iStar2.0-OWL: an Operational Ontology for iStar

Camilo Almendra[1,3], Carla Silva[1], Vitor Souza[2], and Renata Guizzardi[2] and César Bernabé[2]

[1] Centro de Informática – UFPE
{cca3,ctlls}@cin.ufpe.br
[2] Departamento de Informática - Centro Tecnológico – UFES
{vitorsouza,rguizzardi}@inf.ufes.br, cesar.hber@gmail.com
[3] Campus Quixadá – UFC

Abstract. Requirements engineering comprises activities for discovery, analysis and specification of users' needs and goals for a software system. In an early phase of software development, it is essential not to discard alternatives until some reasoning or evaluation is taken. Goal-oriented requirement engineering provides means for dealing with goals, needs and its alternative options for realization. However, analysis of large scale or complex systems requirements may be hard to be accomplished and error prone. Knowledge-based systems are a good tool to assist analysts in carrying out such analysis. This work proposes an operational ontology to represent iStar 2.0 models using OWL-DL.

1 Introduction

Requirements engineering (RE) activities can be improved using ontologies, particularly for reducing ambiguity, inconsistency and incompleteness [4]. Ontology is an explicit specification of a shared conceptualization that holds in a particular context [11]. It aims to organize domain knowledge over individuals, objects, types, attributes, relationships and functions. Once organized in a knowledge base, this information can be easily shared among people involved in the context of the ontology, such as users, customers, requirements analysts, architects and any other personnel involved in Software Engineering (SE).

Two recent literature reviews bring interesting information on the use of ontologies in RE. In [4], Dermeval et al. identify that most studies target the reduction of ambiguity, inconsistency and incompleteness. In second place, there were studies purposing requirements maintenance and evolution, and, in third, studies focusing in domain knowledge representation. In [12], Valaski et al. explore the function of ontologies in RE. The review found that the intended function is well distributed in categories: i) Structuring and recovery of knowledge, ii) Verification and validation, iii) Support to understand/identify concepts, iv) Control/share of vocabulary and v) Integration and transformation models. In the context of requirements modeling, ontologies are an effective approach to validate the consistency of models against a modeling language metamodel and formation rules.

One of the most used Goal-Oriented Requirements Engineering (GORE) languages which also attracts a lot of attention from the research community is i* (iStar). Recently, a steering committee of experts reviewed and released a new version of the language, renamed iStar 2.0 [3]. The new version release aims at facilitating the dissemination of the language by newcomers, by proposing the standardization of constructs and syntax. This work contributes to this community by presenting a formal representation of iStar 2.0 constructs in description logics. Our purpose is to provide requirements engineers with a knowledge base system that can be used to enhance shared understanding and validate consistency of goal models.

This article is organized as follows. Section 2 presents related work. In Section 3, we present the methodology used to build the ontology and its specification. In Section 4, we discuss the implementation of the ontology, languages and tools used, and show an illustrative scenario of use. Finally, we present our conclusions.

2 Related work

Formal representations and ontologies for GORE have been proposed in order to discuss knowledge sharing and reasoning for goal models. Giorgini et al. [6] present a predicate logic-based approach to represent goals and their refinements through AND/OR relationships, and provide qualitative reasoning for goal satisfiability assessment. Their approach uses an algorithm for satisfiability evidence propagation. Borgida et al. [2] use description logics to formally represent a subset of i*'s seminal version, and present evidence-based propagation axioms for goal satisfaction assessment. Negri et al. [9] discuss similarities and divergences among popular goal modeling approaches, and propose a unifying domain ontology. They provide integration with foundational ontologies and seek to clarify the semantics of GORE. However, these works are not focused on providing a ready-to-use ontology implementation. Najera et al. [8] address the problem of integration of iStar variants, before the release of iStar 2.0. They proposed a methodology that uses OWL ontologies as means to integrate different iStar variants. They present an illustrative scenario which shows the transformation of the iStar metamodel into OWL classes and properties. The work did not explore some features of OWL-DL, such as the definition of Domain and Range for object properties. We find these features useful to implement basic model validation, as illustrated in Section 4.1. The iStar 2.0 language guide provides a unified metamodel and set of formation rules. In this work, we seek to formalize these rules.

3 Ontology Specification

In this work, we seek to provide a formal representation for the full scope of iStar 2.0 language guide. We followed the METHONTOLOGY process [5] to design and implement the iStar2.0-OWL. The steps are: (i) Define the scope of ontology and its intended use in GORE; (ii) Conduct literature review to identify

knowledge sources, such as foundational ontologies or related domain-specific ontologies; (iii) Conduct iterative cycles of conceptualization, integration, implementation and verification of ontology prototypes; (iv) Demonstrate ontology usage through an illustrative scenario of requirements analysis and specification.

iStar2.0-OWL is classified as a *Task-Specific Ontology*, because it embodies just the conceptualizations that are needed for carrying out a particular task [11]. The main purpose of the ontology is to validate the consistency of iStar 2.0 goals models. The main source used in this work was iStar 2.0 language guide [3], it provides the conceptual model and constraints. The ontology is implemented in OWL-DL (see discussion in Section 4). The intended end-users are requirements analysts working on elicitation and analysis that need to assess consistency of iStar 2.0 models. The intended scenarios of use are: i) Register actors and their intents, ii) Register decompositions and abstractions, iii) Register dependencies between actors, and iv) Evaluate consistency of models. We guided the specification by stating a set of competency questions that the ontology should be capable of answering based on the models provided. Additionally to answer this questions, the ontology implementation shall be able to identify violations of language guide in the models. These are some of the competency questions used to guide the design of iStar2.0-OWL (see complete list in [1]):

1. Which are the actors of the system?
2. Which are the actors that participate in another actor?
3. Which are the actors that are a kind of another actor?
4. Which are the goals intended by an actor?
5. Which are the qualities expected by an actor?
6. Which are the dependencies of an actor with other actors?
7. Which are the intentional elements that contribute to a quality?
8. Which are the qualities related to a task, goal or resource?
9. Which are the resources needed by a task?
10. Which are all the refinements and sub-refinements for a goal or task?

4 Ontology implementation

The ontology is implemented using the description logics language OWL-DL[4], the rule language SWRL [7], and the query language SQWRL [10]. The ontology was developed using Protégé[5] as supporting tool. The motivation for choosing OWL-DL was two-fold. A comparison analysis of knowledge representation alternatives for goal models indicated OWL as extensible and sufficiently legible, and with good tool support [2]. Also, a literature review indicated OWL as the most common language used for RE-related ontologies [4]. There are a variety of reasoners available to perform evaluation of OWL ontologies, and Protégé tool provides a design environment to build OWL ontologies and to integrate a reasoner. SWRL extends OWL ontologies with an abstract syntax that allows

[4] https://www.w3.org/OWL/
[5] http://protege.stanford.edu/

Table 1. Partial definition and representation of concepts and relationships

Definition	Representation
Actors are active, autonomous entities that aim at achieving their *Goals* by exercising their know-how, in collaboration with other *actors*.	`Class: Actor`
Actors' intentionality is made explicit through the actor boundary which is a container to hold what she **wants**.	`Property: wants`
isA association represents the concept of generalization/specialization.	`Property: isA` `SubPropertyOf: isAssociatedTo`
Intentional elements are the things *actors want*.	`Class: IntentionalElement`
Goal is a state of affairs that the actor wants to achieve.	`Class: Goal` `SubClassOf: IntentionalElement`
Quality is an attribute for which an actor desires some level of achievement.	`Class: Quality` `SubClassOf: IntentionalElement`
Task represents actions that an *actor* wants to be executed, usually with the purpose of achieving some *goal*.	`Class: Task` `SubClassOf: IntentionalElement`
Resource is a physical or informational entity that the *actor* requires in order to perform a *task*.	`Class: Resource` `SubClassOf: IntentionalElement`

specification of semantic rules. SQWRL is based on SWRL and provides SQL-like operators for extracting information from OWL ontologies. As both SWRL and SQWRL are designed to extend OWL expressiveness and are integrated in the Protégé tool, their selection was straightforward.

For ease of reading, we present the ontology definitions and rules together with their representation. The complete ontology is available at [1]. Table 1 partially presents representation for the concepts and relationships in form of OWL axioms (here written in Manchester syntax for legibility). Table 2 partially presents the semantic rules that complement the iStar 2.0 metamodel. The complete version of Tables 1 and 2 can be found in [1].

iStar 2.0 language specifies *Intentional Elements* as things that *Actors* **want**. However, in the specification of the subclasses *Quality* and *Resource*, the terms used are **desire** and **require**, respectively. We prefer to keep the *wants* relationship for all subclass of *Intentional Elements*, as it is specified in the visual metamodel of the language. However, further discussion on the semantics of these relationships are required.

The semantic rules were implemented using SWRL rules, SQWRL queries, property settings in OWL, or a combination of them. In case of OWL property setting and SWRL rules, they are handled as axioms by the reasoner. In cases

we have to use SQWRL queries, we designed them to return empty results if the model is consistent. If the query returns any element of the model, there is an inconsistency related to the returned elements.

SWRL rules have the form of a implication from an antecedent clause to a consequent [7]. The antecedent is a assertion over existing individuals, and the consequent is the inference to be applied if a set of individuals match the antecedent. The assertion and the inference are written using conjunction of atoms, each atom can be a class, a property or a built-in function. Variables are indicated using a question mark (e.g., "`?x`"). For example, a rule asserting that the composition of parent and brother properties implies the uncle property would be written in the form:

```
hasParent(?x,?y), hasBrother(?y,?z) -> hasUncle(?x,?z)
```

SQWRL queries can filter and retrieve individuals in OWL ontologies [10]. The query is formed by an assertion over existing individuals, using SWRL syntax, and a select clause. For example, a query to retrieve every individual that is more than 17 years-old would be written in the form:

```
Person(?p), hasAge(?p, ?age), swrlb:greaterThan(?age, 17)
    -> sqwrl:select(?p)
```

4.1 Model validation example

The first step to validate an iStar 2.0 model is to transform it to individuals (assertions) in OWL. These individuals are then validated together with the iStar2.0-OWL axioms (Table 1) and rules (Table 2). The transformation needs only to consider the concrete (visual) syntax of the models. The mapping is simple, each concrete element of the language is associated with a single Class, and each concrete relationship is associated with a single Object Property. We use as illustrative scenario the Travel Reimbursement model from [3]. Consider the part of the model that relates the "No Errors" quality and "Request Prepared" goal (Figure 1).

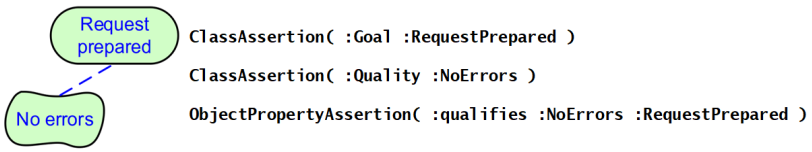

Fig. 1. Transformation of elements and relationships into OWL assertions.

The insertion of the individuals described above, together with iStar2.0-OWL, does not generate any inconsistencies. The list of individuals can be generated directly from a modeling tool or by an intermediary tool that reads the output of the modeling tool and transforms it to OWL individuals.

Table 2. Partial definition and representation of rules

Definition and representation
Only *roles* can be specialized into *roles*. `R: Role(?r2), isA(?r1,?r2) -> Role(?r1)`
Only *general actors* can be specialized into *general actors*. `R: GeneralActor(?g2), isA(?g1,?g2) -> GeneralActor(?g1)`
Agents cannot be specialized via is-a, as they are concrete instantiations. `Q: Agent(?a1), isA(?a2,?a1) -> sqwrl:select(?a2)`
Dependency relationships should not share the same dependum, as each dependum is a conceptually different element. `Q: Dependency(?d1), Dependency(?d2), differentFrom(?d1, ?d2), hasDependum(?d1,?d3), hasDependum(?d2,?d3) -> sqwrl:select(?d1, ?d2, ?d3)`

In a dependency D, if the dependerElmt x exists, then the actor that wants x is the same actor that is D's depender.
`R: Dependency(?d1), hasDependerElmt(?d1,?e1), hasDepender(?d1,?a) -> wants(?a,?e1)`
`P: Property` ***`wants`*** `set as inverse functional (`***`isWantedBy`***`)`

In a dependency D, if the dependeeElmt y exists, then the actor that wants y is the same actor that is D's dependee. `R: Dependency(?d1), hasDependeeElmt(?d1,?e1), hasDependee(?d1,?a) -> wants(?a,?e1)`
The depender and dependee of a dependency should be different actors. `Q: Dependency(?d), hasDepender(?d,?a1), hasDependee(?d,?a2), sameAs(?a1,?a2) -> sqwrl:select(?d, ?a1)`

Legend: R: SWRL rule, Q: SQWRL query, P: OWL property setting.

Suppose a user changes the type of "NoErrors" to *Goal*. The *qualifies* property is defined to have *Quality* class as Domain (that is, if an individual A *qualifies* other individual B, then A is classified as sub-class of *Quality*). In this scenario, an inconsistency is found and explained as follows:

```
DisjointClasses: Goal, Quality, Resource, Task
NoErrors qualifies RequestPrepared
qualifies Domain Quality
NoErrors Type Goal
```

As the *qualifies* property is defined to have as Domain the *Quality* class, and *Goal* and *Quality* are disjoint, an inconsistency is found.

5 Conclusions

This work presented the specification and implementation of an ontology for representation of iStar models. iStar 2.0 language was adopted as basis for extracting concepts, relationships and semantic rules. Concepts and relationships

were formalized in class and properties axioms. Rules were in part represented in axioms, but some required the use of SWRL (Semantic Web Rule Language) for formalization. Some of the language rules could not be defined as axioms or SWRL rules, so they must be represented using query language in future works. We hope this work can contribute to the evolution and adoption of iStar language standard. As future work, we plan to complete the definition and implementation of the concepts using the semantics proposed by Negri et al. [9].

Acknowledgements

The authors thank CNPq for funding the execution of this work.

References

1. Almendra, C., Silva, C., Souza, V., Guizzardi, R., Bernabé, C.: Supplementary material (2019), http://www.cin.ufpe.br/~cca3/istar20ontology/
2. Borgida, A., Horkoff, J., Mylopoulos, J.: Applying knowledge representation and reasoning to (simple) goal models. In: 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE). pp. 53–59 (Aug 2014)
3. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide. arXiv preprint arXiv:1605.07767 (2016)
4. 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 (Nov 2016)
5. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: Methontology: from ontological art towards ontological engineering. In: Proc. AAAI Spring Symposium. pp. 33–40. University of Standford, American Asociation for Artificial Intelligence (1997)
6. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. In: Proceedings of the 21st International Conference on Conceptual Modeling. pp. 167–181. ER '02, Springer-Verlag, London, UK, UK (2002)
7. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member submission **21**(79), 1–31 (2004)
8. Najera, K., Martinez, A., Perini, A., Estrada, H.: An ontology-based methodology for integrating i* variants. In: iStar 2013–Proceedings of the 6th International i* Workshop. p. 1 (2013)
9. 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: Proceedings of 20th Iberoamerican Conference on Software Engineering (CIbSE 2017) (2017)
10. O'Connor, M., Das, A.: Sqwrl: A query language for owl. In: Proceedings of the 6th International Conference on OWL: Experiences and Directions - Volume 529. pp. 208–215. OWLED'09 (2009)
11. Schreiber, G.: Knowledge engineering. Foundations of Artificial Intelligence **3**, 929–946 (2008)
12. Valaski, J., Reinehr, S., Malucelli, A.: Which roles ontologies play on software requirements engineering? a systematic review. In: Proceedings of the International Conference on Software Engineering Research and Practice (SERP). pp. 24–30. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens (2016)