RASO: an Ontology on Requirements for the Development of Adaptive Systems

Cássio Capucho Peçanha, Bruno Borlini Duarte, and Vítor E. Silva Souza

Ontology & Conceptual Modeling Research Group (NEMO)
Department of Computer Science
Federal University of Espírito Santo (UFES) — Vitória, ES, Brazil
cassiocpecanha@gmail.com, bruno.b.duarte@ufes.br, vitorsouza@inf.ufes.br

Abstract. There is growing interest in software that can adapt their behavior to deal with deviations between their outcome and their requirements at runtime. A systematic mapping of the literature on self-adaptation approaches based on requirements models revealed over 200 papers on this subject. However, there is still a lack of a formal and explicit representation of the concepts in this domain, which can lead to problems in communication, learning, problem-solving, interoperability, etc. To make a clear and precise description of this domain, this paper proposes RASO: the Requirements for Adaptive Systems Ontology. RASO was built using a well-established Ontology Engineering method, is grounded on a foundational ontology and reuses concepts from other software-related ontologies. The ontology was evaluated by mapping constructs from the most referenced approaches from the literature to its concepts, thus creating a path for interoperability among them.

Keywords: Adaptive Systems · Requirements · Models · Ontology.

1 Introduction

For the past decades, there is growing interest in self-adaptation — systems that are able to modify their behavior/structure in response to the environment, itself and its goals — as a way to manage the ever increasing complexity and the many other challenges involved in the development and management of current software systems [23]. Some academic efforts in this direction (e.g., [2, 35, 8, 33]) focus on Requirements Engineering (RE) for Adaptive Systems, attempting to address what adaptations are possible and how they can be realized [7]. A systematic mapping of the literature on requirements-based self-adaptation approaches revealed over 200 publications on this subject.

Each of these proposals may use different kinds of models and terms to represent what are the system requirements and prescribe how it should self-adapt in given situations. As result, the vocabulary used by these methods may be very similar, but the semantics of the entities present in their models are not always the same, thus resulting in problems such as concept overloading: the same name being used to identify things that are ontologically different, i.e., that have different natures and identities in the real world. Other problems, such as construct excess, construct redundancy and incompleteness [15] may lead to difficulties in communication, learning, problem-solving, interoperability, etc., especially among Requirements Engineers and other actors involved in the construction of adaptive systems.

An ontology as a computational artifact, i.e., a "formal specification of a shared conceptualization" [3], can help mitigate these problems and has been attracting interest in the RE community [9]. In particular, a *domain ontology* is a conceptual model developed with the goal of making a clear and precise description of domain entities, representing the consensus within a community. More specifically, domain *reference ontologies* are solution-independent specifications and, thus, do not maximize computational properties at the cost of truthfulness to the domain [16].

In a previous work [10], we have proposed a domain reference ontology on the use of *Requirements at Runtime* (RRT), a subject that is at the core of RE for Adaptive Systems [7]. Hence, we set out to extend this ontology towards the domain of requirements-based development of adaptive systems, with the purpose of establishing a clear and precise description of this domain.

In this paper, we present the Requirements for Adaptive Systems Ontology (RASO). RASO was built using a well-established Ontology Engineering method, is grounded on a foundational ontology and reuses concepts from other software-related ontologies. To establish consensus about the domain, we performed a systematic mapping of the literature on requirements-based approaches for the development of adaptive systems and used selected publications as sources for capturing the ontology's concepts. Finally, the ontology was evaluated by verifying if it satisfied its own requirements and validating its concepts against real-world entities extracted from well-known approaches in its domain.

The remainder of the paper is structured as follows: Section 2 describes the ontological foundations (method and reused ontologies) of RASO; Section 3 presents the RASO ontology itself and the process through which it was built; Section 4 explains how RASO was evaluated; Section 5 compares our work to similar proposals in the field; and, finally, Section 6 concludes.

2 Ontological Foundations

The Requirements for Adaptive Systems Ontology (RASO) was built using SABiO: a Systematic Approach for Building Ontologies [11], a well-established Ontology Engineering method. We chose SABiO because it is focused on the development of domain ontologies and has been successfully used for developing several ontologies in the Software Engineering domain (e.g., [5, 30]).

SABiO's development process is composed of five phases — (1) purpose identification and requirements elicitation; (2) ontology capture and formalization; (3) design; (4) implementation; and (5) testing — accompanied by well-known support processes, such as knowledge acquisition, reuse, documentation and evaluation. For a reference ontology, we focus on the first two phases of the method.

At the first phase, requirements for the ontology are elicited in the form of Competency Questions (CQs), which are questions that the ontology should answer [14]. In the following phase, relevant concepts and relations for the domain should be identified and organized. Requirements elicitation and ontology capture for RASO are presented in Section 3.

SABiO explicitly recognizes the importance of using foundational ontologies in the ontology development process to improve the ontology quality, representativity and formality. We thus grounded RASO on the Unified Foundational Ontology (UFO) [15], a well-established foundational ontology. UFO offers a complete set of categories to tackle the specificities of the targeted domain and, as SABiO, it has been successfully employed in the development of several ontologies in the Software Engineering domain (e.g., [5, 30]), including the ones that were reused/extended in this work.

SABiO also suggests that existing ontologies on related domains should be reused. The Software Engineering Ontology Network (SEON) [30] includes a few ontologies that were reused by RASO (which also facilitates the future integration of RASO into the ontology network), namely: the Software Ontology (SwO), the Reference Software Requirements Ontology (RSRO) and the Runtime Requirements Ontology (RRO) [10], all of which are connected to SEON's core ontology: the Software Process Ontology (SPO) [5]. We decided to reuse concepts from these ontologies as their domains are intrinsically related to the domain of requirements for the development of adaptive systems. Moreover, they are all founded on UFO, guaranteeing compatibility at the foundational level. Among these ontologies, RRO has a particular prominent role, given that it describes the use of requirements artifacts at runtime, which is key to our domain. Given how close RASO is of RRO, we could also consider the former as an extension of the latter.

Figure 1 shows fragments of these ontologies, presenting the concepts reused by RASO, using a UML (Unified Modeling Language) class diagram primarily for visualization (for an in-depth discussion and a more formal characterization, refer to [10]). As shown, these concepts derive from the notion of Artifact from SPO, i.e., an Object (in the sense of UFO) intentionally made to serve a given purpose in the context of a software product or organization. We are particularly interested in three types of Artifacts: Software Items — pieces of software produced at a software process —, Documents — any written or pictorial information related to the software development — and Information Items — any relevant information produced at the software process for human use.

As defined in SwO [10], a Software System is a Software Item that *intends to implement* a System Specification, which is a Document containing a set of requirements for a system, defining its desired functions and features in an abstract way, without constraining its behavior. Software Systems are *constituted of* Programs, a Software Item which aims at producing results through execution on a computer, as prescribed by the Program Specification, i.e., a Document that describes the structure and functions of a Program. Finally, a Program is *constituted of* Code, a Software Item representing a set of computer instructions and

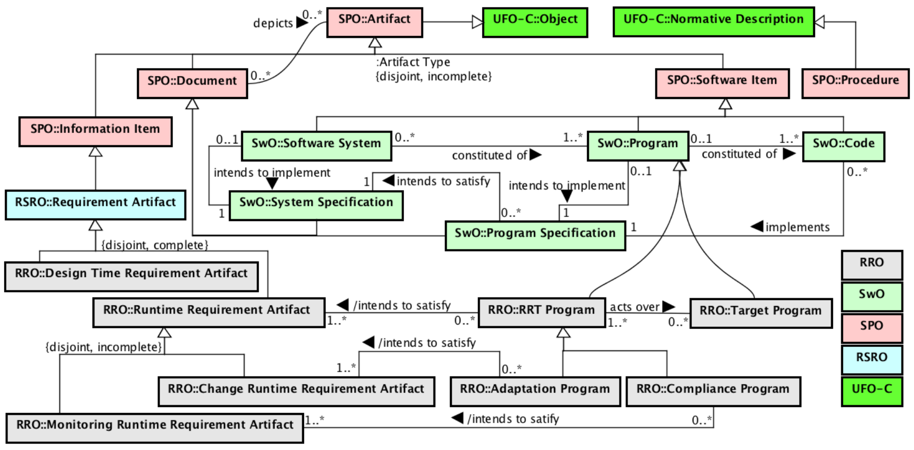

Fig. 1. Fragments of SwO, RSRO and RRO, showing concepts reused by RASO.

data definitions expressed, e.g., in a programming language. A Program is *constituted by* Code, but it is not identical to Code. Code can be changed without altering the identity of its Program, which is anchored to the program's essential property: its intended Program Specification, which the Code *implements*.

Given our focus on requirements-based approaches, we reuse from RSRO the concept of Requirement Artifact, an Information Item that describes a stakeholder's requirement (concepts also present in RSRO, but out of the scope of this paper), most likely as the result of some requirements documentation activity in the Requirements Engineering process. In the context of adaptive systems, we are particularly interested in a Requirement Artifact specified in RRO, namely, a Runtime Requirement Artifact (RRA). Unlike Design Time Requirement Artifacts, RRAs are manipulated by running Programs, i.e., at runtime. Such Programs are defined as RRT (Requirements at Runtime) Programs, which *intend to satisfy* one or more RRAs, *acting over* other running Programs, referred to as Target Programs. RRO specifies two particular types of use of requirements at runtime: Compliance Programs *intending to satisfy* Monitoring RRAs and Adaptation Programs *intending to satisfy* Change RRAs.

3 Requirements for Adaptive Systems Ontology

In this section we present the Requirements for Adaptive Systems Ontology (RASO) a domain ontology about requirements-based approaches for the development of adaptive systems. Following the first steps o SABiO, we defined the purpose of RASO:

1. To serve as conceptual basis to solve interoperability problems among different requirements-based approaches for the development of adaptive systems, thus serving as a well-founded vocabulary to improve understanding

and knowledge sharing in the domain of requirements for adaptive systems, being especially helpful for Requirements Engineers in this domain;
2. To serve as a supporting tool for the creation or reengineering of requirements-based approaches for the development of adaptive systems;
3. To serve as conceptual model for the creation and/or integration of tools that aid in the activities (1) and (2) above.

Given its puposes, requirements for RASO were elicited and documented in the form of Competency Questions (CQs) through a highly iterative process of requirements elicitation/documentation and ontology capture/formalization. The CQs for RASO are: **CQ1:** What is an adaptive system? **CQ2:** What is an adaptive system constituted by? **CQ3:** What kind of control is performed by an adaptive system? **CQ4:** How are adaptive systems specified? **CQ5:** What requirements at runtime does the adaptive system manipulate?

In order to elicit and understand the concepts involved in approaches that use requirements models for the development of adaptive systems, we conducted a systematic mapping of the literature [21] and analyzed the state-of-the-art in this particular field. Figure 2 shows an overview of this process, which is summarized next. A complete description of the mapping is available in https://goo.gl/fpYdpH.

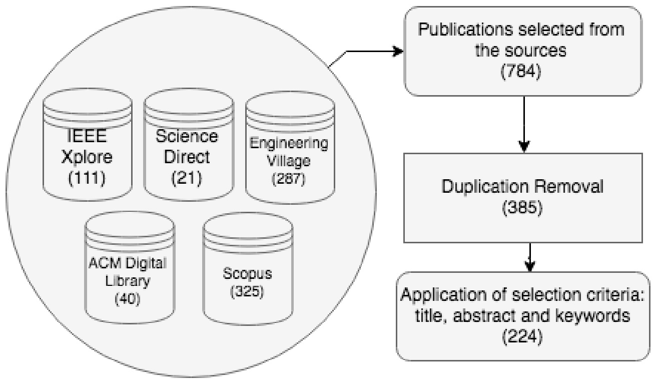
IEEE Xplore (111)
Science Direct (21)
Engineering Village (287)
ACM Digital Library (40)
Scopus (325)
Publications selected from the sources (784)
Duplication Removal (385)
Application of selection criteria: title, abstract and keywords (224)

Fig. 2. Overview of the systematic mapping protocol for RASO.

Based on the requirements for RASO, research questions were defined for the systematic mapping and a search string was elaborated and tested against a set of control articles that were supposed to return when the string was used in the selected publication sources, namely ACM Digital Library, Engineering Village, IEEE Xplore, Science Direct and Scopus. The set of control articles was taken from a previous systematic mapping effort on the broader topic of requirements at runtime [10], by selecting publications that were classified as *Change Requirements*, i.e., in which requirements are used as guidelines to monitoring and as rules on how the system should adapt to keep satisfying its requirements. In an iterative process, the search string was refined with key terms present in the meta-data (title, abstract, keywords) of control articles that were missing from the first executions, resulting in the following final string:

```
("requirements model" OR "requirements engineering" OR "requirements analysis" OR
"requirements reasoning" OR "goal model" OR "gore" OR "goal analysis" OR "goal reasoning")
AND ("adaptive system" OR "adaptive systems" OR "self-tuning" OR "runtime adaptation"
OR "self-adaptive" OR "self-adaptation" OR "self-optimization" OR "self-adaptivity" OR
"software adaptation" OR "self-configuration" OR "self-healing" OR "self-protection")
```

Running the above query, a total of 784 entries returned from the different publication sources, resulting in 385 articles after duplicates (same paper returning in different sources) were merged. We then excluded 161 entries considering inclusion/exclusion criteria, applied in two phases (meta-data only and full article), finally resulting in 224 selected publications. Due to the large number of selected publications, we proposed a new filter in order to select a smaller set of requirements-based self-adaptation approaches, which would be studied as part of ontology capture for RASO. Given that an ontology should provide a conceptualization that is shared by a certain community, we ranked the publications by their popularity (citations), as follows: (1) extract the total number of citations from Google Scholar; (2) divide it by the age in years, normalizing to citations per year; (3) rank the publications. The top 10 papers were selected and used not only for building RASO, but also for validating it (cf. Section 4).

Figure 3 shows the conceptual model of RASO, relating concepts of its domain to the ones reused from the Software Engineering Ontology Network (SEON, cf. Section 2). In what follows, we show how RASO provides a clear and precise description of this domain.

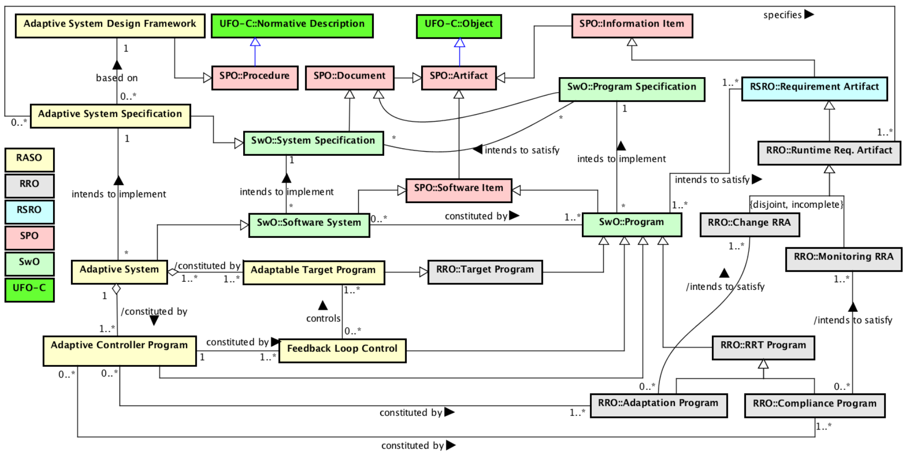

Fig. 3. The Requirements for Adaptive Systems Ontology (RASO).

An Adaptive System is a special kind of Software System, *constituted by* one or more Adaptive Controller Programs and base programs (often referred to in the literature as *target systems* [31]), represented in RASO as Adaptable Target

Programs. For instance, the Meeting Scheduler system, a classic exemplar in the Requirements Engineering community [22], is potentially a Target Program. Once it is integrated into an Adaptive System (in other words, it is made adaptive using any given approach for the development of adaptive systems), it becomes an Adaptable Target Program. Considering the Zanshin approach [31] in particular, this integration consists on implementing a callback API specified by Zanshin.

As Programs, both the Adaptable Target Program and the Adaptive Controller Program *intend to implement* a Program Specification in order to *satisfy* Requirement Artifacts. The former implements what we informally call the *main functions* of the system, e.g., schedule meetings, whereas the latter implements *adaptation functions*, i.e., monitor the main functions and adapt them if needed. The Adaptive Controller Program does this by means of one or more Feedback Loop Controls, i.e., Programs implementing monitoring–adaptation loops (e.g., the MAPE loop [20]) over the Adaptable Target Program [34]. For instance, if the Self-Adaptive Meeting Scheduler (adaptive system composed of base program and controller program) detects a low participation rate in meetings, it could send e-mails to invited participants to properly collect their timetables.

Software Systems *intend to implement* a System Specification [10]. As such, an Adaptive System *intends to implement* an Adaptive System Specification. In particular, besides specifying the main functions of the system, an Adaptive System Specification *specifies* one or more Runtime Requirement Artifacts (RRAs) regarding the system's self-adaptation functions. This particular kind of specification is created *based on* an Adaptive System Design Framework, which SPO [5] considers a Procedure and, at the foundational level, is a type of Normative Description. For instance, in [31], Souza presents an Adaptive System Specification for a Meeting Scheduler Adaptive System, based on the Zanshin Adaptive System Design Framework, containing *Awareness Requirements* (Monitoring RRAs) to monitor some of the Meeting Scheduler's functions (e.g., low participation in scheduled meetings) and *Evolution Requirements* (Change RRAs) to adapt these functions if necessary (e.g., start collecting timetables properly via e-mail).

Finally, the self-adaptation functions in the Adaptive System Specification are implemented by the Adaptive System by means of its Adaptation Controller Programs, which are *constituted by* two particular kinds of RRT (Requirements at Runtime) Programs: Compliance Programs *intend to satisfy* the Monitoring RRAs in the specification and Adaptation Programs *intend to satisfy* the Change RRAs in the specification. Again, using Souza's work as example [31], the Zanshin approach provides an implementation of an Adaptive Controller Program based on the Eclipse(.org) platform and constituted by OSGi(.org) bundles, among which the *Monitoring Bundle* (Compliance Program) monitors the satisfaction of *Awareness Requirements* (Monitoring RRAs) and reports to the *Adaptation Bundle*, which analyze *Evolution Requirements* (Change RRAs) in order to adapt.

Among the concepts presented above, the main ones are the Adaptive System Design Framework (which has the guidelines for the proposed system construction model) and the Adaptive System composition as a junction of an Adaptable Target

Program (target adaptation program) and the Adaptive Controller Program (which will monitor and trigger adaptations when necessary).

Other examples of Adaptive System Design Frameworks were used to instantiate RASO in the next section.

4 Evaluation

In order to evaluate the Requirements for Adaptive Systems Ontology (RASO), we applied verification and validation techniques, as prescribed by SABiO. For verification, SABiO suggests a table that shows the ontology elements that are able to answer the competency questions (CQs) that were raised, thus demonstrating that the ontology satisfies the requirements as they were documented. For validation, the ontology should be instantiated using real-world entities, demonstrating that it fulfills its intended purposes.

Table 1 illustrates the results of verification of RASO regarding the predefined CQs, which can also be used as a traceability tool, supporting ontology change management. The table shows that RASO answers all of its CQs appropriately. Note also that all concepts of RASO are mentioned in Table 1, which shows they are both necessary and sufficient to fulfill the ontology's requirements.

Table 1. Verification of RASO's Competency Questions.

CQ	Concepts and *Relations*
CQ1	Adaptive System is *a subtype of* Software System that *intends to implement* Adaptive System Specification.
CQ2	Adaptive System is *constituted by* Adaptable Target Program (*a subtype of* Target Program) and Adaptive Controller Program (*a subtype of* Program).
CQ3	Adaptable Controller Program is *constituted by* Feedback Loop Control (*a subtype of* Program), which *controls* Adaptable Target Program.
CQ4	Adaptive System Specification is *a subtype of* System Specification which, *based on* Adaptive System Design Framework (*a subtype of* Procedure), *specifies* Runtime Requirement Artifact.
CQ5	Adaptive Controller Program is *constituted by* Compliance Program (*a subtype of* RRT Program), which *intends to satisfy* Monitoring RRA (a *subtype of* Runtime Requirement Artifact), and Adaptation Program (*a subtype of* RRT Program), which *intends to satisfy* Change RRA (a *subtype of* Runtime Requirement Artifact).

For validation, we studied the 8 different approaches described by the top 10 papers with most citations per year from our systematic mapping of the literature on requirements-based approaches for the development of adaptive systems. Publications about these approaches that were outside the top 10 citations per year were also included in this study in order to provide an instantiation table that is as representative and complete as possible. Then, we identified instances of concepts from RASO (and a few key concepts of RRO as well) in each of the approaches, producing the instantiation table shown in Table 2. Note that an m-dash (—) in a cell does not mean that the concept is not present at all in the respective approach, but instead that it is not explicitly named.

The successful instantiation of RASO with entities from these approaches is an indication of the appropriateness of the proposed ontology as a reference

Table 2. Validation of RASO by instantiating its concepts according to popular approaches for the development of adaptive systems.

Concept	**Approach for the Development of Adaptive Systems**			
Adaptive System Design Framework	**Adaptive STS** [8]	**FLAGS** [2]	**LoREM** [13]	**Unnamed approach applied to Necesity** [4]
Adaptive System	Combination of the Smart Home example with the adaptation controller	Combination of the Laundry System example with the adaptation controller	The GridStix system	The Necesity system and the Context Management Middleware
Adaptive System Specification	Plan specifications for the Smart Home example	Goal model specification for the Laundry System example	Goal model specification for GridStix	Adaptive timers algorithm
Adaptable Target Program	Smart Home example implementation	Laundry System example implementation	GridStix's steady-state system implementation	The Necesity system implementation
Adaptive Controller Program	Self-Reconfiguration component	FLAGS framework implementation	Adaptation infrastructure	Context Management Middleware
Feedback Loop Control	Monitor-Diagnose-Reconcile-Compensate (MDRC) cycle	Goal Reasoner	—	—
Compliance Program	Monitor component	Process Reasoner	Monitoring mechanism	Monitoring software
Monitoring Runtime Requirement Artifact	Fulfillment conditions	Condition	Decision-making mechanism	Set of decision rules
Adaptation Program	Reconfiguration component	Adaptator Component	Adaptation mechanism	Adapter component
Change Runtime Requirement Artifact	Activation rules	Adaptive Goal	Adaptive step	Parameters which are set based on the history of use of the system

Concept	**Approach for the Development of Adaptive Systems**			
Adaptive System Design Framework	**QoS-aware Middleware** [25]	**RELAX** [35]	**Tropos4AS** [24]	**Zanshin** [33]
Adaptive System	Combination of a video streaming application with the QoS Middleware	—	Combination of the iCleaner example with the Tropos4AS Middleware	Combination of the Meeting Scheduler example with the Eclipse/OSGi-based implementation
Adaptive System Specification	Qos Specification	Goal model for the Ambient Assisted Living example	Goal model for the iCleaner example	Goal model for the Meeting Scheduler example
Adaptable Target Program	A video streaming application	—	iCleaner example implementation	Meeting Scheduler example implementation
Adaptive Controller Program	QoS Middleware	—	Tropos4AS Middleware	The Eclipse/OSGi-based implementation
Feedback Loop Control	QoS-aware resource management	—	Monitor-Analyze-Plan-Execute (MAPE) loop	Monitor-Analyze-Plan-Execute (MAPE) loop
Compliance Program	Observer component	—	Monitoring BDI agent	Monitoring OSGi bundle
Monitoring Runtime Requirement Artifact	—	Specification using MON operator	Different goal types	Awareness Requirement
Adaptation Program	Component configurator	—	Adaptation BDI agent	Adaptation OSGi bundle
Change Runtime Requirement Artifact	Application adaptation policies	Relaxed requirements	Recovery activities	Evolution Requirement

model of this domain. Table 2 also shows that RASO, as a conceptual model, does not present the problems discussed by Guizzardi [15]:

- *Construct Overload*: the concepts of RASO represent a single entity in the approaches, i.e., they are not overloaded with meaning;
- *Construct Excess*: all concepts of RASO have instances in most of the approaches, i.e., there is no unnecessary concepts in the ontology with respect to the domain;
- *Construct Redundancy*: no two concepts of RASO point to the same entity in any of the approaches, i.e., the concepts of the ontology are not redundant;
- *Incompleteness*: the entities related to the domain of adaptive systems in the different approaches were all mapped to a concept of RASO, i.e., the ontology is complete with respect to its given scope.

As such, it can serve as conceptual basis to solve interoperability problems between different approaches for the development of adaptive systems — Table 2 even serves as starting point for mapping concepts among the 8 selected approaches. RASO can also be used to develop or integrate existing software tools in this domain, as done, e.g., in the domain of software measurement [12].

Another interesting use of the ontology is to perform ontological analysis of the modeling languages used in these approaches, offering recommendations to clarify their semantics and ensure expressiveness, as done, for instance, in the domain of service-oriented enterprise architectures [26]. Table 2 even provides us with hints regarding the 8 selected approaches, as they could also be evaluated with respect to lucidity, soundness, laconicity and completeness [15], as we did with RASO above. For instance, we were not able to instantiate any of the Program concepts for the RELAX approach, as the papers define the specification language but do not provide an implementation. As another example, the QoS-aware Middleware approach does not make explicit their Monitoring Runtime Requirement Artifact. Such feedback could help improve these approaches.

In general, the evaluation of RASO shows that it can act as a well-founded vocabulary of the domain of requirements-based adaptive systems design, improving communication, learning and problem-solving in this domain.

5 Related Work

Qureshi et al. [28] proposed a new version of the Core Ontology for Requirement Engineering (CORE) [17, 19], introducing two new concepts (*context* and *resource*) and relations (*relegation* and *influence*) on top of CORE (an ontology concerning Requirements Engineering in general) in order to properly represent possible changes that might occur in requirements, at runtime. The authors claim that combining the new elements with the ones that are originally used by the goal modeling language Techne [18], they are able to support the definition of the runtime requirements adaptation problem. In comparison with RASO, the ontology of Qureshi et al. does not represent concepts that were found in the approaches selected in our systematic mapping of the literature, such as Monitoring

RRA and Change RRA, for example. Instead, it includes concepts and relations that are not strongly related to this domain (i.e., were not commonly found in the selected approaches, such as *resource* and *relegation*).

Soares et al. [32] propose a core ontology to assist requirements elicitation and specification for adaptive systems, based on the ontology of Qureshi et al. [28] — which they deemed incomplete —, completing it with concepts extracted from the modeling dimensions for adaptive systems [1]. Besides inheriting the aforementioned problems of [28], the ontology of Soares et al. is not properly based on a foundational ontology, is presented only in its operational form (in OWL) and includes concepts pertaining to the area of context-aware systems (claiming they subsume adaptive systems, a claim which we find debatable). On the other hand, RASO is a reference ontology, founded on UFO and focused exclusively on the concepts of the adaptive systems domain.

Reinhartz-Berger et al. [29] propose a conceptual model of software behavior based on the foundational ontology by Bunge [6], which can be used to model the expected behavior of a system (its requirements) and alternative behaviors that the running program can actually perform, thus supporting self-adaptation decisions by comparing alternatives on: (1) how well they meet the requirements and (2) the effort needed to switch behaviors. Their work uses an ontological approach to model the system's behavior in order to foster self-adaptation, i.e., it is a requirements-based approach for the development of adaptive systems. Our work instead provides an ontology for the domain of requirements for adaptive-systems, i.e., it describes concepts used by different approaches (including [29], which could be instantiated like the works included in Table 2).

Finally, regarding the systematic mapping study we present in Section 3, Yang et al. [36] performed a systematic literature review to investigate what modeling methods, activities, quality attributes, application domains and research topics have been studied in the area of requirements engineering for adaptive systems and how well these studies have been conveyed. Our mapping, although more superficial than a review, bears some similarities with their work, such as the research method [21]. We did consider the possibility of using their study as the knowledge base for building RASO instead of performing a new one. However, since this systematic mapping is already five years old, we decided that a new study was needed in order to not neglect the recent advances that were achieved in this research area. Moreover, updating their review would have taken much more effort than producing a new mapping.

Our systematic mapping study showed that a almost a fifth of the requirements-based approaches for the development of adaptive systems make use of ontologies as part of their proposed methods. Among them, CORE [17, 19] is the most cited work. However, as mentioned before, CORE represents the domain of Requirements Engineering in general, whereas RASO focuses on Requirements Engineering for adaptive systems (based on RSRO [10] for the reasons discussed in Section 2). This shows both a growing interest in the use of ontologies in this domain and a gap for ontologies that are specifically tailored for adaptive systems, which RASO intends to fill.

6 Conclusions

This paper presented RASO, the Requirements for Adaptive Systems Ontology, a domain reference ontology about requirements-based development of adaptive systems. RASO was built using the SABiO ontology engineering method, reusing concepts from existing ontologies on Software Engineering and built on the foundational ontology UFO. Requirements elicitation and ontology capture were based on selected publications from a systematic mapping of the literature regarding this domain. The ontology was evaluated by verifying that its proposed conceptual model answers all its Competency Questions and by instantiating real-world entities extracted from the most cited approaches from the systematic mapping of the literature using the concepts defined by RASO.

Given how RASO was elicited and validated, we believe it successfully represents the state-of-the-art on requirements-based approaches for the development of adaptive system, achieving its purposes of serving as a well-founded vocabulary to improve understanding and knowledge sharing in this domain and as conceptual basis to solve interoperability problems or for the creation or reengineering of approaches for adaptive systems development.

As future work, we are considering: (a) studying the approaches returned in our systematic mapping study that did not make the top 10 citations per year cut, in order to validate RASO or complement its conceptual model; (b) providing a more formal characterization of RASO; (c) integrating RASO in the Software Engineering Ontology Network mentioned in Section 2; (d) building an operational version of RASO in order to implement a prototype on interoperability among different requirements-based approaches for adaptive systems; (e) combined with the Goal-Oriented Requirements Ontology (GORO) [27], performing analysis and reengineering of the Zanshin approach, including its modeling language and Adaptive Controller Program implementation; and (f) building an method for evaluating adaptive systems, as well as creating metrics to qualitatively measure these systems.

Acknowledgment

NEMO(.inf.ufes.br) is currently supported by CNPq (process 407235/2017-5), CAPES (process 23038.028816/2016-41), and FAPES (process 69382549/2015). We would also like to thank Ricardo de Almeida Falbo for his contributions.

References

1. Andersson, J., de Lemos, R., Malek, S., Weyns, D.: Modeling Dimensions of Self-Adaptive Software Systems. In: Software Engineering for Self-Adaptive Systems, vol. 5525, pp. 27–47. Springer (2009)
2. Baresi, L., Pasquale, L., Spoletini, P.: Fuzzy Goals for Requirements-driven Adaptation. In: Proc. of the 18th IEEE International Requirements Engineering Conference. pp. 125–134. IEEE (sep 2010)

3. Borst, W.N., Borst, W.: Construction of engineering ontologies for knowledge sharing and reuse (1997)
4. Botia, J.A., Villa, A., Palma, J.: Ambient Assisted Living system for in-home monitoring of healthy independent elders. Expert Systems with Applications **39**(9), 8136–8148 (2012)
5. Bringuente, A.C.O., Falbo, R.A., Guizzardi, G.: Using a Foundational Ontology for Reengineering a Software Process Ontology. Journal of Information and Data Management **2**(3), 511–526 (2011)
6. Bunge, M.: Treatise on basic philosophy, Vol. 3: Ontology I: The furniture of the world (1977)
7. Cheng, B.H.C., et al.: Software Engineering for Self-Adaptive Systems: A Research Roadmap. In: Software Engineering for Self-Adaptive Systems, vol. 5525, pp. 1–26. Springer (2009)
8. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Adaptive socio-technical systems: a requirements-based approach. Requirements Engineering **18**(1), 1–24 (2012)
9. Dermeval, D., Vilela, J., Bittencourt, I.I., Castro, J., Isotani, S., Brito, P., Silva, A.: Applications of ontologies in requirements engineering: a systematic review of the literature. Requirements Engineering **21**(4), 405–437 (2016)
10. Duarte, B.B., Leal, A.L.d.C., Falbo, R.d.A., Guizzardi, G., Guizzardi, R.S.S., Souza, V.E.S.: Ontological Foundations for Software Requirements with a Focus on Requirements at Runtime. Applied Ontology **preprint**(preprint), 1–33 (2018). https://doi.org/10.3233/AO-180197
11. Falbo, R.A.: SABiO: Systematic Approach for Building Ontologies. In: Proc. of the Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering. CEUR (2014)
12. Fonseca, V.S., Barcellos, M.P., de Almeida Falbo, R.: An ontology-based approach for integrating tools supporting the software measurement process. Science of Computer Programming **135**, 20–44 (2017)
13. Goldsby, H.J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Hughes, D.: Goal-Based Modeling of Dynamically Adaptive System Requirements. In: Proc. of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems. pp. 36–45. IEEE (mar 2008)
14. Grüninger, M., Fox, M.: Methodology for the Design and Evaluation of Ontologies. In: IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing (1995)
15. Guizzardi, G.: Ontological foundations for structural conceptual model. Ph.D. thesis, University of Twente, The Netherlands (2005)
16. Guizzardi, G.: On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. Frontiers in artificial intelligence and applications **155**, 18 (2007)
17. Jureta, I., Mylopoulos, J., Faulkner, S.: Revisiting the core ontology and problem in requirements engineering. In: International Requirements Engineering, 2008. RE'08. 16th IEEE. pp. 71–80. IEEE (2008)
18. Jureta, I.J., Borgida, A., Ernst, N.A., Mylopoulos, J.: Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: Requirements Engineering Conference (RE), 2010 18th IEEE International. pp. 115–124. IEEE (2010)
19. Jureta, I.J., Mylopoulos, J., Faulkner, S.: A core ontology for requirements. Applied Ontology **4**(3-4), 169–244 (2009)
20. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (jan 2003)

21. Kitchenham, B.A., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Tech. rep., Keele University, UK (2007), https://www.cs.auckland.ac.nz/ mria007/Sulayman/Systematic_reviews_5_8.pdf
22. van Lamsweerde, A., Darimont, R., Massonet, P.: Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt. In: Proc. of the 2nd IEEE International Symposium on Requirements Engineering. pp. 194–203. IEEE (mar 1995)
23. de Lemos, R., et al.: Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In: Software Engineering for Self-Adaptive Systems II, pp. 1–32. Springer (2013)
24. Morandini, M., Penserini, L., Perini, A., Marchetto, A.: Engineering requirements for adaptive systems. Requirements Engineering **22**(1), 77–103 (2017)
25. Nahrstedt, K., Xu, D., Wichadakul, D., Li, B.: QoS-Aware Middleware for Ubiquitous and Heterogeneous Environments. IEEE Communications Magazine **39**(11), 140–148 (2001)
26. Nardi, J.C., de Almeida Falbo, R., Almeida, J.P.A.: An Ontological Analysis of Service Modeling at ArchiMate's Business Layer. In: Proc. of the 18th International IEEE Enterprise Distributed Object Computing Conference. pp. 92–100. IEEE (2014)
27. Negri, P.P., Souza, V.E.S., Leal, A.L.d.C., Falbo, R.A., Guizzardi, G.: Towards an Ontology of Goal-Oriented Requirements. In: Proc. of the 20th Ibero-American Conference on Software Engineering, Requirements Engineering track (may 2017)
28. Qureshi, N.A., Jureta, I.J., Perini, A.: Requirements engineering for self-adaptive systems: Core ontology and problem statement. In: International Conference on Advanced Information Systems Engineering. pp. 33–47. Springer (2011)
29. Reinhartz-Berger, I., Sturm, A., Wand, Y.: Comparing functionality of software systems: An ontological approach. Data & Knowledge Engineering **87**, 320–338 (2013)
30. Ruy, F.B., Falbo, R.d.A., Barcellos, M.P., Costa, S.D., Guizzardi, G.: SEON: A Software Engineering Ontology Network. In: Proc. of the 20th International Conference on Knowledge Engineering and Knowledge Management. pp. 527–542. Springer (2016)
31. Silva Souza, V.E.: Requirements-based Software System Adaptation. Ph.D. thesis, University of Trento, Italy (2012)
32. Soares, M., Vilela, J., Guedes, G., Silva, C., Castro, J.: Core Ontology to Aid the Goal Oriented Specification for Self-Adaptive Systems. In: New Advances in Information Systems and Technologies, pp. 609–618. Springer (2016)
33. Souza, V.E.S., Lapouchnian, A., Robinson, W.N., Mylopoulos, J.: Awareness Requirements for Adaptive Systems. In: Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 60–69. ACM (may 2011)
34. Souza, V.E.S., Mylopoulos, J.: From Awareness Requirements to Adaptive Systems: a Control-Theoretic Approach. In: Proc. of the 2nd International Workshop on Requirements@Run.Time. pp. 9–15. IEEE (aug 2011)
35. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.M.: RELAX: a language to address uncertainty in self-adaptive systems requirement. Requirements Engineering **15**(2), 177–196 (2010)
36. Yang, Z., Li, Z., Jin, Z., Chen, Y.: A systematic literature review of requirements modeling and analysis for self-adaptive systems. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 55–71. Springer (2014)