensive activity, and knowledge is thought to be an essential asset in an organization. This paper presents our effort in defining ontologies and implementing ontology-based supporting tools for software development environments. Our work followed the pioneering work of Falbo et al. [6] that introduced knowledge in SDEs through ontologies. Based on this work, we have used ontologies to define Domain Oriented Software Development Environments, Enterprise Oriented Software Development Environments and Corporation Oriented Software Development Environments.

Acknowledgements

Prof. Travassos is a CNPq Researcher (grant304234/2018-4) and an ISERN member.

References

1. Rocha, AR, Aguiar TC, Souza, JM (1990) Taba: A Heuristic Workstation for Software development. COMPEURO 90, Tel Aviv, Israel, May.
2. Falbo, R.A, Travassos, G.H. (1997); "Improving Tool's Integration on Software Engineering Environments Using Objects and Knowledge," Proceedings of the SCI'97/ISAS'97, Caracas, Venezuela.
3. Newell, A.(1982); "The Knowledge Level," Artificial Intelligence, 18.
4. Falbo, R.A., Menezes, C.S., Rocha, A.R. (1998); "A Systematic Approach for Building Ontologies." Proc. of the IBERAMIA'98, Lisbon, Portugal
5. Falbo, R.A., Menezes, C.S., Rocha, A.R. (1998); "Using Ontologies to Improve Knowledge Integration in Software Engineering Environments," Proc. SCI'98/ISAS'98, Orlando, USA.
6. Falbo RA, Menezes CS, Rocha AR (1999) Using Knowledge Servers to Promote Knowledge Integration in Software Engineering Environment. Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering: 170-174.
7. Oliveira KM, Zlot F., Rocha AR et al. (2004) Domain-oriented software development environment. Journal of Systems and Software, v. 72, n. 2: 145-161.
8. Guarino N (1998) Formal Ontology and Information System, In Guarino, N. (ed.) Formal Ontology in Information System, IOS Press: 3-15.
9. Mizoguchi R, Ikeda M, Sinitsa K (1997) Roles of Shared Ontology in AI-ED Research, Artificial Intelligence in Education AI-ED 97. IOS Press: 537-544.
10. Mizoguchi, R, Vanwelkenhuysen J, Ikeda M (1995) Task Ontology for Reuse of Problem Solving Knowledge.N.J.I Mars Ed. IOS Press.
11. Breuker J., Van Der Velde W. (1994) CommonKADS Library for Expertise Modeling. Reusable Problem Solving Components. Ed. IOS Press, Ohmsha.
12. OMG - Object Management Group (2004) UML 2.0. Unified Modeling Language, htpp://www.uml.org (last access February 11, 2006).
13. Oliveira KM, Galotta C., Rocha AR, Travassos, GH, Menezes C (1999) Defining and Building Domain-Oriented Software Development Environments. Proceedings of 12th International Conference Software & Systems Engineering and their Applications.
14. Oliveira KM, Rocha AR., Travassos G.H., Menezes C. (1999) Using Domain-Knowledge in Software Development Environments. Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering: 180-187.
15. Villela, K., Santos, G., Schnaider, L., Rocha,A.R.C., Travassos, G.H. (2005) The Use of an Enterprise Ontology to Support Knowledge et al. Management in Software Development Environments, JBCS, vol 11, n 2, 45-60
16. Fox M., Barbuceanu M, Gruninger M (1996) An Organization Ontology for Enterprise Modeling: Preliminary Concepts for Linking Structure and Behaviour. Computers in Industry, v. 29: pp. 123-134.
17. Uschold, M., King M., Moralee S. et al. (1998) The Enterprise Ontology. The Knowledge Engineering Review, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate), v. 13, n. 1: 31-89.
18. Santos, G., Rocha, A.R.C., Travassos, G. H. (2009), Corporation Oriented Software Development Environments In VIII Brazilian Symp. Software Quality Ouro Preto (in Portuguese).

Aprendendo juntos

Crediné Silva de Menezes

Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil
credine@gmail.com

1. Introdução

Conheci Ricardo Falbo em 1991, alguns meses depois de iniciar minhas atividades acadêmicas na Universidade Federal do Espírito Santo (UFES). Nossa aproximação se deu quando participei do corpo docente de um curso de Especialização em Informática Industrial, curso no qual ele era aluno. Foi nesta época que nossas interações começaram. Entremeando conversas sobre os temas da disciplina que eu ministrava, dicas sobre usos e costumes capixabas e, tópicos diversos da área de computação, fomos construindo nossa amizade.

Ricardo atuava no Departamento de Matemática, onde ministrava aulas no curso de Ciência da Computação e, sendo especialista em Análise de Sistemas, mantinha forte interesse em Engenharia de Software, uma área de pesquisa na qual eu já havia trabalhado. Minha vinculação era com o Departamento de Engenharia Industrial e Informática, com atuação docente junto ao Curso de Graduação em Engenharia da Computação. Nossas primeiras conversas sobre Engenharia de Software se deram neste período. As interações prosseguiram pelos anos seguintes, com o assunto engenharia de software ganhando cada vez mais espaço em nossas conversas.

No início do ano de 1993, iniciamos na UFES uma empreitada para criação de um programa de mestrado em informática (PPGI-UFES), que veio a ser implantado em 1994. Antes que isso acontecesse, criamos o Departamento de Informática (DI), onde se juntaram dois grupos de professores que até então estavam lotados nos dois departamentos acima citados. Foi nessa efervescência, em busca de criar um forte grupo de pesquisa em Ciência da Computação, que eu e Ricardo intensificamos nossas conversas, buscando encontrar uma boa oportunidade para o seu ingresso em um programa de doutorado.

As alternativas foram sendo analisadas e numa de nossas análises vislumbramos a possibilidade do Ricardo realizar uma pesquisa em engenharia de software, como parte do seu doutoramento.

2. A cooperação com a COPPE-Sistemas

Em um encontro com a Profa. Ana Regina C. da Rocha da COPPE-Sistemas da Universidade Federal do Rio de Janeiro (UFRJ), minha amiga desde quando fomos contemporâneos na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), ela cursando o doutorado e eu o mestrado, surgiu a possibilidade de parceria em um

projeto de grande porte em Ambientes para Desenvolvimento de Software, liderado por ela, no qual o Ricardo situaria a sua pesquisa de doutorado, com a orientação de Ana Regina e minha coorientação.

Para mim, um recém-doutor, abriu-se uma grande oportunidade para participar de um projeto de pesquisa em um grupo com grande experiência. Para Ricardo a porta de entrada em um programa de doutoramento na área de pesquisa de seu interesse. Convite aceito, compromisso estabelecido, iniciamos nossa cooperação.

O projeto liderado por Ana Regina tinha como meta a construção de Ambientes para o Desenvolvimento de Software e já havia produzido a Estação Taba [18]. O prosseguimento da pesquisa apontava para um novo horizonte, a construção de um framework configurável para a gerência do processo de desenvolvimento de software. Vários componentes haviam sidos produzidos e a tese de doutorado de um dos membros da equipe da COPPE (Prof. Guilherme Travassos) já havia sinalizado para necessidade de caminhar na direção de Um Servidor de Conhecimento que possibilitaria integrar todo o conhecimento sobre o processo de desenvolvimento de software [17].

O processo de desenvolvimento de um software, com bem sabemos, envolve diferentes etapas, cada uma delas com suas especificidades, linguagens e produtos. Isto requer a concepção de estratégias para tratar a interoperabilidade [3]. O problema a ser abordado na pesquisa de Ricardo Falbo foi situado como "integração do conhecimento em um ambiente de gerência do processo de desenvolvimento de software" [2, 5]. Na época, e ainda hoje, o assunto é considerado crítico na gestão de conhecimento em contextos complexos.

Antecedendo o esforço de modelagem do conhecimento necessário para a interoperabilidade foi realizado um estudo para a caracterização do problema, conforme reportado em [4]. Os vários estudos realizados, inspirados em experiências no âmbito da inteligência artificial, sobre processamento de conhecimento, nos levou a selecionar a abordagem baseada em ontologias, na época um caminho novo, tendo por base uma fundamentação teórica capaz de sustentar um processo formal de validação [5, 7].

O projeto foi bem-sucedido, na medida em que a elaboração de uma ontologia sobre o processo de desenvolvimento de software viabilizou a integração pretendida. Esta ontologia se mostrou adequada à instanciação satisfatória de ambientes específicos e para a construção de ferramentas inteligentes para apoiar essas instanciações [6, 8]. Além deste resultado principal, dois outros resultados foram obtidos como produto da Tese, um processo para elaboração de ontologias e uma linguagem gráfica para descrição de ontologias [7].

A oportunidade de trabalhar com um grupo acadêmico de excelência, liderado pela Profa. Ana Regina, que junta qualidade acadêmica com generosidade, deu uma contribuição primorosa ao desenvolvimento de nosso grupo de Pesquisa em Engenharia de Software baseado em Ontologia na UFES. Aprendemos muito com cada um dos componentes do grupo COPPE, alunos e professores com os quais interagimos. Certamente foi uma contribuição de elevado valor para a concretização de nossa empreitada na criação de nosso programa de pós-graduação. Para Ricardo, uma excelente experiência. Desde os primeiros momentos se integrou ao grupo da COPPE onde

participou intensamente do desenvolvimento da pesquisa como um todo e formou uma grande rede de cooperação com seus professores e colegas.

3. O Grupo de Estudos sobre Ontologia

Algumas iniciativas anteriores em nossos campos de atuação nos aproximaram dos estudos sobre Ontologias. A modelagem de conhecimento em ambientes de processamento de conhecimento baseada em lógica e o desenvolvimento de sistemas orientados a objetos serviram como ponto de partida. Em busca de teorias que servissem de fundamentação para essas duas vertentes encontramos suporte na obra de Mario Bunge The *Furniture of the World* [1], um tratado em ontologia, texto no qual o autor apresenta um estudo sobre a *mobília do mundo*. O contato com a obra de Bunge constitui-se assim em uma alavanca para migrarmos da representação da informação para a representação de conhecimento. Esses estudos consistiram em leituras e discussões. As discussões ocorriam semanalmente, nas dependências da biblioteca central da UFES, no Campus de Goiabeiras, onde encontramos as condições apropriadas: silêncio, temperatura amena, isolamento e uma bela paisagem. Era a oportunidade de ficarmos afastados das demais atividades do campus, imersos em nossas elucubrações sobre ontologias. Foram tardes proveitosas que nos propiciaram a compreensão que precisávamos para obter a modelagem pretendida. Outros encontros, na COPPE e em diferentes locais informais, tanto na cidade do Rio de Janeiro quanto em Vitória, foram imprescindíveis para que nos tornássemos grandes parceiros em busca da criação de um modelo de conhecimento adequado ao problema e para consolidar nossas concepções sobre a profissão professor-pesquisador.

Durante o andamento desses estudos a doutoranda Káthia Marçal de Oliveira, hoje professora na Université polytechnique des Hauts-de-France, amiga de Ricardo, com projeto de pesquisa em tema correlato, vislumbrou a possibilidade de também usar a abordagem em sua pesquisa, juntando-se a partir dai ao nosso grupo de estudos. O interesse teve origem nos resultados preliminares obtidos na modelagem da integração de conhecimento. As interações com Ricardo ajudaram-na a conhecer os fundamentos necessários para aplicar os conceitos de ontologia na modelagem de conhecimento que deram o suporte necessário à elaboração de sua tese de doutoramento também com a orientação da Profa. Ana Regina [11, 12, 13, 14, 15, 16].

A partir do ingresso de Káthia, de quem também já me tornara amigo, várias de nossas reuniões passaram a ser feitas em trio, mesmo quando era necessário que ela se deslocasse do Rio de Janeiro para nos encontrar em Vitória. Como os problemas que pesquisavam tratavam de aspectos diferenciados, desdobramos nossos estudos em reuniões com diferentes configurações de participantes. Hoje, fazendo uma retrospectiva, avalio a experiência que juntos realizamos como um caso bem sucedido de aprendizagem cooperativa.

4. O que veio depois do doutoramento

Na volta do doutoramento, Ricardo se empenhou na implantação de uma nova vertente da linha de pesquisa em Engenharia de Software no Programa de Pós-graduação em UFES (PPGI) , tendo como suporte teórico o uso de ontologias para apoiar o desenvolvimento de software. Uma das primeiras inciativas foi o Projeto ODE, onde Ricardo e seus alunos se dedicaram intensamente à concepção e construção de um Ambiente para Desenvolvimento de Software baseado em ontologias. Em torno do ODE foram elaborados vários trabalhos de conclusão de cursos de graduação (Engenharia da Computação e Ciência da Computação) e dissertações de mestrado no PPGI-UFES) .

As contribuições das pesquisas do Ricardo usando ontologias começaram a apresentar resultados no PPGI-UFES já no ano de 2000, quando em parceria com o Prof. José Gonçalves, o Ricardo coorientou o mestrando Giancarlo Guizzardi [10], hoje professor do Departamento de Informática da UFES e um renomado pesquisador em Modelagem Conceitual e Ontologias no cenário Internacional. Desde 2005 Ricardo participa do NEMO (Núcleo Estudos em Modelagem Conceitual e Ontologias) com vários colegas do PPGI, incluindo entre outros os professores Giancarlo Guizzardi, João Paulo Almeida, Renata S. Guizzardi, Monalessa Perini Barcellos e Vítor Estevão S. Souza além de membros de outras instituições. É importante destacar que o NEMO é hoje um conceituado grupo de pesquisa com repercussão internacional.

Por alguns anos, Ricardo e eu, ainda seguimos a nossa cooperação formal no contexto do PPGI, quando orientamos em conjunto algumas dissertações no programa, escrevemos artigos e participamos de bancas de mestrado. A cooperação informal, ou seja, aquela troca de ideias entre amigos prossegue, apesar de hoje serem mais esporádicas, em virtude de estarmos atuando em de áreas de pesquisa distanciadas.

5. Considerações finais

Para encerrar esta homenagem, à qual não podia me omitir, gostaria de reiterar as grandes qualidades deste amigo, um ser humano fraterno, comprometido com a formação de profissionais inovadores na área de computação e de cidadãos com nobres valores de cidadania.

Suas inúmeras e inquestionáveis contribuições acadêmicas sejam como professor ou pesquisador estão registradas e falam por si só: formação de pessoal nos níveis de graduação (35 TCCs) , mestrado (35 dissertações) e doutorado (7 teses) e para a pesquisa em sua área de atuação onde se pode enumerar uma quantidade considerável de artigos científicos (acima de 130) . O impacto de suas contribuições vai além, nestes 25 anos de pesquisa ele ajudou a constituir e a consolidar no país uma linha de pesquisa importante no cenário internacional no campo da modelagem de conceitual que representa um papel imprescindível no desenvolvimento de aplicações inteligentes. A grande rede de amigos e colaboradores que ele soube articular poderá em seus escritos de homenagem, nos ajudar a ter uma visão mais acurada de suas contribuições.

Para mim, que tive a satisfação de compartilhar com ele alguns anos de vida acadêmica fica a certeza de que a oportunidade de trabalharmos em cooperação me propiciou grandes aprendizagens e grandes alegrias.

Certamente todos ainda teremos a oportunidade de aprender muito mais com a generosidade de Ricardo que deixa oficialmente a instituição, mas que certamente continuará a produzir e compartilhar conhecimento científico e lições de vida de elevada qualidade. Certamente fora do contexto formal, com menos sobrecarga e com mais qualidade de vida continuaremos nossas animadas e proveitosas conversas.

Referências

1. BUNGE, M., 1977, Ontology I: The Furniture of the World, volume 3 of Treatise on Basic Philosophy, D. Reidel, Dordrecht, Holland.
2. FALBO, R. A.; MENEZES, Crediné Silva de . (1995) Um Estudo sobre Ambientes de Desenvolvimento de Software com Suporte Baseado em Conhecimento. In: XXI Conferência Latino-Americana de Informática, CLEI'95, 1995, Canela, Rio Grande do Sul. Anais da XXI Conferência Latino-Americana de Informática, CLEI'95, 1995. p. 339-350.
3. FALBO, R. A.; ROCHA, Ana Regina Cavalcanti da .(1996) Requisitos de Ambientes de Desenvolvimento para Suportar Processos de Software. In: VII Conferência Internacional de Tecnologia de Software, 1996, Curitiba - PR. Anais da VII Conferência Internacional de Tecnologia de Software, 1996.
4. FALBO, R. A.; TRAVASSOS, Guilherme Horta (1996). A Integração de Conhecimento em um Ambiente de Desenvolvimento de Software. In: Congreso Argentino de Ciencias de la Computacion, 1996, San Luis, Argentina. Anais do 2do Congreso Argentino de Ciencias de la Computacion, 1996. p. 327-337.
5. FALBO, R. A.; MENEZES, Crediné Silva de .(1997) Improving Tools Integration on Software Engineering Environments Using Objects and Knowledge. In: World Multiconference on Systemics, Cybernetics and Informatics, SCI'97/ISAS'97, 1997, Caracas, Venezuela. Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, 1997.
6. FALBO, R. A.; MENEZES, Crediné Silva de ; ROCHA, Ana Regina Cavalcanti da .(1999) Using Ontologies to Improve Knowledge Integration in Software Engineering Environments. In: World Multiconference on Systemics, Cybernetics and Informatics, SCI'98/ISAS'98, 1998, Orlando. Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, 1999.
7. FALBO, R. A.; MENEZES, Crediné Silva de ; ROCHA, Ana Regina Cavalcanti da.(1998) A Systematic Approach for Building Ontologies. In: Sixth Iberoamerican Conference on Artificial Intelligence, IBERAMIA'98, 1998, Lisboa. Progress in Artificial Intelligence - IBERAMIA'98 (Lectures Notes in Computer Science). Berlin/Germany: Springer-Verlag, 1998. v. 1484. p. 349-360.
8. FALBO, R. A.; MENEZES, C. S. ; ROCHA, Ana Regina Cavalcanti da. (1999a) AssistPro: Um Assistente Baseado em Conhecimento para Apoiar a Definição de Processos de Software. In: XIII Simpósio Brasileiro de Engenharia de Software - SBES'99, 1999, Florianópolis. Anais do XIII Simpósio Brasileiro de Engenharia de Software - SBES'99, 1999. p. 147-162.
9. FALBO, R. A.; MENEZES, Crediné Silva de ; ROCHA, Ana Regina Cavalcanti da.(1999b) Um Servidor de Conhecimento de Processo de Software. In: X Conferência

Internacional de Tecnologia de Software - X CITS, 1999, Curitiba -PR. Anais da X Conferência Internacional de Tecnologia de Software: Qualidade de Software, 1999. p. 209-221.

10. GUIZZARDI, G. Desenvolvimento para e com Reuso: Um Estudo de Caso com um Sistema de Vídeo Sob Demanda. 2000. Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo.
11. OLIVEIRA, K. M.; ZLOT, Fabio ; ROCHA, Ana Regina ; TRAVASSOS, Guilherme H. ; GALLOTA, Catia ; MENEZES, Credine.(2004) Domain-oriented software development environment. Journal of Systems and Software, Estados Unidos, v. 172, n.2, p. 145-161, 2004.
12. OLIVEIRA, K. M.; GALLOTA, C. ; MENEZES, Credine ; TRAVASSOS, Guilherme Horta ; ROCHA, Ana Regina . (1999) Defining and Building Domain-Oriented Software Development Environments. In: 12th International Conference Software & Systems Engineering and their Applications, 1999, Paris, 1999.
13. OLIVEIRA, K. M.; MENEZES, Credine ; TRAVASSOS, G. ; ROCHA, Ana Regina.(1999) CORDIS: Assistência Automatizada no Desenvolvimento de Software em Cardiologia. In: de Simposio en Informática y Salud - 28 Jornadas Argentinas de Informática e Investigación Operativa, 1999, Buenos Aires, 1999. v. 1. p. 38-48.
14. OLIVEIRA, K. M.; MENEZES, Crediné ; TRAVASSOS, Guilherme Horta ; ROCHA, Ana Regina.(1999) Using Domain-Knowledge in Software Development Environments. In: Software Engineering and Knowledge Engineering, 1999, Kaiserlautern. Proceedings of SEKE, 1999. v. 1. p. 180-187.
15. OLIVEIRA, K. M.; ROCHA, Ana Regina ; TRAVASSOS, Guilherme Horta. (1999) O uso da Teoria do Domínio no Processo de Desenvolvimento de Software. In: X Conferencia Internacional de Tecnologia de Software, 1999, Curitiba. Anais do CITS. Curitiba: cits, 1999. p. 223-235.
16. OLIVEIRA, K. M. (1999) Modelo para Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio, Tese de Doutorado. COPPE/UFRGS, Rio de Janeiro.
17. TRAVASSOS, G. H. (1994) O Modelo de Integração de Ferramentas da Estação TABA. Tese (Doutorado). COPPE/UFRJ, Rio de Janeiro.
18. ROCHA, A. R. C.. Taba: Uma Estação de Trabalho Para O Engenheiro de Software. In: XIV CONGRESSO LATINO AMERICANO DE INFORMATICA, 1988. BUENOS AIRES - ARGENTINA.

My Path Experiencing *Falbo's Approach*

Monalessa Perini Barcellos

Ontology and Conceptual Modeling Research Group (NEMO)
Computer Science Department – Federal University of Espírito Santo, Vitória, Brazil
monalessa@inf.ufes.br

Abstract. When I was an undergraduate student, I heard a lot about a certain professor, referred to my colleagues as the *best professor* they had, and I was sorry I did not have classes with him because he had taken a leave to obtain his doctor degree. Years later, I met that professor. I not only met him, but I had the honor of being his first doctorate student. At that moment, I found out that he was much more than the *best professor*. Being advised by him I learned so much as a person and as a student. After finishing my doctorate, I received the immeasurable gift of having this professor as my greatest partner at work and, above all, my great friend. Again, I confirmed that *best professor* was too little to him. What *Ricardo de Almeida Falbo* does makes him much more than that! Working with him is a continuous learning. He makes people better. I can say that from my own experience. More than ten years working with him and being friends with him made me a better person and a better professional. His generosity, competence, patience, excellence, zeal, wisdom, friendship and values make him much more than the *best professor.* He has that "extra" that turns an ordinary person into an *extraordinary person*. To me he always will be a *role model*, my *great friend,* a *mentor*, and a *great master*. This paper is a tribute to him, on the occasion of his formal retirement. Here, I summarize some works we have developed together along the last years.

1 Introduction

Writing this paper was such a pleasure! I spent hours recollecting many years of meetings, discussions, emails, messages, papers, conceptual models and specifications, and all those versions (there were so many! v0.1, v0.2… v0.25…!) of artifacts we produced until finally achieving the "v1.0", as a result we found good. Our meetings and discussions have always been productive and enjoyable. We used to have some fun even when the subject was hard. In fact, seeing the bright side of things is a hallmark of Falbo (and it is contagious!). After a while recollecting moments shared with Falbo along the path we went through together in the last years, I selected some results to report here. This paper is a summary of some works resulting from our partnership in the last years. Many other people are also involved in these works: students, colleagues, partners.

Years ago, a friend of ours found a paper that mentioned what was called "*Falbo's Approach*". Since then, in our research group, we have often used this term as a kind of joke, to play with Falbo. However, there is a "*Falbo's Approach*" indeed. It is not only

about his academical proposals. It is the way Falbo do things. It is the way he works, thinks, acts, values his friends, shares knowledge, sees life, chooses what does and what does not matter. Working with Falbo comes with the privilege of experiencing *Falbo's Approach*.

The works reported in this paper have a subject in common: *ontologies.* An ontology is a formal representation of a common conceptualization of a universe of discourse [1]. It is a useful instrument for reducing conceptual ambiguities and inconsistencies, and for making knowledge structures clearer. Ontologies can be classified according to their generality level into: *foundational ontologies*, which describe very general concepts, such as object, event, etc.; *domain ontologies*, which describe the conceptualization related to a domain (e.g., medicine, law); *task ontologies*, which describe the conceptualization related to a task or process (such as diagnosis and sale); and *application ontologies* that describe concepts dependent on a particular domain and task [1]. *Core ontologies* are between foundational and domain ontologies, providing a precise definition of structural knowledge in a specific field that spans across different application domains in this field [2].

Another important distinction differentiates ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies*. A reference ontology is constructed with the purpose of making the best possible description of the domain in reality, representing a model of consensus within a community, regardless of its computational properties. Operational ontologies, in turn, are designed with the focus on guaranteeing desirable computational properties [3].

The works I report here were developed in the *Ontology and Conceptual Modeling Research Group* (NEMO) and used UFO (*The Unified Foundational Ontology*) [4, 5, 6] as foundational ontology. UFO is based on a number of theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology. It is divided in three parts: an ontology of endurants (objects) [4], an ontology of perdurants (events) [5], and an ontology of social entities [6].

In the last years, I have worked with Falbo in two main areas: *Software Engineering* and *Ontology Engineering*. In the former, we have investigated how to apply ontologies to support software-related processes. In the latter, we have studied how to support ontology development, mainly by facilitating reuse. More specifically, in the Software Engineering context, we have developed several ontologies and applied them to support systems integration, semantic documentation and standards harmonization. In the Ontology Engineering context, we have proposed to develop ontologies as ontology networks and organize ontology patterns in Ontology Pattern Languages [7] in order to favor reuse.

This paper does not intend to present the works in detail. Contrarywise, it aims at giving a general view of some research topics that have been influenced by Falbo and research results that have been produced with his participation. Detailed information about the works mentioned here can be found in the respective references.

The paper is organized as follows. Sections 2, 3 and 4 regard the Software Engineering context, addressing, respectively, the Software Measurement Ontology, my first work with Falbo; Semantic Interoperability; and SEON, a Software Engineering

Ontology Network. Section 5 regards Ontology Engineering, focusing on Ontology Pattern Languages. Last, in Section 6, I present final considerations and personal notes.

2 Software Measurement Ontology

Everything started in 2008. I was working with software measurement and statistical process control in my doctorate research project, advised by professor Ana Regina Cavalcanti da Rocha, at COPPE/UFRJ, and I was looking for something innovative to improve my proposal. At that time, statistical process control was recent in software organizations and there were still many doubts about it. Discussing with Falbo about my research, he introduced me to the *Ontologies* subject and provided me the first materials I read about the topic, which included a paper by Guarino [1], and some papers and a thesis about UFO [4]. He talked to me about well-founded ontologies, the benefits of using them, and also about some applications he had done in the Software Engineering domain. After some discussion, I noticed that I could propose a reference ontology to provide a well-founded conceptualization about software measurement and statistical process control and explore it to help organizations in these practices. That would be great! Moreover, I could improve my research and start a new partnership.

Falbo accepted the invitation to be my co-adviser (Ana Regina Rocha and I are very grateful for that) and we worked for months developing the *Reference Software Measurement Ontology* (RSMO) [8, 9, 10, 11, 12], which is grounded in UFO [4, 5, 6]. I lost count of the number of versions were produced! When we though we have finished, a new question came up and we got back to the ontology. How much I learned in this process!

After working with Falbo as his advisee, I had the honor to work with him as work partners. Once we developed and used RSMO, we noticed that it would also be important to represent knowledge about the software measurement process, highlighting behavioral aspects that are not explicit in domain ontologies. Therefore, we built the *Task Software Measurement Ontology* (TSMO) [13]. RSMO and TSMO were applied in practice to support organizations to understand software measurement and statistical process control and also to help better understanding and thus harmonizing different measurement-related standards [13]. These applications were particularly relevant to software organizations that were interested in implementing practices from CMMI (Capability Maturity Model Integration) [14] levels 4 and 5, and MR MPS (Reference Model for Brazilian Software Process Improvement) [15] levels B and A. Hence, these works contributed to researchers and practitioners.

3 Semantic Interoperability

Over the years, our partnership was extended and involved many other people. We have worked in several research projects and advised together several students. One of the main problems we have explored in these works refers to *semantic interoperability*.

Nowadays, information is one of the most important assets for people and organizations. However, most often, information is spread in different and heterogeneous

sources (e.g., different documents, standards, web pages, models, information systems etc.) and it may not be easy to combine them and obtain useful information that properly supports decision making.

Obstacles for interoperability arise mainly from the fact that the information sources are usually created independently and do not share the same semantics in the used terminology. Different terms can be used to represent the same concept, while the same term can be used to represent different concepts. Without an explicit definition of the involved terms, it is difficult to know the correspondence between concepts from different information sources. However, sharing a terminology is not enough. The entities must share semantics, i.e., the meaning of the used terminologies.

Semantic interoperability, in general, refers to the ability to exchange information based on meaning. In several contexts, semantic conflicts arise because the things being integrated (systems, data, models and so on) do not share a common conceptualization. Thus, to solve semantic conflicts, ontologies can be used as an interlingua to map concepts from different sources.

In the last years, we have investigated semantic interoperability in three main contexts: *systems integration*, *semantic documentation* and *standards harmonization.*

3.1 Systems Integration

Systems integration is one of the biggest challenges faced by organizations to obtain consolidated information from several sources. Taking that into account, Calhau and Falbo developed OBA-SI (*Ontology-Based Approach for Semantic Integration*) [16], which proposes the use of ontologies to assign semantics and support mappings to integrate systems. OBA-SI defines an integration process similar to the software development process, comprising six phases. *Integration Requirements Elicitation* consists of establishing the integration requirements, identifying the business activities that will be supported by the integrated solution, and the systems to be integrated to support those activities. *Integration Analysis* is the phase in which the integration requirements are analyzed and modeled, features to be provided and concepts involved are specified, and the overall behavior of the integrated set of systems is defined. It involves using ontologies as reference models to map the systems models and produce the integration model to be used in the solution. Last, there are the *Design*, *Implementation*, *Tests*, and *Deployment* phases, when the conceptual solution previously modelled is turned into an operational solution [16].

After some experiences applying OBA-SI [16] in practice, we identified some improvement opportunities. First, when we tried to apply OBA-SI to integrate systems to support the software measurement process, we noticed that this domain has some particularities that demand specific activities and ontologies when using OBA-SI. Therefore, we decided to specialize OBA-SI to the software measurement domain. Second, we noticed that although OBA-SI satisfactorily addresses data integration, service and process integration are not so clear. Hence, we carried out investigations and detailed integration at these layers.

Integrating Systems to Support Software Measurement

Typically, software organizations use different systems to support different processes. For example, schedule and budget systems are used to support project management; modeling tools are used to support requirements engineering; and development environments and version control systems are used to support coding and source code management. Although these systems are not usually conceived to support software measurement, many times they store useful data related to the supported processes (e.g., number of defects, time and cost spent on activities, number of lines of code, test failure rate, etc.). In order to properly support the software measurement process, these systems must be integrated. However, this is not an easy task. The heterogeneity between systems to be integrated is the major difficulty. In general, each system runs independently and implements its own data and behavioral models, which are not shared between different systems, leading to several conflicts [17].

Therefore, we developed the *Ontology-Based Approach for Measurement Systems Integration* (OBA-MSI) [18], a systematic approach that uses the Reference Software Measurement Ontology (RSMO) [8, 9, 10, 11, 12] and the Software Measurement Task Ontology (SMTO) [13] to guide systems integration to support the software measurement process. OBA-MSI specializes OBA-SI [16]. OBA-SI can be applied to carry semantic integration of systems in any domain. However, systems integration in the software measurement domain has some peculiarities that are not properly addressed by OBA-SI. For instance, OBA-SI is more suitable for integrating applications that support the same process. However, systems integration to support the software measurement process typically involves applications designed to support different processes that must be integrated to the software measurement process. Besides, in OBA-SI, integration requirements elicitation is superficially addressed. In OBA-MSI, integration requirements elicitation is detailed following a goal-based approach. Figure 1 shows the first phases of the OBA-MSI process[1]. The last phases are the same than OBA-SI. Figure 2 details the first OBAM-SI phase. The complete description of OBA-MI is available at [18].

By specializing OBA-SI to the software measurement domain, three main contributions were given: *(i)* OBA-SI general steps were detailed and specialized to the case of software measurement, resulting in more palatable steps to be followed by users; *(ii)* we defined an integration approach more suitable for integrating systems developed with the aim of supporting very different software processes, but that produce input for the software measurement process; and *(iii)* an ontology framework made up of RSMO and SMTO was provided. One of the most effort-demanding step in OBA-SI is related to select (or develop) and integrate ontologies to be used in the integration initiative. In OBA-MSI, the ontologies to be used are provided and the effort is reduced to understand these ontologies.

[1] In this paper, the words *tool* and *system* are used as synonymous.

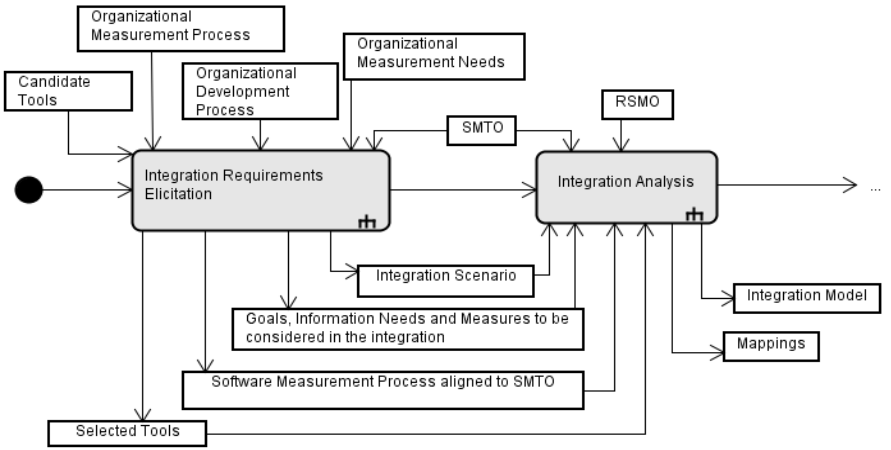

Fig. 1. First phases of OBA-MSI process.

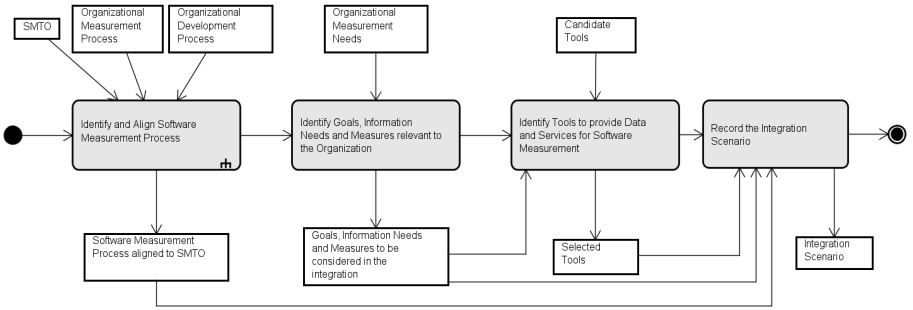

Fig. 2. OBA-MSI - Integration Requirements Elicitation phase.

Integrating Systems at Data, Service and Process Layers

Systems integration can be performed at three layers [17]: data, message, and process. *Data integration* deals with moving or federating data between multiple data stores. It assumes bypassing the application logic and manipulating data directly in the database. *Message (or service) integration* addresses messages exchange between the integrated applications. *Process integration*, in turn, views organizations as a set of interrelated processes and it is responsible for handling message flows, implementing rules and defining the overall process execution.

Our experiences using the first version of OBA-SI [16] showed us that data integration was well addressed, but service and process integration should be improved. When we developed OBA-MSI [18], we explored the Task Software Measurement Ontology (TSMO) [13] to address service and process integration. From these and other experiences, we decided to evolve OBA-SI [16] by improving integration at service and

process layers using task ontologies to assign semantics at these layers. Moreover, we argue that integration at process layer should consider a common understanding about business processes. Therefore, we proposed a *Business Process Ontology* (BPO) [19], a well-founded ontology grounded in UFO [4, 5, 6], which provides a common conceptualization about business process, to be used to support process integration. Figure 3 shows the overview of the Integration Analysis phase of OBA-SI after the improvements we made. A detailed description of this phase and of the current version of OBA-SI is made in [19].

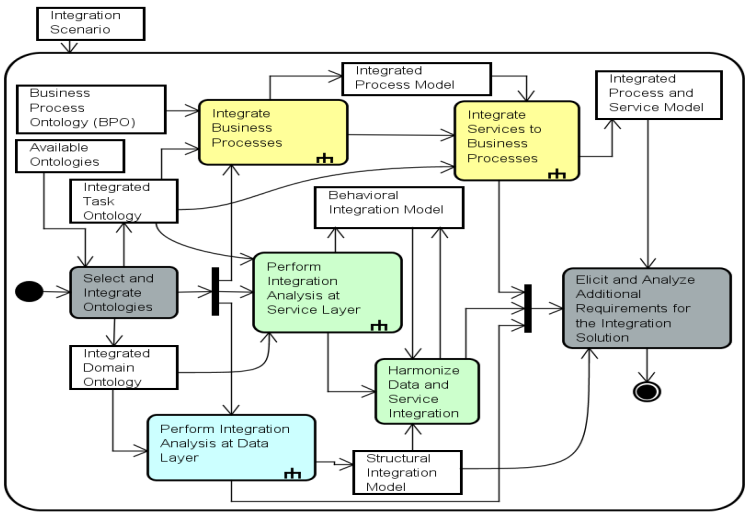

Fig. 3. OBA-SI - Integration Analysis phase addressing all layers [19].

OBA-SI [19] uses domain and task ontologies in different activities. Domain ontologies are used at both data and service layers. At data layer, domain ontologies' concepts, relations and properties are used to assign semantics to classes, associations and attributes of the systems' data conceptual models. For addressing integration at service and process layers, OBA-SI uses task ontologies. At service layer, domain ontology concepts are used to assign semantics to inputs and outputs of the functionalities/services provided by the systems being integrated. Task ontologies, in turn, are used to assign semantics to the systems' functionalities/services and to the business processes' activities. It is possible that more than one domain or task ontology are necessary to cover the domains or processes involved in the integration initiative. In this case, the ontologies must be integrated in a way that results in a single domain ontology and a single task ontology to be used in the integration initiative. Moreover, when both domain and task ontologies are needed, they must be integrated, giving rise to an application ontology, involving both domain and task perspectives.

With the improvements made in OBA-SI, in its current version [19] the integration process is clearer, guiding users on the activities to be performed and allowing users to understand the relations between the integration layers and address only the layers of

interest in a particular system integration initiative. Moreover, semantics assignment at service and process layers was improved, since task ontologies are more suitable for dealing with behavioral aspects than domain ontologies.

3.2 Semantic Documentation

In the context of software projects, documents hold a considerable amount of information that is mainly interpreted by humans [20]. One disadvantage of using documents is the difficulty of obtaining consolidated information from them, especially when information is spread in several documents. Accessing, recovering and managing information recorded in documents usually depend on human intervention and can be laborious and error-prone. Besides, gathering relevant information from different documents can be so wearing that people may tend not to do so [21].

Semantic Web deals with a similar issue by providing a way that both humans and computers can interpret the content of web pages. To reach this goal, web pages are annotated with metadata that describe fragments of the page content so that they become available for computer interpretation [22]. Since ontologies represent a conceptualization about the domain of interest and establish a common vocabulary to be shared, they are of great value to describe metadata, providing a rich formal semantic structure for their interpretation. Therefore, ontologies are often used as a basis for semantic annotation [23].

Semantic Web principles can also be applied to documents (e.g., documents produced in desktop text editors, electronic spreadsheets), giving rise to *Semantic Documentation*. Semantic Documentation combines documents and ontologies in the same way that Semantic Web, i.e., ontology-based metadata are created and then attached to the document content, resulting in a semantic document [24]. By doing that, syntactic information resources are turned into semantic information resources, which can be interpreted by computers. Once information recorded in the documents is automatically retrieved by computers, it is possible to develop tools to provide consolidated and integrated information for end users, decreasing the human effort necessary to obtain such information. Moreover, semantic annotation approach allows relating annotated contents and using the relationships to extract information from several documents, providing a general view that probably could not be gotten without the annotations [21].

Figure 4 illustrates the semantic annotation process for generating a semantic document. The person responsible for adding metadata to documents uses a supporting tool to semantically enrich the document. For each annotation, the tool creates semantic metadata relating a fragment of the document (referring to general information or document content) to an element of the ontology. Usually, metadata are kept in the semantic document. Once documents are annotated, it is possible to extract knowledge and link contents from different documents according to the shared ontology. By integrating content extracted from several documents, it is possible to achieve a more holistic view of the available knowledge [21].

Aiming at supporting semantic documentation, Falbo and Arantes [21] proposed an environment to manage desktop semantic documents, allowing semantic annotation of document templates. Thus, documents produced by using the annotated templates are

automatically annotated, easing end users work, since they can use the annotated templates to create semantic documents without concern with the annotations to be made. Figure 5 shows an overview of the infrastructure for managing semantic documents proposed by Falbo and Arantes [21].

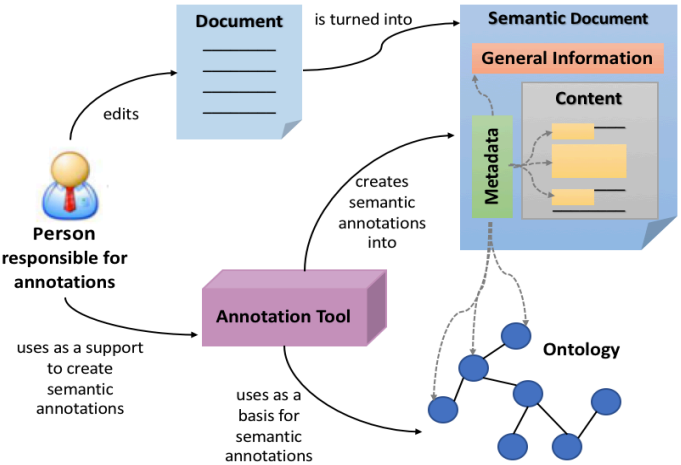

Fig. 4. OBA-SI Creating semantic documents [25].

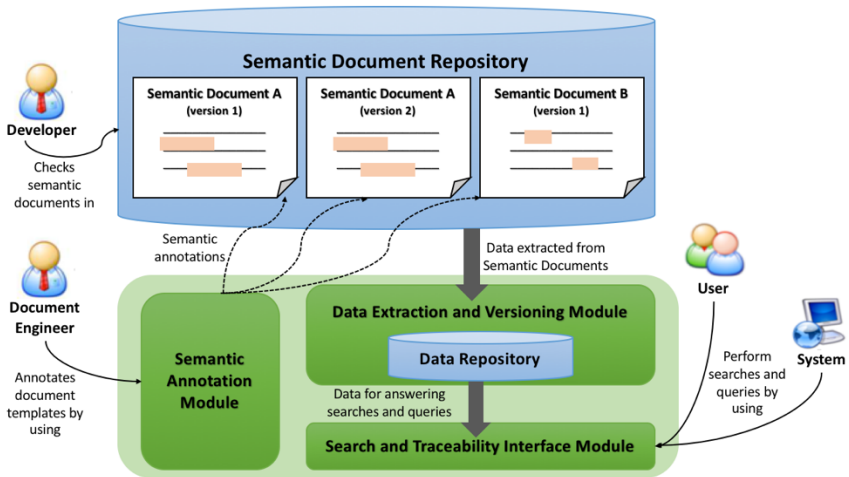

Fig. 5. Overview of the Infrastructure for Managing Semantic Documents [21].

Considering some industrial reports and experiences we had in some organizations, we noticed that many organizations adopt documents and spreadsheets to record information about software projects and to manage them. For example, during a software project, information regarding planning, execution, progress monitoring and control many times is recorded in text documents and spreadsheets (e.g., project plan and status

reports). Due to the importance of these documents for a project, project members should share them in a common environment and be able to access their content in an easy and efficient way [26]. Once documents are annotated, it is possible to build repositories containing semantic documents produced in the projects and develop semantic documentation systems able to retrieve consolidated information from them. Besides, semantic documentation helps store and retrieve knowledge acquired during a project and reuse it in other projects. Thus, we decided to explore the use of semantic documentation in software project management and provide an environment to support organizations in this context.

To provide the domain conceptualization, we developed the *Software Project Management Ontology* (SPMO) [25], grounded in UFO [4, 5, 6]. We used SPMO and extended the infrastructure proposed by Falbo and Arantes [21] to develop the Infrastructure for *Managing Semantic Documents for Software Project Management* (IMSD-SPM) [25], a semantic documentation infrastructure with specific features to support project management aspects. From data recorded in semantic documents and spreadsheets related to scope, time and cost produced along projects, features provide to managers: (i) a consolidated view of project planning regarding scope, time and cost; (ii) dependency matrices, showing dependency relations among project elements; (iii) consolidated information about project execution, pointing out the differences between planning and execution values; (iv) project performance indicators; (v) estimates for project conclusion; (vi) a global view of the performance of several projects, allowing for comparisons among them; and (vii) non-conformities detected in the semantic documents and spreadsheets content.

IMSD-SPM helps software organizations to get consolidated information from data recorded in different documents and spreadsheets. This contributes to well-informed decision making. It is important to point out that problems to retrieve information from documents and spreadsheets could be also addressed by approaches such as corporative architectures and information systems. However, although these approaches are capable of dealing with information recording and extraction, they imply in modifying the way organizations and people perform activities, since they replace documents by information systems. For organizations that use documents and spreadsheets (and prefer to keep this practice), using semantic documentation-based solutions, such as IMSD-SPM, can be an advantageous approach, because it accesses information recorded in the documents, allowing organizations and people to keep the way they perform activities.

3.3 Standards Harmonization

In the Software Engineering domain there are several standards produced by organizations such as ISO (International Standardization Organization), IEC (International Electrotechnical Commission), IEEE (Institute of Electrical and Electronics Engineers), SEI (Software Engineering Institute) and SOFTEX (Association for Promotion of Brazilian Software Excellence). Standards are produced in different moments, by different groups and with different purposes. Often, organizations need to combine different standards to achieve a certain goal. However, the standards do not share a

common conceptualization, and integrating them is not trivial. Thus, standards harmonization is essentially a semantic interoperability problem, as we can observe very interrelated information described in different and sometimes conflicting ways by distinct sources (standards).

We have worked on standards harmonization for several years. For example, in [13] we used TSMO as reference model to analyze and harmonize software measurement standards. After many experiences analyzing different standards and defining software processes that should be in conformance with several of them, we noticed that a systematic approach would be helpful. In 2015, Falbo was advising a doctorate student and the research topic was related to standards harmonization. At that time, we had already carried out an ontological analysis of de ISO/IEC 24744 [27] and participated in an ISO work group together. Thus, Falbo invited me to work as co-adviser of his student. Until then, we had worked together advising master students, but that was our first partnership as co-advisers of a doctorate student. The results were very promising. One of them, *Harmony* [28], is a systematic approach for conducting standards harmonization initiatives that uses the *Software Engineering Ontology Network* [29] (described below) for representing the standards and applying harmonization techniques. Harmony was develop considering knowledge from the literature and harmonization initiatives we experienced in the last years (e.g., [30]). It is based on three main actions: modeling, which refers to representing the standards as models (in opposition to text) for better dealing with them; mapping, which regards mapping the standards models to ontologies serving as a semantic referential; and integration, which concerns building a unified view of the standards by extending the domain view with the necessary standard's specific elements.

4 Software Engineering Ontology Network

Software Engineering (SE) is a wide domain, where ontologies are useful instruments for dealing with knowledge management related problems. When SE ontologies are built and used in isolation, some problems remain, in particular those related to knowledge integration. Along the years, we have produced several ontologies related to SE subdomains, such as software measurement [8, 9, 10, 11, 12], software process [31], requirements [32], configuration management [33], software project management [25], test [34] and others. We experienced some problems, such as the same concept appearing with different meanings in different ontologies and the same term being used to designate different concepts in different ontologies, among others.

In large and complex domains, as 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 [35]. On the other hand, representing each subdomain separately would be too costly, fragmented, and again hard to handle.

D'Aquin and Gangemi [36] point out a set of characteristics that are presented in "beautiful ontologies", from which the following ones can be detached: 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.

Therefore, we started to investigate how to develop and organize ontologies in an architecture that enables integrating existing ontologies and adding new ontologies, keeping consistence between them. Considering the characteristics cited above, we proposed to organize SE ontologies in an ontology network that supports creating, integrating and evolving its ontologies. 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 [35].

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. We proposed to organize the ON in layers. Briefly, in the background, we need a *foundational ontology* to provide the general ground knowledge for classifying concepts and relations in the ON. In the center of the ON, *core ontologies* should be used to represent the general domain knowledge, being the basis for the subdomain networked ontologies. Finally, going to the borders, (sub) *domain ontologies* appear, describing the more specific knowledge.

After defining the architecture, we have developed SEON [29]. SEON provides a well-grounded network of SE reference ontologies, and mechanisms to derive and incorporate new integrated subdomain ontologies into the network. In SEON, UFO [4, 5, 6] lies in the foundation layer, providing the common grounding for all the networked ontologies. At the core layer, there is the *Software Process Ontology* (SPO) [31], which provides a common conceptualization about software process, 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. Two external core ontologies are also integrated to SEON: the *Enterprise Ontology* (EO) [37], which addresses general concepts related to organizations; and the *Core Ontology for Measurement* (COM) [38], which deals with measurement-related general concepts. Finally, at the domain level, there are several ontologies addressing aspects related to SE subdomains.

Figure 6 shows the current status of SEON. Each circle represents an ontology. The circle size varies according to the ontology size (in terms of number of concepts, represented inside the circles in parenthesis). Arrowed lines denote dependencies between networked ontologies, and line thickness represents the coupling level between them (in terms of number of relationships between concepts in different ontologies). The blue circle represents a core ontology. Green circles represent the domain ontologies. The upper-left area comprising four requirements domain ontologies represents the ReqON subnetwork. EO and COM are not represented in the figure because they are external to SEON.

It is important to notice that, even adopting a layered architecture, SEON is a network and each new added node contributes for the whole network. When a new ontology is added, it should reuse existing elements (from a higher or the same layer). Other

ontologies, in turn, may be adapted to keep consistency and share the same semantics along the whole network. Even the core ontologies can evolve to adapt or incorporate new concepts or relations discovered when domain ontologies are created or integrated.

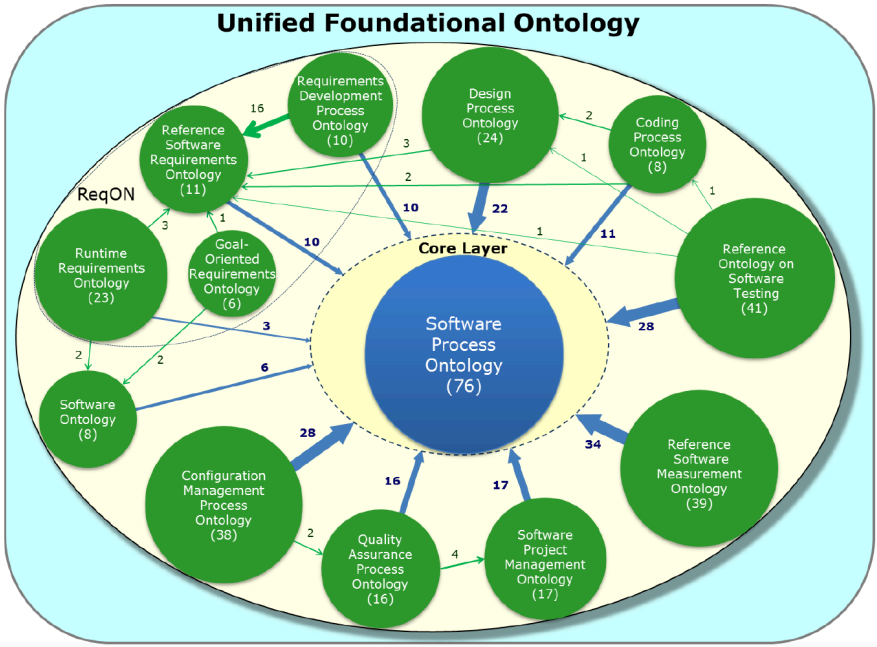

Fig. 6. SEON: The network view [28].

Being an ontology network, SEON is like a living organism and is constantly evolving. It requires a continuous and long-term effort with ontologies being added and integrated incrementally. Therefore, we have been continuously working on SEON. SEON specification is available at nemo.inf.ufes.br/projects/seon, where a machine processable lightweight version implemented in OWL is also available. Some experiences and envisioned applications of SEON are discussed in [29].

5 Ontology Pattern Languages

The works reported in the previous sections are concerned mainly to ontologies applied to Software Engineering. The success of using ontologies to solve knowledge-related or semantic interoperability problems is related to the quality of the used ontologies. The quality of an ontology, in turn, is strongly related to the quality of the languages, methods and tools used to develop it. Thus, it is important to advance on the theoretical and practical support for ontology engineering. In this context, Falbo has contributed to the ontology community by defining methods such as SABiO (*Systematic Approach for Building Ontologies*) [39], which supports ontology development by providing a set

of process and activities to be followed to produce reference and operational ontologies. Moreover, the architecture and framework defined to SEON can be used to define ontology networks related to other areas.

One of the processes prescribed by SABiO is Reuse, which has been pointed out as a promising approach for ontology engineering, helping speed up the development process and improve the quality of the resulting ontologies [40]. In this context, *ontology patterns* are an emerging approach that favors the reuse of encoded experiences and good practices. They are modeling solutions to solve recurrent ontology development problems [41]. We noticed that ontology patterns are often made available in catalogs. This does not favor reuse, since the sense of connection between patterns is lost. From our Software Engineering background, we introduced the *Ontology Pattern Language* (OPL) notion [7]. An OPL is a network of interconnected domain-related ontology patterns that provides holistic support for solving ontology development problems for a specific domain. An OPL contains a set of interconnected patterns, plus a modeling workflow guiding on how to use and combine them in a specific order, and suggesting patterns for solving some modeling problems in that domain [7].

It is important to highlight that we borrowed the term "pattern language" from Software Engineering, where patterns have been studied and applied for a long time. A pattern language, in a Software Engineering view, is a network of interrelated patterns that defines a process for systematically solving coarse-grained software development problems [42, 43]. Thus, we are not actually talking about a "language" properly speaking.

An OPL should indicate explicitly which referenced patterns address mandatory aspects and which ones address optional aspects. To ensure a stable and sound pattern application, referenced patterns should be presented in the suggested application order. Without this explicit procedural guidance, a representation that fits the basic network of the patterns might not provide a suitable process that helps to ensure a sufficiently complete and well-formed ontology.

OPLs are structured to support and encourage the application of one pattern at a time, in the order defined by the pattern sequences that result from the chosen paths through the language. This guideline ensures that the main property of piecemeal growth is preserved: the 'whole' always precedes its 'parts'. A pattern language is of little use if its audience loses the big picture. Conversely, the essential information of each individual pattern within the language must still be preserved [43].

In summary, an OPL should give concrete and thoughtful guidance for developing ontologies in a given domain, addressing at least the following issues: (i) What are the key problems to solve in the domain of interest? (ii) In what order should these problems be tackled? (iii) What alternatives exist for solving a given problem? (iv) How should dependencies between problems be handled? (v) How to resolve each individual problem most effectively in the presence of its surrounding problems?

Considering that core ontologies provide a precise definition of structural knowledge in a specific field that spans across different application domains in this field [2], the knowledge they represent is to be reused in many domains. Thus, we argue that core ontologies are good candidates to be presented as ontology pattern languages. Therefore, aiming to favor reuse, we organized five core ontologies as OPLs [44]: Software

Process OPL (SP-OPL), ISO-based Software Process OPL (ISP-OPL), Enterprise OPL (E-OPL), Measurement OPL (M-OPL), and Service OPL (S-OPL). Figure 7 shows a fragment of SPO from which we identified patterns to create S-OPL. The description and models of the OPLs cited above can be found in [44].

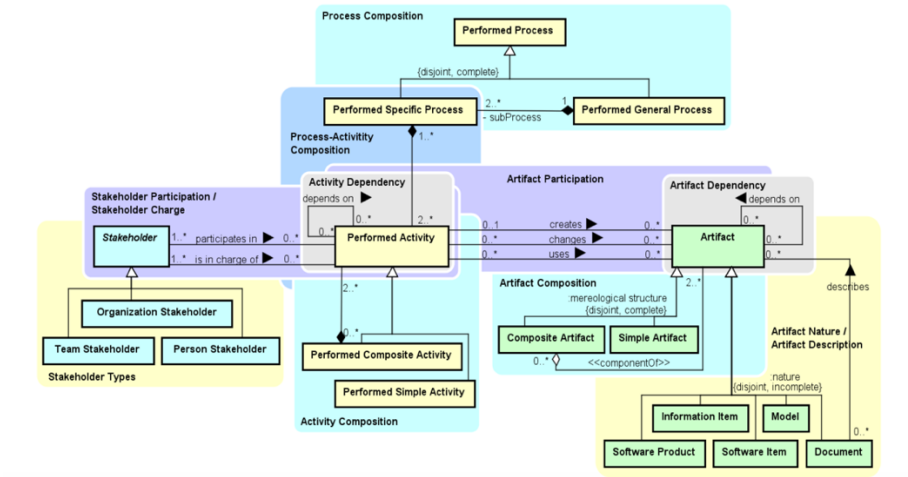

Fig. 7. Fragment of SPO with patterns identification [28].

When developing our OPLs, to create the OPLs diagrams that show how the patterns should be combined and the order in which they should be applied, we used an adaptation of the UML activity diagram. However, we experienced some limitations and there were inconsistencies in the notation we used in different OPLs. Thus, we noticed that it would be necessary to define a proper notation to represent OPLs. Hence, we created OPL-ML (*Ontology Pattern Language Modeling Language*) [45]. For developing OPL-ML, we relied on the results of a systematic mapping of the literature that investigated visual notations for Software Pattern Languages [46]. Moreover, OPL-ML was designed according to the principles of the Physics of Notation (PoN) [47], and following the design process defined by PoN-S (PoN Systematized), as we discuss in [48]. Figure 8 shows a fragment of the process (behavioral) model of S-OPL, which defines the flow to be followed to select the patterns to be applied according to the problem to be solved.

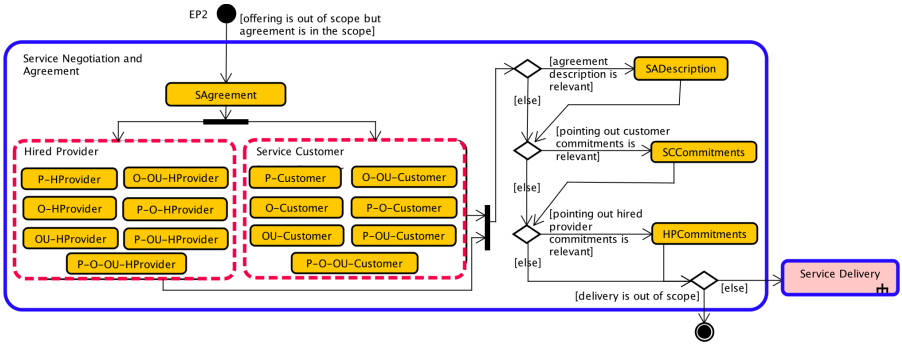

Fig. 8. Fragment of S-OPL – Behavioral Model [45].

Table 1 presents the notation used in the figure.

Table 1. OPL-ML visual notation – Behavioral Model [45].

Element	Symbol
Pattern Application Action Group (black box format)	<< name >>
Pattern Application Action Group (expanded format)	<<name>> <<name>>
Variant Pattern Application Action Group (black box format)	<< name >>
Variant Application Action Group (expanded format)	<< name >>

The complete specification of S-OPL is available at https://nemo.inf.ufes.br/projects/opl/.

6 Personal Notes

This paper was written in honor of *Ricardo de Almeida Falbo*, on the occasion of his formal retirement. It has been more than ten years of partnership. In this paper, I talked about some works on which we worked together during this period. Falbo has contributed to the research topics addressed in this paper, and also to many others. More than that, he has contributed to people's life, in academical and personal sense, helping form better researchers, with better character. Many people (students, colleagues, partners) participated in the works cited in this paper, and I am sure Falbo made a difference in their lives. I can say that from my own experience.

In the path I have followed in the last years, Falbo has played an important role. I have experienced *Falbo's Approach* as his advisee, as partners advising students, as partners in research projects, and most of all, as a friend. We celebrated accepted papers, got frustrated because papers were rejected, discussed research issues, worked a lot, talked about life in general. It is so easy to work with him!

I was once told that we should be able to express our feelings for someone in just three words. In the case of Falbo, my words are *friendship*, *admiration* and *gratitude*.

In this paper, I reported works developed along the path followed in the last years. There is a road ahead. Currently, we are working on ontologies in the Human-Computer Interaction (HCI) domain. Some time ago, a student we advise told us that she was interested in investigating HCI aspects in her doctorate research project. HCI area was new for both of us. Even so, he accepted the challenge of getting into this area and invited me to this project. This has been our more recent adventure together, and we expect to produce interesting contributions to the HCI and Ontology communities. Let's do it, my dear friend!

References

1. Guarino, N.: Formal Ontology and Information Systems. Formal Ontologies in Information Systems, IOS Press, 3 -15 (1998).
2. Scherp, A., Saathoff, C., Franz, T., Staab, S.: Designing core ontologies. Applied Ontology, 6(3), 177-221 (2011).
3. Guizzardi, G.: On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: Proceedings of the 2007 Conference on Databases and Information Systems IV: Seventh International Baltic Conference (DB&IS'2006), pp. 18-39 (2007).
4. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Fundamental Research Series. Centre for Telematics and Information Technology. Netherlands (2005).
5. 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: Proceedings of the 32th International Conference on Conceptual Modeling (ER 2013), pp. 327-341. Hong-Kong, China (2013).
6. 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).
7. Falbo, R.A., Barcellos, M.P., Nardi, J.C., Guizzardi, G.: Organizing Ontology Design Patterns as Ontology Pattern Languages. In: Cimiano P., Corcho O., Presutti V., Hollink L., Rudolph S. (eds) The Semantic Web: Semantics and Big Data. 10th Extended Semantic Web Conference (ESWC 2013). Lecture Notes in Computer Science, vol 7882, pp. 61-75. Springer, Berlin, Heidelberg (2013).
8. Barcellos, M. P., Falbo, R. A., Rocha, A. R.: A Well-Founded Software Process Behavior Ontology to Support Business Goals Monitoring in High Maturity Software Organizations. In: Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW). pp. 253-262. Vitória, Brazil (2010).
9. Barcellos, M. P., Falbo, R. A., Dalmoro, R.: A Well-Founded Software Measurement Ontology. In: 6th International Conference on Formal Ontology in Information Systems (FOIS

2010). Toronto - Canadá. Frontiers in Artificial Intelligence and Applications, 209, pp. 213-226. IOS Press, Amsterda (2010).
10. Barcellos, M. P., Falbo, R. A., Rocha, A. R.: Establishing a Well-Founded Conceptualization about Software Measurement in High Maturity Levels. In: Proceedings of the Seventh International Conference on the Quality of Information and Communications Technology (QUATIC). pp. 467-472. Porto, Portugal (2010).
11. Barcellos, M. P., Falbo, R. A., Rocha, A. R.: Using a Reference Domain Ontology for Developing a Software Measurement Strategy for High Maturity Organizations. In: Proceedings of the 16th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW). pp. 114-123. Beijing, China (2012).
12. Barcellos, M. P., Falbo, R. A., Rocha, A. R.: A strategy for preparing software organizations for statistical process control. Journal of the Brazilian Computer Society. v. 19, p. 1-31 (2013).
13. Barcellos, M. P., Falbo, R. A.: A software measurement task ontology. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC 2013). pp. 311-318. New York: ACM Press. Coimbra, Portugal (2013).
14. SEI, CMMI® for Development, Version 1.3, Pittsburg (2010).
15. Montoni M., Rocha A.R., Weber K.C.: MPS.BR: a successful program for software process improvement in Brazil. Software Process Improve Practice, 14, 289–300 (2009).
16. Calhau, R. F., Falbo, R.A.: An Ontology-Based Approach for Semantic Integration. In: Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference. pp. 111–120. Vitória, Brazil (2010).
17. Izza, S.: Integration of industrial information systems: from syntactic to semantic integration approaches. Enterprise Information System, 3, 1–57 (2009).
18. Fonseca, V. S., Barcellos, M. P., Falbo, R.A.: An ontology-based approach for integrating tools supporting the software measurement process. Science of Computer Programming, 135, 20-44 (2016).
19. Renault, L. D. C., Barcellos, M. P., Falbo, R.A.: Using an Ontology-based Approach for Integrating Applications to support Software Processes. In: Proceedings of the 17th Brazilian Symposium on Software Quality (SBQS). pp. 220-229. ACM International Proceedings Series. Curitiba, Brazil (2018).
20. Lethbridge, T. C., Singer, J., Forward, A.: How Software Engineers Use Documentation: The State of the Practice. IEEE Software, 20(6), 35 – 39 (2003).
21. Arantes, L., Falbo, R. A. An infrastructure for managing semantic documents. In: Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW). pp. 235-244. Vitória, Brazil (2010).
22. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American. 284 (5), 34–43 (2001).
23. Sicilia, M.: Metadata, semantics and ontology: providing meaning to information resources. International Journal of Metadata, Semantics and Ontologies, 1(1), 83 – 86 (2006).
24. Eriksson, H.: The semantic-document approach to combining documents and ontologies. International Journal of Human-Computer Studies, 65(7), 624-639 (2007).
25. Bastos, E.C., Barcellos, M. P., Falbo, R.A.: Using Semantic Documentation to Support Software Project Management. Journal on Data Semantics. vol. 7(2), 107–132 (2018).
26. Talas, J., Gregar, T., Pitner, T.: Semantic wiki in environmental project management. In: IFIP Advances in Information and Communication Technology. pp. 437-444. Brno, Czech Republic (2011).
27. Ruy, F. B., Falbo, R.A., Barcellos, M.P., Guizzardi, G.: An Ontological Analysis of the ISO/IEC 24744 Metamodel, In: 8th International Conference on Formal Ontology in

Information Systems (FOIS 2014). Rio de Janeiro, Brazil. Frontiers in Artificial Intelligence and Applications, 267, pp. 330-343. IOS Press, Amsterda (2014).
28. Ruy, F. B.: Software Engineering Standards Harmonization: An Ontology-Based Approach. Doctoral Thesis. Postgraduate Program in Computer Science. Federal University of Espírito Santo, Brazil (2017).
29. Ruy, F. B., Falbo, R.A., Barcellos, M.P., Costa, S.D., Guizzardi, G.: SEON: A Software Engineering Ontology Network. In: Blomqvist E., Ciancarini P., Poggi F., Vitali F. (eds) Knowledge Engineering and Knowledge Management. EKAW 2016. Lecture Notes in Computer Science, vol 10024, pp. 527-542. Springer, Cham (2016).
30. Ruy, F. B., Souza, E., Falbo, R.A., Barcellos, M.P.: Software Testing Processes in ISO Standards: How to Harmonize Them? In: Proceedings of the 16th Brazilian Symposium on Software Quality (SBQS). pp. 296-310. Rio de Janeiro, Brazil (2017).
31. Briguente, A. C. O., Falbo, R. A., Guizzardi, G.: Using a Foundational Ontology for Reengineering a Software Process Ontology. In: Proceedings of the 16h Brazilian Symposium on Data Base. São Paulo, Brazil (2012).
32. Peçanha, C.C., Duarte, B.B., Souza, V.E.S.: RASO: an Ontology on Requirements for the Development of Adaptive Systems. In: Proceedings of the 21st Workshop on Requirements Engineering (WER 2018). pp. 1–14. Rio de Janeiro, Brazil (2018).
33. Arantes, L.O., Falbo, R.A., Guizzardi, G.: Evolving a Software Configuration Ontology. In Proceedings of the Second Brazilian Workshop on Ontologies and Metamodels for Software and Data Engineering (WOMSDE07). João Pessoa, Brazil (2007).
34. Souza, E.F.D., Falbo, R.A., Vijaykumar, N.L.: ROoST: Reference Ontology on Software Testing. Applied Ontology,12, 59–90 (2017).
35. Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A.: Ontology Engineering in a Networked World. Springer Science & Business Media (2012).
36. d'Aquin, M., Gangemi, A.: Is there beauty in ontologies? Applied Ontology, 6 (3), 165–175 (2011).
37. Falbo, R.A., Ruy, F. B., Guizzardi, G., Barcellos, M.P., Almeida, J.P.A.: Towards an Enterprise Ontology Pattern Language. In: Proceedings of the 29th ACM Symposium on Applied Computting (ACM SAC). pp. 323-330. Gyeongju, Republic of Korea (2014).
38. Barcellos, M.P., Falbo, R.A., Frauches, V.G.V.: Towards a Measurement Ontology Pattern Language. In: Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering. CEUR Workshop Proceedings. Vol. 1301 (2014).
39. Falbo, R.A.: SABiO: Systematic Approach for Building Ontologies. In: Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering. CEUR Workshop Proceedings. Vol. 1301 (2014).
40. Villalon, M.P., Suárez-Figueroa, M. C., Gómez-Pérez, A.: Reusing ontology design patterns in a context ontology network. In: 2nd International Workshop on Ontology Patterns (WOP 2010). CEUR Workshop Proceedings, Vol. 671, pp. 35-49 (2010).
41. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: eXtreme Design with Content Ontology Design Patterns. In: Workshop on Ontology Patterns (WOP). CEUR Workshop Proceedings, Vol. 516, pp. 83-97 (2009).
42. Deutsch, P.: Models and Patterns. In: Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Greenfield, J., Short, K., Cook, S., Kent, S. (eds.), John Wiley & Sons (2004).
43. Buschmann, F., Henney, K., Schmidt, D.C.: Pattern-Oriented Software Architecture: On Patterns and Pattern Languages, John Wiley & Sons Ltd (2007).

44. Falbo, R.A., Barcellos, M.P., Ruy, F.B., Guizzardi, G., Guizzardi, R.S.S.: Ontology Pattern Languages. In: Gangemi, A., Hitzler, P., Janowicz, K., Krisnadhi, A., and Presutti, V., editors, Ontology Engineering with Ontology Design Patterns: Foundations and Applications. IOS Press (2016).
45. Quirino, G., Barcellos, M.P., Falbo, R.A.: OPL-ML: A Modeling Language for Representing Ontology Pattern Languages. In: de Cesare S., Frank U. (eds) Advances in Conceptual Modeling. ER 2017. Lecture Notes in Computer Science, vol 10651, pp. 187-201.Springer, Cham (2017).
46. Quirino, G., Barcellos, M.P., Falbo, R.A.: Visual Notations for Software Pattern Languages: a Mapping Study. In: 32nd Brazilian Symposium on Software Engineering (SBES). pp. 72-81. ACM International Proceedings Series. São Carlos, Brazil (2018).
47. Moody, D.L.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering. 35(6), 1–22 (2009).
48. Teixeira, M. G. S., Quirino, G., Gailly, F., Falbo, R. A., Guizzardi, G., Barcellos, M. P.: PoN-S: A Systematic Approach for Applying the Physics of Notation (PoN). In: Proceedings of the 21st International Conference in Exploring Modelling Methods for Systems Analysis and Design (EMMSAD 2016). pp. 432–447. Ljubljana, Slovenia (2016).

Applying a Collaboration Domain Ontology Pattern Language in Collaborative Editing (or: Collaborating, as we learned from Ricardo Falbo)

Renata Guizzardi[1][0000-0002-5804-5741], Maria Luiza M. Campos[2][0000-0002-7930-612X] and Fernanda Araujo Baião[3][0000-0001-7932-7134]

[1] Ontology and Conceptual Modeling Research Group (NEMO), Department of Computer Science, Federal University of Espírito Santo (UFES), Vitória, Brazil
[2] Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil
[3] Department of Industrial Engineering,
Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil
rguizzardi@inf.ufes.br, mluiza@ppgi.ufrj.br, fbaiao@puc-rio.br

Abstract. Ontology Engineering is a complex task, and Ricardo Falbo has intensively worked on several approaches and methodologies for supporting this task during his career, especially inspired by his expertise from the Software Engineering research area. In particular, Falbo contributed to the topic of Ontology Patterns, by clarifying terminology, proposing how to create them from Reference Ontologies, and creating an application method through what he called Ontology Pattern Language (OPL). In this work, we follow his path – as well as his example on cultivating fruitful and pleasant research collaborations – and propose the use of a catalogue of Collaboration Ontology Patterns (COP) and its corresponding Collaboration Ontology Pattern Language (C-OPL) in the collaborative editing domain. With this, we show how OPs and OPL may be used in practice to create domain ontologies in the domain of collaboration, which is pervasive and relevant for organizations, teams and groups of people in general.

Keywords: Collaboration, Ontology Design Pattern, Ontology Pattern Language.

1 Introduction

Ontologies have been part of mainstream Computer Science for at least two decades, being the core of many systems that rely on reasoning mechanisms to generate information, serving as an interlingua for Semantic Web applications and boosting the interoperability of data silos and systems.

One of the many things that Falbo's work has taught us is that a core aspect of applying ontologies relies on reuse, and here is why. An ontology refers to a domain conceptualization, i.e. a set of concepts and relations defining a given domain, shared by a certain community of users [1]. The main purpose of an ontology is thus to serve as a shared theory to be used to facilitate communication between humans and machines,

defining terminology and providing interoperability. Consequently, if ontologies are not based on reuse, we may achieve a situation in which each person and/or system has her own ontology. And this would make it very difficult to provide interoperability and shared meaning (ultimately, to collaborate).

Another reason for building ontologies on top of other ontologies, i.e. through reuse, is the fact that developing an ontology from scratch is quite a complex task. After all, choosing the "right" concepts and "right" relations to express a domain of discourse is a non-trivial task. How can one pick one word from two synonyms to represent a concept, for instance? How can one be sure that the community of users of such ontology will agree and understand the chosen concepts/relations?

Let us suppose that our arguments convince the reader that reuse is important. Right! But then what is the problem with that? The issue here is that reusing an artifact of any kind puts more pressure on guaranteeing the quality of such artifact. Otherwise, the new artifact built on top of it will share its defects, and we must avoid that.

Falbo and some of his many collaborators [2] proposed a new reuse approach regarding ontology-based conceptual models, namely developing ontologies through reuse of ontology patterns. In this paper, the authors propose a classification of different kinds of patterns, argue how they can be beneficial, and propose a method of patterns application. With some other collaborators [3,4], Falbo extended and clarified these notions.

The main idea in the aforementioned papers is that one is able to build ontologies (and other kinds of conceptual models) with better quality when reusing ontology patterns. But what are ontology patterns? They are excerpts extracted from previously built core ontologies. A core ontology is defined as a model providing a precise definition of knowledge in a specific field that spans across different application domains, e.g. organization, software process, service. Since these domains are quite abstract, it is possible to create ontologies of more specific domains by specializing core ontologies. However, perhaps the new domain ontology does not actually need all concepts of the previously existing core ontology. Hence, dividing such core ontologies in smaller chunks makes sense. These smaller chunks are termed DROPs (Domain Related Ontology Patterns).

The beauty of developing ontologies based on DROPs is that this process not only contributes to the quality of the ontology-to-be, but also leads to a more efficient process in terms of development time. This is due to the fact that the development is based on knowledge already codified in the patterns, which makes the work of the ontology engineer faster and more objective.

This chapter addresses ontology development with reuse, focusing on the collaboration domain. Collaboration is the heart of many human activities (and Falbo has performed it exemplarily during his career). Currently, in any social context, including the business world and academia, it is hard to think of a work that one can accomplish solo. As products and services become more and more complex, caring for the effectiveness of communication, cooperation and coordination processes becomes essential. Moreover, managing such complexities is a problem on its own, and collaboration is the key to provide a smooth approach to such activity.

In [6,7], a Collaboration Core Ontology named CONTO was proposed. Since then, this ontology has been applied in a multitude of works, including in [8]. Based on CONTO, we created a catalogue of collaboration patterns, with the aim of facilitating even more CONTO's reuse. In this process, we of course ended up revising CONTO, adding and or updating concepts/relations to account for more precise definitions that we found useful.

Falbo and colleagues not only proposed the use of ontology patterns, but went much beyond, proposing the use of Ontology Pattern Languages (OPLs)[3,4,5], which may be defined as processes for guiding the use of ontology patterns. According to an OPL, a particular pattern catalogue is organized in an adequate application order, thus guiding the ontology engineer on how to pick and choose between such patterns, and most of all, on how to connect the chosen patterns. In this chapter, besides our proposed Collaboration Pattern Catalogue (COP catalogue), we also present the Collaboration Ontology Pattern Language (C-OPL).

Aiming at illustrating the use of the patterns catalogue and C-OPL, we here present a case of ontology development in the domain of collaborative editing, i.e. a collaboration subdomain. In other words, as expected, the CONTO core ontology [7] (through the patterns that were extracted from it) may be specialized in more specific domains, as the one of collaborative editing that we describe [2].

Last but not least, this chapter also shows the use of the existing OntoUML lightweight editor (OLED)[9] as an automated support for the development of ontologies based on catalogues and OPLs. Throughout his career, Falbo has emphasized the importance of having an engineering view of the Ontology field. Thus, we must admit that even this last result has been inspired in Falbo's approach.

All three contributions of this chapter (the catalogue of collaboration ontology patterns, C-OPL and the computational tool to support ontology development with patterns) represent important advances for the area of collaboration and groupware. Collaboration ontologies may serve, for instance, as the conceptual model of groupware (e.g. for a second version of the Cobrowser proposed at [10]), a new version of the Knowledge-intensive Process Ontology (KIPO)[8], a conceptual model to support for emergency response planning [11], support meaning negotiation about collaboration and their subprocesses [12], among others.

This chapter is organized as follows: Section 2 presents some background information regarding collaboration; Section 3 describes in detail Falbo's approach towards OPs and OPLs, already mentioned in this Introduction; Section 4 presents the research work we developed with basis on Falbo's approach, i.e. the COP catalogue and C-OPL, and points to OLED as an automated support tool; Section 5 illustrates the use of the patterns in the catalogue, by following paths established by C-OPL; Section 6 presents some final remarks; and finally Section 7 concludes this chapter with some personal notes from the authors.

2 Collaboration and its Supporting Processes

In general, one of the most critical processes in the everyday life of organizations is collaboration. Be it e-mail, instant messaging, virtual workspaces, videoconferencing, collaborative text edition, shared white boards or case tools, technology dramatically shortens distances among people and frees up the flow of intellectual capital, enabling organizational members to work, to capture and share knowledge more quickly [6]. Consequently, revenues, operating margins and productivity grow [13], while organizational members gradually learn how to make the most of team collaboration [14].

A ***Collaborative Session (CS)*** is a type of event that (loosely speaking) represents a period of time in which some agents collaborate with each other for a given purpose. There are some important concepts related to a CS, such as: *collaborators*, *goals*, *commitments*, *resources* (as in information and physical artifacts), *coordination* and *communication*. *Collaborators* are the agents, such as people, organizations, or organizational units that can contribute in a meaningful way to achieve the sessions' goals. *Goals* are the states of affairs the participants aim at achieving in the end of this particular CS. *Cooperation* depends on the *commitments* of each collaborator toward the CS. Such *commitments* are regulated by the *collaborative session agreement*, which range from very informal to being formalized in kinds of contracts (which may be termed *collaborative agreement description*).

According to Nguyen et al. [15], the purpose of collaboration artifacts (i.e. *resources*) is to serve as a bridge that connects agents and software. This means that a collaboration artifact provides a shared workspace for the *collaborators*. This creates a kind of record that can help to rebuild the experiences of the CS, which most of times is only "recorded" in the agent's mind. In collaborative work, *cooperation* is a joint effort in a shared space to achieve some *goal*. To avoid implicit rules and uncontrolled behaviors in a CS, participants should know the rules and procedures defined for the CS. Hence, these rules and procedures must be somehow explicit, defined in a way that makes each participant comfortable to use and contribute to the CS in a meaningful way. This motivates the need for a formal *coordination* protocol. *Coordination* is defined in the Merriam-Webster Online Dictionary as the harmonious functioning of parts for effective results. In other words, *coordination* can be described as a layer that mediates *communication* and *cooperation*, to enforce the success of collaboration. *Coordination* services support management and enforcement of group activities [12]. After all, how can people collaborate without *coordination*? Without it, people will usually engage in conflicts or repetitive actions. *Coordination* is hence essential to solve problems that share common objectives, resources and activities [16]. In fact, we claim that *coordination* is one of the most important prerequisites to accomplish the CS *goals*. For an organization, *coordination* is aimed at guaranteeing that a project scope is obtained without exceeding limits of time and cost, while maintaining quality.

Another point worth mentioning is the relationship between collaboration and *communication* [12], which is the basis to any collaborative system. The objective of *communication* is to exchange knowledge among individuals. To transmit content, the *sender* expresses his *intentions* or *goals*, defined by symbols in a *language* that must

be understood by all *receivers*. Moreover, information transmission needs to be accomplished by a *communication media* [12]. A *communicative act* involves *interaction* among individuals, (agents that can be *messages'* senders and receivers), a *dialog event* (an event characterized by the exchange information through messages), a *context* (a situation in which the communication occurs) and a *protocol* (a set of rules that coordinate the communication). Moreover, a *communicative act* generates *commitments* that create new *goals* [17]. And to achieve these *goals*, through a set of individual actions, *coordination* is again needed to carry out the collaborators' *commitments* towards these *goals*.

To sum up, it is difficult to think of *cooperation* without *communication* and *coordination*, *communication* without *coordination* and *coordination* without *communication*. And collaboration is actually regulated by these three processes. This brings us to the 3C framework [12], illustrated in Figure 1.

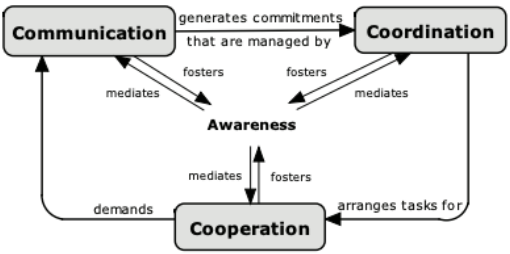

Fig. 1. The 3C Model [12].

3 Acknowledging the importance of Falbo's approach on Ontology Engineering, Ontology Patterns and Domain Ontology Pattern Languages

Ontology Engineering is a complex task. It is for no other reason that Falbo has intensively worked on several approaches and methodologies for supporting this task during his career, especially inspired by his expertise from the Software Engineering research area. In particular, he approached Ontology Engineering as an analogy to Software Engineering, in which a model of the world is built for analysis (i.e., without any implementation concerns such as tractability or efficiency) and further design of a computational system. From the perspective of Ontology Engineering, this conceptual model is later (in a design phase) refined to become an ontology schema after some adaptations required by a specific ontology modeling language/formalism.

Similar to successful Software Engineering approaches and good practices, reuse has been largely advocated for ontologies to be built based on pre-existing models, speeding up the development process and leading to better quality results. In this sense, Ontology Patterns have been appointed as interesting tools to enable reuse.

An Ontology Pattern (OP) describes a particular recurring modeling problem that arises in specific ontology development contexts and presents a well-proven solution for the problem. This definition is actually based on a very analogous one proposed by in the Software Engineering domain. This makes sense, because in Ontology Engineering, OPs play the same role as the ones in Software Engineering.

Using OPs is an emerging approach that favors the reuse of encoded experiences and good practices. Different kinds of OPs support ontology engineers on distinct phases of the ontology development process [2]. Therefore, OPs are specialized into four subtypes: ontology conceptual pattern; ontology architectural pattern; ontology design pattern; and ontology idiom. An ontology conceptual pattern is a self-contained fragment of either a foundational ontology (Foundational OP, or FOP) or a domain ontology (Domain-related OPs, or DROP). An ontology architectural pattern describes how to arrange an ontology (generally a large one) in terms of sub-ontologies or modules, and how to deal with the modular architecture of an ontology network (where the composing ontologies play the role of modules). An ontology design pattern addresses the problems that occur during the ontology design phase. Finally, the ontology idiom describes how to solve problems related to reasoning or expressivity of a specific logical formalism. An ontology idiom, in turn, supports ontology implementation, aiming at solving problems related to reasoning or related to shortcomings in the expressivity of a specific logical formalism.

In [18], Ruy, Guizzardi, Falbo and colleagues explain how to extract DROPs from reference ontologies. In particular, a DROP may assist in building the conceptual model of a new domain ontology. This chapter addresses the application of DROPs in the Collaboration domain. Since DROPs are reusable fragments extracted from reference ontologies that should capture the core knowledge related to a specific domain, they may be seen as a fragment of a core ontology of that domain [3].

Every OP, and DROPs in particular, may be specified using a language to formalize and organize DROPs in a guided application process by explicitly representing the relations between them. In the Software Engineering domain, a pattern language refers to a network of interrelated patterns that defines a process for systematically solving coarse-grained software development problems [19,20]. This approach can also be taken into account in Ontology Engineering, giving rise to Ontology Pattern Languages (OPLs).

An Ontology Pattern Language (OPL) is a network of interconnected DROPs that provides holistic support for solving ontology development problems for a specific domain. An OPL contains a set of interconnected DROPs, plus a modeling workflow guiding on how to use and combine them in a specific order, and suggesting patterns for solving some modeling problems in that domain [3,4].

The notion of OPL provides a stronger sense of connection between DROPs, expressing different types of relationships among them [3]. For instance, to be applied, a pattern may require the previous application of other patterns (dependency); a larger pattern can be composed of smaller ones; or several patterns may solve the same problem in different ways (variant patterns). Those relationships impose constraints in the order in which patterns can be applied. Thus, an OPL provides explicit guidance on how to reuse and integrate related patterns into a concrete conceptual model of a new

domain ontology. In this sense, an OPL is more than a catalogue of patterns. It includes, besides the patterns themselves, a rich description of their relationships, and a process guiding the order to apply them according to the problems to be modeled. OPLs encourage the application of one pattern at a time, following the order prescribed by paths chosen throughout the language. Consequently, by enabling the selective use of DROPs in a flexible way, an OPL releases the ontology engineer from the need to know upfront the whole set of concepts and relations addressed by the OPL. In this way, an OPL also aids managing complexity.

OPLs are among the most important contributions from Falbo's research, applied to several domains. For example, the Measurement Ontology Pattern Language (M-OPL) addresses the core conceptualization about measurement [21]. The Service Ontology Pattern Language (S-OPL) provides a network of interconnected ontology modeling patterns covering the core conceptualization of services [4] and it has been applied in a real case study to model an email service in a big Italian company. The Enterprise Ontology Pattern Language (E-OPL) [5] organizes aspects common to several enterprises and it has been used for building an enterprise ontology on Governmental Brazilian Universities. Also, an ontology pattern language for software process domain (SP-OPL) was used for building a domain ontology about the software measurement process [2].

In the present chapter, we follow on this successful path of research development and illustrate the use of a catalogue of collaboration ontology patterns (COP) and its corresponding Collaboration Ontology Pattern Language, C-OPL. Both COP and C-OPL are described in Section 4.

4 Collaboration Ontology Pattern Language - C-OPL

This section is the core of this chapter, focusing on the proposed Collaboration Ontology Pattern Language (C-OPL). Our idea here is to systematically build knowledge until the culmination of the section where we present C-OPL. Thus, we start in Section 4.1 with the presentation of CONTO, the ontology from which we derived the collaboration patterns; then, in Section 4.2., we move to the illustration of the COP catalogue; and finally, in Section 4.3, we present C-OPL.

4.1 CONTO - An Ontology for the Collaboration Domain

The Collaboration Ontology, CONTO for short [7], follows the 3C Collaboration Model, presented in section 2, thus being subdivided in three ontologies: *Cooperation, Communication* and *Coordination*.

Table 1 presents CONTO's original concept dictionary, defining the concepts modeled in each of the sub-ontologies. In our work of developing the collaboration patterns, we evolved CONTO with extra concepts, which we do not show here, but describe them in connection to the patterns presented in Section 4.2.

Table 1. The concepts modeled in the original version of CONTO [22]

COOPERATION	Agent	Physical Objects that are the collaboration actors.
	Collaborative Role	Social Role that agents play during collaboration
	Collaborative Session	It is an Interaction, i.e. a Complex Action in which at least two Agents are involved for the purpose of collaborating.
	Site	Location where the collaborative session happens. It may be Virtual or Real.
	Collaborative Agreement	Social relator that defines a Collaborative Session.
	Action Contribution	Action which constitute intentional participations performed by the Agents involved in the Collaborative Session.
	Close Commitment	Social Moment established by a Collaborative Role that commits to achieve the Goals of the Collaborative Session, by performing specific Action Universals.
COMUNICATION	Material Contribution	An Action Contribution that physically change the world.
	Communication Act	An Action Contribution that carries out the information that should be exchanged in a session.
	Perception	An Action Contribution that consists in the reception of the information communicated through the Communicative Acts.
	Message	The propositional content of a Communicative Act.
	Idiomatic Language	Language using an idiom for its representation.
	Sender	Agent who sends the message.
	Receiver	Agent who perceives the message.
	Communicative Interaction	Complex Action composed of exactly one Communicative Act and one or more Perceptions.
COORDINATION	Protocol	Normative Description defining the rules that govern the Collaboration Group and, thus, being recognized by this group. The Protocol defines the Collaborative Roles (e.g. leader, participant, etc.) and the Action Contribution Universals presente in the Collaborative Session.
	Collaborative Group	Social Agent composed of the Physical Agents that participate in the Collaborative Session.
	Exclusive Resource	Resource that cannot be used simultaneously.
	Sharable Resource	Resource that can be used simultaneously.

4.2 Collaboration Ontology Patterns Catalogue

Table 2 presents part of the Collaboration Ontology Patterns Catalogue (COP Catalogue).

The full specification of each OP is composed of this extra information:

- variant patterns, indicating which other patterns (if any) may be used in place of the specified pattern, typically to allow the definition of two similar patterns, a simple and a more complex one. From the patterns in this table, only CSR and CSRS (in lines 9 and 10, respectively) are variant to one another.
- intent, expressing the objective of the pattern.
- rationale, explaining the reason behind the definition of the pattern.
- axioms, formalizing the pattern.

Table 2. Ontology Patterns that compose the COP Catalogue

Acronym	Name	Competency question
1) CA	Collaborative Agreement	What is the collaborative agreement that rules the collaborative session?
2) CACC	Collaborative Agreement Commitment Composition	Which commitments are part of an agreement in a collaborative session?
3) CCIC	Collaborative Commitments Inheres in Collaborator	What are the collaborative commitments that inhere in the collaborator?
4) CSCo	Collaborative Session Collaborator	What are the agents that participate in the collaborative session?
5) CCAC	Collaborative Commitment for Action Contribution	What is the collaborative commitments that causes the action contribution?
6) CSAC	Collaborative Session Action Contributions	What are the action contributions that compose the collaborative session?
7)ACPCo	Action Contribution Performed by Collaborator	Which collaborator performed the action contribution?
8)ACPPS	Action Contribution Pre and Post Situations	What are the action contribution's pre-situations? What are the action contribution's post-situations?
9) CSR	Collaborative Session Resource	What are the resources that participate on the collaborative session (with the resource participation event represented)?
10)CSRS	Collaborative Session Resource Simple	What are the resources that participate on the collaborative session?
11)RT	Resource Type	What is the type of a resource?
12)RC	Resource Composition	Can this resource be decomposed? What are the components of a resource?
13)RN	Resource Nature	Can a given resource be shared?
14) CSL	Collaborative Session Location	Where does the collaborative session happen?
15) LT	Location Type	What is the location type?
16) CAD	Collaborative Agreement Description	What is the description of the collaborative agreement kind?
17)CSAD	Collaborative Session Agreement Description	What is the collaborative agreement description used in the collaborative session?

18) CAC	Collaborative Agreement Composition	Can this agreement be decomposed? What are the components of this agreement?
19)ACPD	Action Contribution Plan Description	What is the plan description associated to the action contribution?
20)CCDR	Collaborative Commitment Defined Role	Which is the role associated to a commitment?
21)CRRPA	Collaborative Role Right-to-Perform Action	Which are the permitted actions for a role?
22)CoPCR	Collaborator Plays Collaborative Role	Which is the role played by a collaborator?

For reasons of space, we show here only a simplified specification of 22 of the patterns in the COP Catalogue[1].

To illustrate a full pattern description, Figure 2 shows the specification of one of the patterns of Table 2, namely the RN ontology pattern.

RN – Resource Nature

Name: Resource Nature (RN)

Variant Patterns: --

Intent: To represent a resource in terms of shareability.

Rationale: A resource in a collaborative session can be either shareable or exclusive

Competency Questions:

Can this resource be shared in this collaborative session?

Axioms:

A14 - $Exclusive\ Resource\ (r) \rightarrow \forall p, p'\ participation\_of\ (p,r) \wedge participation\_of\ (p',r) \wedge (p \neq p') \rightarrow \neg(overlaps\ (ti\ (p), ti\ (p'))$

A15 – $Sharable\ Resource\ (r) \rightarrow \neg Exclusive\ Resource\ (r)$

Fig. 2. The full specification of the RN pattern

The conceptual model of each pattern is undoubtedly one of the most important pieces of information, since they function as building blocks for the creation of new ontologies. Figure 3 and 4 shows the conceptual models of the first 13 patterns described in Table 2, as these were used in our application example. To be perfectly clear, one of such patterns (namely CSRS, in line 10) was not actually used in the application

[1]For the full specification of all OPs in the COP Catalogue, we refer the reader to the repository available at https://github.com/nemo-ufes/ontouml-lightweight-editor/ (search for document entitled "Collaboration Patterns Catalogue and C-OPL Specification.pdf")

case, because it is a variant pattern w.r.t pattern CSR, in line 9. However, for reasons of transparency, we decided to include also CSRS's conceptual model. The correspondence between the patterns and their respective conceptual models is obtained by the numbers identifying each model and each line of the table. The other patterns were just listed for comprehension purpose, as they are mentioned in the C-OPL process represented in Figure 5.

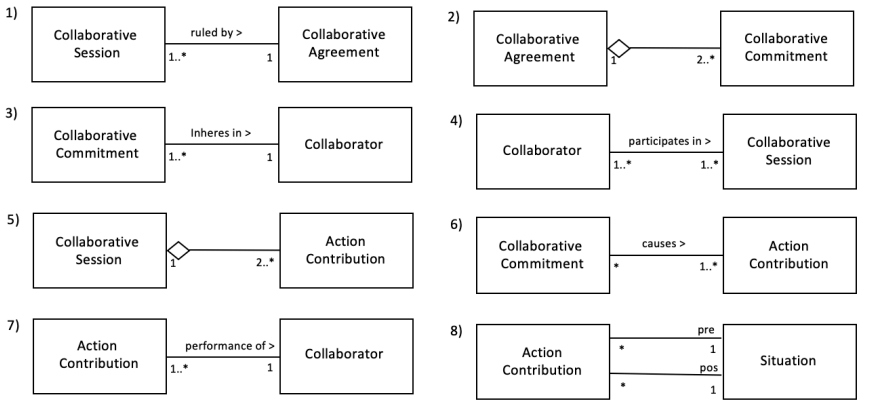

Fig. 3. Conceptual models of the presented ontology patterns (part I)

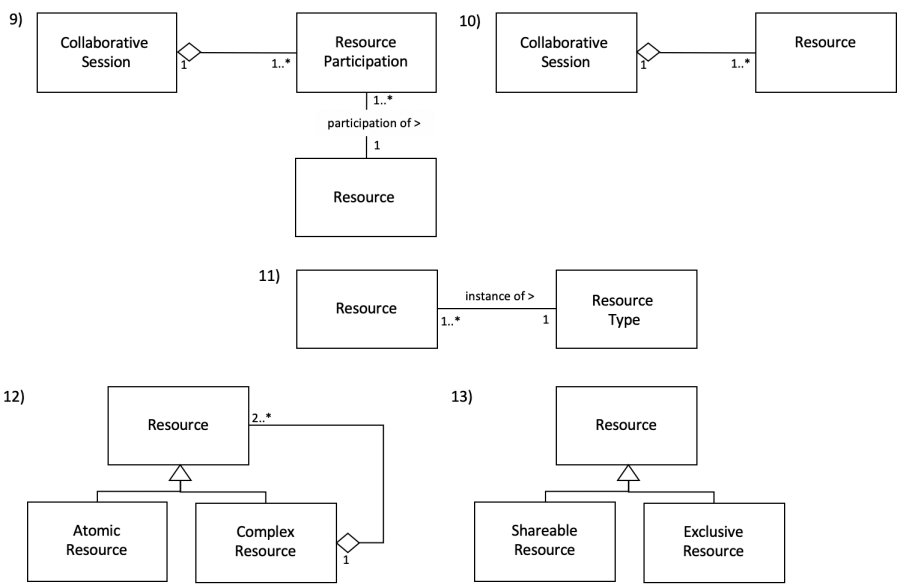

Fig. 4. Conceptual models of the presented ontology patterns (part II)

4.3 Collaboration Ontology Patterns Use Process

In this section, we present an excerpt of the C-OPL. We do not consider productive to present the whole pattern language in this chapter[2].

Besides the set of patterns, C-OPL is complemented by a process intended to guide ontology engineers in using the collaboration DROPs created from CONTO. Figure 5 shows the process model of C-OPL using the OPL-ML specification language. proposed by Quirino, Barcellos and Falbo [23]. According to this language, (i) DROPs are represented by action nodes (labeled rounded rectangles); (ii) initial nodes (solid circles) represent entry points in the OPL, that is, language DROPs that can be used without solving other problems first; (iii) final nodes (solid circle with external halo) represent output points in the OPL, that is, DROPs that can be used to terminate language paths; (iv) control flows (lines with arrows) represent allowable sequences in which DROPs may be used; (v) decision points (diamond shaped symbols) represent a decision to be taken in a path, depending on some conceptual aspect to be evaluated as relevant or not to the specific target domain; (vi) junction / division nodes (line segments) represent the conjunction of paths (junction) or independent and parallel paths (division); and (vii) variant patterns are represented by an extension of the original UML notation (dotted lines with arrows[3]), that is, patterns that can be used to solve the same problem in different ways.

The C-OPL process model presented in this article is only a partial outline of the full process, focusing on the *cooperation* component of CONTO, and not contemplating the use of other DROPs associated to *coordination* and *communication*, nor to some complementary cooperation details, because of space limitations. Needless to say, all patterns composing the example of section 5 are present, along with other relevant ones.

We begin using an OPL by focusing on the entry points present in the language process. Here we discuss three entry points of C-OPL related to *cooperation*. The ontology engineer must choose one of these entry points, depending on the collaboration ontology being developed. EP1 should be chosen when the requirements of the new ontology include defining where the collaboration session takes place. EP2 should be chosen when there is a need to discuss collaboration regardless of where it occurs. but when agreements and commitments might be relevant for a particular ontology. EP3 is the option when interest is not in collaboration commitments and agreements or session location, but when the collaborative session might be defined through the collaborators, their roles, action contributions and associated resource participations. Although not discussed here, other entry points should be chosen when requirements focus on coordination or if they contemplate communication aspects.

The cooperation part of C-OPL is organized according to 4 blocks where the collaborative session and collaborators are the central concepts (see CSCo pattern). These

[2]Again we suggest the reader to access the repository available at https://github.com/nemo-ufes/ontouml-lightweight-editor/ (search for document entitled “Collaboration Patterns Catalogue and C-OPL Specification.pdf”), which also contains the C-OPL specification.

[3]Due to a limitation of the tool we used to design C-OPL, we present the variant patterns using a red double-ended arrow

blocks were defined based on their relevance for the domain and implicit (loose) dependences: (1) details of the location where the collaboration takes place; (2) aspects related to the agreement and associated commitments assumed by the collaborators and their corresponding roles, when these are defined; (3) collaborators, their action contributions and their association to roles and action types (when required); and (4) resources details, including their types, composition and nature, resources participations and their association to action contributions.

The C-OPL process is highly influenced by the flexibility requirement, as collaboration scenarios and applications are extremely diverse. For this reason, C-OPL has only a few mandatory elements, i.e. OPs that are reached no matter what path you take. One particular example of mandatory element is the CSCo pattern, which corresponds to the EP3 entry point, where one starts if *location*, *agreements* and *commitments* were not considered relevant for a specific domain.

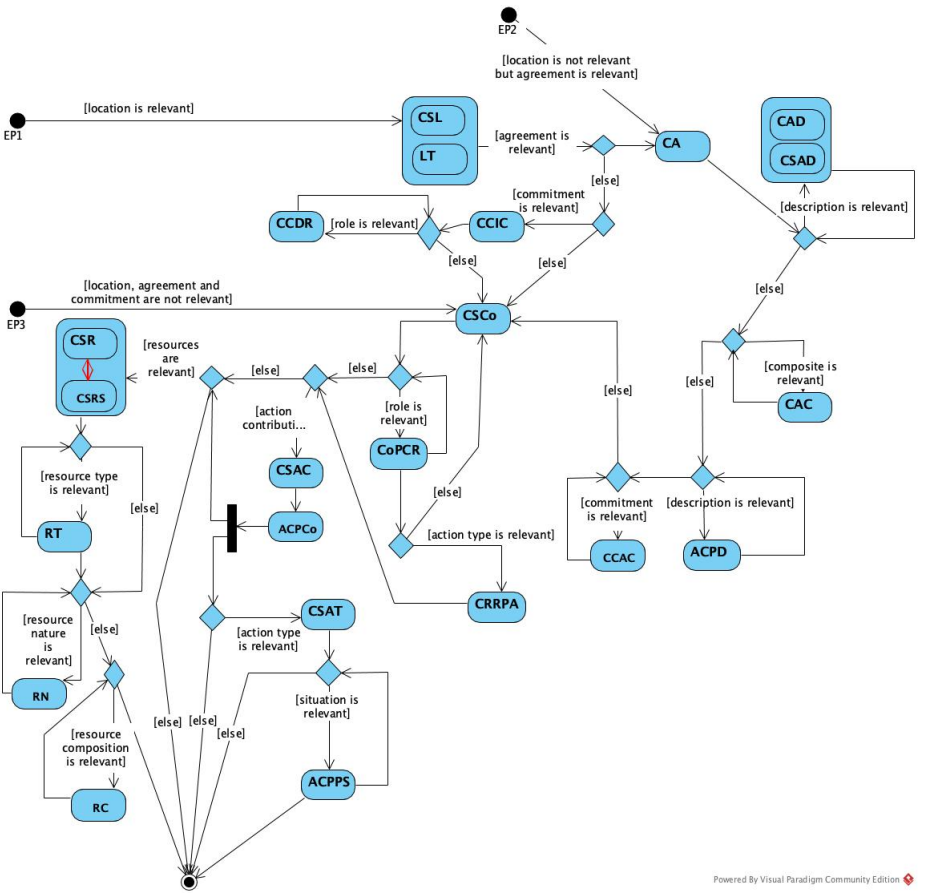

Fig. 5. Partial C-OPL process related to cooperation aspects

This means that by starting from EP3, one begins by representing the *collaborators* that participate in the collaborative session and proceeds to evaluate the use of other optional aspects such as *action contributions*, *role attribution* and *resource participation*.

4.4 Supporting tool

OLED (OntoUML lightweight editor) is a model-based environment to support Ontology Engineering using OntoUML, in particular the tasks of formalization, verification, validation and implementation [9]. OLED was designed to aggregate all the aforementioned technologies developed for OntoUML in the long-term research project conducted by the Ontology and Conceptual Modeling Research Group (NEMO)[4]. In order to support the management and use of DROPs, OLED was extended to support the creation of pattern catalogues, allowing OPs edition and reuse, and, in particular, the Collaboration Ontology Patterns Catalogue (COP Catalogue) [22]. Figure 6 shows the COP Catalogue, with available OPs listed on the toolbox located on the left side of the OLED interface.

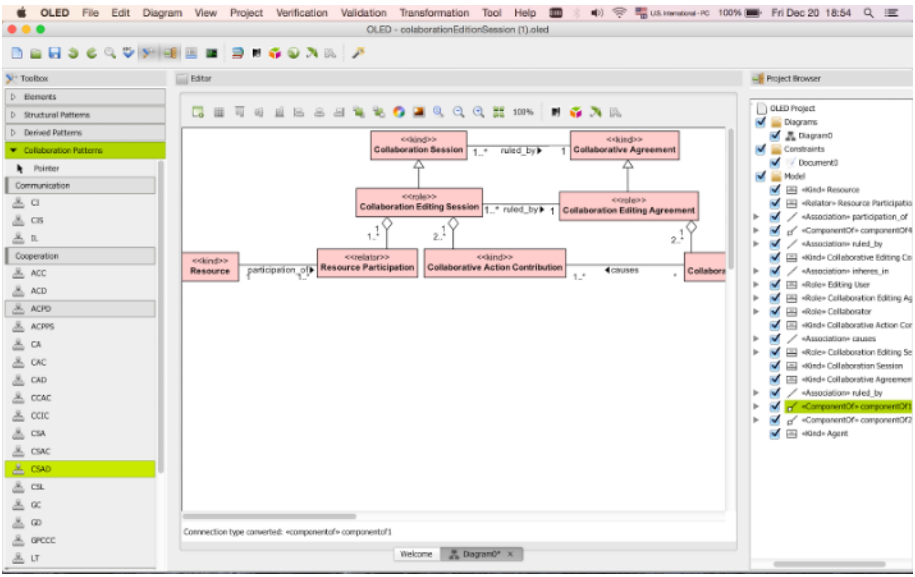

Fig. 6. OLED interface with the Collaboration Ontology Patterns Catalogue

We emphasize that the COP Catalogue is only an example connected to the content of this chapter, since with the developed extension, one may define catalogues of patterns in any domain. For knowing more on how to develop pattern catalogues using OLED, please see [22,24].

[4]For more information on the project and to download the editor, see https://nemo.inf.ufes.br/projects/oled/

5 Applying C-OPL in the domain of Collaborative Editing

The use of C-OPL will be exemplified through a simple and very common collaboration scenario, but still rich enough to show how the OPL process guides the ontology engineer on the use of the C-OPL patterns.

Collaborative software tools and applications allow people to interact, cooperate and share knowledge, working together on a common virtual space in a simple and efficient way, largely increasing productivity. Specifically, we consider features available on collaborative editing systems, like Google Docs, where users can work, either synchronously or asynchronously, over the same resource, such as a document or a spreadsheet. We concentrated on the main collaboration functionalities to build the collaborative editing sub-ontology, basically including the *actions* that the users perform, engaged on *roles* and *commitments* they agree upon to execute editing actions to evolve a particular *resource*. Hence, we do not address *communication* and *coordination* features, such as awareness, although many of them have already been addressed by C-OPL.

Considering this scope, a small set of competency questions (CQs) were defined, and we have identified the patterns that could help answering those questions. In many Ontology Engineer methodologies, one starts from the CQs and thus, the relation to the patterns comes only later. In fact, with the use of a catalogue of patterns, after defining the first version of the CQs, one may use the CQs specified for each pattern to guide creation/update of CQs for the domain of the ontology-to-be. For reasons of space, Table 4 presents only the final versions of the CQs, along with the patterns that may be applied to respond each of them.

Table 3. Competency questions of our scenario and the OPs that are able to respond them

Competency Question	**OP**
CQ01. What kinds of agreements are there in a collaborative editing session?	CA
CQ02. Which commitments are part of an agreement in a collaborative editing session?	CACC
CQ03. What are the collaborative commitments that inhere in the editing user?	CCIC
CQ04. Who is involved in a collaborative editing session?	CCIC
CQ05. What are the collaborative commitments that cause actions?	CCAC
CQ06. Which action are performed on a collaborative editing session?	CSAC
CQ07 Which are the actions played by an editing user?	ACPCo
CQ08a What are the versions of a document?	ACPPS
CQ08b. What changes were made in a resource comparing to its versions?	ACPPS
CQ09. Which resources participate in a cooperative editing session?	CSR
CQ10. What kinds of resources are the target of a collaborative editing session?	CSRT
CQ11. Are there resources composed of other resources?	CSRC
CQ12. Can this resource be shared in this collaborative editing session	CSRN

In what follows we have applied C-OPL to generate a model aligned with these competency questions.

In our application case, EP2 was the chosen entry point since location was not relevant at this moment for our initial goal. Hence, CA (Collaborative Agreement) was the

first pattern applied, as we needed to represent the Collaborative Editing Agreement among Editing Users. As commitments were relevant as well, the associated Collaborative Editing Commitments were defined (CACC – Collaborative Agreement Commitment Composition) and their associated Editing Users (CCIC – Collaborative Commitment Inheres in Collaborator). As action contribution is relevant, we used CCAC (Collaborative Commitment for Action Contribution). To simplify, as roles were not considered relevant, we followed straight to apply CSCo (Collaboration Session Collaborator) pattern and, as we needed to represent Action Contributions for the session, CSAC (Collaboration Session Action Contribution) was used. Then, as possible parallel tracks, we have one block of activities for complementing the representation of action contributions and another block to represent the activities associated to resources. We applied ACPCo (Action Contribution Performed by Collaborator) pattern to represent which actions were performed by each Editing User. In the sequel, following the path, we have chosen ACPPS (Action Contribution Pre and Post Situation), representing the changes made to a resource, as each action contribution makes the resource go from one state (pre situation) to another state (post situation). This modeling construct supports the important versioning feature of Resource (represented by the version-of relation in Resource). For modeling the Resources Participation and their respective Resources which are the targets of the Collaborative Editing Session, we followed the path to apply CSR (Collaborative Session Resource) pattern. Then, to represent different perspectives of the resources, three patterns were applied: (i) RT (Resource Type); (ii) RC (Resource Composition); and (iii) RN (Resource Nature). With these, we could represent, for example, two types of resources according to RT: Document and Folder, where the Document is a part-of the Folder (an Atomic and a Complex Resource, respectively, using RC. Besides, according to RN, they should be defined as a Shared Resource, so that the Editing Users could work on them in a Collaborative Editing Session. Finally, the copy-of and version-of relationships, specifically identified for this application domain, were added to Resource. The determined-by relationship between Resource Participation and Action Contribution was also added. The resulting model is presented in Figure 7.

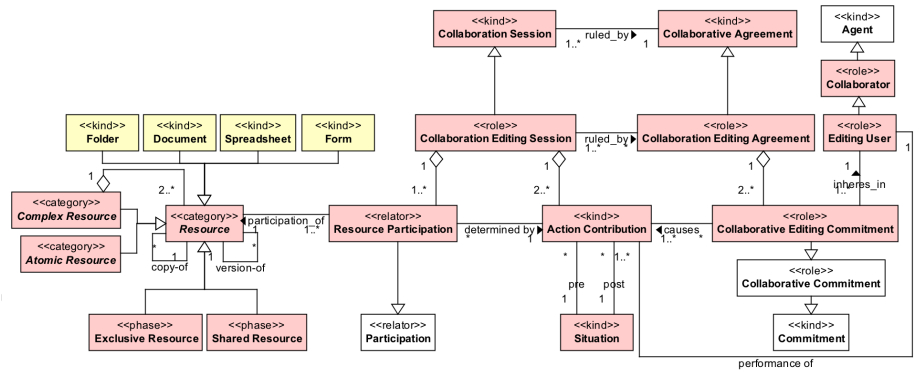

Fig. 7. Fragment of the Collaborative Editing sub-ontology

In Fig. 7, we applied the following color code: white for UFO foundational concepts; rose for OP concepts; and yellow for domain concepts. Note that when developing the model, some fragments were a direct use of C-OPL patterns. Others had to be specialized in the domain from existing ones. In fact, only four concepts needed to be added to complete the Collaborative Editing domain ontology. And, as expected, some relationships (namely, **copy-of** and **version-of**) were also specific of this domain and were added to the model in the end. Moreover, an important information here regards the reuse approach. Falbo and colleagues [2] define two approaches for reusing ontology patterns: by analogy and by extension. To build the ontology shown in Fig.7, we applied the latter.

6 Final Remarks

In the past 30 years, Falbo has been a mentor and an inspirer of so many researchers and practitioners within the areas of Software Engineering and Ontology Engineering.

In this chapter, we described our work on the COP catalogue and on C-OPL, both aiming at providing support for the development of higher quality domain ontologies in the Collaboration field. This work without a doubt would not have been possible without the support of Falbo's approach regarding the use of ontology. He and his colleagues have set a solid basis in which our work could be framed, and hopefully many other works will follow these same footsteps.

We highlight that the Collaboration field is a very important domain, crosscut to other areas. Thus, this seemed to us as an ideal scenario to illustrate the application of OPs and OPLs as we learned from Falbo. Not only this may assist ontology engineers on producing domain ontologies about collaboration and its underlying processes. Our work may also assist in the development of other areas that, for one reason or another, rely on collaboration and may, thus, build on top of the defined patterns.

Our future work is still vast. We aim at defining, for instance, priorities among the paths of C-OPL, distinguishing between mandatory and optional paths, as well as addressing *coordination* and *communication*, not presented here. It is also in our research agenda to improve the language to describe OPLs. Last but not least, we hope to apply the COP ***catalogue*** and C-OPL in many application cases, improving these artefacts along the way; and possibly evaluate them thoroughly through a real case study.

It is hard to find the right words to thank Falbo for all his hard work and encouragement. He inspires us through the excellency of ***the content of his work***, as well as through the ***processes he chose to conduct it***, and ***the way he did it***. We may thus make an analogy here, saying that we have been using ***Falbo's patterns***, following ***Falbo's OPL***, and turning ***collaborators into friends***.

7 Personal Notes

Renata: I met Ricardo when he was coming back to UFES after his PhD and I was finishing my undergraduate studies. I remember that when he first arrived, he gave us

a tutorial about Software Analysis and Design. And in the first remark he made, he summarized in such a brilliant way the distinctions between software analysis and software design that I thought *"Ok, I've just had two semesters to learn about analysis and design but, after this sentence, I feel that I learned more"*. He then became my professor when I joined the PPGI/UFES master's program, and he has been one of the best and most organized professors that I have ever had. What I also admire about him is his determination to do high quality work in a non-stressful way, combining the long weeks of work with weekends of joy in the beach with his beautiful wife, Jo, and his great son, Pedro. Every week, we (students) would note that he arrived in class with a nice tan, and usually wearing white shirts with touristic motives and comfortable tennis shoes. I felt very honored when I finally finished my PhD and I became his colleague at UFES, and we started collaborating more closely on our research. Teaching or doing research work together, Ricardo has always been very serene, smart and friendly. Anyway, I could keep talking more and more about Ricardo's qualities, but if I would have to pick one feature, then I would say that Ricardo is ultimately a very kindhearted person. And most of all, I am proud to call him my friend.

Maria Luiza: I've learned to call him Falbo. Although it is his surname, it always sounded to me like his nickname, as he is just the sort of people you become very close as soon as you meet. I have not been fortunate enough to directly work with him, but I consider it already a privilege when we occasionally interact in seminars and events. From the moment we met, I soon became an admirer not only of his research achievements but, more importantly, of a very special, kind and generous human being.

Fernanda: I was a second year PhD student when Ricardo got his PhD. While at COPPE Labs, in Rio de Janeiro, I remember his tenderness with everyone surrounding him and his everlasting smile. We did not share the same research interests at that time (I started my career as a fan of Databases), and it was a great pleasure to me that our research interests intersected when I evolved from Conceptual Data Modeling to Ontology and Ontology Engineering. Since then, I started to collaborate with him and all his colleagues from NEMO/UFES, and it has been an honor to share some deep, fruitful and extremely pleasant discussions with them. I am your fan, Ricardo.

References

1. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, PhD Thesis, University of Twente, The Netherlands. (2005)
2. Falbo, R. A., Guizzardi, G., Gangemi, A., and Presutti, V.: Ontology patterns: Clarifying concepts and terminology. In: Proc. of the 4th Workshop on Ontology and Semantic Web Patterns, CEUR, Sydney, Australia. (2013)
3. Falbo, R., Barcellos, M.P., Nardi, J.C., Guizzardi, G.: Organizing Ontology Design Patterns as Ontology Pattern Languages, 10th Extended Semantic Web Conference (ESWC 2013), Monpellier, France. (2013)

4. Falbo, R.A., Barcelos, M.P., Ruy, F.B., Guizzardi, G., Guizzardi, R.: Ontology Pattern Languages, Ontology Engineering with Ontology Design Patterns: Foundations and Applications. In: A. Gangemi, P. Hizler, K. Janowicz, A. Krisnadhi, V. Presutti (eds), IOS Press, pp. 133-159 (2016)
5. Falbo, R., Ruy, F., Guizzardi, G., Barcellos, M., Almeida, J.P.: Towards an enterprise ontology pattern language. Proc. of the 29th Symposium on Applied Computing (SAC 2014), pp. 323-330, Gyeongju, Republic of Korea (2014)
6. Oliveira, F.F., Antunes, J., Guizzardi, R.: Towards a Collaboration Ontology. In Proc. of the 2nd Workshop on Ontologies and Metamodels in Software and Data Engineering. pp. 97-108. João Pessoa/PB, Brazil (2007)
7. Oliveira, F. F.: Uma ontologia de colaboração e suas aplicações. (in Portuguese) Masters Dissertation, Universidade Federal do Espírito Santo, Brazil. (2009)
8. França, J., Netto, J., Carvalho, J., Santoro, F., Baião, F., Pimentel, M.: KIPO: the knowledge-intensive process ontology. Software & Systems Modeling, 14(3), pp. 1127-1157. (2015)
9. Guerson, J., Sales, T.P., Guizzardi, G., Almeida, J.P.A.: OntoUML Lightweight Editor: A Model-Based Environment to Build, Evaluate and Implement Reference Ontologies, In: Proc. of the 19th IEEE Enterprise Computing Conference (EDOC 2015). pp. 144-147. Demo Track, Adelaide, Australia, (2015)
10. Santos, R.O., Oliveira, F.F., Gomes, R.L., Martinello, M., Guizzardi, R.S.S.: Lightweight Collaborative Web Browsing. In: Polgar, J., Adamson, G. (eds.) Web Portal Design, Implementation, Integration, and Optimization, IGI Global, pp. 17-32. (2013)
11. Gentil P.B., Campos M.L.M., Borges M.R.S.: Construction and Evaluation of a Collaboration Observation Model. In: Baloian N., Burstein F., Ogata H., Santoro F., Zurita G. (eds) Collaboration and Technology. CRIWG 2014. LNCS, v. 8658. Springer (2014)
12. Fuks, H., Raposo, A., Gerosa, M., Pimentel, M., Lucena, C. "The 3c collaboration model." In Encyclopedia of E-collaboration, pp. 637-644. IGI Global (2008).
13. IBM Global Services: Using collaboration to enable the innovators in your organization (2006) Available at: https://docplayer.net/16076799-Using-collaboration-to-enable-the-innovators-in-your-organization-welcome-to-the-ibm-cio-podcast-series-i-m-your-host-jeff-gluck.html (captured in Dec 2019)
14. Avery, C., Walker, M., Murphy, E. Teamwork is an Individual Skill: Getting your work done when sharing responsibility. San Francisco: Berrett-Koehler (2001)
15. Nguyen, A.V., Rekik, Y., Gillet, D.: A framework for sustaining the continuity of integration in Web-based learning environment for engineering education. In: Proc. of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2005). AACE, Montreal, Canada (2005)
16. Ellis, C. A., Gibbs, S. J., and Rein, G.: Groupware: some issues and experiences. Commun. ACM, (34)1, pp. 39-58. (1991)
17. Winograd, T., Flores, F.: Understanding Computers and Cognition. AddisonWesley, USA. (1987).
18. Ruy, F., Guizzardi, G., Falbo, R., Reginato, C., Santos, V.: From Reference Ontologies to Ontology Patterns and Back, Data & Knowledge Engineering, (109), pp. 41-69 (2017)
19. Deutsch, P.: Models and Patterns. Software Factories: Assembling Applications with Patterns, Frameworks, Models and Tools. (2004)
20. Buschmann, F., Henney, K., & Schimdt, D.: Pattern-oriented Software Architecture: on patterns and pattern language (Vol. 5). John Wiley & Sons. (2007)
21. Barcellos, M., Falbo, R., Frauches, V.: Towards a Measurement Ontology Pattern Language. Proc. Of the ONTO.COM/ODISE workshop, co-located with the 8th International Conference on Formal Ontology and Information Systems (FOIS) (2014)

22. Souza, A.K., Guizzardi, R., Campos, M.L.M., Guizzardi, G.: Extending an Ontology Editor for Domain-related Ontology Patterns Reuse: An Application in the Collaboration Domain. In: Proc. IX Brazilian Ontology Research Seminar (ONTOBRAS). pp. 47-58. CEUR, Curitiba, Brazil, (2016)
23. Quirino,G., Barcellos, M.P., Falbo, R.D.A.: OPL-ML: A Modeling Language for Representing Ontology Pattern Languages, In: S. de Cesare and U. Frank (Eds.): ER2017 Workshops. LNCS 10651, pp. 187-201. Springer, Cham (2017)
24. Souza, A.K.: Suporte automatizado para desenvolvimento de ontologias utilizando padrões ontológicos de domínio. Masters Dissertation, Universidade Federal do Espírito Santo, Brazil. (2017)

O Uso da Abordagem SABiO na Construção do Overview de OntoSaúde

Danielli dos Reis Costa, Maria das Graças da Silva Teixeira,
Silvia das Dores Rissino, Thayza Sacconi Guarnier

Universidade Federal do Espírito Santo (UFES), São Mateus-ES, Brasil

{maria.teixeira, silvia.rissino}@ufes.br,
{daniellic9, thayzasaconi}@gmail.com

Resumo. A parceria entre as áreas de Ciências da Saúde e Ciência da Computação está solidamente estabelecida. No momento, há uma demanda de aplicações de análise de dados de Saúde. Uma análise de dados de qualidade requer uma compreensão profunda da semântica do domínio explorado. Essa semântica pode ser obtida através de ontologias. O presente projeto aplica a abordagem SABiO para desenvolver um overview de uma ontologia na área da Saúde, denominada OntoSaúde, bem como artefatos que utilizam tal ontologia. A ontologia de referência é mapeada para um modelo de dados utilizando a ferramenta REDCap, resultando em um instrumento de coleta de dados. Os dados coletados são exportados em formatos que permitem o seu uso em variadas ferramentas de análise de dados. Para fins de avaliação inicial, os dados exportados são utilizados na geração de gráficos no software Excel. As primeiras impressões e resultados são promissores.

Palavras-chaves. SABiO, Engenharia de Ontologias; Ontologias em Saúde.

1 Introdução

Nos últimos anos, o interesse na área de Modelagem Conceitual tem aumentado, não só na Ciência da Computação como em outros campos. E tal comunidade tem-se voltado cada vez mais para ontologias [1]. O termo Modelagem Conceitual Baseada em Ontologias (*Ontology-Driven Conceptual Modeling*) é aplicado para denominar modelos conceituais desenvolvidos com fundamentação em teorias ontológicas. Tais modelos têm as mais variadas aplicações, seja como artefato no desenvolvimento de um software, seja para capturar a semântica de um modelo de dados, seja na avaliação e desenvolvimento de linguagens de modelagem e de seus métodos de desenvolvimento, como por exemplo na fundamentação do processo de design de linguagens de modelagem denominado Física das Notações Ontologizada e Sistematizada (*Physics of Notation Ontologized and Systematized*) (PoNTO-S) [2].

À medida em que o uso de ontologias se expande, aumenta a necessidade de metodologias e tecnologias para desenvolvimento de tal artefato. Nesse sentido, pode ser citada a *Unified Foundational Ontology* (UFO) [1], uma ontologia de fundamentação aplicada no desenvolvimento tanto de ontologias núcleo quanto de

ontologias de domínio, que nos últimos anos tem se destacado como uma fundamentação sólida e consistente, além de possibilitar expansões em seu metamodelo. UFO fornece uma base de qualidade para fundamentar um modelo conceitual baseado em ontologia (ou simplesmente, ontologia de referência), e é preciso responder uma outra questão: há um processo de desenvolvimento de ontologias que possa ser utilizado em conjunto com UFO? A resposta a tal pergunta é sim. Pelo uso do *Systematic Approach for Building Ontologies* (SABiO) [3].

Trabalhar com ontologia requer escolher um domínio a ser explorado. Esse domínio não foi escolhido de forma aleatória, pois atualmente a demanda de sistemas na área de Saúde está crescendo sem uma devida organização do conhecimento. Escolheu-se, então, explorar o domínio de Saúde Coletiva, o qual é uma subárea da grande área da Ciências da Saúde, conforme proposto pela CAPES. A Saúde Coletiva é composta pela Epidemiologia, Saúde Pública e Medicina Preventiva.

Foi estabelecido como objetivo do trabalho: desenvolver o overview de uma ontologia de referência na área de Saúde Coletiva, e o seu mapeamento para uma ferramenta informacional.

Para implementar o objetivo estabelecido foi aplicada a metodologia SABiO, que resultou na geração da ontologia de referência denominada OntoSaúde e guiou o mapeamento dos conceitos ali caracterizados para um conjunto de variáveis que compõe um instrumento de coleta de dados. A implementação do instrumento de coleta ocorreu no ambiente REDCap, uma ferramenta de uso cada vez mais disseminada por profissionais da área de Saúde. Além de habilitar a construção e uso de instrumentos de coletas de dados, o REDCap também possibilita a exportação dos dados coletados. Uma simulação de coleta de dados foi realizada através do REDCap e os dados coletados foram exportados e utilizados para originar uma série de gráficos, possibilitando uma análise simplificada dos dados simulados. Para construir os gráficos foi utilizada a ferramenta Excel.

A motivação para escolha do objetivo e da metodologia de trabalho aqui estabelecidos foi um questionamento: a aplicação conjunta de SABiO, UFO, REDCap e uma ferramenta simples de análise de dados é possível e gera um resultado válido?

Esse texto está dividido nas seguintes seções: A seção 2 apresenta um breve referencial teórico dos conceitos necessários para compreender esse texto. A seção 3 contém uma descrição detalhada de como o overview de uma ontologia da área de Saúde foi desenvolvida através do uso de SABiO. Por fim, a seção 4, descreve as considerações finais desse trabalho.

2 Referencial Teórico

2.1 Modelagem Conceitual Baseada em Ontologias

Um dos maiores propósitos da modelagem conceitual é facilitar a comunicação entre usuários e analistas de um determinado artefato. Analistas desenvolvem um modelo conceitual (tipicamente diagramático) para documentar a sua compreensão de um determinado domínio. Tal modelo permite que os usuários verifiquem se a interpretação feita pelo analista reflete a realidade como percebida pelos usuários e façam uso do modelo em suas atividades [14]. Atualmente, mais e mais os papéis de analista e usuário

se mesclam, destacando que no fim das contas há duas tarefas para um modelo conceitual: a sua leitura e a sua escrita, e que ambas as tarefas podem ser realizadas por uma variedade de perfis, inclusive da mesma pessoa, além de serem utilizadas em um amplo leque de atividades.

No entanto, uma coisa não muda: modelos conceituais devem ser uma representação expressiva e acurada do domínio que pretendem representar, já que são representações de conhecimento. Uma das maneiras de se obter essa representatividade é através de ontologias. O *matching* entre modelos conceituais e ontologias já está estabelecido, sendo conhecido como **Modelos Conceituais Baseados em Ontologias**, ou simplesmente, ontologias de referência.

Ontologias de referência são aplicadas em variadas áreas da Ciência da Computação, tais como Engenharia de Software, Inteligência Artificial, Banco de Dados. E nos mais variados domínios, tais como Direito, Saúde, Redes.

Essa variedade de possibilidades, conduz a uma variedade de classificações de ontologias. Uma das mais adotadas foi introduzida por Guarino [15]. Guarino classifica as ontologias de acordo com o seu nível de generalidade, definindo as seguintes categorias: (1) Ontologias de Nível-topo ou Ontologias de Fundamentação, descrevem conceitos gerais que são independentes de um domínio ou tarefa em particular, tais como objetos, tempo e espaço; (2) Ontologias de domínio descrevem um vocabulário específico a um domínio (por exemplo, a indústria de alimentos), especializando os termos de uma ontologia de fundamentação; (3) Ontologias de Tarefa, descrevem uma atividade em geral (por exemplo, um processo de produção de alimentos), também através da especialização de uma ontologia de fundamentação; (4) Ontologias de aplicação especializam conceitos relacionados a um determinado domínio e tarefa.

Uma classificação complementar foi proposta em [16]. Tal proposta foca em uma perspectiva estrutural, adicionando as chamadas Ontologias Núcleo, que se localizam entre as ontologias de fundamentação e de domínio. Uma ontologia núcleo estrutura conceitos de um domínio mais amplo (por exemplo, Direito) que pode ser expandida para diferentes domínios (como por exemplo o Direito Criminal).

O termo ontologia de referência, apresentado com mais detalhes na seção seguinte é perpendicular às categorias previamente citadas, pois refere-se ao uso de ontologias como modelos conceituais, que é uma visão independente da tecnologia que irá ser utilizada para implementar artefatos incluindo a própria ontologia. O complemento de tal classificação são as chamadas ontologias operacionais.

UFO é uma ontologia de fundamentação que tem sido utilizada como base para ontologias de domínio [1]. UFO está organizada em 3 camadas: UFO-A (de objetos), UFO-B (de eventos) e UFO-C (sociais).

O desenvolvimento de ontologias tem sido estudado através da área conhecida como Engenharia de Ontologias, responsável por propor métodos, técnicas e ferramentas que podem ser utilizados não só para desenvolver ontologias e artefatos que as utilizam como base, como também para preservar e expandir tais ontologias e artefatos correlacionados. A seguir encontra-se descrito SABiO, a abordagem selecionada pelos pesquisadores para aplicação no projeto descrito parcialmente neste texto.

2.2 SABiO: Abordagem Sistemática para Construção de Ontologias (*Systematic Approach for Building Ontologies)*[1]

A metodologia escolhida para uso no projeto descrito nesse texto é a *Systematic Approach for Building Ontologies* (SABiO)[3], desenvolvida pelo Prof. Ricardo de Almeida Falbo. SABiO é uma abordagem sistemática para a construção de ontologias baseada na metodologia proposta em [4] e adicionada de alguns itens, tais como uso de linguagens gráficas, classificação de axiomas e uso de questões de competência. O processo descrito por SABiO é composto por atividades e diretrizes para guiar a sua aplicação.

A escolha de tal metodologia não foi aleatória. SABiO tem sido aplicada com sucesso no desenvolvimento de uma série de ontologias, tais como a ROoST (*Reference Ontology on Software Testing*)[5] e uma ontologia de referência para Autorização Orçamentária e Execução da Despesa Pública no Brasil [6]. Além disso, SABiO pode ser integrada com o uso de UFO como ontologia de fundamentação, é suficientemente genérica para ser aplicada em diversas áreas, e pode ser adaptada (estendida) para características específicas de um projeto.

Antes de descrever as atividades de SABiO é útil estabelecer alguns conceitos presentes em tal processo, sendo eles:

- **Ontologia de referência de domínio**. Uma ontologia de domínio que é construída com o objetivo de fazer a melhor descrição do domínio com relação a um determinado nível de granularidade e ponto de vista. Também pode ser referenciada como Modelo Conceitual Baseado em Ontologia.
- **Ontologia Operacional**. Uma ontologia desenvolvida em uma linguagem de implementação de ontologias, como OWL.
- **Modelo Relacional de Dado**s. Um modelo de dados que adota a metodologia relacional, como a descrita por Peter Chen [7].
- **Questão de Competência**. Cada questão relevante sobre o domínio do problema a ser respondida pela ontologia criada.
- **Axiomas de Derivação.** Permitem que novas informações possam ser derivadas a partir do conhecimento prévio.
- **Axiomas de Consolidação**. Definem os limites para definição de um objeto ou estabelecimento de relações.

Além disso, também é válido caracterizar os papéis dos envolvidos no processo, sendo eles:

- **Especialista em Domínio.** É um especialista no domínio do problema a que se refere a ontologia e que fornece o conhecimento que deve ser modelado e implementado na ontologia do domínio. No caso do projeto aqui descrito, são os especialistas da área da Saúde;
- **Usuário de Ontologia.** Representa todas as pessoas que pretendem usar a ontologia para um determinado propósito. No caso do projeto sendo descrito, são tanto os usuários do banco de dados a ser desenvolvido quanto os profissionais da Saúde e os próprios pesquisadores do projeto.

---

[1] O texto dessa seção foi descrito baseado em [3] a não ser quando explicitamente utilizada outra referência. Por isso, evitou-se usar repetidamente a referência em questão.

- **Engenheiro de Ontologia.** É o responsável pelo desenvolvimento da ontologia de referência.
- **Designer/Projetista.** Responsável pelo design (projeto) da ontologia operacional e/ou do modelo relacional de dados.
- **Programador.** Responsável pela implementação do artefato criado pelo projeto.
- **Testador**. Responsável por testar a qualidade dos artefatos implementados.

Descrição do Processo de Desenvolvimento da(s) Ontologia(s)

A Fig. 1 apresenta um *overview* da metodologia SABiO, adaptada para as necessidades deste trabalho. Sendo a adaptação realizada[2]: a fase de implementação pode gerar diferentes modelos, tanto uma ontologia operacional quanto um modelo relacional de dados, ou mesmo outro tipo de modelo de dados.

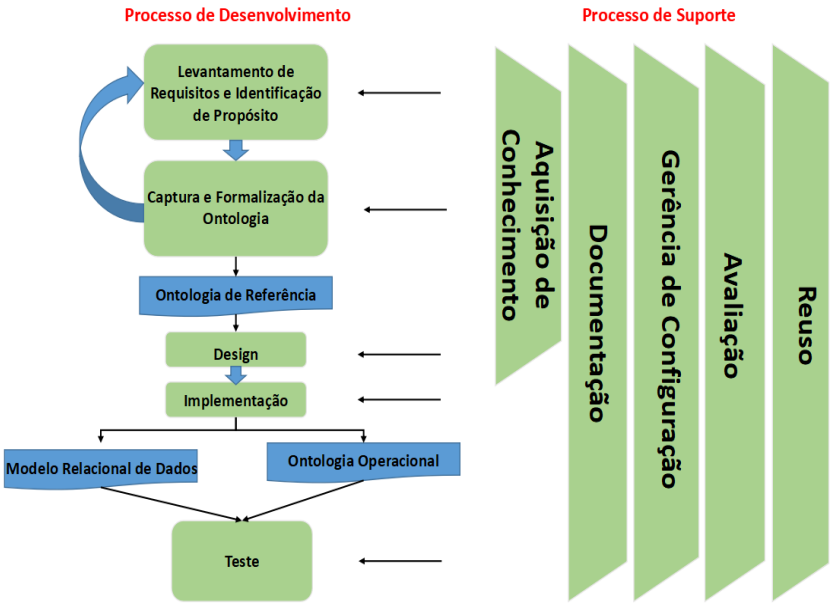

Fig. 1: Processo de desenvolvimento SABiO. Inclusão do artefato: Modelo Relacional de Dados

O processo de desenvolvimento do SABiO compreende cinco fases principais: (1) Identificação de propósito e levantamento de requisitos; (2) Captura e formalização; (3) Design; (4) Implementação; (5) Teste.

Sobre a fase de Identificação de Propósito e Levantamento de Requisitos. Suas atividades são: Identificação de usos e propósito, Identificação de sub-ontologias e Identificação de questões de competência (aqui também se incluem os requisitos do

[2] SABiO considera que a ontologia operacional pode ser um modelo relacional de dados. No entanto, para melhor diferenciar esses 2 formatos de modelos optou-se por tornar a diferenciação entre os dois tipos de artefatos de implementação mais destacada.

problema). Os participantes previstos para executarem as atividades são: Engenheiro de ontologia, especialista de domínio e possíveis usuários da ontologia. As atividades devem ser aplicadas de forma iterativa até que se chegue a um resultado satisfatório.

Se a ontologia demonstrar grande complexidade e necessitar de modularização poderão ser criadas sub-ontologias a partir de uma ontologia núcleo.

Sobre a fase de Captura e Formalização. Suas atividades são: Modelagem Conceitual, Elaboração do Glossário de Termos, Definição de Axiomas Informais e Definição de Axiomas Formais. Os participantes previstos para executarem as atividades são: Engenheiro de Ontologias e Especialista de Domínio. O Engenheiro de ontologias identifica os conceitos principais e relações, organizando-os em taxonomias, de modo a se evitar termos que possam ser mal interpretados. Conceitos primitivos não podem ser definidos em termos de outros conceitos. Eles devem ser descritos em linguagem natural com exemplos evitando ambiguidade e inconsistências. Possível modularização e hierarquização de modelos também deve ser estudada, e, se necessário, desenvolvidas. Reuso de artefatos ontológicos é muito importante e é utilizado nessa etapa, ajudando processos de integração, reengenharia e mapeamento. Existe a possibilidade de axiomas complementares aos conceitos serem criados, e isso deve ser feito com base nas questões de competência. Existem dois tipos de axiomas a serem considerados: axiomas de derivação e axiomas de consolidação. Sugere-se o uso de UFO como a fundamentação para a ontologia de referência.

Sobre a fase de Design. Suas atividades são: Elicitação de requisitos não funcionais, Definição de plataforma de implementação, Design Arquitetural e Design Detalhado. Os participantes previstos para participarem em tais atividades são: Engenheiro de ontologia e Designer de ontologias.

Sobre a fase de Implementação. Suas atividades são: Implementação. Os participantes previstos para participarem em tais atividades são: Projetista e Programador.

Para encerrar a sequência de atividades tem-se a fase de Teste. Suas atividades são: Teste de sub-ontologias, teste de integração e teste de qualidade de modelo e qualidade de dados. Os participantes previstos para executarem tais atividades são: Testador de ontologias e usuário da ontologia. Os testes se referem a validação e verificação da ontologia operacional e/ou do modelo relacional de dados através de um conjunto finito de testes, a ser previamente elaborado com base nas questões de competência. Testes de qualidade do modelo visam aferir a qualidade do modelo construído enquanto os testes de qualidade dos dados têm como objetivo principal detectar campos nulos, erros de entrada de dados, duplicação e inconsistências de dados.

As fases de SABiO são complementadas por processos de suporte, sendo eles: (1) Aquisição de conhecimento, (2) Reuso, (3) Documentação, (4) Gerenciamento de configuração, (5) Avaliação. Tais processos estão descritos a seguir.

Sobre o processo de Aquisição de Conhecimento: A maior parte do processo de aquisição do conhecimento acontece nas fases iniciais do processo de desenvolvimento. Aqui, recorre-se aos métodos e técnicas convencionais de aquisição de conhecimento, tais como *brainstorming*, análise de documentos e entrevistas com os especialistas. É um processo fortemente relacionado ao processo de reuso.

O termo informal "não reinventar a roda" representa o processo de reuso e suas inúmeras vantagens. Para a construção de ontologias podem existir quatro tipos de reuso: ontologias de domínio, ontologias núcleo, ontologias de fundamentação e

padrões de ontologias. O reuso de ontologias pode ser total ou parcial. As técnicas utilizadas podem ser técnicas de mesclagem de ontologias, mapeamento e reengenharia. Reuso de ontologias núcleo é em geral feito para fins de especialização - os conceitos e relações mais gerais da ontologia núcleo são estendidos para caracterizar uma conceitualização mais específica do domínio. Ontologias de Fundamentação podem ser reutilizados, assim como as ontologias núcleo, para fins de especificação, mas também por analogia. No reuso por analogia, os conceitos e relações são implicitamente usados para derivar a estrutura da ontologia de domínio. Também pode-se usar ontologias de fundamentação para analisar fragmentos da ontologia de domínio. No que se refere ao reuso através de padrões, existem quatro tipos principais de padrões de ontologias: conceitual, arquitetural, design e de programação. Padrões conceituais de ontologias são usados durante a atividade de modelagem conceitual, e focam somente nos aspectos conceituais, sem preocupação com a tecnologia ou linguagem usadas para se derivar a ontologia operacional. Padrões de ontologias de fundamentação são fragmentos reusáveis de ontologias de fundamentação, podendo ser aplicados em qualquer domínio - são usados através da analogia entre o padrão e o problema em mãos. Já os padrões arquiteturais de ontologias são padrões que descrevem como organizar uma ontologia em termos de sub-ontologias e/ou em módulos. Esses padrões podem ser usados durante a identificação de propósito e a fase de levantamento de requisitos, e na fase de *design*. Padrões de *design* de ontologias são usados durante a atividade de *design* detalhado, e existem dois tipos: lógico e de raciocínio. Padrões de raciocínio são para problemas específicos de forma a melhorar o raciocínio com ontologias, como, por exemplo, o processo de busca semântica nas bases de dados. Por sua vez, padrões lógicos são relacionados a problemas de expressividade do formalismo usado para implementação da ontologia. Ajudam a resolver problemas de *design* quando a linguagem não suporta diretamente certas construções lógicas. Durante os testes de ontologias, os casos de teste podem ser reutilizados.

O processo de documentação ressalta o fato de que todos os resultados do desenvolvimento de uma ontologia devem ser documentados, incluindo propósitos, intenções de uso, questões de competência, modelos conceituais, descrições textuais dos conceitos e relações, axiomas formais e informais, requisitos não funcionais, arquitetura, soluções de *design*, casos de teste e resultados de teste. Até mesmo resultados dos processos de suporte. Assim, a documentação é um processo que ocorre em paralelo aos demais. Preferencialmente, devem ser definidos *templates* de documentos e quais os documentos básicos a serem criados. Entre os documentos elaborados destaca-se o glossário de termos, que complementa os conceitos e relações presentes no(s) diagrama(s) que representa(m) a ontologia, ao descrever as definições de tais termos

O processo de Gerenciamento de Configuração é um importante lembrete de que os principais documentos propostos pela metodologia bem como como as fontes de códigos operacionais devem ter suas configurações gerenciadas. Assim uma vez aprovadas, devem ser submetidas ao gerente de configuração onde elas serão controladas pelas mudanças, versões e entregas.

O processo de Avaliação aparece tanto no seu sentido de testes, para encerrar as fases de desenvolvimento das ontologias e dos artefatos que a utilizam como um processo de suporte em paralelo que ocorre durante todas as fases de desenvolvimento. O processo de avaliação de uma ontologia possui duas perspectivas: verificação e validação.

Verificação visa assegurar que a ontologia está sendo construída corretamente, de forma que os artefatos de uma atividade estejam de acordo com as especificações definidas pelas atividades anteriores. Validação pretende assegurar que a ontologia certa está sendo construída, ou seja, que a ontologia supre todas as especificações propostas.

Tanto a verificação quanto a validação devem começar cedo no processo de desenvolvimento. A atividade de teste é executada para avaliar a qualidade da ontologia, assim como identificar os defeitos e problemas. Teste de Ontologia consiste em testar o comportamento de uma ontologia operacional num conjunto finito de casos de teste, contra um comportamento esperado. Porém, outras atividades também são executadas durante o desenvolvimento, como a revisão técnica, cujo propósito é avaliar um produto intermediário do processo de desenvolvimento para determinar se é apto para sua necessidade.

Sobre a verificação, o foco é nos dois principais aspectos: se os critérios de qualidade são satisfeitos e se os padrões estão sendo corretamente aplicados. Sobre os critérios de qualidade um se destaca, a competência. Durante a fase de captura e formalização da ontologia de referência deve-se verificar se a ontologia está de acordo com os requisitos e questões de competência identificados. Isso pode ser feito através de revisões técnicas, uma tabela indicando quais elementos da ontologia são capazes de responder cada questão. O propósito dessa tabela vai além da verificação, podendo ser usado como ferramenta de rastreabilidade, dando suporte a gerência de mudanças (assunto não abordado diretamente pelo SABiO).

As questões de competência são essenciais na verificação da completude da ontologia, especialmente quando considerando os axiomas. Em um domínio particular pode-se identificar um bom número de conceitos, relações e propriedades para escrever axiomas. Porém, deve-se ter atenção para não incluir na ontologia mais elementos do que o necessário. O conjunto de elementos deve ser necessário e suficiente para expressar as questões de competência e para caracterizar suas soluções, de forma que elementos que não contribuem para responder essas questões devam ser excluídos.

Sobre a validação é essencial a participação dos especialistas de domínio e dos usuários. Os usuários têm que avaliar se a ontologia é adequada para sua necessidade. O uso de notação gráfica para validar a ontologia de referência é importante, desde que os usuários normalmente não são treinados para ler as especificações formais. Outra maneira de validar uma ontologia de referência é através da instanciação, representando situações do mundo real.

2.3 Ferramentas para Design e Implementação

Um Instrumento para Coleta de Dados Baseado em Ontologias

Um instrumento de coleta é um formulário usado para coletar dados a serem posteriormente analisados a fim de auxiliar na tomada de decisões em um determinado domínio. Eles são utilizados em várias áreas, e particularmente úteis na Saúde Coletiva.

O instrumento de coleta descrito neste trabalho possui as seguintes características [13]: importância, necessidade, brevidade, não ter ambiguidade, analisável, validade e universo único, além de que seria utilizado para simular, com dados fictícios, uma coleta de dados primária.

Como o principal objetivo deste trabalho é mostrar como um modelo de ontologia pode facilitar a construção de um instrumento de coleta, não são detalhadas e nem considerados as técnicas para construção de instrumentos de coleta, apenas considera-se o modelo de ontologia proposto, descreve-se brevemente a ferramenta usada para criar o instrumento de coleta, e por fim, na seção descrita é apresentado como foi construído o instrumento de coleta a partir da overview da OntoSaúde.

Para a construção do instrumento de coleta baseado em ontologia foi utilizado o REDCap (*Research Electronic Data Capture*), software do tipo EDC[3], baseado em navegador Web, orientado por metadados, utilizado para o desenvolvimento e implementação de ferramentas de capturas de dados que apoiam pesquisas clínicas e translacional. Criado em 2004 na Universidade de Vanderbilt, apresenta-se como um ambiente gratuito para os Parceiros do REDCap Consortium (https://projectredcap.org), além de ser fácil de usar, flexível e robusto [8].

O REDCap possibilita criar e gerenciar pesquisas e bancos de dados on-line, sendo utilizado para coleta de vários tipos de dados (incluindo ambientes compatíveis com 21 CFR Parte 11[4], FISMA[5] e HIPAA[6]). O REDCap é voltado especificamente para suportar a captura de dados online ou offline e o gerenciamento de dados em pesquisa científica, principalmente pelo fato de ter conformidade regulatória, o que possibilita a guarda segura dos dados produzidos, além de permitir auditoria através do uso de trilhas rastreáveis de manipulação de dados e de atividades dos usuários [9].

A capacidade de gerenciar banco de dados online está diretamente relacionada aos tipos de Sistemas Gerenciadores de Banco de Dados (SGBDs) que podem trabalhar com REDCap. O SGBD MySQL é o mais indicado por ter licença do tipo GLP,por sua facilidade uso, padronização e robustez. Neste trabalho utiliza-se a versão estável 8.0.16. [10].

O ambiente disponibilizado pelo REDCap apresenta procedimentos de exportação automatizados através do download dos dados para arquivos de planilhas eletrônicas no formato CSV, para arquivos PDF e para pacotes estatísticos como SPSS, SAS, Stata e R. Esta característica facilita, ao final da fase da coleta, que as análises preliminares sejam realizadas diretamente no ambiente do REDCap [11].

O grande motivador para utilização do REDCap neste trabalho, é que o REDCap abre a possibilidade de utilizar ontologia em seu desenvolvimento, o que indica uma preocupação com relação ao uso de ontologia, mesmo que de maneira bem básica. Nesse caso, é uma ontologia operacional disponibilizada através do BioPortal. O fato de usar o MySql como SGBD, possibilita ter um modelo de dados relacional, com

---

[3] EDC (*Electronic Data Capture*): Sistema de captura eletrônica de dados projetado para coleta de dados clínicos em formato eletrônico para uso principalmente em ensaios clínicos em humanos.

[4] Parte 11, é a parte do Título 21 do Código de Regulamentos Federais que estabelece os regulamentos da Administração de Medicamentos e Alimentos dos Estados Unidos sobre registros e assinaturas eletrônicas.

[5] FISMA (*Federal Information Management Act*): Lei federal americana de gestão de segurança da informação.

[6] HIPAA (*Health Insurance Portability and Accountability Act*): conjunto de conformidades que organizações de saúde passaram a seguir para proteger suas informações digitais internas.

flexibilização de regras de integridade relacional, mas sem perder a essência da integridade de dados.

Um Instrumento para Análise de Dados Baseado em Ontologias
O REDCap armazena os dados coletados através do instrumento de coleta em uma base de dados interna, possibilitando construir relatórios baseados em consultas simples.

Os dados armazenados podem, também, serem exportados em diferentes formatos, tais como CSV e XLS, possibilitando análise dos dados por uma variedade de ferramentas disponíveis atualmente no mercado, como por exemplo o PowerBI e o Excel. Espera-se que a análise de dados se baseie nas questões de competência identificadas durante o desenvolvimento da ontologia de referência.

2.4 Domínio da Saúde

Cada vez mais os conceitos de ser saudável ou não saudável tem sido alterados. Ao ponto de que uma pessoa com alguma doença sem cura ser considerada saudável, desde que sua doença esteja sob controle. O que se sabe com certeza é que a grande área de Saúde tem recebido cada vez mais atenção. Inclusive cada vez mais se destaca a integração entre Saúde e Computação.

É possível observar a área da Saúde sobre dois focos: Biológico e Social. No lado Biológico procura-se entender o que é uma doença, sua manifestação, causas, sintomas, diagnóstico e formas de tratamento, entre outros conceitos. O lado Social é o lado que envolve o serviço de saúde, e todas as variáveis envolvidas no tratamento e acompanhamento que deve ser oferecido seja a um indivíduo com problemas de saúde, seja à sociedade.

A Organização Pan-Americana da Saúde no Brasil (OPAS) lançou uma coletânea de módulos que falam sobre doenças, com enfoque em epidemiologia, que visa auxiliar os profissionais de saúde a melhorar o planejamento e a gestão em saúde, além de promover o conhecimento e a aplicação dos conceitos. Esses módulos formam a principal fonte de informação no domínio da saúde para elaboração da ontologia e estão descritos em [13].

Devido ao caráter de overview da ontologia a ser criada os conceitos utilizados das fontes de informação são bastante abrangentes e possuem a capacidade de serem expandidos no futuro, de forma que conceitos como Tratamento, Doença e Serviço de Saúde possam dar origem a conceitos mais detalhados de forma que seja mais fiel às descrições nas fontes de informação de domínio.

3 Overview de OntoSaúde

Dado o tamanho e complexidade da área de Saúde, para ela ser melhor explorada optou-se pelo desenvolvimento de uma ontologia, chamada **OntoSaúde**, em diferentes níveis de abstração. O primeiro a nível a ser desenvolvido é um overview sobre a área como um todo, considerando tanto o lado biológico quanto o social. Essa é a ontologia que está descrito no presente texto, e cujos demais níveis de abstração estão em processo de construção através do mesmo método.

A seguir estão descritas cada fase de SABiO aplicada ao overview de OntoSaúde.

Levantamento de Requisitos e Identificação de Propósito

Na primeira etapa foram identificados os propósitos e usos para a ontologia, bem como algumas questões de competência necessárias de serem respondidas ao final do projeto. Abaixo se encontram descritos tais itens.

O **propósito** da ontologia de referência OntoSaúde é apresentar um modelo conceitual (entidades, relações, atributos e restrições) caracterizado a partir do Módulo de Princípios de Epidemiologia para o Controle de Enfermidades (MOPECE)/2010 [12], que descreve um fragmento do domínio da Saúde Coletiva, buscando caracterizar as relações de uma pessoa e seus possíveis problemas de saúde.

O overview aqui descrito possibilita expansões para conceitos mais específicos da saúde, como por exemplo, de doenças transmissíveis, a serem apresentados em trabalhos futuros.

Quanto ao **uso**, a ontologia de referência será utilizada para a comunicação e aprendizado de interessados na área de saúde coletiva, bem como para geração de ferramentas tecnológicas para uso dos profissionais envolvidos no projeto.

Mesmo para formular o overview uma série de **questões de competência** já foram formuladas, entre elas:

- Qual porcentagem de homem e mulher que tem mais doença?
- Qual porcentagem de homem e mulher que sofre mais acidente?
- Qual a porcentagem de homem e mulher morto por consequência de acidente?
- Qual a porcentagem de homem e mulher morto por consequência de doença?
- Qual a porcentagem de doenças que possuem forma de contágio? E que não possui?
- Quais informações básicas de pessoas são necessárias para um Tratamento de Saúde?
    - Estado da Pessoa, Faixa etária, Diagnóstico, Sexo.
- Quanto ao Diagnóstico, quantas pessoas não saudáveis não possuem diagnóstico?

Captura e Formalização da Ontologia

Na segunda etapa de SABiO foi realizada a modelagem da ontologia de referência, conforme o objetivo de representar um overview da subárea de Saúde Coletiva, que engloba Epidemiologia, Saúde Pública e Medicina Preventiva. Com o propósito de ser overview, a ontologia está representada com poucos detalhes, porém esses são suficientes para estabelecer uma visão geral do domínio.

O overview de OntoSaúde se encontra na Fig. 2. Um glossário de termos acompanha tal diagrama, mas não está aqui descrito por questões de espaço.

A ontologia proposta apresenta como conceitos centrais Pessoa e Problema de Saúde. O conceito de Pessoa (um ser humano) é caracterizado por suas fases de vida (Bebê, Criança, Adolescente, Adulto e Idoso), as classificações quanto ao sexo de nascimento (Homem e Mulher), o estado de vida (Vivo e Morto), bem como suas partes constituintes (Corpo e Mente).

A principal relação entre os conceitos centrais é que uma Pessoa pode ou não apresentar um Problema de Saúde. Através desse relacionamento, implica-se que uma

Pessoa pode ser Saudável ou Não Saudável (não tem um Problema de Saúde). O conceito de Pessoa Saudável é entendido como sendo de pessoas que não possuem nenhum problema de saúde ou pessoas que tem algum problema de saúde que está sob controle. Essa representação foi identificada a partir da classificação dos problemas de saúde definida na área das Ciências da Saúde.

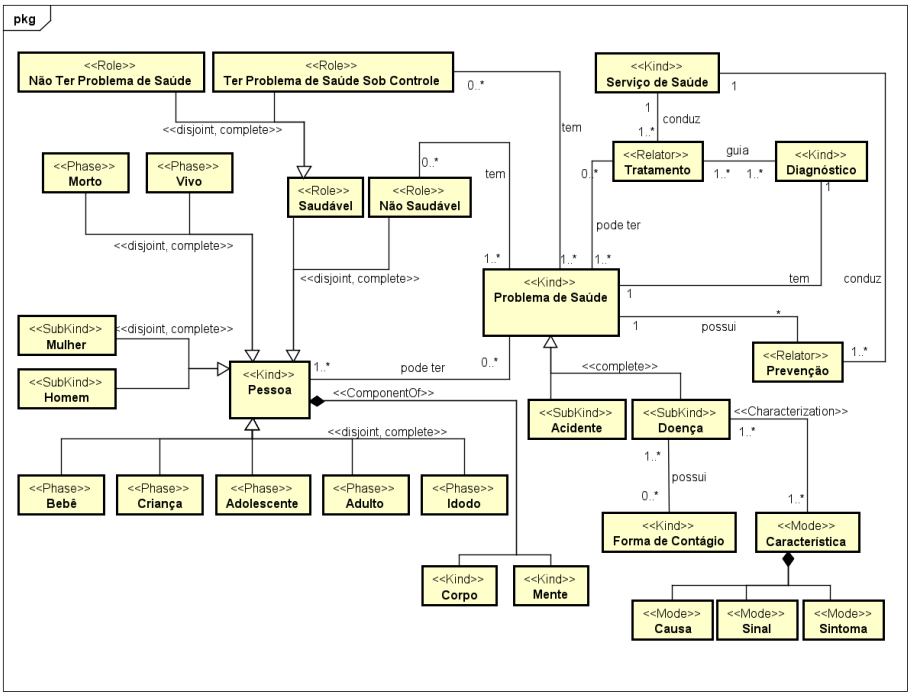

Fig. 2: Overview da OntoSaúde – uma ontologia de referência representativa da Saúde Coletiva[7].

Com relação ao Problema de Saúde, este pode ser subdividido em Acidente e Doença. Por ser o foco da Saúde Coletiva, a Doença é mais explorada. Uma Doença pode possuir uma Forma de Contágio (a maneira de contrair a Doença) e têm algumas Características definidas, sendo elas Causa, um conjunto de Sinais e um conjunto de Sintomas. Além disso, um Problema de Saúde pode ser minimizado através de Formas de Prevenção, sendo que essas são conduzidas por um Serviço de Saúde, que também é responsável por realizar um Tratamento em uma Pessoa que possui um Problema de Saúde. O Tratamento é estabelecido e guiado depois que um Diagnóstico é realizado.

O diagrama representado na Fig. 2 também apresenta os estereótipos originados a partir de UFO que caracterizam cada conceito presente no diagrama. Esses estereótipos fornecem uma série de decisões de modelagem que guiaram a própria elaboração da ontologia.

[7] O diagrama foi desenvolvido utilizando a ferramenta Astah Professional 7.0.0.

Uma vez que a ontologia de referência foi construída, o próximo passo seguido foi o uso de tal ontologia através do seu mapeamento para um modelo de dados. Esse passo está descrito na próxima subseção.

Design e Implementação

O instrumento de coleta desenvolvido é um questionário virtual (survey) com 34 (trinta e quatro) questões, sendo a maioria fechadas ou encadeadas, mas existindo algumas abertas. Observa-se que, o REDCap cria dois atributos adicionais, um é o identificador (ID_pessoa) da pessoa, o qual é gerado automaticamente pela ferramenta e, ao final do instrumento) é criado um atributo ao final do instrumento, para que o gerente do projeto identifique se o registro está completo, verificado ou incompleto, o que facilita o controle de qualidade dos dados.

O objetivo do instrumento é coletar informações básicas de Saúde Coletiva sobre uma população. O questionário desenvolvido foi baseado na ontologia representada na Fig. 2 e construído em ambiente REDCap (versão 9.2.5 – disponível em https://redcap.ufes.br).

O processo de implementação iniciou-se com criação do projeto Avaliação de Pessoas no REDCap e, na sequência, a criação do instrumento de coleta denominado Avaliar_Pessoa. A Tabela 1 apresenta parte do dicionário de dados do Instrumento de Coleta Avaliar_Pessoa, destacando algumas das variáveis que foram identificadas a partir dos conceitos caracterizados na ontologia.

Tabela 1. Exemplo de variáveis do dicionário de dados do instrumento de coleta Avaliar_Pessoa Gerado automaticamente pelo REDCap

Field Type	*Field Label*	*Choices, Calculations, OR Slider Labels*
text	Identificação da Pessoa:	Este campo é o campo ID do registro, que é o primeiro campo no projeto. Este campo é especial porque é usado para armazenar os nomes dos registros no seu projeto. Portanto, o campo ID do registro não pode ser excluído ou movido, mas apenas editado.
radio	Estado da Pessoa:	1, Vivo | 2, Morto
text	Data de Nascimento:	Este campo é solicitado para que seja calculado a idade da pessoa sendo avaliada.
calc	Idade da pessoa:	rounddown(datediff([dt_nascimento_pessoa],"today","y","dateformat", false))
radio	Classificação da fase da vida da pessoa pela idade:	1, Bebê (Menor de 2 anos) 2, Criança 3, Adolescente 4, Adulto 5, Idoso
calc	Idade do óbito da pessoa:	rounddown(datediff([dt_nascimento_morto], [dt_obito_pessoa], "y", "dateformat", false))
radio	Classificação da fase da vida da pessoa pela idade quando do óbito:	1, Nascido morto 2, Bebê (Menor de 2 anos) 3, Criança

Field Type	*Field Label*	*Choices, Calculations, OR Slider Labels*
		4, Adolescente 5, Adulto 6, Idoso
radio	A doença que levou a óbito teve alguma forma de contágio?	1, Sim | 2, Não

A seguir são apresentadas algumas figuras do instrumento de coleta Avaliar-Pessoa, sob o ponto de vista de um usuário que irá preencher tal formulário.

A Fig. 3 apresenta a primeira parte do instrumento de coleta. O Estado da Pessoa apresenta duas opções, que dependendo da escolha conduz a uma segunda etapa do instrumento. A Fig. 4 apresenta a segunda parte do instrumento de coleta, sendo acionada quando a opção do Estado da Pessoa é “Morto”.

Fig. 3. Primeira parte do instrumento de coleta

A partir do design e implementação do instrumento de coleta, ou seja, da entrada de dados, os pesquisadores iniciaram o design e implementação de alguns relatórios, elaborados tendo como base as questões de competência identificadas, e as variáveis definidas no instrumento de coleta. A Fig. 5, por exemplo, está relacionada com a questão de competência “Qual a relação entre as pessoas mortas e vivas por sexo”?. Os dados do gráfico são aqueles resultantes da simulação realizada durante os testes descritos na subseção seguinte. Pelo gráfico, poderia-se concluir que há um maior percentual de homens mortos do que mulheres, bem como há um maior percentual de mulheres vivas registradas do que homens.

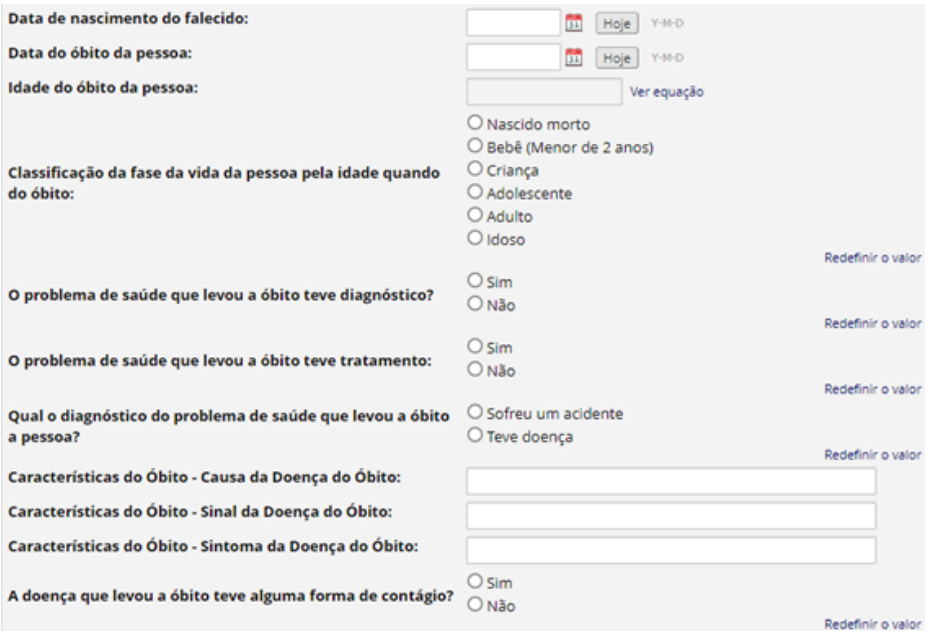

Fig. 4. Segunda parte do instrumento de coleta - Estado da Pessoa "Morto".

Fig. 5. Gráfico representativo da quantidade de pessoas nos estados de vivo e morto caracterizadas por sexo de nascimento.

Testes

Uma vez projetados e implementados os instrumentos de entrada e saída de dados, iniciou-se a próxima etapa de SABiO, ou seja, os testes. Nesse momento, optou-se por realizar uma simulação de dados.

A simulação dos dados para a análise se deu com o instrumento de coleta sendo preenchido diversas vezes pelas mesmas pessoas, a fim de que fossem gerados dados fictícios, ou seja, referentes não a "pessoas reais", mas simulados pelos autores deste trabalho que desejavam, ao final do trabalho, fazer uma análise inicial de dados, bem

como avaliar a coerência dos artefatos gerados (ontologia de referência, instrumento de coleta de dados, relatórios para análise de dados).

Ao final da coleta dos dados, foi possível fazer o download de uma planilha em formato CSV para posterior análise usando a ferramenta licenciada Excel, gerando, então, os gráficos que estão dispostos nas figs. 5 e 6.

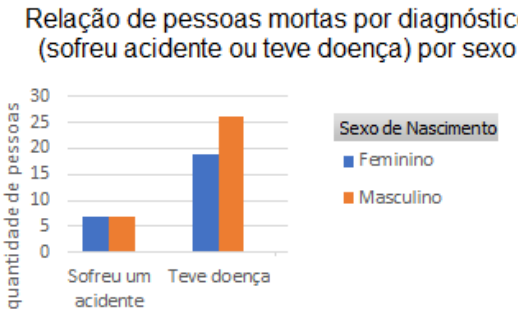

Fig. 6. Gráfico representativo da quantidade de pessoas, caracterizadas por sexo de Nascimento, no estado de "morto" e o problema de saúde (acidente e/ou doença) que conduziu a esse óbito.

4 Considerações Finais

A área de Saúde tem se voltado cada vez mais para ferramentas computacionais a fim de realizar análises de dados de qualidade a partir do imenso conjunto de dados que a área tem a sua disposição. A intenção básica é que decisões da área, tal como a implementação de políticas públicas de Saúde, possam ser tomadas a partir de informações de qualidade que guiem as decisões necessárias. Assim, o presente trabalho propôs a aplicação da metodologia SABiO na criação de um overview de uma ontologia de referência, denominada OntoSaúde, bem como o mapeamento dos conceitos da ontologia para um conjunto de variáveis organizados através de um instrumento de coleta de dados. Para tal mapeamento foi utilizado o ambiente REDCap, o que também tornou possível a exportação de dados preenchidos através do instrumento de coleta para que os mesmos possam ser processados através de uma ferramenta de análise de dados. Nesse caso, de maneira simplificada, gráficos foram desenvolvidos no software Excel.

Em um breve levantamento de publicações em bases de dados especializadas em literatura acadêmica e na Internet, não foram encontrados trabalhos que utilizam, em conjunto, a ontologia de fundamentação UFO e a ferramenta REDCap. No entanto, no desenvolvimento deste trabalho, percebeu-se que o uso de SABiO, colocando em conjunto a ontologia de fundamentação UFO para gerar uma ontologia de referência e a ferramenta REDCap, para gerar artefatos de implementação mapeados a partir da ontologia de referência, produziu um resultado promissor. Os artefatos gerados são apenas iniciais e trabalhos futuros irão investigar se o indício aqui encontrado é verdadeiro.

Como trabalho futuro está previsto o aprofundamento dos conceitos da ontologia OntoSaúde, a partir da overview apresentada. A intenção é abranger conceitos mais específicos da Saúde Coletiva e de classificações de doenças, como por exemplo, as doenças infecciosas e transmissíveis. Como consequência tanto o instrumento de coleta como os relatórios de análise de dados deverão ser expandidos para acomodar as novas características de OntoSaúde. Além disso, o próprio uso do SABiO será melhor explorado, através da intensificação do uso dos processos de suporte, principalmente o de avaliação e o de reuso.

References

1. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Centre for Telematics and Information Technology, University of Twente, The Netherlends (2005).
2. da Silva Teixeira, M. das G.: An Ontology-Based Process for Domain-Specific Visual Language Design, (2017).
3. Falbo, R.: SABiO: Systematic Approach for Building Ontologies. CEUR Workshop Proc. 1301, (2014).
4. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95 (1995).
5. de Souza, É.F., de Almeida Falbo, R., Vijaykumar, N.L.: ROoST: Reference Ontology on Software Testing. Appl. Ontol. 12, 59–90 (2017). https://doi.org/10.3233/AO-170177.
6. Fonseca, L., Almeida, J.P.A., Falbo, R.: Uma Ontologia de Referência para Autorização Orçamentária e Execução da Despesa Pública. iSys - Rev. Bras. Sist. Informação. 11, 4–53 (2019).
7. Chen, P.: The Entity-Relationship Model -Toward a Unified View of Data. ACM Trans. Database Syst. 1, (2001). https://doi.org/10.1145/320434.320440.
8. Harris, P., Taylor, R., Thielke, R., Payne, J., Gonzalez, N., Conde, J.: Research Electronic Data Capture (REDCap) - A Metadata-driven Methodology and Workflow Process for Providing Translational Research Informatics Support. J. Biomed. Inform. 42, 377–381 (2008). https://doi.org/10.1016/j.jbi.2008.08.010.
9. REDCap Consortium: REDCap, Project-redcap.org, last accessed 2019/11/27.
10. MySQL, MySQL Documentation. dev.mysql.com/doc/, last accessed 2019/11/20.
11. Ducan Jr, W.D., Cobb, J., Levey, A.I., Gutman, D.A.: REDLetr: Workflow and Tools to Support the Migration of Legacy Clinical Data Capture Systems to REDCap. Int. J. Med. Inform. 93, 103–110 (2016).
12. Organização Pan-Americana da Saúde, Ministério da Saúde do Brasil. Módulos de Princípios de Epidemiologia para o Controle de Enfermidades. Módulo 2: Saúde e doença na população, (2010).
13. Bryson, G. L., Turgeon, A. F., Choi, P. T.: The science of opinion: survey methods in research. Can. J. Anesth. Can. d'anesthésie. 59, 736–742 (2012). https://doi.org/10.1007/s12630-012-9727-3.
14. Parsons, J.; Cole, L. What do the pictures mean? Guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. Data & Knowledge Engineering, v. 55, n. 3, p. 327–342, 2005.
15. Guarino, N. Formal Ontology in Information Systems. Formal Ontology in Information Systems (FOIS). p.3–15, 1998.
16. Scherp, A.; Saathoff, C.; Franz, T.; Staab, S. Designing Core Ontologies. Applied Ontology, v. 6, n. 3, p. 177–221, 2011.

Personal Note on Ricardo Falbo

Nicola Guarino

ISTC-CNR Laboratory for Applied Ontology, Trento, Italy
nicola.guarino@cnr.it

Ricardo Falbo is the person who is responsible of what are probably the most thorough, insightful, and successful applications of my old ideas on the role of formal ontology in information systems, namely the OntoUML system and the UFO ontology. It was him, indeed, who suggested Giancarlo Guizzardi to study my early papers, when he was still at the beginning of his master studies. Several years later, when Giancarlo invited me to visit the NEMO group, I had a chance to meet him personally, and I was impressed by his vision, his scientific curiosity and his openness to interdisciplinary research. A great scientific leadership!

The FrameWeb Approach to Web Engineering: Past, Present and Future

Vítor E. Silva Souza[0000−0003−1869−5704]

Ontology & Conceptual Modeling Research Group (NEMO)
Department of Computer Science
Federal University of Espírito Santo (UFES) - Vitória, ES, Brazil
vitor.souza@ufes.br
http://www.inf.ufes.br/~vitorsouza/

Abstract. The use of software frameworks is a popular method of reuse, especially in the context of Web-based Information Systems (WISs) development, as they share a very similar basic infrastructure that is generalized and implemented in many different state-of-the-practice frameworks available. Moreover, there has been a growing interest in publishing data on the Web in a machine-processable format known as *linked data* in order to enable automatic processing of the huge amount of data available today on the Internet. The goal of the *FrameWeb* project is to provide methods, languages and tools to aid developers in the construction of WISs that take advantage of the architectural foundation offered by such frameworks and facilitate the publication of linked data from such WISs. In this paper, we review the history of the *FrameWeb* project, describe the approach in its current (end of 2019) form and devise plans for its near future.
This paper has been written to honor Ricardo de Almeida Falbo on the occasion of his formal retirement as a professor of the Department of Computer Science at the Federal University of Espírito Santo (UFES). The research project described herein would not have existed without him. Hence, this paper shows the fruits that came from a seed planted by Ricardo more than a decade ago.

Keywords: Reuse · frameworks · Web Engineering · method · language · tools · FrameWeb

1 Introduction

Software reuse has been practiced since programming began, using, e.g., libraries, domain engineering, design patterns, componentry, etc. [15]. A popular method of reuse is the use of software frameworks (e.g., Hibernate [6]) or platform architectures (e.g., Java™ Enterprise Edition [11]), which are middleware on/with which applications can be developed [15]. The use of such frameworks[1] helps to avoid the continual rediscovery and reinvention of basic architectural patterns and components, reducing cost and improving the quality of software by using proven architectures and designs [25].

[1] In this paper, the term *framework* is used both in its traditional sense—a reusable set of libraries or classes for a software system—and in the sense of *platform architectures* mentioned above.

This is particularly evident in the context of Web-based Information Systems (WISs) development, which is the focus of this paper. WISs are data-centric applications deployed on the Internet or an intranet, in which functionality and data have greater importance than content presentation. Such systems are usually developed on top of a solid Web infrastructure which commonly includes a Front Controller [5], a Dependency Injection mechanism [14], an Object/Relational Mapping [6] solution to communicate with the database, and so on.

Despite their popularity, until recently, and to the best of our knowledge, none of the Web Engineering [22] methods and modeling languages proposed in the literature considered the existence of such frameworks before the coding phase of the software process. Given how these frameworks affect the architecture of a WIS, this fact motivated us to propose *FrameWeb*, a Framework-based Design Method for Web Engineering [27, 28]. *FrameWeb* incorporates concepts from well established categories of frameworks (such as the ones above) into a set of architectural design models, improving developer communication and project documentation.

On a different, but related front, an increasing number of people and organizations are choosing to share their data on the Web, contributing to a *data deluge*. This phenomenon creates problems such as how to provide access to data so it can be most easily reused; how to enable discovery of relevant data; or how to integrate data from different and formerly unknown data sources [18]. A solution that has been gaining momentum in recent years is the publication of *linked data* [7], a set of technologies that lay the foundation for what researchers have been calling *The Semantic Web* [8] for the past two decades.

According to the Semantic Web vision, making data available on the Web in such a machine-processable format, would allow the creation of software agents that could help us through the data deluge, executing tasks that are repetitive, impractical or even impossible to accomplish nowadays. One of the main issues with this vision is that the current level of adoption by data publishers and application developers is not enough for us to harness all the advertised benefits of this new *Web of Data*.

In this context, *FrameWeb* provides a systematic method based on well-founded conceptual models, coupled with tools that automate certain parts of the process, facilitating the task of integrating a WIS into the Web of Data and, thus, promoting the adoption of linked data. Although a small contribution regarding the broader problem of realizing the Semantic Web vision, we can nevertheless harness the benefits of linked data, even if such vision has not been (or will never be) reached.

Since its initial proposal [27, 28], the *FrameWeb* approach has evolved in a number of ways [21, 10, 9, 4, 24], involving many undergraduate and graduate students in a research project.[2] The goal of the *FrameWeb* project is to provide methods, languages and tools to aid developers in the construction of WISs that take advantage of the architectural foundation offered by such frameworks and facilitate the publication of linked data from such WISs.

In this paper, we review the history of the *FrameWeb* project in Section 2 — what was its initial proposal and how it evolved —, describe the approach in its current (end of 2019) form in Section 3 — what can the approach help me accomplish now — and

[2] See https://nemo.inf.ufes.br/projects/frameweb/.

devise plans for its near future in Section 4 — what can we expect as future work in the project. Finally, Section 5 concludes the paper with some personal notes.

2 Past: the FrameWeb Story

FrameWeb's initial proposal [27, 28], developed between 2005 and 2007, focused on three specific frameworks based on my previous experiences in developing Web-based Information Systems (WISs) in practice: Struts,[3] Spring[4] and Hibernate.[5] These frameworks established, later, the initial set of *framework categories* that *FrameWeb* would support:

- **Front Controller frameworks** (e.g., Struts): frameworks of this kind implement a slightly modified version of the Model-View-Controller pattern [16], adapted to the Web and are, thus, also known as *MVC frameworks*. When using such a framework, a WIS manages all requests from clients using an object known as Front Controller. Based on its configuration,[6] this object decides which class (called a *controller class*) will respond to the current request. Then, it instantiates an object of that class, calls one of its methods and, based on the method's return value, the Front Controller decides the appropriate view to present as result, such as a Web page, a PDF report, a file download, etc. For instance, in the Java EE set of standards, JavaServer Faces[7] is a Front Controller framework;
- **Dependency Injection frameworks** (e.g., Spring): frameworks of this kind allow the developer to program to interfaces, i.e., when classes depend on objects of other classes to perform a certain task, have the dependent class relate only to the interface of its dependencies, and not to a specific implementation of that service [14]. Such dependencies are specified in the framework's configuration and, when a certain object is created (which is also performed by the framework), all of its dependencies are automatically injected and satisfied. For instance, in Java EE, Contexts and Dependency Injection for Java[8] is the standard Dependency Injection framework;
- **Object/Relational Mapping frameworks** (e.g., Hibernate): frameworks of this kind offer automatic and transparent persistence of objects to tables of a relational database management system (RDBMS) using meta-data that describe the mapping between both worlds [6]. Such frameworks became very popular (and not

[3] https://struts.apache.org

[4] https://spring.io/projects/spring-framework

[5] https://hibernate.org/orm/

[6] Frameworks are usually configured using specific files or annotations in the classes themselves. Often, sensible defaults help keep such configuration to a minimum.

[7] JSF, http://jcp.org/en/jsr/detail?id=344. Strictly speaking, however, JSF (and the other Java EE standards) are *specifications* that can be implemented by many frameworks (e.g., Mojarra and MyFaces implement JSF). When using a Java EE certified application server, however, this is not explicit to the developer. As such, we will refer to these standard specifications as being frameworks themselves.

[8] CDI, http://jcp.org/en/jsr/detail?id=346

only on WISs) due to what has been called the *object-relational impedance mismatch* [19], i.e., a set of problems that arise due to the combination of the object-oriented paradigm (popular choice for software development) and the relational paradigm (popular choice for data storage). For instance, the Java EE standard for Object/Relational Mapping is the Java Persistence API.[9]

FrameWeb, thus, incorporates the concepts from these frameworks into design models. Initially, it started with two main contributions to the architectural design phase of the software process: (i) the definition of a basic architecture (detailed in Section 3) for better integration with these kinds of frameworks; and (ii) a UML profile (lightweight extension) for the construction of four different design models that bring the concepts used by the frameworks to the models.

Figure 1 illustrates some of the proposed extensions in a **Navigation Model**, which is the *FrameWeb* model that incorporates concepts from Front Controller frameworks. UML stereotypes are used to differentiate Web pages (⟨⟨page⟩⟩), templates (⟨⟨template⟩⟩, used to render Web pages), forms (⟨⟨form⟩⟩) and controller classes (no stereotype). Dependency associations with constraints indicate how the different components interact and, thus, guide the configuration of the Front Controller framework.

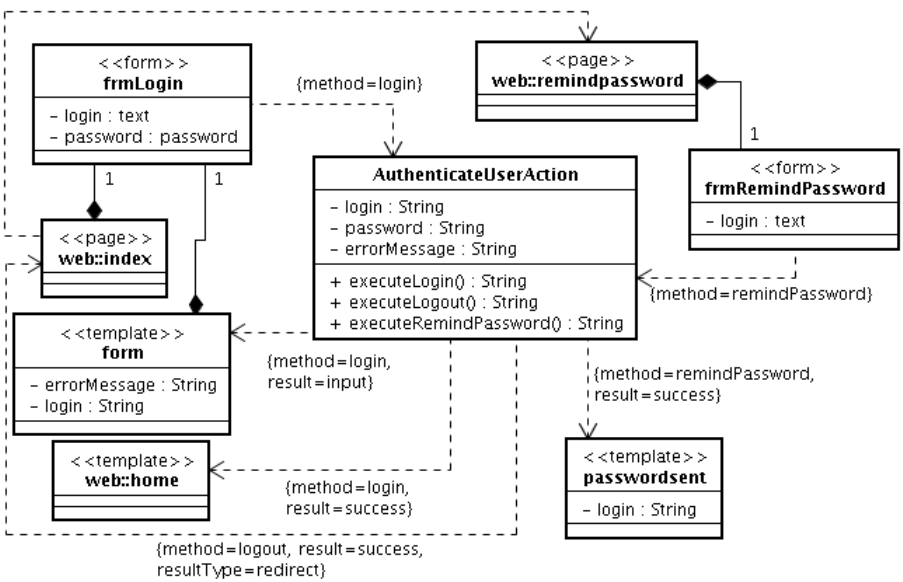

Fig. 1. Navigation Model for log in, log out and remind password features of a WIS [28].

This Navigation Model indicates that the index page of the WIS should have a form frmLogin with login and password fields, whose respective types (text and password)

[9] JPA, http://jcp.org/en/jsr/detail?id=338

refer to visual components from the *tag library* used by the framework. Once the user fills in and submits this form, the framework should respond with the AuthenticateUserAction controller, in particular its executeLogin() method (Struts suggested a standard execute prefix to all controller methods). If this method returned input (presumably due to some issue with the user input), the form template should render an error message related to the login attempt and the user may try again. Instead, if it returns success, the user should be directed to the home template. *Log out* and *remind password* features work analogously. Note that when components have attributes with the same name (e.g., frmLogin.login and AuthetnicateUserAction.login) it means that the framework should take care of this binding (e.g., have the contents of the login field in the form copied to the login attribute of the controller).

FrameWeb also prescribed three other models, all of them based on the UML Class Diagram: the **Domain Model** (later renamed **Entity Model**), the **Persistence Model** and the **Application Model**. It also offered an extension of the method, called *S-FrameWeb*, that prescribed the use of the Object Management Group's (OMG) Ontology Definition Metamodel (ODM) [1] in order to guide the creation of a vocabulary in OWL (W3C's Web Ontology Language)[10] representing the classes from the domain model of the WIS. Further, a component compatible with the Struts framework was implemented in order to output instances of this vocabulary based on the data from the WIS.

The original proposal of *FrameWeb* provided software engineers with interesting tools to organize and document the architecture of a WIS, giving precise instructions to developers on how they should write the code and configure the frameworks. However, it suffered from a few drawbacks:

(i) Its proposed models were based on specific instances of the supported framework categories (namely, Struts, Spring and Hibernate), with no guarantees they would fit appropriately if another set of frameworks (although from the same categories, say JSF, CDI and JPA) were used;
(ii) Although using UML lightweight extensions provides the advantage of allowing designers to use their UML case tool of choice, it does not prevent them from including UML constructs in the models that were not intended by the *FrameWeb* approach, or to use the ones that were intended, but in an inappropriate way;
(iii) Further, general-purpose UML tools will not validate the specific rules proposed by the *FrameWeb* approach for its models, nor provide code generation support for the kind of application that these models represent (e.g., web::index in Figure 1 would be generated as a class, not a Web page).
(iv) Finally, the method is focused on a particular architecture for WISs and the state-of-practice on Web development has evolved, producing different architectures (which use different kinds of frameworks), e.g., Progressive WebApps, Single-Page Applications, the use of microservices and front-end frameworks, etc.

Using Model-Driven Development (MDD) [23] techniques, we thus formalized a domain-specific language for *FrameWeb* models, whose abstract syntax is the metamodel illustrated (at a high-level of abstraction) in Figure 2. We decided to keep the concrete syntax the same as before, as the UML is a language that is familiar to most

[10] https://www.w3.org/TR/2012/REC-owl2-overview-20121211/

software developers. Hence, our meta-model depends on a *Partial UML Meta-model*, which contains the parts of UML that are used by *FrameWeb*. The meta-model is then divided in five components, one for each of the proposed *FrameWeb* models and a *Framework Meta-model* component, which allows us to specify rules and modeling constructs that are specific to the set of frameworks with which a given WIS will be built [21].[11]

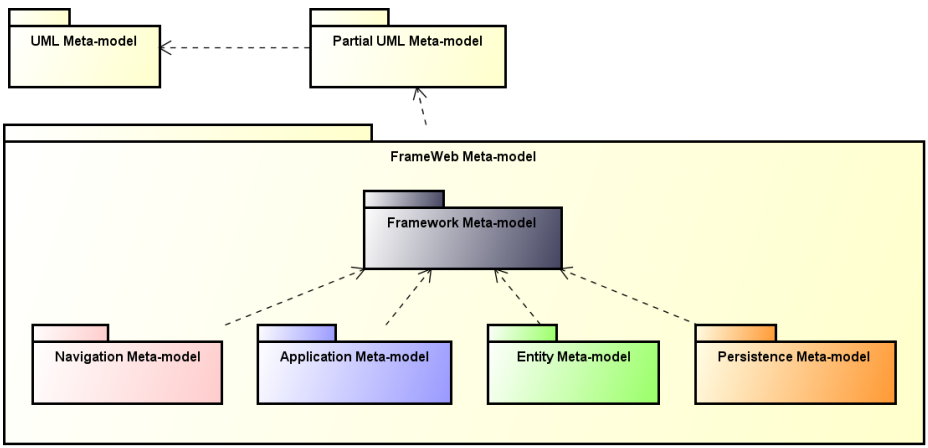

Fig. 2. Overview of the meta-model that defines the *FrameWeb* language [20].

Next, we proposed to replace the *S-FrameWeb* extension with a new one we called *FrameWeb-LD*. The new extension suggested the use of the higher-level, well-founded language OntoUML [17] for ontology capture and formalization and replaced the primitive linked data support that had been specifically built for the Struts framework with the Ontology-Based Data Access (ODBA) tool D2RQ,[12] which serves as a linked data adapter layer over the relational database already in use by the WIS. *FrameWeb-LD* extended the *FrameWeb* meta-model, providing new constructs that allow designers to link concepts of the WIS's domain with external linked data vocabularies, an important step to connect one's data to the Web of Data (or Semantic Web). Further, it provided a tool to help developers generate the operational ontology schema and the ODBA configuration [10].[13]

Once the *FrameWeb* language was defined, it became possible to build tools based on it. Using the infrastructure provided by the Eclipse Modeling Framework (EMF) [29] and the Sirius project [30], a first version of the *FrameWeb Editor* was developed. Figure 3 shows the top-level view of a *FrameWeb* model open in the editor. At the right-hand side a palette provides designers only with the constructs that are allowed in the model being edited; at the bottom, a list of properties allows one to set the attributes

[11] This is the result of the work of Masters student Beatriz Franco Martins Souza.
[12] http://d2rq.org/.
[13] This is the result of the work of Masters student Danillo Ricardo Celino.

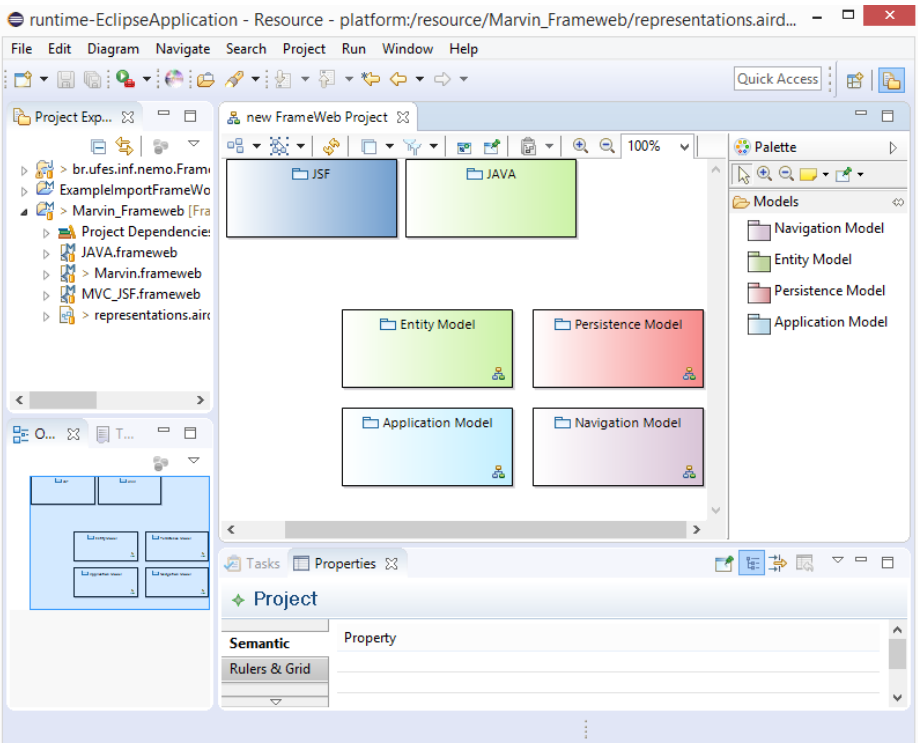

Fig. 3. The first version of the *FrameWeb Editor* [9].

of different model components; at the left-hand side, an overview of the project; in the center, the model being edited, in this case the project overview. Double-clicking one of the model components opens it for edition. Note how two platform-specific settings (Java and JSF) have been imported in the project, showing how the editor supports the extensibility of the method [9].[14]

Another tool built on top of the *FrameWeb* meta-model is the *FrameWeb Code Generator*. This tool reads the model created with the *FrameWeb Editor* and, for each element that represents an artifact of code (e.g., controller classes and Web pages in Navigation Models such as the one in Figure 1), it uses a template for that particular artifact in the specific framework/platform of choice (e.g., a controller class in JSF), filling in the blanks with data extracted from the model (e.g., the controller's name, attributes, methods, etc.) [4, 32]. After the first version of the *FrameWeb Code Generator* was implemented and integrated with the *FrameWeb Editor*, both tools were converted into Eclipse plug-ins with the purpose of integrating them with Web development projects in this IDE. A developer can now turn on the *FrameWeb* facet in their regular Eclipse

[14] This is the result of the work of undergraduate student Silas Louzada Campos.

project, design the models and have the code generated directly into the project structure.[15]

FrameWeb has also evolved in the direction of supporting new categories of frameworks. A feature that is very commonly implemented in WISs using frameworks is that of *authentication & authorization*, or *role-based access control*. To add support for Security Frameworks to *FrameWeb*, its meta-model was modified in order to extend its language syntax, with modifications also implemented in the graphical editor and the code generator. This now allows developers to specify authentication & authorization features in architectural design models using a generic language, generating code to their framework of choice, thanks to *FrameWeb*'s extensibility characteristics [24].[16]

The evolution of *FrameWeb* so far has scratched the surface regarding the aforementioned drawbacks of the method. With respect to not being generic enough (drawback (i)), the definition of the *FrameWeb* language using MDD techniques has improved the method in this sense, but further studies (discussed in Section 4) are necessary to assure that the proposed language is, in fact, generic considering the supported categories of frameworks. On the other hand, such language definition solves drawback (ii), as developers now have a clear specification of how to write a *FrameWeb* model. Related to that, tool support (drawback (iii)) has also improved, but is an ongoing work that needs further development and polishing (part of which probably should take place outside Academia to guarantee a minimum level of quality required by the Industry). Finally, *FrameWeb* is still very much focused on a particular architecture (drawback (iv)). The inclusion of a new category of framework paves the way for further modification of the method's modeling language in order to support further kinds of frameworks and, as a later step, different WIS architectures.

3 Present: Developing WISs with FrameWeb

In this section, we describe what can be accomplished with *FrameWeb* at the moment (December 2019). Both the method and its tools are constantly being developed, thus some of the contributions described in the previous section have yet to be incorporated into the IDE (Integrated Development Environment).

In what follows, we first present what can already be done with the aid of *FrameWeb* tools (Subsection 3.1), then we talk about two features that have not yet been integrated: support for security frameworks (Subsection 3.2) and linked data publication (Subsection 3.3).

3.1 Tool-supported WIS Development

With *FrameWeb*, one can design Web-based Information Systems (WISs) that fit into the architecture shown in Figure 4. Based on the Service Layer pattern [13], this architecture divides the system in three layers: presentation, business and data access.

[15] This is the result of the work of Masters student Nilber Vittorazzi de Almeida and undergraduate students Breno Leite Zupeli and Lucas Ribeiro Mendes Silva.

[16] This is the result of the work of Masters student Rodolfo Costa do Prado.

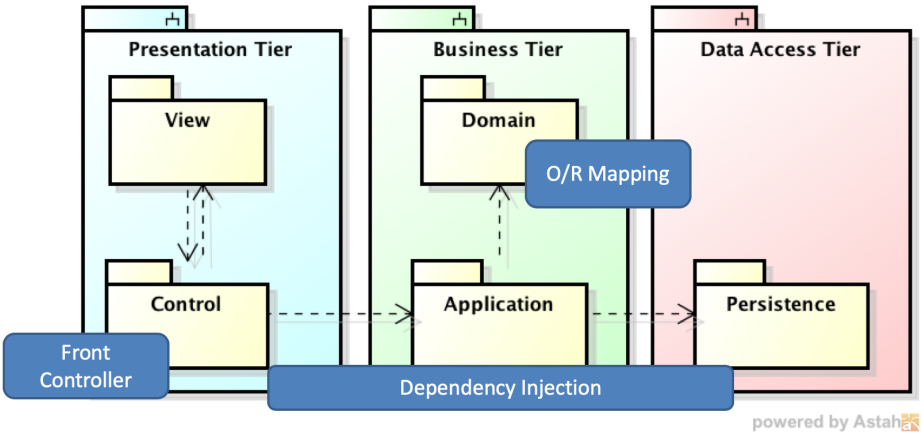

Fig. 4. *FrameWeb*'s supported architecture.

In the Presentation Tier, the View package holds the Web pages, stylesheets, client-side scripts and other user interface artifacts. At the Control package, controller classes handle the requests made by components of the View package, using the infrastructure of the Front Controller framework, and call services offered by the Application package.

In the Business Tier, the Application package contains the classes that are responsible for implementing the system's functionalities, whose dependencies (with Control and Persistence) are wired by the Dependency Injection framework. Application classes manipulate objects from the Domain package and persist them via the Persistence package. The Domain package contains the classes that represent the problem domain, plus annotations that guide the Object/Relational Mapping framework in persisting their data.

Finally, the Data Access Tier consists solely of the Persistence package, which contains the DAO (Data Access Object [5]) classes, responsible for the persistence, i.e., using the Object/Relational Mapping framework services for storing/retrieving objects in/from the relational database. This last tier/package is optional and its responsibilities could be merged into the Application package if desired. However, concentrating all data access operations regarding a given domain class into a single DAO class (which is the essence of the DAO pattern) helps with the maintainability of the code.

To develop a WIS with *FrameWeb*,[17] we should begin by installing its tools, as follows: first, install Java and the Eclipse IDE for Java EE Developers; then, install Sirius through the *Eclipse Marketplace* that can be accessed inside the IDE; finally, using the *Install New Software* feature of Eclipse and pointing it to the *FrameWeb* plug-in update site, install the Code Generator and the Graphical Editor *FrameWeb* tools.

Once the tools are installed, we can create a regular Web project in Eclipse, using the frameworks of our preference. In this section, we will illustrate the use of *FrameWeb*

[17] More detailed instructions can be found in a tutorial that can be accessed through the project's website: https://nemo.inf.ufes.br/projects/frameweb/.

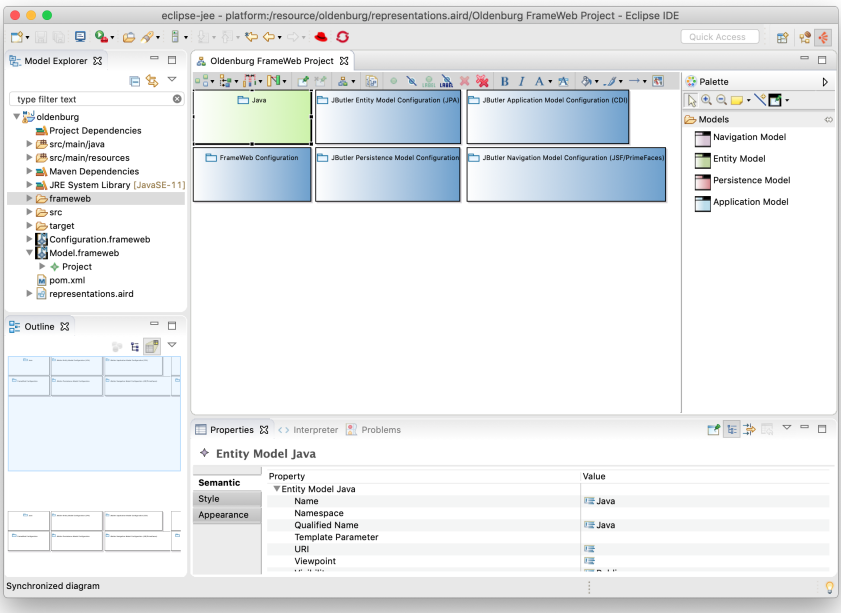

Fig. 5. A project in the Eclipse IDE with the *FrameWeb Editor* facet activated.

with a simplified conference management system,[18] focusing on a single functionality, namely: author registration. Once the project is created, we need to activate the *FrameWeb Editor* facet for that project, which will result in the inclusion of a blank *FrameWeb* model and configuration in that project. Switching to the Sirius perspective, those models can be opened in the *FrameWeb Editor*, as demonstrated in Figure 5.

At the top-right corner of the figure we can see that the Sirius perspective is active. At the left-hand side, the *Model Explorer* view shows our project's files. The files *Configuration.frameweb* and *Model.frameweb* were created when the *FrameWeb* facet was activated. By expanding the latter and double-clicking the *Project* item in the *Model Explorer*, we open the *FrameWeb Editor* in the center of the IDE, as shown. At the right-hand side, the palette allows us to create the four kinds of model the method supports.

The boxes that are already in the model represent *Architecture Definition Files* (divided in *Language* and *Framework Definition Files*) that were imported to the project before the screenshot of Figure 5 was taken. As previously shown in Figure 2, the *FrameWeb* language defines a *Framework Meta-model* component, which allows us to use,

[18] We envisioned a WIS that could be used by professors of Research Methodology classes to simulate a conference-like setting in which students could peer-review each others' papers, like a simplified EasyChair (https://easychair.org). We called it *Oldenburg*, in honor of the philosopher who is seen as the 'father' of modern scientific peer review, according to Wikipedia.

in our models, elements that are specific to the chosen platform/frameworks. At the project's source code repository,[19] we can copy a *Language Definition File* from the languages folder and a set of *Framework Definition Files* from the frameworks folder into our Eclipse project and the *FrameWeb* tools will automatically include them in the model.

In practice, *Language Definition Files* include a list of primitive types and classes from the API of the programming language of choice (e.g., Java has int, double, String, etc.) to be used as types of attributes and parameters in different models. In turn, *Framework Definition Files* include tags from the visual component library of choice (e.g., JSF component library PrimeFaces[20] has dataTable, inputText, password, etc.) to be used in Navigation Models and templates for code generation. For every combination of frameworks we want to use, a set of files should be created and imported. For instance, the frameworks/jbutler folder at the source code repository offers definition files for projects that use the JButler[21] utility classes together with the Java EE standards JSF, CDI, JPA and visual component library PrimeFaces.

We now demonstrate the design of the author registration feature of our running example. Figure 6 shows the Entity Model with the Author class and its object/relational mappings. Most of the mapping relies on sensible default values (e.g., table names are the same as class names, column names are the same as attribute names, column types are inferred, etc.), but string size and date precision are explicitly specified. The diagram does not show any ID or versioning (optimistic locking) attribute because they are inherited from a JButler utility class.

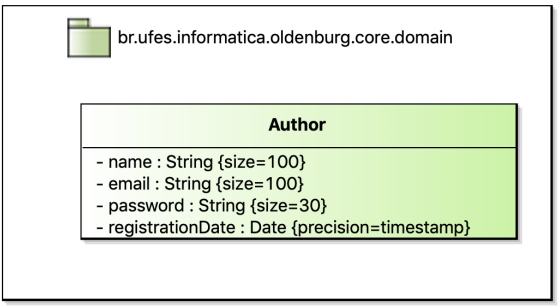

Fig. 6. *FrameWeb* Entity Model for our running example.

The persistence of objects of the Author class is handled by the AuthorDAO, shown in Figure 7. Most of the basic persistence operations (e.g., retrieve all, retrieve by id, save, delete, etc.) are inherited from a JButler utility class, therefore are not shown. The DAO is divided into interface and implementation, and the semantics of the *FrameWeb* language states that the former should declare the signatures of all public methods of the

[19] https://github.com/nemo-ufes/FrameWeb
[20] http://primefaces.org
[21] https://github.com/dwws-ufes/jbutler

latter, allowing us to use a simplified notation for the interface. By relying on JButler for the basic operations, the DAO only shows a method that is specific to our WIS: retrieving an author given her e-mail, required to check if someone is registering with an e-mail that has already been used.

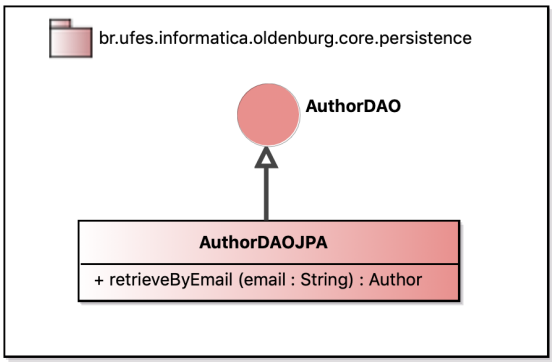

Fig. 7. *FrameWeb* Persistence Model for our running example.

The author registration feature is represented in the Navigation Model of Figure 8. Web pages in the core/registration/ path are used in this scenario, starting with the index page, which contains the registrationForm with inputText and password fields (from PrimeFaces). Once the form is submitted, the Front Controller copies the contents of the fields to attributes of RegistrationController (note that the fields with author. prefix are copied to internal attributes of the author object in the controller) and the register() method is called. Depending on the outcome, the user may be presented the success or the emailinuse pages, which require that the Front Controller bring some data (author.name and author.email respectively) back to the view.

Finally, the Application Model shown in Figure 9 completes the architecture with the RegistrationService which, like the DAO before, is divided in interface and implementation. The RegistrationController from the Navigation model depends on this service which, in turn, depends on the AuthorDAO to properly perform its register() method. The Dependency Injection framework will satisfy both dependencies when needed.

Once the models have been created, we can generate code for it. When doing it for the first time, we should click on the FrameWeb Configuration item of our project in the editor (as shown in Figure 5) and set a few properties, such the as *Class* and *Page Extensions* (e.g., .java and .xhtml), the *Framework Definition Path* — where the code generation templates are located — and the *Src* and *View Paths*, which is where classes and Web pages, respectively, will be generated. After that, right-clicking any blank space in the *FrameWeb Editor* and selecting *Generate Source Code* will create all the classes and Web pages from our models right into the structure of our project in Eclipse itself.

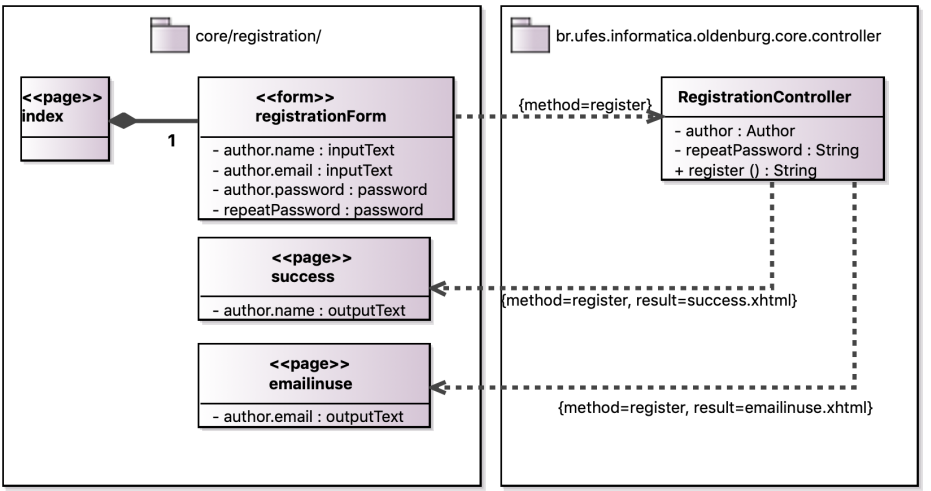

Fig. 8. *FrameWeb* Navigation Model for our running example.

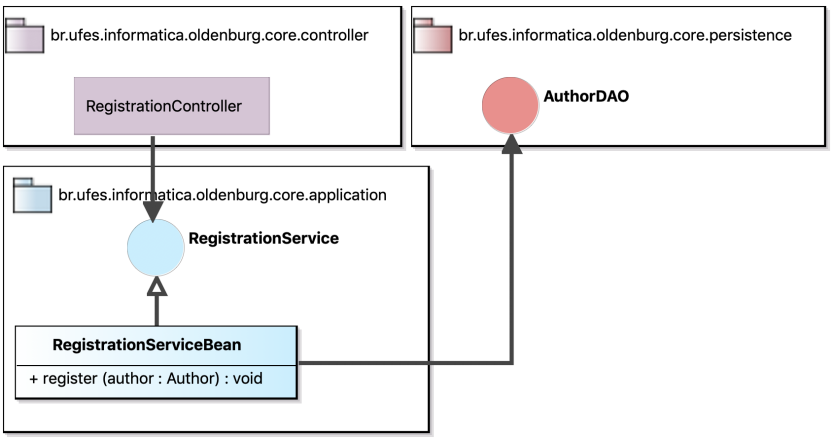

Fig. 9. *FrameWeb* Application Model for our running example.

Listing 1 shows one of the templates used in our running example. Code between {{ and }} are replaced by elements extracted from the Navigation Model. Further, {% and %} can be used to insert control flow directives like loops and conditionals in the template. The rest of it is standard Java code that defines a class that extends JSFController from JButler, is annotated with @Model to be referred to in Web pages, has association with service classes annotated with @EJB (dependency injection annotation), attributes defined with their respective accessor/mutator (get/set) methods and the skeletons (stubs) for other methods the controller might have. The result of applying such template to the Navigation Model of Figure 8 (plus running *Organize Imports* and *Format* features from Eclipse) is shown in Listing 2.

We can see that, apart from comments, we only need to implement the register() method to complete this particular artifact of code. Although further experiments are in order, recent tests with a different set of frameworks showed that the *FrameWeb Code Generator* generated between 68% and 94% of the lines of code of a simple WIS when compared with the final solution, after manual editing [26]. We consider this a good result in terms of cost (of modeling) vs. benefit (of less code to write).

3.2 Role-based Access Control

One recent *FrameWeb* extension that has not yet made its way to the Eclipse plug-in is the support for security frameworks that implement Role-based Access Control (RBAC). This extension of the *FrameWeb* language (i.e., meta-model) allows developers to specify authentication & authorization features in Entity, Application and Navigation models using a generic language and generating code to a security framework of choice.

Role-Based Access Control (RBAC) [12] is a basic model for authorization inside an application that is founded on the separation between *actors* and the *actions* available to them in the system. This separation is made by adding the concept of *roles*. In RBAC, any *permission* to run an action inside the application can only be associated with a role. Actors do not acquire permissions directly, instead they are given roles that aggregate a collection of permissions. With this configuration, the assignment of permitted actions to users inside a system is made with both simplicity and flexibility [24].

A Security Framework provides as reusable infrastructure a set of features concerned with the security of an application, such as authentication, authorization, cryptography, session management, etc. The proposed *FrameWeb* extension focuses on *authentication*, i.e., certifying that a user is who she says she is; and *authorization*, i.e., verify if the user has the right to perform an action, given her authenticated credentials. *FrameWeb* models can now define: (a) the domain classes that represent users, roles and permissions; (b) aspects of the Web pages and forms that will trigger the authentication; (c) which permissions are required by each service method or entire classes, to implement authorization.

Figure 10 shows an Entity Model that defines users (⟨⟨AuthUser⟩⟩ stereotype), roles (⟨⟨AuthRole⟩⟩) and permissions (⟨⟨AuthPermission⟩⟩) in a generic way, i.e., they could be used in or adapted to any WISs. For instance, to use them in *Oldenburg*, we could connect User to Author or have the latter annotated with ⟨⟨AuthUser⟩⟩ instead, use the author's email as the ⟨⟨AuthUserName⟩⟩, and so on.

Listing 1. Template for a controller class.

```
package {{ package.Name }};

import javax.ejb.EJB;
import javax.enterprise.inject.*;
import br.ufes.inf.nemo.jbutler.ejb.controller.JSFController;

/** TODO: generated by FrameWeb. Should be documented. */
@Model
public class {{ class.Name }} extends JSFController {
  /** Serialization id (using default value, change if necessary). */
  private static final long serialVersionUID = 1L;

  {% for association in associations %}
  /** TODO: generated by FrameWeb. Should be documented. */
  @EJB
  private {{ association.TargetMember.Type.Name }} {{ association.
       TargetMember.Type.Name | lower_first }};
  {% endfor %}

  {% for attribute in attributes %}
  /** TODO: generated by FrameWeb. Should be documented. */
  private {{ attribute.Type.Name }} {{ attribute.Name }};
  {% endfor %}

  {% for method in methods %}
  /** TODO: generated by FrameWeb. Should be documented. */
  {{ method.Visibility.Name }} {% if method.MethodType is null %}void{% else
       %}{{ method.MethodType.Name }}{% endif %} {{ method.Name }}({% for
       parameter in method.OwnedParameters %}{{ parameter.Type.Name }} {{
       parameter.Name }}{% if loop.last == false %}, {% endif %}{% endfor %})
       {
    // FIXME: auto-generated method stub
    return{% if method.MethodType is not null %} null{% endif %};
  }
  {% endfor %}

  {% for attribute in attributes %}
  /** Getter for {{ attribute.Name }}. */
  public {{ attribute.Type.Name }} get{{ attribute.Name | capitalize }}() {
    return {{ attribute.Name }};
  }

  /** Setter for {{ attribute.Name }}. */
  public void set{{ attribute.Name | capitalize }}({{ attribute.Type.Name }}
       {{ attribute.Name }}) {
    this.{{ attribute.Name }} = {{ attribute.Name }};
  }
  {% endfor %}
}
```

Listing 2. Generated code for a controller class.

```
package br.ufes.informatica.oldenburg.core.controller;

import javax.ejb.EJB;
import javax.enterprise.inject.Model;

import br.ufes.inf.nemo.jbutler.ejb.controller.JSFController;
import br.ufes.informatica.oldenburg.core.application.RegistrationService;
import br.ufes.informatica.oldenburg.core.domain.Author;

/** TODO: generated by FrameWeb. Should be documented. */
@Model
public class RegistrationController extends JSFController {
  /** Serialization id (using default value, change if necessary). */
  private static final long serialVersionUID = 1L;

  /** TODO: generated by FrameWeb. Should be documented. */
  @EJB
  private RegistrationService registrationService;

  /** TODO: generated by FrameWeb. Should be documented. */
  private Author author;

  /** TODO: generated by FrameWeb. Should be documented. */
  private String repeatPassword;

  /** TODO: generated by FrameWeb. Should be documented. */
  public String register() {
    // FIXME: auto-generated method stub
    return null;
  }

  /** Getter for author. */
  public Author getAuthor() {
    return author;
  }

  /** Setter for author. */
  public void setAuthor(Author author) {
    this.author = author;
  }

  /** Getter for repeatPassword. */
  public String getRepeatPassword() {
    return repeatPassword;
  }

  /** Setter for repeatPassword. */
  public void setRepeatPassword(String repeatPassword) {
    this.repeatPassword = repeatPassword;
  }

}
```

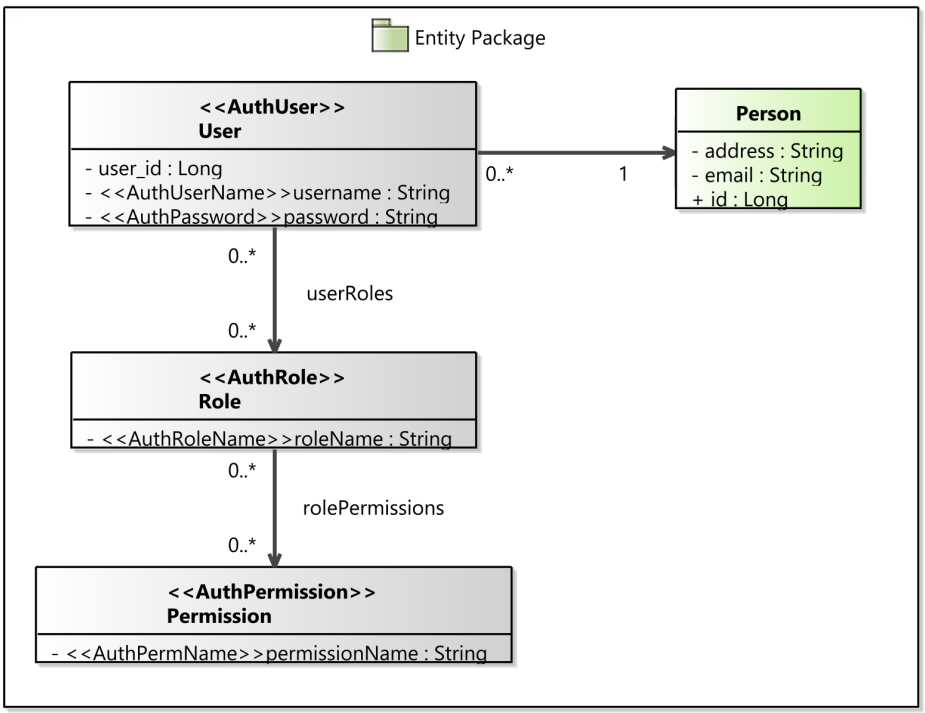

Fig. 10. *FrameWeb* Entity Model with RBAC features [24].

Figure 11 shows a Navigation Model that specifies how authentication will be implemented. The model represents the login page (⟨⟨authPage⟩⟩ stereotype), the form with fields for user credentials (⟨⟨authForm⟩⟩), as well as processing (⟨⟨AuthMethod⟩⟩), success (⟨⟨AuthSuccessUrl⟩⟩) and failure (⟨⟨AuthFailureUrl⟩⟩) URLs. This information will guide the security framework in performing authentication. Note that the processing URL actually refers to a method of the controller class so the security framework will use the URL that activates this method as the processing URL.

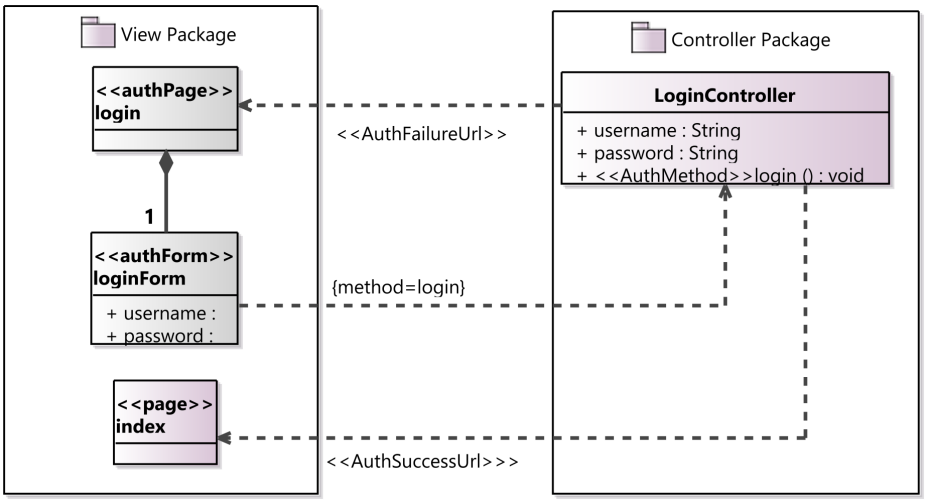

Fig. 11. *FrameWeb* Navigation Model with RBAC features [24].

Finally, Figure 12 shows an Application Model with authorization settings. Permissions are expressed using UML constraints as concrete syntax. Service class PersonServiceImp requires a permission named PERM_PERSON to be accessed. Service methods delete() and update() further require permissions named PERM_PERSON_DEL and PERM_PERSON_UP, respectively.

The RBAC extension for the *FrameWeb* method has been implemented, not only regarding the modifications in the meta-model but also with respect to tool support. Therefore, we can produce the models with security features using the *FrameWeb Editor*, as demonstrated by figures 10–12, and generate code with the *FrameWeb Code Generator*, as shown in [24]. However, this has been implemented on a separate code base,[22] and, thus, needs to be carefully merged into the code of the *FrameWeb* Eclipse plug-ins.

[22] https://github.com/Rodolfocostapr/Experimento-Frameweb-Sec

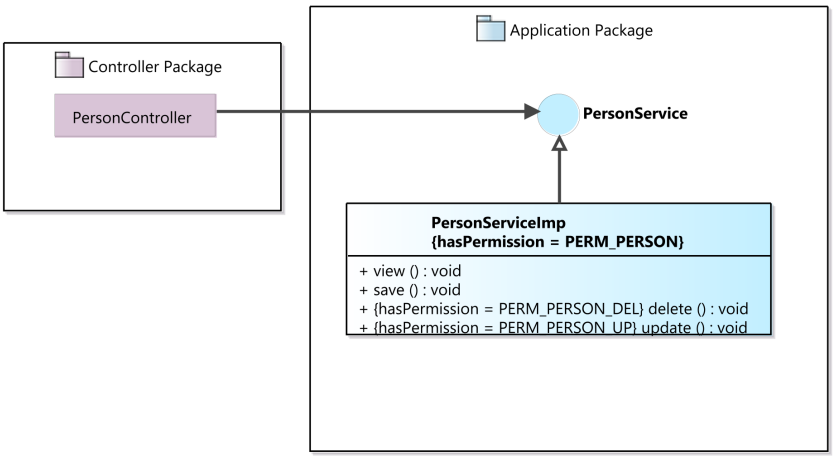

Fig. 12. *FrameWeb* Application Model with RBAC features [24].

3.3 Linked Data Support

Another extension of the *FrameWeb* language/meta-model that has not yet been incorporated into the *FrameWeb* tools is *FrameWeb-LD* [10]. Such extension allows developers to specify how the data from the WIS relates to well-known vocabularies from the Semantic Web, with the purpose of integrating them into the Web of Data [18]. Figure 13 shows an example of Entity Model with linked data mappings added to the domain classes.

The figure illustrates a system that manages researchers from a postgraduate program and their publications in order to produce reports on their research productivity. Although not shown in the diagram, vocabulary identifiers (IDs) are associated to their respective URIs, e.g., foaf is associated with http://xmlns.com/foaf/0.1/ (Friend of a Friend vocabulary) and dblp with http://dblp.rkbexplorer.com/id/ (DBLP Computer Science Bibliography dataset).

Then, concepts from external vocabularies are shown using their vocabulary IDs as UML namespace (e.g., foaf::Person). They can be related to classes from the WIS via UML associations, navigable towards the external class, representing an RDF triple: the class from the WIS is the subject, the external one is the object and the predicate is specified as a constraint. In the example, Researcher is owl:equivalentClass to dblp:Person. As a syntactic sugar, the rdfs:subClassOf relation between a class from the WIS and one from an external vocabulary can be represented by a UML inheritance association. In the example, Researcher is rdfs:subClassOf foaf:Person.

Triples concerning attributes of classes are represented using constraints in the form *predicate=object*. In the example, Researcher.name is owl:equivalentProperty to dblp:primaryFullPersonName. Constraints in associations between classes from our WIS establish relations among object properties (in the same way constraints in attributes establish relations among data type properties). In the example, the association between Publication and Venue is rdfs:subPropertyOf dblp:publicationType. Last, but

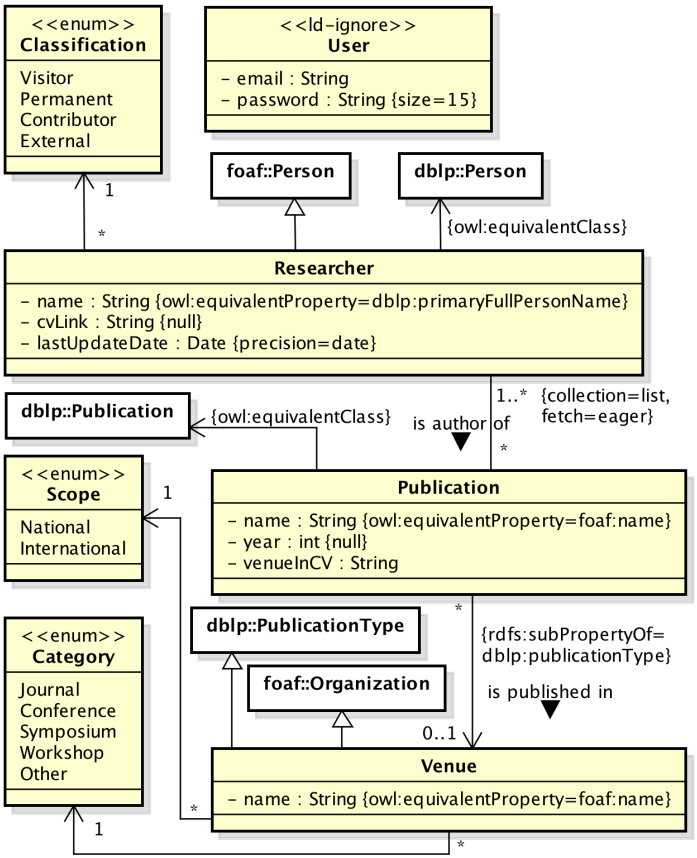

Fig. 13. *FrameWeb-LD* Entity Model with linked data mappings [10].

not least, data from all classes are to be published as linked data, unless the ⟨⟨ld-ignore⟩⟩ stereotype is used (either to exclude specific attributes or entire classes). In the example, the User class is excluded from the linked data set to be published.

Once all the mappings have been included in the Entity Model, tool support[23] can aid developers in producing code for an Ontology-based Data Access (ODBA) solution such as D2RQ, which creates a layer on top of the relational database and offers triple-store features (derreferenceable URIs for navigation, a SPARQL endpoint for querying, etc.) based on a semi-automatic conversion from the database schema to RDF. Listings 3 and 4 show excerpts from the OWL operational ontology and D2RQ mapping generated by *FrameWeb-LD*'s tool support. Some of the mappings of Figure 13 can be identified in these generated artifacts.

[23] The tool is called ReMaT and is available in a stale branch in *FrameWeb*'s source code repository: https://github.com/nemo-ufes/FrameWeb/tree/breno/

Listing 3. Excerpt from operational ontology in OWL generated by *FrameWeb-LD*'s tool support [10].

```
<owl:Class rdf:about="http://dev.nemo.inf.ufes.br/owl/c2d.owl#Publication">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
       Publication</rdfs:label>
  <rdfs:subClassOf rdf:resource="http://dblp.uni-trier.de/rdf/schema
       -2015-01-26#Publication"/>
</owl:Class>
<owl:Class rdf:about="http://dev.nemo.inf.ufes.br/owl/c2d.owl#Venue">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Venue</
       rdfs:label>
  <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Organization"/>
  <rdfs:subClassOf rdf:resource="http://dblp.uni-trier.de/rdf/schema
       -2015-01-26#PublicationType"/>
</owl:Class>
<owl:ObjectProperty rdf:about="http://dev.nemo.inf.ufes.br/owl/c2d.owl#
     isPublishedIn">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
       isPublishedIn</rdfs:label>
  <rdfs:domain rdf:resource="http://dev.nemo.inf.ufes.br/owl/c2d.owl#
       Publication"/>
  <rdfs:range rdf:resource="http://dev.nemo.inf.ufes.br/owl/c2d.owl#Venue"/>
  <rdfs:subPropertyOf rdf:resource="http://dblp.uni-trier.de/rdf/schema
       -2015-01-26#publicationType"/>
</owl:ObjectProperty>
```

Listing 4. Excerpt from the relational-to-RDF mapping file generated by D2RQ and *FrameWeb-LD*'s tool support [10].

```
@prefix c2d: <http://dev.nemo.inf.ufes.br/owl/c2d.owl#>

# Table Researcher
map:Researcher a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:class c2d:Researcher;
  d2rq:classDefinitionLabel "Researcher";
  rdfs:subClassOf foaf:Person;
  owl:equivalentClass dblp:Person;
  .
map:Researcher_name a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Researcher;
  d2rq:property vocab:Researcher_name;
  d2rq:propertyDefinitionLabel "Researcher name";
  owl:equivalentProperty dblp:primaryFullPersonName;
  d2rq:column "Researcher.name";
  .
```

4 Future: where is FrameWeb Going?

FrameWeb is an ongoing research project with undergraduate and graduate students working on different aspects of the proposal. In the previous sections of this paper, we have already mentioned limitations of the approach that need to be addressed in future work. Section 3, for instance, mentioned the support for security frameworks and linked data not being incorporated into the *FrameWeb* Eclipse plug-in, which is on our short-term plans for the future.

Section 2 discussed drawbacks that have motivated recent proposals for the evolution of *FrameWeb*: (i) the language not being generic enough; (ii) not having a precise language specification; (iii) lack of tool support; and (iv) being tailored to a specific

architecture and particular framework categories. As discussed, the evolution of *FrameWeb* only partially addresses these challenges and, thus, there are many opportunities for future work to be taken from these limitations.

Regarding the ***FrameWeb* language generality** and its **precise specification**, the definition of the meta-model and the application of the method in other platforms and with different frameworks have contributed towards these aspects. However, to properly understand each category of framework supported by the method, a more systematic study of the different frameworks of each category is required.

As such, we intend to build ontologies for each supported category, using an ontology engineering approach to try and make sure such reference model properly represents a consensus among the mostly used frameworks. Then, the *FrameWeb* meta-model can be reviewed and adjusted based on the ontology, possibly leading to changes in the *FrameWeb* language.

This is an ongoing effort, conducted under a separate research project that aims at building a *Software Frameworks Ontology Network* (SFWON).[24] The network already includes the *Object/Relational Mapping Ontology* (ORM-O) [31], which was built based on the *Relational Database System Ontology* (RDBS-O) [2] and the *Object-Oriented Code Ontology* (OOC-O) [3], both part of SFWON.

Regarding the **lack of tool support**, the development of the *FrameWeb Editor* and the *FrameWeb Code Generator* is an ongoing effort, with many ideas for new developments, for instance:

- The use of the *FrameWeb* tools in the context of Web Development and Semantic Web courses offered in our university have identified bugs and improvements in the tools' usability, reliability, etc.[25] that need to be fixed so the tools can be put to further use and test;
- The editor currently has no support for the creation of Architecture Definition Files, making it harder for organizations/developers to include support for their platform/frameworks of choice in *FrameWeb*;
- Being based on the Eclipse IDE, the tools integrate best with projects for programming languages and platforms that are supported by this IDE and its plug-ins. Support for different development environments could be offered;
- Modern Rapid Application Development tools (e.g., JHipster[26]) automatically generate considerably more code, especially regarding basic Create, Retrieve, Update and Delete (CRUD) functionalities. These tools, however, are not as extensible (i.e., able to support different frameworks) as *FrameWeb*, but our tools need support for easier generation of CRUD features and other artifacts that are common to WISs;
- Currently, code generation works only in one direction, thus generating code again overwrites any changes that might have been manually performed in previously generated files. Support for preserving manual changes or even more advanced reverse engineering features could be added.

[24] https://nemo.inf.ufes.br/projects/sfwon/
[25] https://github.com/nemo-ufes/FrameWeb/issues
[26] https://www.jhipster.tech

Regarding ***FrameWeb*'s supported architecture and framework categories** further efforts similar to the one presented in Section 3.2 to include support for new categories of frameworks are in order. Such efforts could include the definition of an ontology for the new category of frameworks or postpone the creation of the ontology as a later step (e.g., the support for security frameworks was proposed based on the most used frameworks in the Java platform [24]). As new types of frameworks are included in the method, new architectures can also be proposed.

Finally, we need to perform more extensive and systematic experiments in order to evaluate *FrameWeb* in all of its aspects. Although each proposal conducted their own validation through proofs of concept and small experiments, we do not yet have properly evaluated the entire *FrameWeb* proposal in terms of its usefulness, ease of use, efficacy, etc.

With the *FrameWeb* project, I intend to continue to honor professor Ricardo de Almeida Falbo by carrying on a research agenda that he helped create — more accurately put, would not exist without him — for many years to come. Hopefully, this project will continue to contribute to the qualification of students involved in it, a legacy that Falbo should be proud of.

5 Personal Notes

I met Ricardo as my professor of Requirements Engineering during an undergraduate course in Computer Science at UFES in 2002, the year in which he also became my advisor on a "Scientific Initiation" (undergraduate research scholarship) project. He also supervised my (research-oriented) undergraduate final project in 2004 and accepted me as a Masters student in 2005, continuing to supervise me until 2007. After my PhD abroad, I came back to Brazil and became a professor at the Computer Science Department of UFES in 2013, thus Falbo became my colleague until his retirement in 2019. I am proud to say that during this entire time Ricardo has been, and continues to be, a great friend.

The topic of this paper, the *FrameWeb* method, was born during my Masters course, under the supervision of Falbo [27]. However, the topic of Web Engineering was not connected to my previous undergraduate research projects under his supervision, nor was it one of the particular areas that Ricardo was focusing his research. Instead, it was motivated by my previous experiences in software development projects for the Web and Falbo decided to accept it as the research topic of one of his supervised Masters students. As a professor now myself, I see how altruistic this gesture was at the time and it is fair to say that this contributed to the researcher I came to be, which I think was Ricardo's intention all along (but we will have to ask him).

Professor Ricardo, thank you for all that you have done for me. You are in great part responsible for (as modest as they may be) my academic accomplishments. You are definitely one of my role models and I hope (and work hard) to be to my supervised students as good an advisor as you were to me. Congratulations on a successful career of inspiring people like me.

References

1. OMG: Ontology Definition Metamodel (ODM) Specification, v. 1.1 (formal/14-09-02), http://www.omg.org/spec/ODM/1.1/ (2014)
2. de Aguiar, C.Z., Falbo, R.A., Souza, V.E.S.: Ontological Representation of Relational Databases. In: Proc. of the 11th Seminar on Ontology Research in Brazil (ONTOBRAS 2018). pp. 140–151. CEUR, São Paulo, SP, Brazil (2018)
3. de Aguiar, C.Z., Falbo, R.d.A., Souza, V.E.S.: OOC-O: A Reference Ontology on Object-Oriented Code. In: Proc. of the 38th International Conference on Conceptual Modeling (ER 2019). pp. 13–27. Springer, Salvador, BA, Brazil (2019)
4. de Almeida, N.V., Campos, S.L., Souza, V.E.S.: A Model-Driven Approach for Code Generation forWeb-based Information Systems Built with Frameworks. In: Proc. of the 23rd Brazilian Symposium on Multimedia and the Web (WebMedia 2017). pp. 245–252. ACM, Gramado, RS, Brazil (oct 2017)
5. Alur, D., Crupi, J., Malks, D.: Core J2EE Patterns: Best Practices and Design Strategies. Prentice Hall / Sun Microsystems Press, 2nd edn. (2003)
6. Bauer, C., King, G.: Hibernate in Action. Manning, 1 edn. (2004)
7. Berners-Lee, T.: Linked Data - Design Issues,http://www.w3.org/DesignIssues/LinkedData.html (last access: May 7th, 2015) (2006)
8. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284**(5), 34–43 (2001)
9. Campos, S.L., Souza, V.E.S.: FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb. In: Anais do 16º Workshop de Ferramentas e Aplicações, 23º Simpósio Brasileiro de Sistemas Multimedia e Web (WFA/WebMedia 2017). pp. 199–203. SBC, Gramado, RS, Brazil (oct 2017)
10. Celino, D.R., Reis, L.V., Martins, B.F., Souza, V.E.S.: A Framework-based Approach for the Integration of Web-based Information Systems on the Semantic Web. In: Proc. of the 22nd Brazilian Symposium on Multimedia and the Web. pp. 231–238. ACM (nov 2016)
11. DeMichiel, L., Shannon, B.: JSR 342: Java(TM) Platform, Enterprise Edition 7 (Java EE 7) Specification, https://jcp.org/en/jsr/detail?id=342 (last access: April 29th, 2015) (2013)
12. Ferraiolo, D., Cugini, J., Kuhn, D.R.: Role-based access control (RBAC): Features and motivations. In: Proc. of 11th Annual Computer Security Application Conference. pp. 241–248 (1995)
13. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley, 1 edn. (2002)
14. Fowler, M.: Inversion of Control Containers and the Dependency Injection pattern, http://www.martinfowler.com/articles/injection.html (last access: September 29th, 2016) (2004)
15. Frakes, W.B., Kang, K.: Software reuse research: Status and future. IEEE Transactions on Software Engineering **31**(7), 529–536 (2005)
16. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: No TitleDesign Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1 edn. (1994)
17. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Phd thesis, University of Twente, The Netherlands (2005)
18. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool Publishers, 1 edn. (2011)
19. Ireland, C., Bowers, D., Newton, M., Waugh, K.: A classification of object-relational impedance mismatch. In: 2009 First International Conference on Advances in Databases, Knowledge, and Data Applications. pp. 36–43. IEEE (2009)

20. Martins, B.F.: Uma abordagem dirigida a modelos para o projeto de Sistemas de Informação Web com base no método FrameWeb. Ph.D. thesis, Dissertação de Mestrado, Universidade Federal do Espírito Santo (2016)
21. Martins, B.F., Souza, V.E.S.: A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In: Proc. of the 21st Brazilian Symposium on Multimedia and the Web. pp. 41–48. ACM (2015)
22. Murugesan, S., Deshpande, Y., Hansen, S., Ginige, A.: Web Engineering: a New Discipline for Development of Web-Based Systems. In: Murugesan, S., Deshpande, Y. (eds.) Web Engineering - Managing Diversity and Complexity of Web Application Development, chap. 1, pp. 3–13. Springer (2001)
23. Pastor, O., España, S., Panach, J.I., Aquino, N.: Model-driven development. Informatik-Spektrum **31**(5), 394–407 (2008)
24. do Prado, R.C., Souza, V.E.S.: Securing FrameWeb: Supporting Role-based Access Control in a Framework-based Design Method for Web Engineering. In: Proc. of the 24th Brazilian Symposium on Multimedia and the Web (WebMedia '18). pp. 213–220. ACM, Salvador, BA, Brazil (2018)
25. Schmidt, D., Stal, M., Rohnert, H., Buschmann, F.: Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects. Wiley (2013)
26. Silva, L.R.M.: Integração do Editor de Modelos de FrameWeb à IDE Eclipse. Tech. rep., Relatório Final de Pesquisa, Programa Institucional de Iniciação Científica, Universidade Federal do Espírito Santo (2019)
27. Souza, V.E.S.: FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web. Tech. rep., Universidade Federal do Espírito Santo (2007)
28. Souza, V.E.S., Falbo, R.A., Guizzardi, G.: Designing Web Information Systems for a Framework-based Construction. In: Halpin, T., Proper, E., Krogstie, J. (eds.) Innovations in Information Systems Modeling: Methods and Best Practices, chap. 11, pp. 203–237. IGI Global, 1 edn. (2009)
29. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF - Eclipse Modeling Framework. Addison-Wesley, 2nd editio edn. (2008)
30. Viyovic, V., Maksimovic, M., Perisic, B.: Sirius: A rapid development of DSM graphical editor. In: Intelligent Engineering Systems (INES), 2014 18th International Conference on. pp. 233–238. IEEE (2014)
31. Zanetti, F.L., de Aguiar, C.Z., Souza, V.E.S.: Representação Ontológica de Frameworks de Mapeamento Objeto/Relacional. In: Proc. of the 12th Seminar on Ontology Research in Brazil (ONTOBRAS 2019). CEUR, Porto Alegre, RS, Brasil (2019)
32. Zupeli, B.L., Souza, V.E.S.: Integração de um Gerador de Código ao FrameWeb Editor. In: Anais Estendidos do 24º Simpósio Brasileiro de Sistemas Multimedia e Web - Workshop de Ferramentas e Aplicações (WFA/WebMedia 2018). pp. 109–113. SBC, Salvador, BA, Brazil (2018)

Carta para Ricardo Falbo

Érica Ferreira de Souza

Departamento Acadêmico de Computação
Programa de Pós-Graduação em Informática (PPGI)
Universidade Tecnológica Federal do Paraná, Cornélio Procópio, PR, Brasil
ericasouza@utfpr.edu.br

O prof. Falbo teve influência na vida de muitos alunos. Como meu coorientador, também influenciou sobremaneira no meu doutorado. Com ele, tive um aprendizado enorme durante o desenvolvimento do meu trabalho e isso tornou possível a construção da minha carreira na área acadêmica.

Lembro do dia da minha defesa de proposta de doutorado. Naquele momento, ele era membro externo da banca examinadora. Durante a arguição, de forma inesperada, fechou a cópia da proposta que estava em suas mãos e disse que ia conversar sobre o projeto sem ficar olhando para o documento. Então, começou a recomendar o que eu deveria, de fato, fazer para alcançar todos os objetivos do meu projeto. Sempre muito educado em suas falas, me senti acolhida e passei a ter confiança no que estava me propondo a fazer. Foi quando meu o orientador, prof. Dr. Nandamudi L. Vijaykumar, carinhosamente conhecido como Vijay, o convidou para participar do projeto e, a partir de então, o prof. Falbo passou a ser meu coorientador de doutorado.

A orientação foi quase toda à distância, pois eu estava no Instituto Nacional de Pesquisas Especiais (INPE), em São José dos Campos/SP e o prof. Falbo na Universidade Federal do Espírito Santo (UFES), em Vitória/ES. Foram inúmeros e-mails trocados e reuniões por Skype. Tive a oportunidade de estar na UFES algumas poucas vezes para reuniões presenciais. Na UFES, conheci outros professores e alunos (alguns hoje ex alunos e colegas de profissão) com os quais até hoje mantenho contato.

No começo foi bem desafiador entender tudo sobre Ontologias e Gestão do Conhecimento. Convergir tudo aquilo para Teste de Software, que era minha principal área de pesquisa, não foi nada fácil. O prof. Falbo foi muito paciente e motivador em momentos de desespero. O tempo foi passando e tudo começou a fazer sentido. Começamos a produzir.

Nosso esquema de trabalho era muito interessante. Eu me sentia bem trabalhando a noite, por ser mais silencioso. Então, decidi trocar o dia pela noite. Eu trabalhava na tese até às 4h00 da manhã e enviava tudo o que produzia. Já de manhã, bem cedinho, o prof. Falbo revisava, provavelmente enquanto eu dormia. Quando eu acordava, as anotações da revisão já estavam em minha caixa postal. Eu, implementava as correções e encaminhava para o meu orientador (prof. Vijay) fazer a sua revisão. E assim, iniciava uma nova rodada de produção e revisões. Esse processo perdurou por um bom tempo e se mostrou muito eficiente.

No final do doutorado tive momentos bem difíceis. Perdi meu pai na reta final. Já não tinha mais cabeça para nada. Foi quando tivemos um artigo internacional aceito.

Uma alegria imensa. No entanto, eu não havia solicitado, para a fomentadora que pagava minha bolsa de estudos, o recurso financeiro para pagar a inscrição do evento antes da renovação da bolsa. Dessa forma, teria que esperar passar o período de renovação de bolsa para o dinheiro ser liberado e, com isso, o prazo para inscrição já teria passado. Fiquei aflita com tudo aquilo, pois eu não tinha dinheiro para cobrir essa despesa até o dinheiro da bolsa ser liberado. Naquele momento, ambos (profs. Falbo e Vijay) se prontificaram e arcaram com os gastos da inscrição e me tranquilizaram em relação ao momento difícil que eu estava passando. Para mim, aqueles gestos foram muito importantes.

O dia da defesa da tese chegou. Confesso que não lembro muito bem de alguns detalhes desse dia. Minha ansiedade falou mais alto. Mas lembro do prof. Falbo, ao final, contando toda nossa trajetória para desenvolver o projeto. Foram muitas batalhas e uma grande vitória.

Dias depois me vi passando de aluna para professora em uma Universidade Federal, orientando projetos de graduação, pós-graduação e investindo em pesquisas, fruto do meu doutorado. Foi uma troca de papeis bem rápida. E mais uma vez o prof. Falbo deu todo o apoio. Desde então, tivemos muitos trabalhos em parceria. Ele contribuiu e ainda contribui em muitos dos trabalhos que desenvolvo com meus alunos e outros parceiros de pesquisa.

Ao pensar no prof. Falbo, me vem a lembrança de uma pessoa motivadora, sincera e humana. Estou muito feliz e sinto-me extremamente honrada por ter sido convidada a escrever esta carta, pois vejo como uma oportunidade de agradecer pelo tanto que ele contribuiu para minha vida profissional.

Prof. Falbo, obrigada por toda a paciência, empenho e visão prática com que sempre me orientou, desde o início do doutorado. Muito obrigada por me mostrar o certo sem nunca me desmotivar.

A Influência do Professor Ricardo Falbo na Formação de Docentes do Instituto Federal do Espírito Santo

Julio Cesar Nardi[1], Victorio Albani de Carvalho[1], Archimedes Alves Detoni[2] e Fabiano Borges Ruy[3]

[1] Instituto Federal do Espírito Santo - Campus Colatina, ES, Brasil
[2] Instituto Federal do Espírito Santo - Campus Santa Teresa, ES, Brasil
[3] Instituto Federal do Espírito Santo - Campus Serra, ES, Brasil
{julionardi, victorio, archimedes, fabianoruy}@ifes.edu.br

Resumo. Neste capítulo abordamos a importância que o professor Ricardo de Almeida Falbo, membro titular do Departamento de Informática da Universidade Federal do Espírito Santo (UFES) e referência nas áreas de Engenharia de Software e Ontologias, teve na formação de servidores do Instituto Federal do Espírito Santo (IFES). Para tanto, apresentamos os relatos de nossa trajetória de formação, passando por momentos em que tivemos contato com o professor Falbo seja em disciplinas, seja em orientações no nível de graduação, mestrado e/ou doutorado. Por meio desses relatos, evidenciamos aprendizados, oportunidades e inspirações que o professor Falbo nos possibilitou. Acreditamos que, de alguma forma, por meio desses relatos, representamos outros servidores (técnicos-administrativos e docentes) do Ifes que também foram alunos e/ou orientandos do professor.

Palavras-chaves: reconhecimento, formação profissional, orientação, Ifes.

1 Introdução

Criado em 2008, o Instituto Federal do Espírito Santo (IFES) foi o resultado da união do Centro Federal de Educação Tecnológica do Espírito Santo (CEFETES) com as Escolas Agrotécnicas Federais, mas possui suas raízes mais profundas na Escola de Aprendizes Artífices do Espírito Santo, criada em 1909. O IFES busca promover educação profissional pública de excelência, integrando ensino, pesquisa e extensão, e oferece desde cursos de nível técnico até o nível de doutorado, possuindo, aproximadamente, 36 mil alunos [1].

O IFES possui 21 campi em funcionamento e mais o Centro de Referência em Formação e em Educação a Distância (Cefor), abrangendo, assim, todas as microrregiões capixabas e possuindo uma capilaridade que faz com que cada um de seus campi possa, além do ensino, propor soluções tecnológicas alinhadas aos problemas locais/regionais [1]. Considerando toda essa estrutura e distribuição geográfica, um dos desafios do IFES é a formação dos seus servidores para que possam atuar em iniciativas de ensino, pesquisa e extensão nos seus vários níveis.

A Universidade Federal do Espírito Santo (UFES), por sua vez, com mais de 60 anos de existência, tem desempenhado um papel fundamental na formação de profissionais em diversas áreas do conhecimento. A UFES é um ator importante no desenvolvimento do Estado do Espírito Santo, pois promove formação gratuita e de qualidade desde o nível de graduação até os níveis de mestrado e doutorado.

Dentre os vários servidores da UFES, colocamos em evidência, neste capítulo, o professor Ricardo de Almeida Falbo. Falbo é professor dessa instituição desde 1989, membro titular do Departamento de Informática, referência nas áreas de Engenharia de Software e Ontologias e um dos fundadores do Núcleo de Estudos em Modelagem Conceitual e Ontologias (Nemo) [2], o qual é considerado um dos principais grupos de pesquisa da área no mundo. Em seus 30 anos de atuação como professor e pesquisador, Falbo é autor de mais de 180 publicações (entre capítulos de livros, artigos em periódicos e trabalhos em eventos), orientou 60 trabalhos de conclusão de graduação, 32 trabalhos de iniciação científica, 35 dissertações de mestrado e 6 teses de doutorado. Para além de professor e orientador, Falbo tornou-se referência pela sua maneira de lecionar e de orientar, sendo uma figura muito querida pela maneira com que se relaciona com colegas pesquisadores e alunos.

Durante seus anos de atuação na UFES, o professor Falbo orientou e/ou lecionou para vários alunos que hoje são servidores (técnicos-administrativos ou docentes) do IFES. Tais servidores estão em, pelo menos, 8 (oito) das 22 (vinte e duas) unidades do IFES espalhadas pelo Espírito Santo. Dessa maneira, destacamos a contribuição do professor Falbo na expansão e consolidação de alguns campi do IFES (em especial, aqueles relacionados à área de Computação).

Neste capítulo evidenciamos a importância que o professor Falbo teve na formação de profissionais que hoje atuam no IFES. Apresentamos, assim, os relatos de 4 (quatro) docentes que atuam em 3 (três) campi: Colatina, Santa Teresa e Serra. Em cada um dos nossos relatos, apresentamos os principais eventos que tiveram a participação do professor Falbo e como ele influenciou a consolidação de ações de ensino, pesquisa, extensão no IFES.

O restante deste capítulo está organizado como a seguir: a Seção 2 apresenta os relatos de cada um de nós docentes; e a seção 3, por fim, apresenta nossas considerações finais.

2 Relatos de Ex-Orientados

Esta seção apresenta nossos relatos, 4 (quatro) docentes do Ifes que foram alunos orientados pelo professor Ricardo Falbo durante seus cursos de graduação, mestrado e/ou doutorado.

2.1 Falbo: Oportunização e Inspiração

Meu nome é Julio Cesar Nardi. Desde 2006, sou docente do Instituto Federal do Espírito Santo (IFES) – Campus Colatina e lotado na Coordenadoria de Informática.

Em 1999, aos meus 19 anos, ingressei no Curso de Bacharelado em Ciência da Computação na Universidade Federal do Espírito Santo (UFES), onde tive a oportunidade de conhecer excelentes mestres, que contribuíram muito para minha formação profissional e pessoal. Eu vinha de uma cidade do interior do Estado do Espírito Santo (Colatina) e a UFES, com tudo o que ela representaria na minha vida, ainda não estavam claros para mim. Dentre as pessoas que foram fundamentais para minha formação profissional e pessoal, destaca-se, sem dúvida, a figura do professor Ricardo de Almeida Falbo.

Embora tendo ingressado na UFES em 1999, o primeiro contato com o professor Falbo foi em 2001 na disciplina de Análise de Sistemas e, posteriormente, na disciplina de Projeto de Sistemas. Eu já tinha um interesse pela área de Desenvolvimento de Sistemas (por vir do Curso Técnico em Processamento de Dados da antiga Escola Técnica Federal do Espírito Santo – ETFES) e logo passei a me aproximar ainda mais dessa área, em especial, pela maneira com que o professor Falbo ministrava os conteúdos e pela organização e didática com que conduzia as aulas. A metodologia que o professor Falbo aplicava em suas disciplinas tinha forte relação com o que eu, muito tempo depois, passei a conhecer como Aprendizagem Baseada em Projetos (*Project Based Learning*). Por meio dos trabalhos práticos propostos, os alunos podiam exercitar e vivenciar a prática da construção de sistemas, em especial, durante as fases iniciais de análise e projeto de desenvolvimento de software. Ressalto aqui que essa organização e didática influenciaram muito minha maneira de lecionar, quando passei a atuar como docente.

No início de 2003 procurei o professor Falbo a fim de me inscrever no edital de seleção de alunos de iniciação científica que ele costumava realizar. (Destaco aqui a primeira oportunidade colocada pelo professor Falbo diante de mim). Tive a felicidade de ser aprovado e realizei um projeto relacionado ao desenvolvimento e manutenção de ferramentas diagramáticas UML (Unified Modeling Language) no contexto do Ambiente ODE (Ontology-based Software Development Environment) [3][4]. Essa foi uma grande oportunidade para mim, pois graças a ela passei a discutir e aplicar conceitos como ontologia, modelo, meta-modelo, diagrama, instanciação e especialização, dentre outros. Tive meu primeiro contato com o conceito de "ontologia". Sem dúvida não tinha toda a compreensão do conceito e do potencial que tal termo carregava, mas da maneira com que o professor Falbo o apresentava, passei a ter grande interesse. Nessa época, graças às orientações do professor, tive meus primeiros contatos com textos de pesquisadores como Nicola Guarino [5][6], Thomas Robert "Tom" Gruber [7][8], Riichiro Mizoguchi [9], B. Chandrasekaran [10], Vladan Devedžić [11] e Asuncion Gómes-Pérez [12][13]. Ainda realizando o projeto de iniciação científica, o professor Falbo me deu a oportunidade de desenvolver meu projeto final de curso. Tal projeto seguiu a mesma linha de pesquisa abordada durante a iniciação científica, mas com foco na modelagem de estados segundo a UML, o que me permitiu aprofundar meus conhecimentos na área de Modelagem Conceitual.

No ano de 2004 terminei meu curso de graduação e me candidatei a uma vaga no Programa de Mestrado em Informática do Departamento de Informática (DI) da UFES. Eu tinha interesse em continuar minhas pesquisas na área que já vinha trabalhando com o professor Falbo. Fui aprovado no processo seletivo do mestrado e tive mais uma vez

a oportunidade de trabalhar com o professor Falbo. Durante o mestrado (2004 a 2006), me dedicando integralmente ao curso como bolsista Capes, desenvolvemos uma abordagem de gerência de conhecimento à engenharia de requisitos no ambiente de desenvolvimento de software ODE [14][15]. Tal abordagem, dentre outros, utilizava ontologias de domínio para derivação de requisitos e modelos de classe, dando suporte às fases iniciais de desenvolvimento de software. Durante o mestrado, tive a oportunidade de realizar o estágio docência na disciplina de Projeto de Sistemas do Curso de Ciência da Computação do UFES. Mais uma vez pude contar com o professor Falbo, que me orientou sobre os procedimentos em sala de aula, sanando minhas dúvidas a respeito de como atuar. Portanto, tive minha primeira experiência com a docência. Gostei do que vivenciei.

No mesmo ano em que finalizei o mestrado (2006), prestei concurso público para uma vaga de docente de Informática do Instituto Federal de Educação do Espírito Santo (IFES) – Campus Colatina, à época Centro Federal de Educação Tecnológica do Espírito Santo (CEFETES). Finalizar o curso de mestrado em tempo seria essencial para que eu pudesse ser aprovado no pleito, uma vez que a titulação contaria ponto no processo seletivo. Destaco aqui, a dedicação e atenção que o professor Falbo teve comigo, em especial, nas últimas semanas pré-defesa de mestrado, a fim de que eu pudesse defender a tempo de apresentar meu título à banca do concurso público. O fato é que, ao me candidatar àquela seleção, eu já tinha o desejo de que minha carreira profissional fosse na docência aliada à pesquisa científica. Muito desse desejo veio do fato de ter como referência profissional e pessoal o professor Falbo.

Ingressando no IFES em 2006, me dediquei às disciplinas relacionadas à Análise e Projeto de Sistemas. Ao ministrar essas disciplinas, trouxe todo o exemplo vivenciado com o professor Falbo. Inclusive, passei a utilizar seu material de aula. Assim, no contexto do IFES – Campus Colatina, o nome “Falbo” ou “professor Ricardo Falbo” passou a ser respeitado e admirado, ainda que muitos dos meus alunos não tenham tido a oportunidade de conhecê-lo pessoalmente. Para além da docência, iniciei meus estudos e pesquisas. Trabalhei por 4 (quatro) anos chegando a 2010, quando finalizou meu período probatório e, então, eu pude me candidatar a algum curso de doutorado.

No início de 2010, comecei a analisar alguns programas de pós-graduação em Computação, em especial, no Rio de Janeiro. Nessa época, a UFES ainda não possuía o curso de doutorado em Computação. Minhas conversas foram avançando com professores e representantes de alguns programas até que recebi a grata notícia de que a UFES havia aprovado seu Curso de Doutorado em Ciência da Computação junto à Capes. Prontamente preparei-me para participar do edital de seleção e tive a felicidade de ser aprovado, principalmente, por poder trabalhar novamente sob orientação do professor Falbo. O curso de doutorado foi uma experiência ímpar, com a qual pude amadurecer, de fato, como pesquisador. Agradeço imensamente ao professor Falbo pelas orientações, pelos incentivos e pela oportunidades colocadas diante de mim. Agradeço, inclusive, a oportunidade de ter o professor João Paulo A. Almeida como (co-)orientador. O convite partiu do Falbo e o professor João prontamente aceitou. Por intermédio desses dois orientadores, tive a oportunidade de colaborar com importantes pesquisadores de fora do Brasil como Luís Ferreira Pires, Marten Van Sinderen e Nicola Guarino, dentre outros. Desenvolvemos trabalhos relacionados à integração

semântica de aplicações de software [16][17], ao *design* e uso de ontologias de serviços [16][18], e à análise ontológica de linguagens de modelagem de arquiteturas organizacionais [19][20]. Ao desenvolver tais trabalhos consolidei linhas de pesquisas em que tenho trabalhado atualmente. Com a conclusão do doutorado, tive o privilégio de ser o primeiro aluno a obter o título de Doutor em Ciência da Computação da Ufes, sendo também o primeiro orientando do professor Ricardo Falbo a obter tal título no âmbito Programa de Pós-Graduação em Informática da Ufes.

Terminado o Curso de Doutorado em Ciência da Computação, retornei para minha instituição de origem (IFES - Campus Colatina), onde retomei minhas pesquisas e minhas aulas no contexto do Curso de Bacharelado em Sistemas de Informação. Ontologias, Ciência de Serviços, Modelagem Conceitual e Modelagem de Arquiteturas Organizacionais têm sido meus temas de interesse no desenvolvimento de projetos de pesquisa e de extensão universitária. Tenho buscado alinhar tais interesses aos de outros colegas de instituição e juntos temos buscado estabelecer parcerias a fim de desenvolver soluções para as demandas locais/regionais, como estabelece a Política Institucional do IFES. Assim, a Coordenadoria da Área de Informática do IFES – Campus Colatina tem se desenvolvido a cada dia e tem buscado contribuir para o desenvolvimento da região. Atualmente, ofertamos, além do Curso de Bacharelado em Sistemas de Informação (integral), o Curso Técnico em Informática para Internet Integrado ao Ensino Médio (matutino), o Curso Técnico Concomitante em Montagem e Suporte em Informática (noturno), o Curso de Pós-Graduação lato sensu em Conectividade e Tecnologias da Informação (gratuito e aos finais de semana), além de cursos de extensão (com foco em comunidades de baixa renda e/ou escolas públicas da rede municipal e estadual). Dessa forma, temos buscado atuar nos vários níveis de ensino e no caminho de promover educação gratuita e de qualidade como política de redução das desigualdades. Considerando esse contexto, retomo à importância que o professor Falbo teve como educador. Além de mim, pelo menos outros 9 (nove) docentes, de um total de 14 (quatorze) lotados na Coordenadoria da Área de Informática, tiveram Falbo como professor e/ou orientador.

A partir do relato de eventos vivenciados por mim ao longo de minha formação profissional e pessoal, busquei destacar a importância do professor Falbo como educador e como aquele que oportuniza. Busquei destacar mais essas características do que aspectos científicos e técnicos, pois esses já são reconhecidos amplamente pela Indústria e pela Comunidade Científica. Embora eu tenha usado na narrativa fatos da minha história pessoal como direcionamento do enredo, tenho certeza de que outros colegas orientados e alunos do professor Falbo passaram por oportunizações e aprendizados tão grandes ou ainda maiores que os relatados por mim. Por vezes, nós orientados conversávamos sobre a abordagem do professor Falbo e como ele tinha a capacidade de simplificar e direcionar as questões. Dizíamos entre nós: “chegamos até a sala dele [Falbo] aflitos e cheio de dúvidas; depois da orientação, saímos da sala com o dobro de trabalho, mas saímos rindo e felizes”. Sua capacidade de lidar de maneira séria e organizada com o trabalho, mas sem perder o bom humor e o frescor que a vida requer era realmente inspiradora. O trabalho desenvolvido por esse mestre permitiu a formação de vários servidores do Instituto Federal do Espírito Santo (IFES), os quais podem, então, continuar suas respectivas missões de educar, oportunizar e inspirar.

2.2 Falbo: uma referência de docente

Meu nome é Victorio Albani de Carvalho. Sou professor do Instituto Federal do Espírito Santo (IFES), Campus Colatina, desde 2008. Meu primeiro contato com o professor Ricardo de Almeida Falbo foi em 2001, como aluno das disciplinas de análise de sistemas e de projeto de sistemas do curso de ciência da computação da UFES. Enquanto aluno de graduação, ainda em fase inicial de curso, eu procurava uma área com a qual me identificasse e que viesse a ser meu foco principal no curso. A estratégia didática de conduzir o aprendizado através do desenvolvimento de trabalhos práticos propiciou meus primeiros contatos com a prática de modelagem conceitual, que veio a se tornar minha principal área de interesse durante a graduação, sendo também uma área central de minha pesquisa de doutorado. Anos depois de ser aluno de Falbo nessas disciplinas, ao estudar práticas pedagógicas buscando aperfeiçoar minha atuação como docente, descobri que, através de sua estratégia de ensino focada na construção de artefatos, Falbo me apresentava, mesmo sem eu saber, à prática pedagógica construcionista [21].

Em 2002, já decidido a direcionar meus estudos às áreas de engenharia de software e modelagem conceitual, me candidatei no edital de seleção de alunos para orientação de trabalho de conclusão de curso do professor Falbo e tive a felicidade de ser selecionado. O projeto desenvolvido tinha por objetivo construir ferramentas de apoio à modelagem de arquitetura física no contexto do ambiente ODE [4]. Além de ser meu primeiro contato com um projeto de pesquisa, o desenvolvimento deste trabalho me levou a conhecer, discutir e aplicar diversos conceitos que vieram a ser fundamentais em minha formação, a saber: ontologia, meta-modelo, diagrama, classe, meta-classe, instanciação, especialização, dentre outros. Ademais, foi minha primeira experiência como orientado de Falbo, através da qual pude perceber que o papel do orientador vai muito além de indicar materiais e técnicas a serem aplicadas. Durante todo o processo Falbo se mostrou um grande incentivador e, acima de tudo, um amigo.

Ainda em 2002 fui convidado para uma entrevista que resultou em meu primeiro emprego, no qual trabalharia com modelagem multidimensional, implementação e manutenção de data warehouse. Foi uma experiência enriquecedora onde pude aplicar conceitos de modelagem conceitual sob uma ótica diferente. Depois de contratado, vim a saber que o convite para a entrevista partiu de uma indicação do professor Falbo. Mais uma influência direta de Falbo sobre minha trajetória.

Em 2003, quando terminava a graduação, surgiu o interesse em ingressar em um mestrado, mas não queria abrir mão da experiência no mercado de trabalho. Então fui conversar com Falbo que me alertou para as dificuldades que enfrentaria para conciliar o mestrado com o trabalho, mas ofereceu todo o apoio e incentivo que eu precisava para me candidatar a uma vaga no Programa de Mestrado em Informática do Departamento de Informática (DI) da UFES. Fui selecionado e tive uma nova oportunidade de trabalhar com o professor. Devido a oportunidades que via no mercado de trabalho, expus a Falbo minha vontade de buscar conhecimentos na área de gerência de projetos de software. Como sempre, tive todo o apoio do professor e, durante o mestrado (de 2004 a 2006) desenvolvemos uma infra-estrutura que combinava técnicas de gerência de conhecimento a técnicas de decisão em grupo para apoiar atividades de gerência de

projetos no contexto do ambiente de desenvolvimento de software ODE [22][23][24][25]. Concluir o mestrado conciliando estudos e trabalho só foi possível graças à motivação, à compreensão e ao apoio que Falbo dedicou durante todo o projeto.

No ano de 2007, recebi um convite para atuar como docente em cursos de graduação. Estava empolgado com a oportunidade, porém extremamente inseguro devido a não ter experiência em docência. Então, mais uma vez, recorri a Falbo que, como de costume, me apoiou e incentivou, além de me orientar sobre como atuar em sala de aula. Seu apoio e orientação foram fundamentais para que eu aceitasse o desafio e iniciasse a carreira docente. No ano seguinte, prestei concurso público para docente do Instituto Federal do Espírito Santo (IFES) – Campus Colatina, fui selecionado e passei a me dedicar integralmente à carreira docente.

Em 2012, quando estava finalizando o período de estágio probatório no IFES, comecei a aventar a possibilidade de ingressar em um programa de doutorado. Como não poderia ser diferente, a primeira pessoa que procurei para conversar foi o professor Falbo. Cheguei para conversar com algumas grandes áreas de interesse e muitas dúvidas sobre o que seria projeto em nível de doutorado. Depois de algumas conversas, Falbo indicou que o professor mais adequado para orientar trabalhos nos temas que demonstrei interesse seria João Paulo A. Almeida. Então, Falbo me apresentou ao professor João Paulo e, em menos de um mês, eu iniciava o curso de doutorado em ciência da computação da UFES. Durante o doutorado desenvolvemos trabalhos focados em modelagem conceitual baseada em ontologias e em modelagem multi-nível. Os conceitos básicos de (meta)modelagem e ontologias que me foram apresentados por Falbo na graduação e no mestrado foram fundamentais para o desenvolvimento dos trabalhos. Mas, a influência de Falbo em minha trajetória durante o doutorado foi muito além. Mesmo não sendo meu orientador, em vários momentos durante o curso recorri a Falbo para conversar sobre as angústias inerentes ao trabalho de pesquisa e sempre pude contar com seu apoio e seus conselhos.

Além de estar presente em toda minha trajetória acadêmica, Falbo é um dos grandes responsáveis por despertar meu desejo de ser professor e pesquisador. Ele é, sem dúvidas, minha maior referência profissional e tenho certeza que também o é para muitos outros colegas docentes. Atualmente me dedico exclusivamente à carreira docente na coordenadoria de informática do IFES – Campus Colatina. Essa coordenadoria oferece quatro cursos nos níveis técnico, de graduação e de pós-graduação lato sensu, ofertando, anualmente, em torno de 120 vagas para alunos nos diferentes níveis de ensino. Nossa equipe é formada por 14 professores, dos quais mais da metade foram alunos e/ou orientandos do professor Falbo. Notas de aula e apostilas de análise de sistemas, de projetos de sistemas e de engenharia de software produzidas por Falbo são adotadas como referências bibliográficas complementares nas disciplinas dos cursos técnicos e de graduação. Assim, não é exagero considerar que, mesmo nunca tendo vínculo empregatício com o IFES – Campus Colatina, o trabalho do professor Falbo está presente na formação de cada um dos alunos dos cursos de informática deste campus.

2.3 Falbo: influenciando pelo exemplo

Meu nome é Archimedes Alves Detoni, sou docente do IFES - Campus Santa Teresa desde 2009. Entendo que outro exemplo, dentre tantos, da importante contribuição do Falbo na multiplicação do conhecimento e da sua influência acadêmica, em especial para os servidores do IFES, seja o processo da minha formação durante o doutorado em Ciência da Computação na UFES, entre 2014 e 2019.

Minha primeira interação com o Falbo ocorreu durante o curso de mestrado em Automação, em 1999, quando tive a oportunidade de participar de suas aulas de Engenharia de Software, como uma disciplina optativa. Havia me graduado no curso de Engenharia da Computação em 1998, que possuía baixa carga horária de disciplinas da área de análise, projeto e processos de desenvolvimento de software. Dessa forma, avaliando minha formação acadêmica, percebo que aquela breve interação com o Falbo, apenas 45 horas no segundo semestre de 1999, produziu impactos significativos nas minhas escolhas acadêmicas posteriores.

Minha atuação como docente começou em 2003, lecionando inicialmente em algumas faculdades privadas, ministrava disciplinas de programação e, posteriormente, de análise e projeto de software. Na evolução dessa caminhada, já como docente do Campus Santa Teresa do IFES, percebi a necessidade de me qualificar a fim de atuar efetivamente como pesquisador. Dessa forma, em 2014 (13 anos após haver concluído o mestrado), fui admitido como estudante no curso de Doutorado em Ciência da Computação da UFES, optando por fazer parte do Nemo [2] e conduzir pesquisa na área de modelagem conceitual e ontologias.

Durante o doutorado, fui fortemente influenciado pelas ideias e metodologias propagadas pelo Falbo, iniciando pela disciplina de Engenharia de Ontologias, seguida pelas de Ontologias em Engenharia de Software e Metodologia de Pesquisa (nesta última, tive a oportunidade de atuar como um auxiliar). A sua proposta didática imprimia nas aulas uma dinâmica extremamente participativa, provocando nos estudantes uma postura ao mesmo tempo crítica e investigativa. Sua conduta era sempre marcada pela organização didática, pelo zelo com que conduzia a exposição dos assuntos e pela exigência por qualidade das atividades produzidas pelos estudantes. Contudo, Falbo demonstrava uma incrível habilidade de não tornar esses aspectos pesados ou enfadonhos, pois se expressava sempre de forma simpática, bem humorada e paciente em relação às dúvidas e reclamações dos seus estudantes e orientados. Especificamente no meu caso, que fui um aluno e orientado um tanto questionador (e reclamão, preciso admitir - por esse motivo recebi dele o apelido de “Chorão”), o Falbo sempre tinha uma palavra de estímulo e uma forma leve de conduzir as situações a ele apresentadas.

No segundo ano do doutorado, meu orientador, prof. João Paulo A. Almeida, fez a proposta do Falbo acompanhar meu trabalho como co-orientador, o que acolhi com muita satisfação, tendo em vista que já vínhamos trabalhando juntos em um projeto de pesquisa e na elaboração de artigos científicos. A partir desse ponto, tive a oportunidade de estreitar ainda mais a relação com ele e ser influenciado de forma mais intensa sobre os diversos aspectos que envolvem o desenvolvimento de uma pesquisa de qualidade. Preciso aqui destacar a importância das contribuições e orientações recebidas do Falbo,

juntamente com as do prof. João Paulo, especificamente para o desenvolvimento da pesquisa que foi objeto da tese do meu doutorado, a abordagem EArly-OE (*Enterprise Architecture-driven early Ontology Engineering*) [26] assim como na elaboração do seu texto final.

Enfim, quero engrossar os relatos dos demais colegas, unânimes em destacar suas boas experiências e percepções quanto à relevância da atuação do mestre Ricardo Falbo para nossa formação acadêmica e consequente melhoria da qualidade dos servidores e do potencial de pesquisa do IFES.

2.4 Falbo: 20 anos de orientação

Meu nome é Fabiano Borges Ruy, sou professor do Instituto Federal do Espírito Santo (IFES), Campus Serra, desde 2008.

Esta seção é dedicada a destacar a importância do Professor Ricardo de Almeida Falbo, o Falbo, em minha trajetória acadêmica e profissional, além de sua influência nas atividades em meu campus. Hoje sou professor, com muito orgulho, e escrevo deste ponto de vista. Considero que uma das maiores satisfações que um professor pode ter é o reconhecimento de seus alunos por seu trabalho. Às vezes isso acontece em uma disciplina, por meio da atenção e respeito prestados. Às vezes ocorre num trabalho ou artigo bem feito, que deixa o professor orgulhoso que seus ensinamentos surtiram efeito. Às vezes numa formatura, com uma justa homenagem da turma ao professor. Às vezes só acontece anos depois, quando o ex-aluno, já exercendo sua vida profissional, encontra o professor e diz: "aquele 'negócio' que você me ensinou é muito útil, me salvou lá no trabalho". Acredito que, de todas essas formas, já tenha expressado meu reconhecimento a ele. Esta sessão segue para fazê-lo mais uma vez.

Sou de Linhares, do interior do Estado e em 1998 tive a felicidade de ingressar na Universidade Federal do Espírito Santo, a UFES, no curso de Ciência da Computação. Para quem veio de escola pública, a Universidade é um novo mundo, cheio de desafios e conhecimento. Mal sabia que neste ambiente iria passar ainda muitos anos de minha vida. Já no meio do curso, em 2000, cursei a disciplina de Análise de Sistemas com o Falbo. Embora naquele momento o conteúdo não parecesse ser o foco da graduação, bastante técnico até então, depois vim a entender bem a utilidade daquele "monte de caixinhas interligadas". Naquela época o Falbo era mais novo do que sou hoje e já um professor experiente e reconhecido entre colegas e alunos. Em suas aulas demonstrava extrema tranquilidade e segurança, seus trabalhos eram bastante práticos e estimulavam a busca por problemas reais. Impressionava sua capacidade de conhecer cada trabalho, muitas vezes até melhor que os próprios alunos do grupo. No semestre seguinte, cursei ainda Projeto de Sistemas com ele, nos mesmos moldes.

Em 2001, comecei uma Iniciação Científica, junto com o colega Gleidson Bertollo, sob sua orientação. Foram as primeiras iniciações orientadas pelo Falbo. À época, ele já havia orientado diversos trabalhos de conclusão de curso, que produziram ferramentas CASE, e o plano era integrá-las em um único ambiente. Assim nascia o ODE (*Ontology-based software Development Environment*) [4], inspirado em sua tese de doutorado, resultado de meses de muito esforço quebrando pedra, vários protótipos integrados, e muitas reuniões em que entrávamos com dúvidas e saíamos com muito

mais trabalho, porém empolgados. No ano seguinte, fizemos uma parceria com uma empresa para alguns experimentos com o ODE e pude prosseguir na pesquisa. Mais alunos se juntaram ao grupo e surgiu o LABES (Laboratório de Engenharia de Software), comandado pelo Falbo, único professor da linha na época. Em 2003 concluí meu projeto de graduação, desenvolvendo infraestruturas para a integração de dados e conhecimento em ODE [27]. Após a apresentação, fizemos uma grande mesa com alunos em um bar próximo a UFES, com o Falbo presente, como sempre foi de seu costume.

Os anos finais de curso foram muito produtivos em se tratando de uma graduação. Sob a orientação do Falbo, escrevemos e publicamos alguns artigos sobre ODE (e.g. [4], [28], [29]) em eventos nacionais e internacionais. Tive a oportunidade de apresentar dois deles, em Manaus e em Valdivia, no Chile, experiências fantásticas em termos de conhecimentos científicos, contatos com importantes pesquisadores e também de conhecer novos lugares.

Ainda em 2003 entrei para o Mestrado em Informática, sob a orientação do Falbo. A proposta agora era tratar a questão semântica em ODE [30][31][32]. Tive direito a uma bolsa do mestrado, que me permitiu total dedicação aos trabalhos no LABES. Lembro de uma das primeiras reuniões, quando Falbo perguntou se a minha intenção com o mestrado era me tornar professor. Disse, seguro, que não! Ledo engano. Em poucos meses estava eu assistindo suas aulas em uma pós-graduação, onde fui monitor. Em seguida, fiz o estágio docência na disciplina de Projeto de Sistemas. Ambas experiências muito enriquecedoras, contando com a supervisão do Falbo.

O Mestrado foi um período muito próspero em trabalho e aprendizado. Pude consolidar meus conhecimentos da graduação, focando em um sistema real, o ODE, onde obtinha e aplicava conhecimentos de programação, modelagem, banco de dados, ontologias, gestão de projetos etc. Tínhamos uma equipe fantástica no LABES com Julio Nardi, Rodrigo Dal Moro, Silvano Buback e Lucas Oliveira, dentre outros. Pessoas super capacitadas que tenho como amigos até hoje. O LABES funcionava muito bem: as pessoas motivadas, trabalhando em um projeto comum, desenvolvendo habilidades profissionais e ainda fazendo pesquisa, todos sob a orientação do Falbo. Muitos foram os resultados em termos de publicações, evolução do ambiente ODE e projetos. Eu estava tão envolvido com as atividades do laboratório, que não concluí o texto da dissertação antes que a bolsa acabasse. Logo comecei a trabalhar em uma empresa de software, o que tomou a maior parte do meu tempo e foco, praticamente paralisando o andamento da dissertação. Foram meses até que eu conseguisse retomar. E nestes meses, novamente aprendi com o Falbo, que a motivação é um recurso mais efetivo que a pressão nos processos de orientação e ensino. Os prazos do mestrado eram claros; ele, de forma muito justa, não aceitaria um trabalho de menor qualidade; e me manteve motivado a concluir um bom trabalho. A defesa foi em 2006, no último dia possível, para dar tempo de concluir todos os detalhes. E a comemoração novamente com os amigos, sob "supervisão" do Falbo em um bar.

Em 2008 eu já estava convencido a ser professor. Falbo me aconselhou então a fazer um concurso do IFES. Fiz, passei, e atuo no Campus Serra, na linha de Engenharia de Software. Já ministrei as disciplinas de Análise e Projeto de Sistemas inúmeras vezes, sempre ao estilo do Falbo! Não só eu, mas tantos outros colegas que também foram

ensinados por ele: Vanessa Battestin, Rodrigo Calhau, Paulo Sérgio, Felipe Frechiani e Carlos Azevedo. Foi o formato que aprendemos, que funciona, e por mais que façamos um ajuste ou outro, a essência se mantém. Certa vez, outro colega, Francisco Rapchan, bem disse em uma reunião com pedagogos: "nós somos professores, mas muitos de nós não tem formação para ministrar aulas, não estudamos as técnicas, nós, no máximo, imitamos um professor que gostávamos". De fato, em termos de didática e organização, o Falbo é uma referência, não só na UFES e IFES, como em outras instituições. Sua abordagem é amplamente utilizada [33].

Passando-se os semestres e ganhando experiência com as disciplinas, em 2012 criamos algo diferente no IFES. Aliando a vocação mais tecnológica e aplicada do Instituto, a experiência anterior no LABES e o anseio de ver os alunos aplicando seus conhecimentos, gerando resultados práticos além dos trabalhos de sala de aula, fundamos o LEDS - Laboratório de Extensão em Desenvolvimento de Soluções. Coordenei o laboratório por um ano, até conseguir meu afastamento para cursar o doutorado. Não tive dúvidas quanto à escolha do orientador.

Então, em 2013, sete anos após a conclusão do mestrado, tive o prazer de voltar a trabalhar com o Falbo, quando meu projeto foi aprovado no Doutorado em Ciência da Computação da UFES. Foram 4 anos (e alguns meses) até a defesa. Um período de grande aprendizado em variados aspectos. Falbo me incentivou a escrever artigos, em inglês, desde o primeiro período. E sempre os revisou com a máxima prioridade. Conseguimos ter publicações em eventos importantes [34][35][36][37], journals [38] e até um *best paper* [39]. Tivemos muitas reuniões, discussões, sempre bem focadas e produtivas, sem perder o bom humor. Por meio de seus contatos conheci vários pesquisadores (aqueles que a gente citava desde a graduação) e tive o prazer de trocar ideias sobre meu trabalho com alguns deles. Viajei, apresentei artigo na Coréia, fiz estágio na Austrália e sanduíche na Espanha.

Durante a defesa de minha tese [40], nos agradecimentos, tive o prazer de dizer que foram 4 anos tranquilos. Não foram fáceis, *muito pelo contrário*, de muito trabalho, mas sempre bem focado e no caminho certo. Tínhamos um plano no início e que, embora as atividades variassem um pouco, os objetivos foram mantidos e alcançados. O Falbo, em sua experiência, sabia bem onde deveríamos chegar e sempre me guiou corretamente, uma orientação impecável!

Voltando ao IFES em 2017, a área de Engenharia de Software, nos vários cursos do campus, tem sido dominada por ex-alunos do Falbo. No mestrado, recém-aberto, assumi a linha de Modelagem Conceitual por minha experiência do doutorado. Além disso, hoje o LEDS é o maior laboratório do Campus Serra, atualmente com 6 professores (5 deles seus ex-alunos) e cerca de 25 alunos, desenvolve projetos reais, com clientes, prazos e boas práticas de Engenharia de Software. É uma referência no IFES, tendo gerado inclusive unidades em outros campi. O *fato concreto é* que alguns ingredientes do LEDS vieram do LABES e do NEMO, da metodologia aplicada pelo Falbo, do estilo de orientação que aprendemos, e das experiências que os professores integrantes tiveram em sua formação acadêmica.

Concluo com uma frase que ouvi há tempos, dizendo que "na vida é preciso apenas um bom professor para que se encontre a motivação necessária aos estudos". Em todos estes anos como estudante, e foram muitos, tive vários bons professores, mas o Falbo

é, sem dúvida, o que mais me faz lembrar desta frase. E a tomo como exemplo hoje, para que eu possa ser um bom professor para ao menos alguns alunos.

3 Considerações Finais

Na última década, o IFES passou por um grande processo de expansão e interiorização com a criação de 18 (dezoito) novos campi por todo o Estado do Espírito Santo, demandando, assim, a contratação de muitos servidores (técnicos administrativos e docentes). Concomitante à expansão, houve também um processo de verticalização do ensino com a criação de vários cursos de graduação e de pós-graduação, o que gerou a necessidade de capacitação do corpo docente do IFES. A UFES, como instituição referência de ensino superior do Estado, teve fundamental importância no processo de formação do corpo docente do IFES.

Nesse contexto, este capítulo destaca a importância que o professor Ricardo de Almeida Falbo, membro do Departamento de Informática da UFES desde 1989, teve na formação de servidores do IFES. Apresentamos relatos de nossas próprias trajetórias de formação, mas buscamos destacar que, para além de nossas formações pessoais, nosso propósito é evidenciar fatos que ratificam a grande importância de Falbo na formação dos profissionais do IFES em geral.

Como titular da cadeira de Engenharia de Software da UFES há mais de 20 anos, Falbo influenciou a formação de grande parte dos profissionais dessa área no Estado, sendo possível encontrar casos de sucesso de seus ex-alunos na Academia e na Indústria. Os fatos deixam claro que Falbo é um grande professor e orientador. Sua influência, entretanto, vai além do exercício docente. Nos relatos que apresentamos neste capítulo, procuramos demonstrar nossa admiração por esse grande profissional, mas, acima de tudo, por esse grande amigo, incentivador e parceiro. Acreditamos que falamos em nome de diversos outros colegas que também tiveram a oportunidade de conviver e aprender com ele.

4 Referências

1. IFES – Instituto Federal do Espírito Santo: Institucional (https://ifes.edu.br/o-ifes)
2. NEMO: Núcleo de Estudos em Modelagem Conceitual e Ontologias (https://nemo.inf.ufes.br/pt-br/).
3. Falbo, R.A.: Integração de Conhecimento em um Ambiente de Desenvolvimento de Software. COPPE-UFRJ (1998).
4. Falbo, R.A., Natali, A.C.C., Mian, P.G., Bertollo, G., Ruy, F.B., ODE: Ontology-based software Development Environment. IX Congreso Argentino de Ciencias de la Computación, 1124-1135, La Plata, Argentina (2003).
5. Guarino, N.: The Ontological Level. Presented at the (1994).
6. Guarino, N.: Formal Ontology and Information Systems. In: Formal Ontology and Information Systems. pp. 3–15, Trento, Italy (1998).
7. Gruber, T.R.: Toward Principles for the Design of Ontologies used for Knowledge Sharing. Int. J. Human-Computer Stud. 43, 5/6, (1995).

8. Gruber, T.R.: A Translation Approach to Portable Ontologies. Knowledge Acquisition. 5, 2, 199–220 (1993).
9. Ikeda, M. et al.: Task ontology: Ontology for building conceptual problem solving models. In: Proceedings of {ECAI98}. pp. 126–133 (1998).
10. Chandrasekaran, B. et al.: What Are Ontologies, and Why Do We Need Them? IEEE Intell. Syst. 20–26 (1999).
11. Devedzic, V.: Ontologies: Borrowing from Software Patterns. ACM Intell. Mag. 14–24. (1999).
12. Gómez-Pérez, A., Martins, J.P.: Some Issues on Ontology Integration. In: IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden. (1999).
13. Benjamins, V.R. et al.: Knowledge Management through Ontologies. In: 2nd International Conference on Practical Aspects of Knowledge Management, Basel, Switzerland (1998).
14. Nardi, J. C.: Apoio de Gerência de Conhecimento à Engenharia de Requisitos em um Ambiente de Desenvolvimento de Software. Universidade Federal do Espírito Santo (UFES) (2006).
15. Nardi, J. C., Falbo, R. de A.: Apoio de Gerência de Conhecimento na Engenharia de Requisitos. In: VI Simpósio Brasileiro de Qualidade de Software. pp. 63–77, Porto de Galinhas - Ipojuca - Recife - Brasil. (2007).
16. Nardi, J.C., Falbo, R. A., Almeida, J. P. A.: Towards a Commitment-based Reference Ontology for Services. In: 17th IEEE International Enterprise Distributed Object Computing Conference (EDOC). pp. 175–184, Vancouver, Canada (2013).
17. Nardi, J. C.; Falbo, R. A.; Almeida, J. P. A.: Foundational Ontologies for Semantic Integration in EAI: A Systematic Literature Review. In: 12th IFIP Conference on e-Business, e-Services, e-Society (I3E 2013). pp. 238–249, Athens, Greece (2013).
18. Nardi, J.C.: A Commitment-Based Reference Ontology for Service: Harmonizing Service Perspectives. Federal University of Espírito Santo (2014).
19. Nardi, J. C.; Falbo, R. A.; Almeida, J. P. A.: An Ontological Analysis of Service Modeling at ArchiMate's Business Layer. In: 18th IEEE International Enterprise Distributed Object Computing Conference (EDOC), Ulm - Germany (2014).
20. Nardi, J. C.; Falbo, R. A.; Almeida, J. P. A.: Revealing Service Commitments in Service-Oriented Enterprise Architecture. In: The Sixth Workshop on Service oriented Enterprise Architecture for Enterprise Engineering. pp. 286–295, Ulm - Germany (2014).
21. Valente, J. A. Computadores e Conhecimento: Repensando a educação. Campinas: Unicamp/NIED. (1998).
22. Carvalho, V.A.: Gerência de Conhecimento e Decisão em Grupo: um Estudo de Caso na Gerência de Projetos. Universidade Federal do Espírito Santo (UFES) (2006).
23. Carvalho, V.A., Coelho, A.G., Falbo, R. A.: Apoio Automatizado à Gerência de Riscos Cooperativa. In: X Workshop Ibero-americano de Engenharia de Requisitos e Ambientes de Software. Isla Margarita - Venezuela. (2007).
24. Arantes, L.O., Carvalho, V.A., Falbo, R. A.: EstimaODE: Apoio a Estimativas de Tamanho e Esforço no Ambiente de Desenvolvimento de Software ODE. In: V Simpósio Brasileiro de Qualidade de Software (SBQS). Vila Velha - Brasil. (2006).

25. Falbo, R. A., Machado, B.N., Carvalho, V.A.: Uma Infra-estrutura para Apoiar a Elaboração Colaborativa de Artefatos de Software. In: V Simpósio Brasileiro de Sistemas Colaborativos (SBSC). Vila Velha - Brasil. (2008).
26. Detoni, A. A.: EArly-OE: Atividades Iniciais de Engenharia de Ontologias Apoiadas em Modelos de Arquitetura Organizacional. Universidade Federal do Espírito Santo (UFES) (2019).
27. Ruy, F.B., Infra-estruturas de Apoio à Integração de Dados e Conhecimento em ODE. Projeto de Graduação, Universidade Federal do Espírito Santo, Vitória, 2003.
28. Falbo, R.A., Guizzardi, G., Natali, A.C.C., Bertollo, G., Ruy, F.B., Mian, P.G., Towards Semantic Software Engineering Environments. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, SEKE'2002, Ischia, Italy, 2002.
29. Ruy, F.B.; Bertollo, G.; Falbo, R.A. Apoio Baseado em Conhecimento à Integração de Processo em ODE. In: 3a Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería de Conocimiento, Valdivia, 2003.
30. Ruy, F.B. Semântica em um Ambiente de Desenvolvimento de Software. Dissertação de Mestrado, Universidade Federal do Espírito Santo, Vitória, 2006.
31. Ruy, F.B., Bertollo, G., Falbo, R.A., Knowledge-based Support to Process Integration in ODE. CLEI Electronic Journal, Volume 7, Number 1, 2004.
32. Falbo, R.A., Ruy, F.B. and Dal Moro, R. Using Ontologies to Add Semantics to a Software Engineering Environment. In International Conference on Software Engineering and Knowledge Engineering (SEKE'2005), pp. 151-156, 2005.
33. Ismail, M.A., Yaakob, M. and Kareem, S.A. Domain Specific Ontology Construction: Falbo's Approach. In International Conference on Electrical Engineering and Informatics, 2007.
34. 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 '2014), pp. 323–330, 2014.
35. Ruy, F.B., Falbo, R.A., Barcellos, M.P., Guizzardi, G. An Ontological Analysis of the ISO/IEC 24744 Metamodel. In Proceedings of the 8th International Conference on Formal Ontology in Information Systems (FOIS'14), Rio de Janeiro, Brazil, 2014.
36. Ruy, F.B., Reginato, C.C., Santos, V.A., Falbo, R.A. and Guizzardi, G. Ontology engineering by combining ontology patterns. In Proceedings of the 34th International Conference on Conceptual Modeling (ER'2015). pp.173-186. Springer International Publishing, 2015.
37. Ruy, F.B., Falbo, R.A., Barcellos, M.P., Costa, S.D., Guizzardi, G. SEON: A Software Engineering Ontology Network. In Proceedings of 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16), Bologna, Italy, pp.527-542, 2016.
38. Ruy, F.B., Reginato, C.C., Santos, V.A., Falbo, R.A. and Guizzardi, G., From Reference Ontologies to Ontology Patterns and Back. Data & Knowledge Engineering (DKE), 2017.
39. Ruy, F.B., Falbo, R.A., Barcellos, M.P. and Guizzardi, G. Towards an ontology pattern language for harmonizing software process related ISO standards. In Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC'2015). pp.388-395. ACM, 2015.
40. Ruy, F.B. Software Engineering Standards Harmonization: an Ontology-based Approach. Doctorate Thesis. Federal University of Espírito Santo, Vitória, 2017.

The EPUB version of this book was designed with calibre. The cover was designed in Inkscape using the Khand open source font family and a photo shared at PxHere with CC0. The editors would like to thank all those involved in these free open source projects for their dedication. The editors would also like to thank Patrícia Dockhorn Costa and the NEMO senior members for their support in producing this book.

This volume celebrates the career of Prof. Ricardo de Almeida Falbo on the occasion of his formal retirement. The volume includes reflections from collaborators and former students, casting light on his academic work and contributions.

The chapters show how Falbo's original contributions have influenced a number of developments in the application of Software Engineering practices to Ontology Engineering and in the application of Ontologies in Software Engineering.

A few personal notes are also offered, and some of the technical essays also include personal notes commemorating his captivating qualities.

Contributors:

Ana Regina Rocha · Archimedes A. Detoni · Crediné Silva de Menezes
Danielli dos Reis Costa · Érica Ferreira de Souza · Fabiano B. Ruy
Fernanda Araujo Baião · Giancarlo Guizzardi · Gleison Santos
Guilherme Horta Travassos · João Paulo A. Almeida · Julio C. Nardi
Karina Villela · Káthia Marçal de Oliveira · Maria das Graças Teixeira
Maria Luiza M. Campos · Monalessa Perini Barcellos · Nicola Guarino
Renata Guizzardi · Silvia das Dores Rissino · Thayza Sacconi Guarnier
Victorio A. Carvalho · Vítor E. Silva Souza