GORO 2.0: Evolving an Ontology for Goal-Oriented Requirements Engineering

César Henrique Bernabé[1], Vítor E. Silva Souza[1], Ricardo de Almeida Falbo[1], Renata S. S. Guizzardi[1], and Carla Silva[2]

[1] Ontology and Conceptual Modeling Research Group (NEMO)
Department of Computer Science, Federal University of Espírito Santo (UFES), Brazil
{chbernabe,vitorsouza,falbo,rguizzardi}@inf.ufes.br
[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Brazil,
ctlls@cin.ufpe.br

Abstract. Goal-Oriented Requirements Engineering (GORE) gained prominence by covering some of the limitations of traditional Requirements Engineering (RE). As a result, many GORE modeling languages have been proposed since this field emerged. Aiming at providing formal semantics to the concepts of GORE, the Goal-Oriented Requirements Ontology (GORO) was proposed as a common vocabulary for this domain. However, the first version of GORO lacks important concepts and its applicability was not demonstrated in practice. In this paper, we present GORO 2.0, an evolution of the first version of GORO that overcomes several limitations of its first version, presenting new concepts such as obstacles, conflicts and contributions.

Keywords: Goal-Oriented Requirements Engineering · Goal Modeling · Ontology.

1 Introduction

Goal-Oriented Requirements Engineering (GORE) emerged in the mid-1990s and became popular for overcoming some of the limitations of traditional Requirements Engineering (RE). For example, goals provide precise criteria for requirements completeness and adequate rationale and justification for a requirement's existence [15]. They are also an efficient tool for identification and negotiation of conflicts [16]. As a result, many GORE modeling languages have been proposed since this field emerged [12].

The multitude of languages and their constructs motivated the creation of the Goal-Oriented Requirements Ontology (GORO), which was proposed with the aim of providing formal semantics to the concepts of GORE [20]. As a consequence, GORO can be used to enable interoperability between models from different GORE languages as it provides a common vocabulary about the GORE domain (and, therefore, improves the communication between stakeholders). Moreover, GORO allows previous and new modeling languages to clearly specify their semantics by grounding their concepts in a formal reference ontology.

By providing a common vocabulary, the ontology can also support modelers to create ontologically correct models.

The first version of GORO, however, suffers from some limitations, namely: (i) the ontology was captured from and had its concepts mapped to only three GORE languages (*i** [24], KAOS [7] and Techne [3]) and considered only a subset of concepts of these languages; (ii) it lacks integration with other ontologies on the Software Engineering (SE) domain to strengthen its semantic foundations; and (iii) its applicability was not properly demonstrated as, for example, using a model conversion tool. Hence, we evolved GORO into a new version, hereafter *GORO 2.0*, in order to overcome the aforementioned limitations.

This paper presents GORO 2.0 and is organized as follows: Section 2 briefly summarizes the GORE domain; Section 3 presents the method used to build GORO 2.0; Section 4 presents GORO 2.0; Section 5 compares our ontology with related work; and Section 6 concludes the paper.

2 GORE Modeling Languages

NFR [19] was the first GORE language proposed (1992) and brought the concept of goals as desirable qualities in a system. It introduced the concept of contribution between Softgoals (goals without clear criteria of satisfaction). In 1993, KAOS was proposed and redefined goals as states of affairs desired by stakeholders, categorizing them as Goal (not sufficiently refined to be assigned to a stakeholder), Expectation (under the responsibility of a human agent) and Requirement (under the responsibility of a software agent). It also introduced the concepts of Operation (a task/plan that can be performed to achieve a goal), Domain Property (a presupposition about the system context considered to be true in certain situations) and Obstacle (an undesired behavior in the context).

In 1995, Yu formalized the specification of *i**, which focuses on the representation of stakeholders' interests within the organizational context. The *i** core concept is the Actor, which depends on others to accomplish goals and perform tasks. The language also highlighted the differentiation between Goal and Softgoal: the former would have a clear satisfaction criteria, while the latter did not. In the following year, GBRAM [2] emerged and defined a method for goal analysis in which the concept of Scenarios, a description of a system and its environment, is used to identify Goals and Obstacles.

In 2004, GSN [14] was proposed with focus on security systems, such as information security, air traffic control and safety systems. In the same year, Tropos [4], a variation of *i**, emerged and brought the concept of Capability as the "ability of an actor of defining, choosing and executing a plan for the fulfillment of a goal". In 2009, the first version of Techne was presented. Based on an ontology, Techne made a more precise differentiation between Hard and Softgoals, as the latter can be restricted through Quality Constraints.

In 2010, GRL [1], a variation of *i**, introduced a differentiation of the OR-Decomposition relation (exclusive and inclusive) and the concept of Correlation (a relation of side effects rather than desired impacts as in the contribution rela-

tion). Finally, in 2016, *i** was revised and its second version, now spelled iStar, had some elements and relationships removed, and new elements that were popularly used by the community were added. For instance, Softgoal was renamed as Quality; and the means-end and task-decomposition links were grouped in the Refinement Link.

3 Method

GORO 2.0 was built using the Systematic Approach to Building Ontologies (SABiO) [8], an Ontology Engineering method, successful in the development of domain ontologies, particularly in SE. To provide a solid semantic foundation, GORO 2.0 is based on the Unified Foundational Ontology (UFO) [10], and reuses existing ontologies, such as the Common Ontology for Value and Risk (COVR) [23] and the Reference Software Requirements Ontology (RSRO), which is part of the Software Engineering Ontology Network (SEON) [22].

In order to improve domain coverage, GORO 2.0 was created based on the modeling languages mentioned in Section 2, which were studied and analyzed to extract concepts that, in fact, belong to the GORE domain. GBRAM, GRL, i*, KAOS, Techne and Tropos were first selected based on a literature review [12]. NFR was added to the list as it is cited in Van Lamsweerde's guided tour on GORE [15]. Finally, when searching for related works (cf. Section 5), we also identified GSN. The selected languages were validated with domain experts who advised us to consider *i** and *iStar* as different languages, given the perceptible differences between them. GORO, as its name states, is focused on Requirements. Hence, we do not consider other languages that use goal related constructs but are not specifically GORE modeling languages.

Regarding scope, we have applied two criteria for the inclusion of a construct: it must I_1: appear in more than two GORE modeling languages; and I_2: be considered a GORE concept by domain experts — a group of five academic professionals with more than ten years of experience. We applied I_1 in order to exclude constructs that were not GORE, but actually extra features of specific languages. In order to verify if different languages' elements shared the same meaning, we also consulted the group of experts. It is worth to highlight that GORO 2.0 is concerned with the part of the GORE domain that is covered by the languages selected according to the described heuristics. This decision has been made because one of the purposes of this work is to provide interoperability among the selected GORE languages. As a consequence, other concepts pertaining to GORE domain, but not covered in the selected languages, were not considered to be part of GORO 2.0.

To evaluate GORO, we conducted three activities: verification, validation and an application-based evaluation. To check whether GORO satisfies its own requirements, we verified if its conceptual model can answer all of the proposed Competency Questions (CQs). To validate GORO's domain coverage, we mapped concepts of the GORE modeling languages listed in Section 2 to the concepts of the ontology. Finally, to assess the feasibility of GORO in enabling

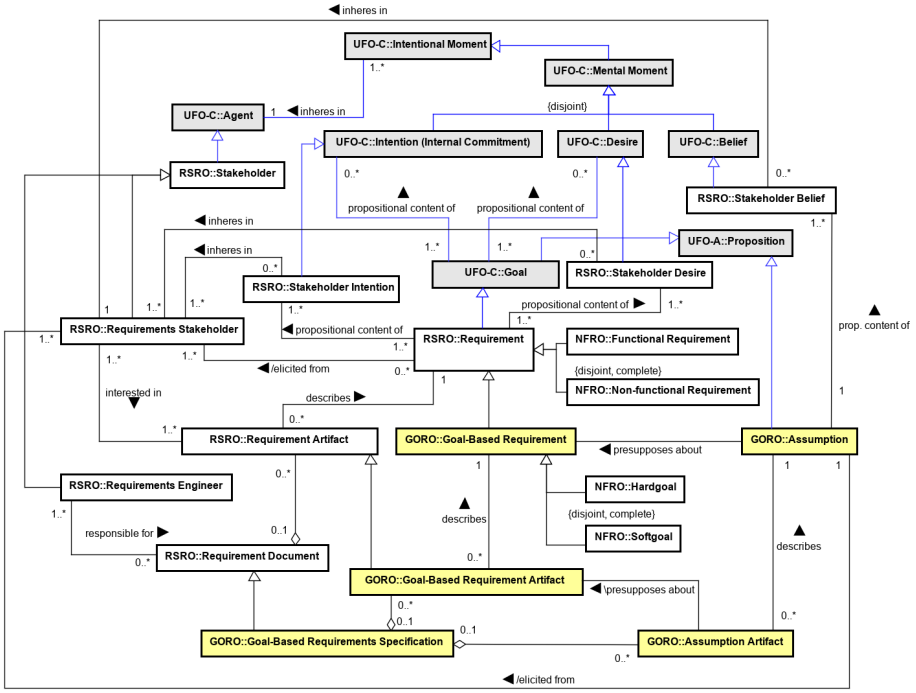

Fig. 1. GORO 2.0's first module: mental moments, goals and assumptions.

interoperability between GORE languages, we implemented a model conversion tool that uses the ontology as an interlanguage. The CQs, the concepts mapping and the conversion tool source code (and conversion examples) are available at https://nemo.inf.ufes.br/projects/rose/.

4 GORO 2.0

Figure 1 presents the module of GORO 2.0 that defines concepts related to mental moments existentially dependent on a single individual, which can be classified as Beliefs, Desires and Intentions. Agent's beliefs are assumed to be true in a given set of situations. Given that desires and intentions are both related to agents' goals, the difference between them is actually related to the fact that the former is only a will of an agent towards a state of affairs (situation) in reality, whereas the latter is an intended state of affair (situation) for which the agent commits to pursuing, causing the agent to perform actions [10].

GORO 2.0 inherits the Stakeholder definition from RSRO: a Stakeholder can be a Requirements Stakeholder, when in the role of the person that provides needs and expectations for the product, or a Requirements Engineer, when in the role of conducting the requirements development activities.

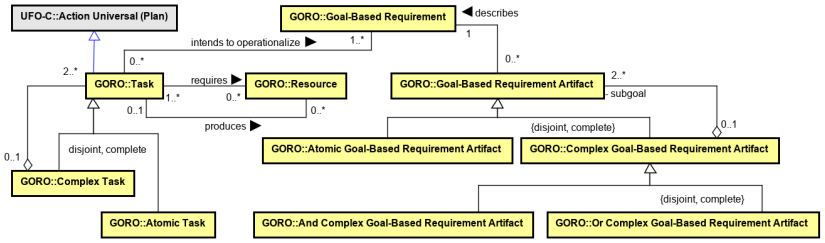

Fig. 2. GORO 2.0's second module: tasks, goals and relations.

A Goal is the *propositional content of* an Intention/Desire, which inheres in an Agent, *supertype* of Stakeholder. Therefore, a Requirement is a goal *elicited from* a stakeholder's intention/desire. A Requirement can be a Non-functional Requirement or a Functional Requirement. When applying a GORE approach to a Requirements Engineering process, a traditional Requirement becomes a Goal-Based Requirement, which can be a Hardgoal or a Softgoal. Both definitions are extracted from [17] and represented in GORO 2.0 with the NFRO prefix.

By combining two perspectives, we end up with four different classifications for a goal-based requirement [17]: Functional Requirement & Hardgoal, Functional Requirement & Softgoal, Non-functional Requirement & Hardgoal and Non-functional Requirement & Softgoal, implicitly represented in Figure 1. Hence, GORO 2.0 is compatible with NFRO, making adaptations where necessary. We highlight that such adaptations are now incorporated in NFRO. A Goal-Based Requirement Artifact *describes* a Goal-Based Requirement in the same way that a Requirement Artifact *describes* a Requirement, differentiating a documented requirement from a requirement that exists only in the stakeholder's mind. It is important to note that, in GORO 2.0, an Assumption still has the same classifications proposed in GORO 1.0 [20], not shown here due to space limitations and for not being a contribution of this paper.

Figure 2 shows GORO 2.0 second module. A Task *intends to operationalize* a Goal-Based Requirement. Tasks can be Complex Tasks, when composed of two or more Tasks, or Atomic Tasks otherwise. A Task can *require* or *produce* a Resource. As with Tasks, a Goal-Based Requirement Artifact can also be complex or atomic. Complex Goal-Based Requirement Artifact (or Complex GBRA) is further refined into Or/And-Complex GBRA, which are satisfied when at least one/all of their components are satisfied. GORO does not allow tasks to be refined into goals. Yu [24] argues that the refinement between goals and tasks is a way to capture the transition between the problem domain (goal) and the solution domain (task). In addition, according to him, refining a task into a goal would be natural in the analysis and modeling cycle, which generally iterates between these two domains. However, by ontologically analyzing these concepts, the relationship between a task and a goal is not a "refinement". Rather, the task analysis shows that new goals should be considered in the model. In other words, task analysis may motivate the "emergence" of new goals, possibly better characterized if different

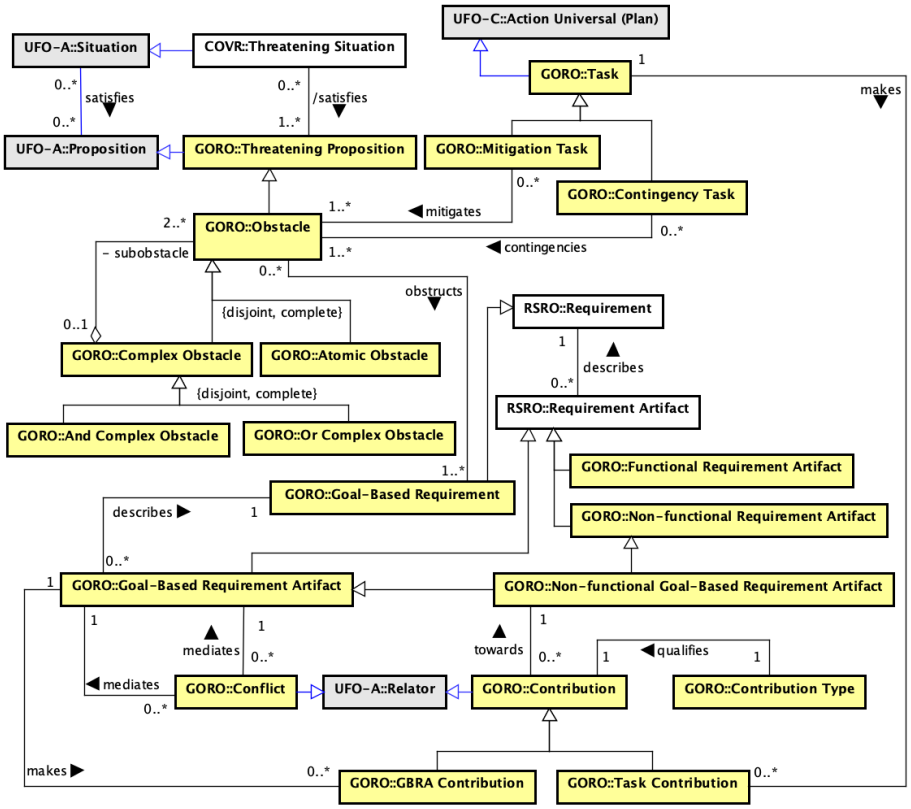

Fig. 3. GORO 2.0's third module: obstacles, conflicts and contribution.

models are created for the different analysis cycles. This is an example of how the ontological analysis performed with GORO may have methodological impact.

Figure 3 introduces concepts not previously considered in GORO 1.0, namely: obstacle, conflict and contribution. Van Lamsweerde [16] defines obstacle as a dual notion of goals: "while goals capture desired conditions, obstacles capture undesirable (but nevertheless possible) ones". We argue that obstacles can be equated, here, to the definition used by the Common Ontology for Value and Risk (COVR) [23], i.e., a condition that may be satisfied in certain situations in which something of human value has been put at stake and the outcome is uncertain. Thus, an Obstacle is seen as a Threatening Proposition, which *satisfies* a Threatening Situation and *obstructs* a Goal-Based Requirement satisfaction.

An Obstacle, according to Van Lamsweerde [16], can be *mitigated* by an agent's goal. We consider that what mitigates or contingencies a risk is an action (task) and not a goal. Although KAOS uses goals to mitigate obstacles, the task that intends to operationalize this goal is indirectly mitigating the obstacle. We also argue that this task definition is overloaded and, thus, propose two distinct

types of actions (tasks): contingency (action taken after a Threatening Situation to decrease damage) and mitigation (can reduce or prevent the risk rate of an event to happen). A Threatening Proposition *satisfies* a Threatening Situation in the same way as a Proposition *satisfies* a Situation, hence, the former relation is derived from the latter (denoted by a / symbol). Like goals, an Obstacle can be decomposed in complex/atomic ones, and Complex Obstacles are further refined in Or/And Complex Obstacles, with analogous satisfaction rules.

A conflict happens when two or more goals cannot be achieved in the same solution set of a domain problem [16]. In other words, given goals G_1 and G_2 and a model M, there is no solution set S of M that contains both G_1 and G_2. In GORO, Conflicts are modeled as a *relator* between Goal-based Requirement Artifacts, which potentially *conflicts* with another. It is important to emphasize the difference between conflict and obstacle: the former describes situations in which two goals cannot be achieved in the same solution, although both are desired by stakeholders, whereas the latter describes an undesired state of affairs.

In GORO 2.0, contributions are represented by a Contribution relator that stands between a Goal-Based Requirement Artifact or a Task and a Non-functional Goal-Based Requirement Artifact, which is a non-functional requirement used in a GORE approach. We argue that contributions should only have non-functional requirements as targets because: (i) in the case of total contributions, a negative contribution to a functional requirement would be semantically similar to a Conflict, while a positive contribution would have the same meaning of Complex GBRA or operationalization (*intends to operationalize*); (ii) in the case of partial contributions, it does not make sense to partially satisfy/deny a GBRA which, in turn, has a precise satisfaction criteria. Several GORE languages have certain types of *contribution* relations: *i**, iStar and the NFR Framework, for instance, have some types of contributions, e.g. *make*, *help*, *hurt* and *break* [24, 6, 19]. *Make* and *Break* are positive and negative contributions that sufficiently satisfy a non-functional requirement, respectively, whereas *Help* and *Hurt* are partial positive and negative contributions.

5 Related Works

The initial set of related work was raised based on relevant references of the area, such as Horkoff *et al.* [12] and Guizzardi *et al.* [11]. We consider as related works: (i) the ones that use ontologies as basis for analysis or construction of GORE languages, (ii) the ones that proposed metamodels with the purpose of unifying concepts of goal modeling languages.

Regarding the use of ontologies: the Core Ontology for Requirements Engineering (CORE) [13] has as main objective to review the conceptualization of several RE elements and is the foundation of Techne. However, it is based on a foundational ontology in which essential aspects of conceptual modeling (e.g., material relations and relational properties) have not received sufficiently detailed attention [20]. Guizzardi *et al.* [11], in turn, use UFO as a reference model

to analyze *i** and its many variants, therefore aiming to promote interoperability between them, but the work is constrained to the *i** family of languages.

Regarding the use of metamodels: in the work of Fayoumi *et al.* [9], the concepts of eight modeling languages are raised and organized in a metamodel, in which the main objective is the interoperability between GORE models. The work of Lucena *et al.* [18] presents a metamodel created to unify two variants of *i** (its original version and Tropos), considering similarities and differences between them. The work of Cares & Franch [5] defines a *supermetamodel* created on the basis of different variations of *i** (GRL and Tropos), which is validated through a translation algorithm that uses the XML-based iStarML format to depict the relation between tools. Patricio *et al.* [21] propose a unified GORE language called Unified Goal-oriented Language (UGL), which incorporates concepts of *i**, GRL and KAOS and whose metamodel is based on existing metamodels of *i** and KAOS. We argue that, unlike ontologies, metamodels do not provide sufficient semantic foundation to explain complex domain concepts. Metamodels are not efficient enough to promote interoperability between languages because, although they are powerful structures for defining the syntax of a language, they suffer for several limitations in relation to semantic clarifications [11].

6 Conclusions

In this paper, we defined GORO 2.0, a domain reference ontology about Goal-Oriented Requirement Engineering, built based on GORO 1.0, by including concepts related to GORE that had not yet been covered in its previous version. Nine goal modeling languages were chosen based on both literature review and experts' opinion. Their concepts were analyzed and those considered GORE concepts were included. Then, these same concepts were mapped to GORO 2.0 in order to verify and validate the new ontology. Further, a GORO-based tool that converts between two GORE languages (*iStar* and KAOS), was developed as a proof-of-concept. Evaluation results were not presented here due to space constraints, but are available at https://nemo.inf.ufes.br/projects/rose/. GORO 2.0 was built with a strong foundation as it was based on both relevant literature on GORE and on UFO [10]. It also reuses concepts from other ontologies, namely COVR [23] and RSRO. As the latter is part of SEON [22], GORO 2.0 becomes part of this ontology network as well.

By performing validation on GORO, in addition to verifying domain coverage, we were able to notice a few issues in the design of the analyzed languages. Regarding the relations between elements defined in each language, for instance, we could identify that some of them are overloaded. GORO defines decomposition of Goal-Based Requirement Artifacts (GBRA) (Fig. 2), Tasks (Fig. 2) and Obstacles (Fig. 3); the *Conflict* relator between GBRA (Fig. 3); Contribution relation between a GBRA and a Non-functional GBRA (Fig. 3); and finally, an Operationalization relation between a Task and a GBRA (Fig. 2). It was identified that some elements were, at the same time, both an And-Complex GBRA aggregation (when a GBRA is AND decomposed into other GBRAs) and an

operationalization relation (when a GBRA is operationalized into Tasks) or both an Or-Complex GBRA aggregation and an operationalization relation. This is the case, for instance, of Techne's Inference relation.

In terms of interoperability, it is important to mention that, in some cases, elements of a given language cannot be directly converted into elements of another. In this case, we plan to propose conversion patterns as future work. Currently, the tool proposed in this paper creates a log with the elements that were not converted, leaving the user to make the best decision regarding the new model.

In future works, we also intend to: (a) extend the model conversion tool, adding support for more GORE languages and improving its user interface; (b) use GORO 2.0 to make a systematic ontological analysis of GORE languages, verifying possible inconsistencies, construct overload, and other opportunities of improvement in such languages; (c) through the activities performed in (b), propose ontology-based modeling patterns to ensure consistency in the creation of GORE models; (d) use the ontology to identify and incorporate other GORE concepts that the current modeling languages do not cover; (e) use GORO as base for the abstract syntax of a more complete GORE language; and (f) improve the validity of GORE language constructs definition (which was interpreted by our domain experts group), by analysing models on the same subject with the help of GORO.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. NEMO(.inf.ufes.br) is currently supported by CNPq (processes 407235/2017-5, 433844/2018-3), CAPES (process 23038.028816/2016-41), and FAPES (process 69382549/2015).

References

1. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating goal models within the goal-oriented requirement language. International Journal of Intelligent Systems **25**(8), 841–877 (2010)
2. Anton, A.: Goal-based requirements analysis. In: Proc. of the 2nd Intl. Conference on Requirements Engineering (RE). pp. 136–144. IEEE Comput. Soc. Press (1996)
3. Borgida, A., Lapouchnian, A., Ernst, N., Liaskos, S., Jureta, I., Mylopoulos, J., Lapouchnian, A., Liaskos, S., Mylopoulos, J.: Techne : A(nother) Requirements Modeling Language. Tech. rep., Dept. Comput. Sci., University of Toronto (2010), ftp://www.cs.toronto.edu/dist/reports/csri/593/techne-techrep-v1.pdf
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems **8**(3), 203–236 (may 2004)
5. Cares, C., Franch, X.: A metamodelling approach for i* model translations. In: International Conference on Advanced Information Systems Engineering. Springer (2011)

6. Dalpiaz, F., Franch, X., Horkoff, J.: iStar 2.0 Language Guide. CoRR **abs/1605.07767** (2016)
7. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming **20**(1-2), 3–50 (apr 1993)
8. Falbo, R.A.: SABiO: Systematic approach for building ontologies. In: Proc. of the 1st Joint Ws. on Ontologies in Conceptual Modeling and Inf. Systems Engineering. vol. 1201. CEUR (2014)
9. Fayoumi, A., Kavakli, E., Loucopoulos, P.: Towards a Unified Meta-Model for Goal Oriented Modelling. In: Proc. of the 12th European, Mediterranean & Middle Eastern Conf. on Information Systems (EMCIS). pp. 1–10 (2015)
10. Guizzardi, G., 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: Proc. of the 11th IberoAmerican Conf. on Software Eng. (CIbSE) (2008)
11. Guizzardi, R., Franch, X., Guizzardi, G., Wieringa, R.: Using a foundational ontology to investigate the semantics behind the concepts of the i* language. In: Proc. of the 6th Intl. i* Workshop (iStar). vol. 978, pp. 13–18. CEUR (2013)
12. Horkoff, J., Aydemir, F.B., Cardoso, E., Li, T., Maté, A., Paja, E., Salnitri, M., Piras, L., Mylopoulos, J., Giorgini, P.: Goal-oriented requirements engineering: an extended systematic mapping study. pp. 1–28. Springer (2017)
13. Jureta, I.J., Mylopoulos, J., Faulkner, S.: A core ontology for requirements. Applied Ontology **4**(3-4), 169–244 (2009)
14. Kelly, T., Weaver, R.: The goal structuring notation—a safety argument notation. In: Proc. of Dependable Systems and Networks 2004 Ws on Assurance Cases (2004)
15. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Proc. of the 5th IEEE Intl. Symposium on Requirements Engineering. pp. 249–262. IEEE Comput. Soc (2001)
16. van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. IEEE Transactions on Software Engineering **26**(10), 978–1005 (2000)
17. Li, F.L., Horkoff, J., Mylopoulos, J., Guizzardi, R.S., Guizzardi, G., Borgida, A., Liu, L.: Non-functional requirements as qualities, with a spice of ontology. In: 2014 IEEE 22nd Int. Requirements Engineering Conf. (RE). pp. 293–302. IEEE (2014)
18. Lucena, M., Santos, E., Silva, C., Alencar, F., Silva, M.J., Castro, J.: Towards a unified metamodel for i (2008)
19. Mylopoulos, J., Chung, L., Nixon, B.: Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. IEEE Transactions on Software Engineering **18**(6), 483–497 (jun 1992)
20. Negri, P., Souza, V., Leal, A., Falbo, R., Guizzardi, G.: Towards an ontology of goal-oriented requirements. In: Proc. of the 20th Ibero-American Conf. on Software Engineering (CIbSE) (2017)
21. Patricio, P., Amaral, V., Araujo, J., Monteiro, R.: Towards a Unified Goal-Oriented Language. In: Proc. of the 35th Annual Computer Software and Applications Conf. pp. 596–601. IEEE (2011)
22. Ruy, F.B., Falbo, R.A., Barcellos, M.P., Costa, S.D., Guizzardi, G.: SEON: A Software Engineering Ontology Network. In: Knowledge Engineering and Knowledge Management. LNCS. vol. 10024, pp. 527–542. Springer (2016)
23. Sales, T.P., Baião, F., Guizzardi, G., Almeida, J.P.A., Guarino, N., Mylopoulos, J.: The Common Ontology of Value and Risk. In: Proc. of the 37th International Conference on Conceptual Modeling (ER). LNCS. vol. 11157, pp. 121–135 (2018)
24. Yu, E.S.K.: Modelling strategic relationships for process reengineering. Ph.D. thesis, PhD thesis, University of Toronto (1996)