Modeling Stories for Conceptual Model Assessment

Bernardo F. B. Braga and João Paulo A. Almeida

Ontology and Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo (UFES), Vitória ES, Brazil
{bfbbraga,jpalmeida}@inf.ufes.br

Abstract. Conceptual modeling is a challenging activity and assessing the quality of conceptual models is key to ensure that they may be used effectively as a basis for understanding, agreement and construction of information systems. Stories have always been used as means of communicating complex affairs and we argue that they may be used effectively to assess models and reveal modeling decisions to those that cannot understand the modeling language. This paper proposes an approach to assess conceptual models by creating narratives about a subject domain. These narratives employ concepts of the conceptual model and are formalized as abstract stories. These stories guide model simulation, supporting the validation of the conceptual model. Contrasting simulation with the intended conceptualization is the basis for model assessment.

Keywords: Conceptual Modeling, Storytelling, Model Assessment, Ontology

1 Introduction

In a broad perspective, conceptual modeling has been characterized as "the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication" [12]. These formal descriptions are called conceptual models and are built using artificial modeling languages.

Conceptual models may be used as basis for information systems such as the semantic web and its applications. Therefore, assessing their quality is key to ensure they may be effectively put to use. Assessing model quality is a challenging activity, in particular assessing whether the model corresponds to the modeler's original intention, and whether it reflects accurately the conceptualization of a subject matter expert. This is aggravated by the fact that frequently subject matter experts do not know the modeling language and modelers know little or nothing beforehand about the subject matter. Helping communication between these parties motivated our efforts into building tools and techniques for conceptual model assessment.

Here, we build on previous efforts by approaching model assessment using model transformation and a lightweight formal method. In our previous approach [1,3,9], an ontology-based conceptual model is translated to the Alloy logic-based language [11] that presents valid instances of the model or may search for assertion counter examples. This allows the inspection of model instances (in what could be considered a model "simulator") and therefore allows the assessment of the consequences of mod-

eling choices. So far, the generation of model instances in this approach is based purely on a random strategy. This means that the modeler cannot control the validation process. While this is useful to detect problems in the conceptual model (e.g., "edge cases" [18]), the simulation still has an overwhelmingly large number of possible instantiations. In order to control the model assessment process, we explore in this paper a technique that allows the modeler to guide the simulation through storytelling.

The rest of this paper is organized as follows: in section 2, we position storytelling and conceptual modeling as complementary means to transfer knowledge about reality. In section 3, we present our approach to creating stories and formalizing them, using as a running example a model in the software configuration domain. In section 4, we discuss some related work, and, finally, in section 5, we present conclusions and topics for further investigation.

2 Relating Storytelling and Conceptual Modeling

According to [7], "there is little doubt that narrative thought developed earlier in human history than scientific and logical thought". The ability to narrate gives us the possibility to reenact real-world events eliciting the imagination of the listeners, giving them experiences that they never had themselves. Early in the history of mankind, oral storytelling culture produced collective, standardized narrative versions of reality, particularly of past events; having become what we call the dominant "myths" of a society. Myths reflect the earliest form of integrative thought. In contrast with myths, theories are "very large, externally nested cultural products" which only emerged much later, as our culture allowed the externalization of memory [7].

Similarly to storytelling, conceptual modeling is also used for transferring knowledge. Nevertheless, the concrete representation of this knowledge takes a very different form. Although a conceptual model also represents a view of some subject matter, it does so in a very structured manner, using a formal language to describe the categories of entities that are assumed to exist in a subject matter and how these entities relate to each other. We take ontology-based conceptual models to be a particular means to represent a theory about a subject domain, formally capturing admissible states of affairs [10] using invariants i.e. logical assertions or rules that are held to always be true.

Our approach in this paper aims to leverage the value of storytelling as means for transferring knowledge, not substituting but enriching ontology-based conceptual modeling. In this approach both subject matter experts and modelers create natural language narratives using the concepts that appear in the conceptual model. The modeler translates these natural language narratives into Formal Stories using a story specification language that makes explicit reference to the concepts in the conceptual model. These formal stories constrain the generation of valid instances of the model to generate Formal Narratives (simulations) that conform to the specified formal story. By complementing a natural language narrative with a formal narrative, one can exemplify how the domain was modeled. That means modelers may assess whether their intentions were correctly expressed in the model by exemplifying model features

and “testing” their correctness with a subject matter expert. Also, this allows subject matter experts to assess the content of a model regardless of their knowledge of the modeling language: guiding which elements of a natural language narrative correspond to formalized knowledge. This helps to bridge the communication between modelers and subject matter experts. We integrate the support for story modeling in the model assessment tool ecosystem developed at our research group. We thus assume models are defined using the ontologically well-founded OntoUML profile [10], which provides a clear semantics for a fragment of UML class diagrams.

3 Creating stories for model assessment

Our approach aims to validate existing conceptual models using a mix of informal and formal storytelling. In **Fig. 1** we summarize our approach, showing three of its elements: (i) the natural language narrative, (ii) the formal story (anchored in the conceptual model) and (iii) the formal narratives (roughly a simulated story). Typically, natural language narratives about the subject matter are recorded. These natural language narratives are partially formalized regarding their semantic content (including the specification of which classes are instantiated from the conceptual model), using in this activity the specification language we defined. The product of that activity is called a formal story, which may partially define valid instantiations of the model. Formal stories are used to constrain the model simulation, resulting in what we call a formal narrative (a.k.a. model simulation).

In order to demonstrate the application of the technique, we introduce a running example in the domain of Software Configuration Management. We use as a starting point a previously published conceptual model for this domain extracted from [4]. This model is presented briefly in section 3.1. In section 3.2, we discuss the development of natural language narratives, providing a narrative for our running example. In section 3.3, we present a Story Specification Language and the informal narrative of our example is represented as a formal story. This formal story is simulated in section 3.4, demonstrating how formal narratives may support model assessment.

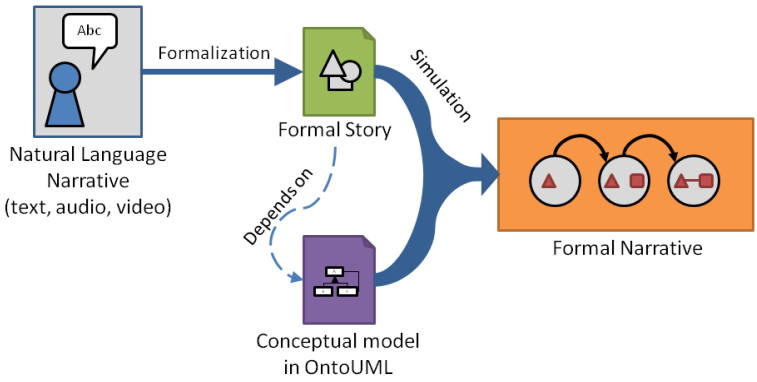

Fig. 1. An overview of the approach

3.1 Running Example

We use as running example a fragment of a model extracted from [4]. The diagram in **Fig. 2** specifies different kinds of **Items** that can be versioned: **Software Tools** and **Artifacts** such as **Source Code**, **Document** and **Diagram**. Classes stereotyped as Kinds are classes that apply necessarily to their instances and define a principle of identity for them. Categories (e.g., **Item**) are classes that also apply necessarily to their instances (i.e. are Rigid), but subsume instances with different principles of identity. An Item that has been selected by a **Configuration Manager** assumes the role of a **Configuration Item**. Configuration Manager is the role a **Person** assumes in the context of that selection. Roles are Anti-Rigid (a.k.a. dynamic) classes i.e. they apply contingently to instances. The relationship between the Configuration Manager and the Configuration Item is reified as a **Configuration Selection.** The Person class is omitted from the diagram and appears in italics on the top classes that specialize it.

Each Configuration Item is characterized by some **Version.** Version is stereotyped as Mode, meaning they are existentially dependent and inhere in the thing they characterize. In this case, Versions can only exist in Configuration Items. Versions are part of some **Branch.** Branches, on the other hand are part of some **Repository.** Stereotyped as Collectives, their instances are collections formed by uniform parts. Versions can be **submitted for change**, when **requested**. A **Developer** is a **Person** that may **Check Out** versions, **modify** them and **Check In Modifications** (a checked-in modification is called a **Registered Modification**)**.** Versions that are checked out are **Checked-Out Versions** and generate **Copies.** A **Copy** that has been modified assumes the role of **Modified Copy**, and when checked-in, makes the requested **change implemented**. A **verifier** may assess an **implemented change**, making it **verified**.

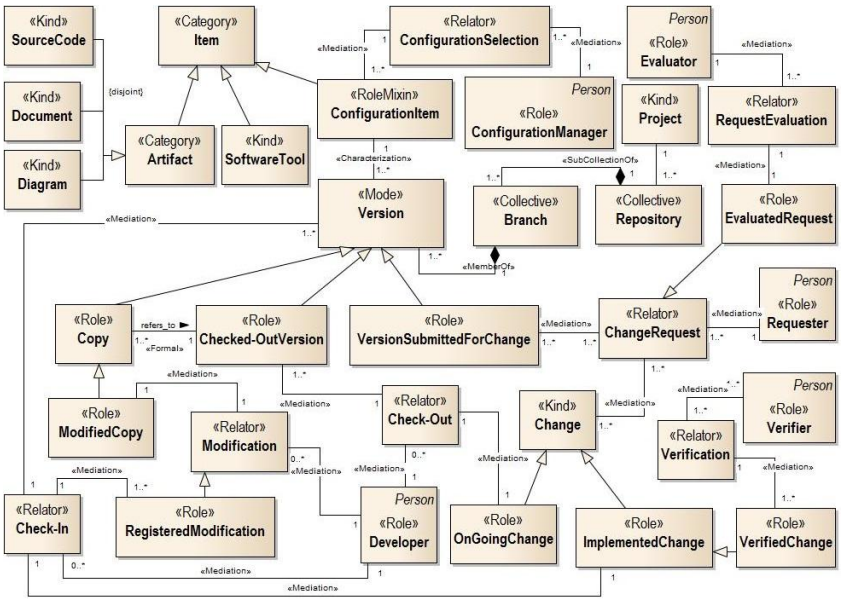

Fig. 2. A model for Software Configuration Management

3.2 Natural Language Narrative

Producing some natural language narratives about the domain can be the first step in our approach. The activity of creating these narratives and validating them is done between subject matter experts and modelers. Either of them may create the narrative. With regard to the scope of a narrative, in the case of modeler-authored narratives, the modeler may exercise fragments of the model he/she suspects may be incorrect i.e. he/she imagines a real-world scenario where the concepts of such fragment are instantiated. In the case of subject matter expert authored narratives, the subject matter expert narrates real life events about a fragment of the model requested by a modeler. The narratives help the modeler to understand how these concepts are exercised in their real context.

Drawing from our running example, we produced the following natural language narrative. It exercises the classes of the model presented in Figure 2. Whenever a class is used in the narrative, it is highlighted in bold. This narrative is the basis for the formal story presented in section 3.3 which will in its turn be used to generate formal narratives (simulations) in section 3.4.

*"John, Mary, Fred and Thomas work at OntoSoft company as **developers**. They are working on an information system for a bakery to manage its finances and supply-chain processes. The system they are producing already manages the finance aspects, and currently they are developing new **artifacts** (such as **diagrams**, **documents** and **source code**) to manage the supply-chain processes. Thomas is the **Configuration Manager** and he **selects** some of the **artifacts** they created to be part the project's repository, where they are **version**-controlled.*

*As the team focuses efforts on the bakery's supply-chain processes, Fred finds a deadlock in a process diagram for buying raw materials and files a **change request** for it, describing the problem he found and the **change** that should be implemented. John **evaluates** the request and **checks out** the **diagram** in the version control system to **modify** it. After doing the necessary adjustments, he **checks in** the **modified version** and Mary is assigned to **verify** whether John has met the **change request**.*

*Mary **verifies** the **code** and notices that John's modifications introduced bugs in the already-approved finance processes. These **changes** have a deep impact in the approved parts of the software so Mary rejects the **version** and asks John to **branch** the **project** and try again from a different angle."*

3.3 Formal Stories

Stories are abstract representations of a narrative. Elsewhere, these concepts are alternatively called *Fable* and *syuzhet* [15], respectively. Here, Formal Stories are abstract representations of both Natural Language Narratives, discussed in the previous section, and Formal Narratives, which will be discussed in the next section.

There are two ways to create such stories. In the first case, they may be based on an existing Natural Language Narrative. In this alternative, the modeler captures what happens in the story using the concepts present in the conceptual model. When formalizing an existing natural language narrative much detail is lost since formal stories

only contain semantic aspects of the narrative that are relevant to the conceptual model. However, this process may create information that is more precise than their natural language counterparts. Inconsistencies, ambiguities and suppositions are removed in this stage, making the modeler commit to a certain interpretation of the story's semantic content. The formal story acts like an explanation, revealing the elements involved in the story and its unfolding.

In the second case, a modeler may take the reverse approach: first create formal stories and posteriorly elaborate a natural language narrative based on it. By narrating this story to a subject matter expert, the modeler may validate his understanding of the domain. This is especially helpful for checking edge cases, as it is common practice in the testing of computer algorithms [18].

Formal Stories are model instances of our special-purpose language, whose metamodel is presented in Fig. 3. In this language, the user may specify *nodes* and *links* between *nodes*. Each node may be assigned to instantiate some Rigid classes from the conceptual model, while links instantiate Associations. Individuals (nodes and links) can be present in *worlds* and a world sequence represents the unfolding of the story (the world sequence is represented using "next" and "previous" relations). A world is a snapshot of the story, capturing the state of things in a particular point in the story. As the story progresses, elements may be created, changed or destroyed. Change is represented as classification statements that may be made about the nodes, which specify contingent characteristics of it, i.e., the Anti-Rigid classes a node instantiates.

The relations in this model capture the "facts" that the modeler asserts about the story. The modeler can assert a fact (e.g., "John" is an instance of "Developer") or assert its negation (e.g., "Mary" is not an instance of "Developer") by using the appropriate relations. Whenever the model is silent with respect to a particular choice, e.g., when nothing is said about whether "John" is a developer, the simulator will allow both options, meaning either case can appear in a formal narrative of such story. This is useful to partially formalize a narrative and simulate to see the possible rearrangements of states of affairs generated by the simulator. Later, a story may be revisited to constraint it further, specifying more details.

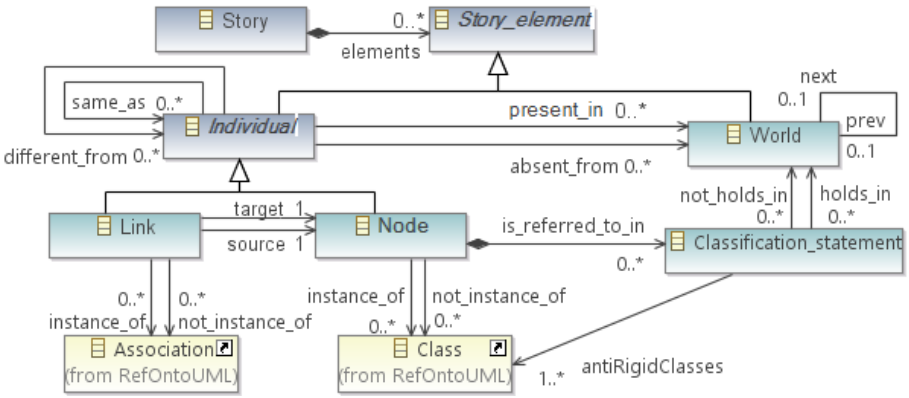

Fig. 3. Metamodel of the formal story language

Formalizing our running example, John and his peers are represented as Nodes that are **instance of** Person and each **is referred to in** a classification statement (Developer). The Items are also nodes and their classifications statements specify they are Configuration Items. All of those statements **hold in** every world of the story. A classification statement about Fred instantiating Requester, does **not hold** in the first world of the story and **holds in** the last two worlds exemplifying dynamic classification. That statement enforces that, in every simulation, John will always be instance of Requester in the last two worlds and will never be in the first. Other nodes defined include a selection and a check-out. To specify that these are actually Thomas' selection and John's checkout, we must specify links between Thomas and the selection, as well as between John and the checkout. We could specify the type of link instantiated but in this case there is only one type of relationship between person and each of these classes, meaning the simulator will assert the correct type of link, so there is no need for specifying it in the formal story.

Figure 4 is a screenshot of the prototype application, showing part of the formal story we just described. The tool represents this formal story internally as an instance of the abstract syntax metamodel presented in Fig. 3 (using code generated by EMF). The tree table specifies the story elements (Nodes and Links) in each row and the Worlds on the columns. Each field determines if the element exists (a checkmark), does not exist (an x), or if it is left unspecified (an empty box); for each world column. The classes each story element instantiates, as well as the anti-rigid classes for the classification statements, can be defined in the list below the story elements panel.

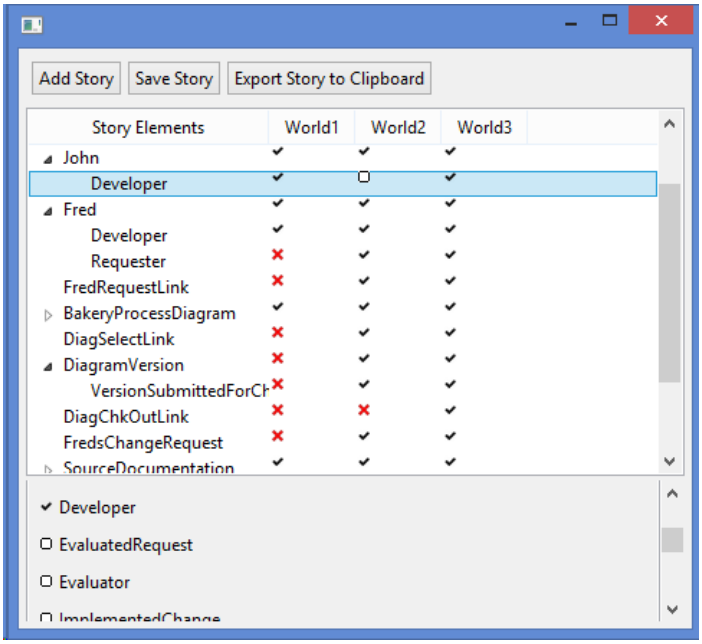

Fig. 4. The Formal Story Specification interface

3.4 Formal Narrative Generation

The generated narratives allow the assessment of what is possible according to the model's constraints, confronting the modeler and the stakeholders with the consequences of modeling choices. Counter-intuitive simulations of the story hints to modeling issues. Here we discuss a small sample of the issues that were identified in the simulation of our story and concern the quality of the conceptual model of Fig. 2.

Fig. 5 shows the first world in a simulation of our story. It shows not only those elements explicitly mentioned in the natural language narrative but also reveals other elements which are required to exist given the conceptual model. We have noticed that, similar to Fig. 5, every single simulation of the story had in its first world simultaneous check-ins, check-outs and modifications. Inspecting the model closely, we found that the minimum cardinalities of several relations create a cycle of mandatory entities. This means that any check-in must be associated with a check-out. As a consequence, a brand-new repository with no check-outs cannot be represented in this model. Note that it is not the *story* that requires check-outs in the first world, but these elements were included in the formal narrative by *logical necessity* by the Alloy Analyzer in order to show as a simulation of the story that is conformant with the conceptual model. The cycle in the model was most likely not detected by the authors of [4]. Identifying this by inspecting the conceptual model directly is not trivial as it involves 10 classes and requires navigation of several relationships. Relaxing the minimum cardinalities would break the cycle.

Other issues concern role interaction. In Fig. 5, Thomas's modifications were checked in by John. Is this possible in the domain? If not, then the model is under constrained, which could be fixed by requiring the modifier of a checked-in version to be the same person that checked it out (e.g., we could write a temporal OCL invariant [9] reflecting this domain rule). Further, in Fig. 5, Thomas selects the configuration items but it is John who checks them in. Again, if needed, invariants could be required to specify that a person who selects items is the same who checks them in for the first time. Finally, in Fig. 5, the set of items that were selected together and checked-in together, have versions belonging to different branches, a situation which could be presented to domain experts in order to assess validity.

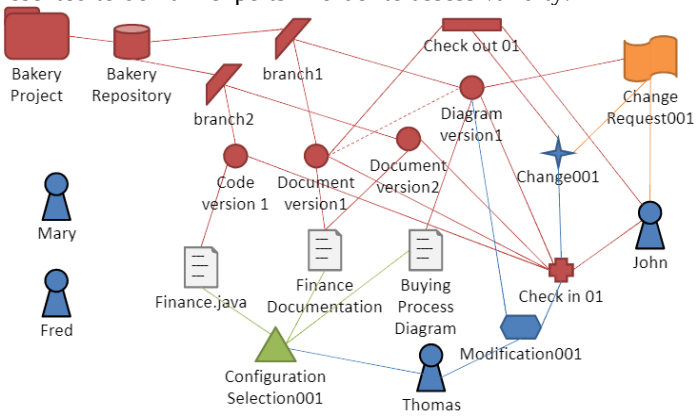

Fig. 5. The first world (snapshot) of a simulation

4 Related Work

There are many applications of storytelling and narratives in computer science, with many purposes different than ours (which is a posteriori assessment). [15] reviews some of these approaches. E.g. there are symbolic annotation tools [7], metadata for news stories [14] and means of assessing database systems [6]. One particular approach, Cucumber [16], share some of our goals by aiming to bridge communication between subject matter experts and developers. Differently from our approach, their technique consists of elaborating short stories that exemplify systems features with the purpose of driving development. The technique shows a promising direction for future work in expanding our approach to use stories to guide model development (and not only a posteriori assessment).

Other model transformations have been defined for OntoUML to other languages besides Alloy, including transformations to OWL [17] and SVBR [5]. The Alloy transformations were specified initially in [1,3], and later merged and improved in [13]. We build on these previous approaches by allowing the modeler to guide the simulations and inspect them intentionally.

5 Final Considerations

We have presented a technique to incorporate storytelling in an existing model validation approach to improve communication between modelers and experts, as well as facilitating model assessment. Formalizing natural language narratives allows the simulation of the model for validation. Natural language narratives have details which are not represented in the conceptual model and the process of formalizing natural language narratives or interpreting formal narratives in terms of a natural language narrative adds detail to the interpretation of the theoretical logical constructs. Analyzing examples allows an intuitive understanding of the model and the consequences of abstract definitions. To analyze the model by itself one must unfold in their own mind the possibilities and interactions between classes. The mental workload of performing this analysis is offloaded to the Alloy Analyzer, shifting the focus of the modeler to the validation task.

While we have applied the approach on a number of models and performed qualitative evaluations, there is still work to be done on systematically evaluating the approach and specifying quality criteria that could be quantitatively measured.

Currently, there are limitations with respect to the scalability of the analysis, given that the approach based on the Alloy Analyzer becomes intractable when the size of the model grows. The tool we use allows the modeler to select fragments of a larger model for assessment to cope with that. However, further investigation is required to assess whether fragments of models are a sound basis for overall model assessment.

While the Alloy instance visualizer does provide customization of elements using different shapes and colors, further work is required to incorporate visualization techniques described in [2] to generate better diagrams. Further work also includes a re-

verse transformation from formal narratives to formal stories, allowing the use of a simulation as a template for the definition of a formal story.

Acknowledgments. This research is funded by the Brazilian Research Funding Agencies CNPq (grants number 311313/2014-0, 485368/2013-7 and 461777/2014-2) and CAPES/CNPq (402991/2012-5).

6 References

1. Benevides, A.B., Guizzardi, G., Braga, B.F.B., Almeida, J.P.A.: Validating modal aspects of OntoUML conceptual models using automatically generated visual world structures. Journal of Universal Computer Science, 16, 2904–2933 (2011)
2. Braga, B.F.B.: Cognitive effective instance diagram design. Graduation Thesis, Federal University of Espírito Santo (2011)
3. Braga, B.F.B., Almeida, J.P.A., Guizzardi, G., Benevides, A.B.: Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal method. Innovations in Systems and Software Engineering, v. 6 (2010)
4. Calhau, R.F.: Uma Abordagem Baseada em Ontologias para a Integração Semântica de Sistemas, Master Thesis, Federal University of Espírito Santo (2011)
5. Carraretto, R.: A modeling infrastructure for OntoUML. Graduation Thesis, Federal University of Espírito Santo (2010)
6. Ciarlini, A.E.M., Furtado, A.L.: Understanding and Simulating Narratives in the Context of Information Systems. 21st International Conference on Conceptual Modeling (ER), Proceedings. v. 2503, 291–306. Springer (2002)
7. Donald, M.: Origins of the modern mind: Three stages in the evolution of culture and cognition Cambridge, MA: Harvard University Press (1991)
8. Elson, D.: Scheherazade, available at http://www.cs.columbia.edu/~delson/software.shtml
9. Guerson J., Almeida, J.P.A.: Representing Dynamic Invariants in Ontologically Well-Founded Conceptual Models, 20th EMMSAD, Sweden (2015)
10. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Telematica Instituut, The Netherlands (2005)
11. Jackson, D.: Software Abstractions-Logic, Language, and Analysis. The MIT Press (2012)
12. Mylopoulos, J.: Conceptual Modeling, Databases, and CASE: An Integrated View of Information Systems Development; Conceptual Modeling and Telos; Wiley (1992)
13. Sales, T.P.: Ontology Validation for Managers, MSc Thesis, Federal University of Espírito Santo, UFES (2014)
14. Wilton, P., Tarling, J., Mc Ginnins, J.: Storyline Ontology; available at http://www.bbc.co.uk/ontologies/storyline
15. Winer, D.: Review of Ontology Based Storytelling Devices. In Language, Culture, Computation. Computing of the Humanities, Law, and Narratives, 394–405. Springer (2014)
16. Wynne, M., and Hellesoy, A.: The cucumber book: behaviour-driven development for testers and developers. Pragmatic Bookshelf (2012)
17. Zamborlini, V., and Guizzardi, G.: On the representation of temporally changing information in OWL. In: 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), 283–292. IEEE (2010)
18. Zimmerman, J.: Unit Testing. Principles of Imperative Computation (2012)