An Essential Textual Use Case Meta-model Based on an Analysis of Existing Proposals

Fábio Levy Siqueira and Paulo Sérgio Muniz Silva

Escola Politécnica da Universidade de São Paulo, Av. Prof. Luciano Gualberto, tr.3, n°158, 05508-900 São Paulo, Brazil
levy.siqueira@usp.br, paulo.muniz@poli.usp.br

Abstract. Several studies have proposed meta-models for textual representations of use cases. Each of these meta-models has different goals and viewpoints, with either varying concepts and relationships or different semantics for the same concept. In Model-Driven Engineering approaches where there is no compatible proposal or a more appropriate viewpoint, or where there is no intention to limit the approach to a specific use case format, it might be difficult to choose between these proposals. Aiming at a generic textual representation, this paper presents a meta-model based on an analysis of 20 studies, chosen through a survey, that propose templates or meta-models. The more common elements of these studies are represented in a meta-model, together with their more frequent attributes. The goal is to create an essential and easily extendable meta-model that can be used directly in Model-Driven Engineering activities.

Keywords: essential, meta-model, specification model, textual, use case.

1 Introduction

The use case is a popular representation of requirements [14] that describes a set of scenarios. It was originally created as part of the *Objectory* process [12] for object-oriented software development. Other object-oriented processes that were based on *Objectory* also employ use cases as their central concept, such as *Unified Process* [13], and ICONIX [22]. Despite that close relationship with object-oriented processes, nowadays use cases can be used by other processes that take different software development paradigms into account [14]. One of these possibilities is to employ use cases in Model-Driven Engineering (MDE) approaches as models for functional requirements, using them as inputs or outputs in transformations.

A meta-model, i.e., a model that specifies the language used by other models [5], is necessary in order to employ use cases in the context of MDE. Because the use case is described in the Unified Modeling Language (UML) [20], one possible alternative would be to use the meta-model proposed by this standard. Therefore, there are several possibilities to express the behavior of a use case using the UML meta-model, for example, pre- and post-conditions, activity diagrams, state diagrams, or interaction diagrams [20]. However, the UML does not specify a textual representation for use

cases, which, according to some authors [6, 9, 14], is the most appropriate representation.

There is no standard meta-model for use case textual representation, but there are several proposals [11, 16, 18, 23, 25, 26, 29, 31]. These meta-models are usually based on only one or two use case formats, although there are several possibilities, such as [3], [4], [6], [8], [9], [14], [15], [17], [21], [22], [24], [27] and [28]. Even though there are some similarities among these meta-model proposals, there is no agreement on the semantics for the basic concepts, or even what the basic concepts would be. Furthermore, each meta-model has different goals and viewpoints, which might unnecessarily constrain a MDE approach or its result. For example, this paper is part of a research that aims to automatically transform an enterprise model into use cases. Because this research focuses on the transformation, any use case meta-model could be used as an output; at the same time, none of them is specific to this context. As a result, if an arbitrary meta-model is used, the transformation may be limited to its viewpoint, or it is even possible that an important transformation rule may not be defined because the chosen meta-model does not have a common concept. In situations such as this, a simple and generic meta-model that could be extended in the future, if necessary, would be a better option than an arbitrary meta-model.

With the purpose of creating an essential textual representation of use cases, this study proposes a meta-model based on a survey of 20 use case templates or meta-models. It represents the most frequent elements of these proposals, considering a simple semantic analysis. The proposed meta-model can be used in a Model-Driven Engineering context where a use case meta-model independent of a specific approach is necessary. In addition, this model can be extended to consider specific details of an approach or even specific details of a transformation.

This paper is organized as follows: in section 2, an overview of use cases is presented. Next, in section 3, an analysis of existing use case representations is presented, and the survey method and the most common elements found in the surveyed templates or meta-models are described. In section 4, an abstract syntax of the proposed meta-model is presented, where methods for extension and related works are discussed. Finally, in section 5, the conclusions, limitations, and extensions of this work are discussed.

2 Use Case

A use case is defined as "the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system" [20, p.596]. Consequently, it describes an interaction between entities external to the system, represented by the roles executed by them – the actors – [20], and the system under consideration – called the subject – to accomplish a desired result.

As it describes possible ways to use a system, the use case is a scenario-based approach; more specifically, it can be seen as a set of scenarios [1]. Like other kinds of scenario-based approaches, the use case can represent both requirements and specifications [2, 6, 9, 15]. Depending on its scope and detail (or goal level [9]), it can also be used to describe domain knowledge or implementation details. For instance, a

business use case [9, 15] describes the business context or even the idealized environment.

Among the reasons or benefits for employing use cases, some studies highlight that: they are easy to write and read [15], they are written in the users' language (or from users' perspective) [15], they relate requirements activities to design and implementation activities [15], a scenario described by them can almost directly be used as a test script [9, 15], they help to define the scope of the system [14]. Although some of these reasons are related to the ease of creation or comprehension of use cases, there is no agreement on how they should be represented. Even in the UML there are several options for expressing their behavior: by a particular kind of *Behavior* (a meta-class), by preconditions and post-conditions, indirectly by *Collaboration* (another meta-class), or by natural language. Within these options, some authors, such as [6], [9], and [14], consider that natural language is the most appropriate way to express use case behavior, mainly because it must be understood by developers and stakeholders in general. Nonetheless, the UML does not specify the nature of this textual representation: whether unrestricted or structured natural language – and if it is to be structured, what format it should have[1].

3 Analysis of Textual Representations

Although there is no standard that specifies what textual representation of use case behavior should look like, several studies propose templates or meta-models. Despite the fact there are some similarities in these about what should be represented, there are also several differences, as each representation has different goals and viewpoints. For instance, actions are usually represented in a flow of events – also called a scenario, course, path, or episode. The proposals suggest different names and types for these elements: normal course, alternative course, basic path, alternative path, exception path, main scenario, optimistic flow, bounded alternative flow, etc. In some cases, there is only a syntactical difference, but in others, the difference is semantic. Therefore, in order to work with use cases, it is necessary to choose a template or meta-model and understand it.

In some situations, the choice for a specific textual representation is a consequence of the method used or the viewpoint under consideration. In others, the choice is arbitrary. One example is situations where there is not a compatible proposal or a more appropriate viewpoint; another example is situations where only the basic elements of a use case should be taken into account. These situations are common in Model-Driven Engineering proposals that employ use case meta-models as inputs or outputs in transformations. The focus of such proposals is on the concept or validity of the transformation, and not on specific details regarding the models. In particular, this paper was conceived in the context of research that aims to transform a requirements model, represented as an enterprise model, into a specifications model, represented as use cases. As the goal is the transformation, any meta-model for textual

[1] It should be noted that the lack of a specification for a textual format is not a UML flaw. As the UML is a standard for a graphical modeling language, specifying a textual representation does not come within its scope.

representation could be used. Consequently, the choice between the existing proposals would be arbitrary, as none is more adequate than any of the others for this purpose. However, if in future it is decided to use a different representation, problems could arise as some concepts applied might not have equivalence or they might have different relationships.

When the choice of a meta-model for a textual representation of a use case seems to be arbitrary, the ideal solution would be a simple representation with the essential use case elements. Being essential, it could be easily understood and extended with other concepts. In addition, this representation could be changed by absorbing concepts defined by other proposals without many syntactic or semantic changes. Therefore, this study proposes an essential meta-model for use cases to be used as a functional specification, based on the common elements defined in existing templates and meta-models.

One approach to obtaining this essential textual representation of use cases would be to choose or to define something similar to an intersection of all existing templates and meta-models, taking into consideration the semantics of the elements and aspects pertaining to conflict resolution. The resulting model would only have elements in common to all representations. However, if an important concept is not defined in a single representation, the intersection would not represent it. Considering that templates and meta-models have different hypotheses, as well as specific goals and viewpoints, it is very probable that some elements important to a meta-model are not represented in another, or, even if they are, they can have different semantics (more limited, for example). Consequently, the resulting intersection would be a very limited meta-model, or even an empty one.

Due to the problems in the intersection of meta-models, the meta-model will be created here by taking into consideration the most frequent elements in several proposals. In other words, the elements that are used in more than half of the representations will be represented in the resulting meta-model. The main problem with this approach is that the analysis involved is a kind of match operation – an operation that maps model elements, finding commonalities between them [7]. Even when a semantic analysis (and not merely syntactical) is undertaken, it is difficult to conclude that an element is semantically equivalent, or even similar, to another one since the semantics of use cases is usually described informally. As a result, it would be almost impossible to obtain common elements considering that, in a semantic analysis, any small semantic discrepancy makes an element be considered different. To avoid this problem, elements with similar semantics will be considered as the same, and the most frequent name will be chosen. Although there are some arbitrary decisions, this simplification allows matching elements without a significant loss of meaning.

3.1 Survey method

Since the meta-model will consider the most frequent elements in existing proposals, one important decision is what proposals to consider. Although an analysis of all representations seems to be the best choice, in practice it is very difficult to guarantee that every existing proposal will be considered. To avoid an arbitrary decision about what studies to use, this paper will consider the studies found in a survey of some

important article databases (ACM, Elsevier, IEEE, and Springer) and a search on Amazon.com for books on the subject[2]. In the selected article databases, we carried out a search for papers that had the words "use case" and "template" or "meta-model" (or "metamodel") in their titles or abstracts. Only papers that have the proposal of a template or a meta-model for a textual representation as their objectives were analyzed. This constraint was designed to avoid simplified use case descriptions that were used only to test, analyze, or apply an approach. In addition, to avoid repeated descriptions, when there was more than one paper by a single research group, only the most recent one was considered. The results of this survey are presented in Table 1.

Table 1. Results of the survey in the article databases.

	IEEE	ACM	Springer	Elsevier
Number of results	36	30	23	7
Results considered	5	2	4	1

Because there are several important books about use cases, the survey also took into account books available on Amazon.com. Using "use case" as a keyword and limiting the search to the "Books" department and the "Computer & Internet" category, 109 books were found. Because it is more difficult to access and analyze books, only the first 10 books ordered by relevance were analyzed. Of that list, only 2 of them do not propose a clear template or meta-model: in [1] there are only general guidelines for writing adequate scenario descriptions with use cases, and [10] discusses quality aspects related to use cases.

Considering the survey method, 20 templates or meta-models for textual representations of use cases were analyzed. Some of these studies discuss a textual representation of use cases in the context of Requirements Engineering (RE). Leffingwell and Widrig [15] discuss RE activities emphasizing the use of use cases, proposing a specific use case template, and guidelines to create and employ use cases during the other RE activities. On a different tack from Leffingwell and Widrig, Wiegers discusses RE activities without proposing a specific approach [28]. The use case is presented as a technique to aid in the understanding of functional requirements.

Some other studies are specifically about use cases, discussing their foundation, guidelines, and templates in detail. [6], [9] and [24] focus on use case writing, and in [9] a meta-model for a textual representation is also proposed. Besides discussing how to write use cases and proposing a template, several topics about the software development process employing use cases are also discussed in [3].

There are also several studies that propose use case templates with specific approaches. Rosenberg and Stephens [22] propose that a simple use case template be used in a software development process. Focusing only on RE, Kulak and Guiney [14] propose an RE method that is driven by use cases, thus proposing a use case template. Other studies propose templates be used in a subset of RE activities: in [27] a method to transform stakeholders' requests into use cases is proposed, presenting a use case template and guidelines; [8] proposes a use case template (using controlled natural language) that can be transformed into process algebra – allowing use case

[2] This survey considers the search results obtained on October 07, 2010.

verification and also automatic test case generation; in [17] an approach to create use cases considering 4 categories of aspects is proposed; and in [4] a use case template that allows use case model merging is proposed. Finally, in [21] a use case template specifically for the automotive domain is proposed, involving embedded systems.

Even if a template defines the elements of a textual representation of a use case, that information is described in more detail in a meta-model. However, studies that propose meta-models generally only present their abstract syntaxes, and do not present their semantics clearly. Furthermore, each meta-model considers different views and goals: in [11] the scope is a requirements management tool; the meta-model presented in [18] provides multiple perspectives of use cases; in [31] a meta-model is proposed that can be configured to allow several specific features; [25] has a linguistic approach, using grammar to represent use case actions and proposing three different representations of use cases; [23] uses a use case meta-model to discuss use case refactoring; [16] proposes a meta-model that considers several concepts of the UML activity model; and [29] seeks to automatically transform use cases into analysis models.

3.2 Analysis

The analysis of the most frequent concepts took into account the studies obtained in the previously discussed survey: [3], [4], [6], [8], [9], [11], [14], [15], [16], [17], [18], [21], [22], [23], [24], [25], [27], [28], [29], and [31]. Due to a lack of space, it will not be possible to present all the tables used for the analysis.

Initially, only the meta-classes or concepts directly related to the use case meta-class or concept, and their attributes, were considered. This was to avoid analyzing sub-elements of uncommon elements. In the studies, 55 different concepts are defined, for example, the source of the use case, related information, priority, creation date, trigger, and scope. The elements proposed by 10 or more studies are as follows, with their usual semantics and most common terminology:

- *name*: an identification of the use case (proposed by 14 studies);
- *description*: a brief text that represents the goal of the use case (proposed by 14 studies);
- *actor*: a type of role played by an external entity that interacts with the system under consideration [20] (proposed by 11 studies);
- *precondition*: a condition of the subject that must hold before the use case starts (proposed by 15 studies);
- *post-condition*: a condition of the subject that must hold after the basic flow ends[3] (proposed by 14 studies);
- *basic flow*: the most common and/or successful flow of events (proposed by 15 studies); and
- *alternative flow*: a flow of events that describes a deviation in another flow of events (proposed by 14 studies).

[3] There were two equally frequent semantics for post-conditions (with 6 studies on each): success guarantees (when it must hold only for the basic flow), or minimum guarantees (when it must hold for all flows). The meta-model considers the semantics of the former.

In addition, a flow of events is defined here as an ordered set of steps executed by actors or the system under consideration to achieve the goal of the use case.

After considering these elements, their sub-elements were then analyzed. In the 7 elements analyzed, there were sub-elements frequently proposed only in "basic flow" and "alternative flow". In "basic flow" there were "steps" (16 times) - also called actions or events - with an associated "number" (15 times). Similarly, in "alternative flow" there were "steps" (16 times) but also "extension points" (11 times) and "conditions" (16 times). The usual semantics of these elements are:

- *condition*: an expression that can be evaluated as true or false;
- *step*: a unity of behavior that represents the interaction between an actor and the system under consideration; and
- *extension point*: the step where the deviation used by an alternative flow occurs, pertaining to a condition.

With regards to the steps, several studies define a specific format to describe them. For instance, Cockburn suggests that the step should be written as "subject + verb + direct object + prepositional phrase" [9]. Other studies propose even more detail, especially in [8] and [25], where patterns and grammar to describe the steps are presented. However, in 10 of the studies analyzed, the method for describing a step is not discussed - they just state that steps should be described textually. Also, there is no agreement about what should be permitted with regards to their organization or sequencing. Those variations are not always made as explicitly as the other elements, and writing a flow of events is sometimes simply expressed as "good practice". Therefore, in Table 2, the analysis is about whether the studies are in favor, against, or indifferent - when there is no clear position - to conditions, loops, parallelism (steps executed in parallel), and ad-hoc blocks (where steps are executed in no particular order). When most of the studies are indifferent to a specific type of organization or sequencing, it is considered that it should not be represented. As a result of this analysis, it was found that only loops and conditions are frequently defined. Thus, these concepts are defined as:

- *conditional statement*: a statement that specifies a condition to execute an ordered set of steps; and
- *loop statement*: a statement that specifies a condition to execute an ordered set of steps repeatedly while certain condition holds true.

Also, a statement is defined here as a type of step with a condition and an ordered set of steps.

Table 2. The position of the analyzed studies on certain types of organization and the sequencing of steps.

	In favor	Against	Indifferent
Condition	12	4	4
Loop	12	1	7
Parallelism	7	0	13
ad-hoc block	3	0	17

Several studies also propose different types of steps. For example, in [6], [8], [9], and [11] steps executed by the system under analysis are separated from steps

executed by actors; and in [23] they are divided among stimulus, response, and action. However, these types of steps are not frequently proposed. Finally, some studies also propose heuristics to restrict the content of a flow of events. For instance, in [9] it is suggested that there should be 3 to 10 steps in the main flow, while in [3] it is suggested that there should be only one level of loop or conditional statements in a flow of events, and in [11] it is suggested that there should be almost the same number of actor steps and system steps. However, as those heuristics vary according to the proposal, they were not considered.

4 Use Case Meta-Model

The elements obtained in the analysis described above can be organized into a meta-model, which can be used in Model Driven-Engineering (MDE) activities. According to Bézivin [5, p.41], a meta-model "describes the various kinds of contained model elements, and the way they are arranged, related, and constrained." A meta-model is essential to MDE activities, as it allows tools to adequately apply operations to a model. From the different possibilities of representing a meta-model, this study will use a representation that divides a meta-model into concrete syntax, abstract syntax, and semantics. For example, this representation is used by OMG standards, such as the UML [20]. Although the use case meta-model should be described by these three models, this paper will only describe its abstract syntax and semantics. The concrete syntax will be omitted because there are several possibilities that could be used. Even though a textual concrete syntax seems to be more appropriate – as a textual representation of use cases is proposed –, even a graphical concrete syntax can be used in some situations. With regards to the other models, the abstract syntax will be described using MOF class diagrams [19], as is commonly used. The semantics is described in section 3.2, using natural language and considering the usual semantics proposed by the analyzed studies. This representation was chosen because formal semantics would make it difficult to apply the use case meta-model – and may even hinder some of the benefits presented with regards to a textual representation. In future studies, it is possible to describe this more precisely, by using a formal semantic representation that enables use case simulation, for example.

Finally, to create this meta-model, there is an important decision to be made: whether it should be compliant with the UML standard or not. As the use case is defined in the UML, it seems a natural choice to use the standard, thus extending its meta-model. However, the only extension mechanism available in the UML is the profile, which adds constraints to the meta-model, but does not change those already defined [20]. Therefore, the use case textual meta-model should be created by extending the use case package or by defining a new type of *Behavior*. Both solutions would make the meta-model more complex than would be expected for an essential meta-model. For example, existing constraints in the standard would be taken into account, along with the inclusion and extension concepts, and the UML action semantics for the steps pertaining to the flow of events (as considered in [26]). Thus, it was decided that the meta-model would not be compatible with the UML. A meta-model consistent with the UML standard can be created in a future study.

4.1 The proposed meta-model

In Fig. 1, part of the meta-model abstract syntax is presented, represented as an MOF class diagram. In addition to the meta-classes representing the concepts found in the analysis, four other meta-classes were also created: *UseCase*, *FlowOfEvents*, *Statement*, and *Action*. *UseCase* represents a use case, enabling more than one of them to be specified in a single model in conformance to this meta-model. As the concepts *Name* and *Description* are normally specified as a text, they were considered attributes of this meta-class. The *FlowOfEvents* meta-class was created to simplify the meta-model, representing that both *BasicFlow* and *AlternativeFlow* have similar semantics and possess an ordered set of *Steps*. Similarly, the meta-class *Statement* represents the generalization of loop statements and conditional statements. Finally, the *Action* meta-class was created to represent that a *Step* is described textually. However, for extension purposes and because the *Statement* is a kind of *Step*, a separate meta-class was created – rather than just adding this attribute to *Step*. As a *Step* is called an *Action* in some studies, this name was used for the new meta-class.

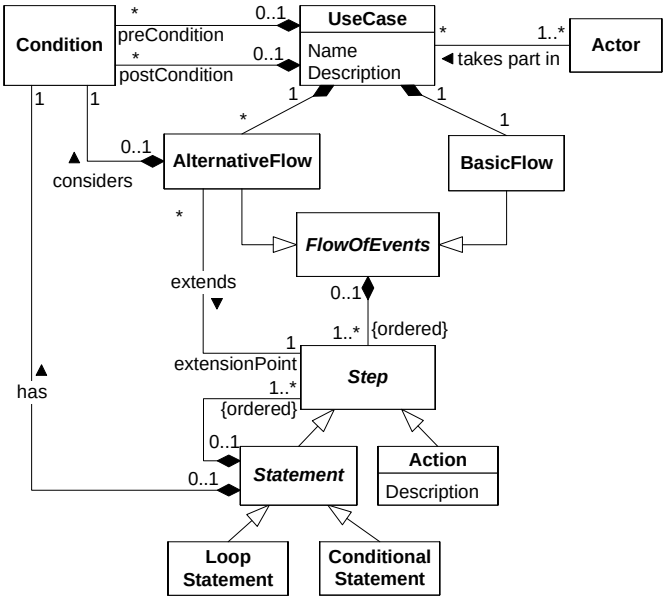

Fig. 1. The proposed use case meta-model.

In the meta-model, each *UseCase* can have one or more *Actors*, while each *Actor* can take part in several *UseCases*. Each *UseCase* can have several preconditions or post-conditions (association between *UseCase* and *Condition*). To describe the behavior of the use case, there are two specializations of the abstract meta-class *FlowOfEvents*: the *BasicFlow* and the *AlternativeFlow*. Each *FlowOfEvents* has an ordered set of *Steps* (the step number is represented by its order). A *Step*, which is an abstract meta-class, can be an *Action* or a *Statement*. An *Action* has a description, as it was found in the analysis that there is no single frequent format to represent the *Step*. A *Statement* is an abstract meta-class that represents a *ConditionalStatement* or

LoopStatement. As a *Statement* can involve more than one *Step*, it has an ordered set of *Steps*. Finally, the *AlternativeFlow* has an extension point (association between the *AlternativeFlow* and *Step*), considering a *Condition* (association between *AlternativeFlow* and *Condition*). There are also two constraints that must be considered in this meta-model: (1) a *Step* has to be on a *FlowOfEvents* or (exclusively) on a *Statement*; and (2) that an *AlternativeFlow* cannot extend one of its *Steps*.

4.2 Meta-model extension

Because the meta-model only considers the most frequent concepts, it has few meta-classes and almost none of them have attributes. However, if necessary, the meta-model proposed here can be extended. The recommended extension mechanism involves specializing the existing meta-classes - for example, *Condition* can be specialized to use a specific language, or the *Action* can be specialized to consider a specific type. However, there is no constraint on how this meta-model should be extended. It was not intended to define a rigid meta-model, but an essential meta-model that can be used in different situations. Therefore, if necessary, a user can extend it by modifying existing classes and associations.

A recommended extension to the proposed meta-model is to define a concrete syntax. For example, within the research that motivated this paper – the transformation of an enterprise model into use cases from the perspective of functional requirements -, the concrete syntax was specified using XSD. But the meta-model user can use any other concrete syntax deemed appropriate for the intended use.

4.3 Related Work

There are several studies that propose a use case template or meta-model based on other proposals. For instance, Zelinka and Vranic [31] propose a meta-model based on two representations; Somé [26] analyzes 2 templates to create a UML-compliant meta-model, focusing on consistence with the UML specification; the Petterson, Ivarsson, and Öhman [21] template is based on 6 existing proposals; and Anda, Sjøberg, and Jørgensen [2] briefly analyze 6 existing templates. However, only a few proposals are analyzed in these studies, and there is no clear criterion to define the new use case representation. In a similar vein to this study, Yue, Briand, and Labiche [30] review 16 use case approaches, and in [29] they propose a template to facilitate their transition to analysis models. Therefore, they do not propose a use case template or meta-model based on the most frequent elements, using their most frequent semantics.

5 Conclusion

This paper presents an essential meta-model for a textual representation of use cases, by expressing their abstract syntax (using MOF) and semantics (in natural language).

This meta-model considered the most common elements described by 20 proposals found in a survey of five article and book databases.

Due to a lack of space, it was neither possible to present all the tables used in the analysis nor all its details. For the same reason, the constraints to the abstract syntax were not represented in OCL, and an example of concrete syntax was not proposed. Furthermore, this study has some limitations. First, the meta-model is not compliant to the UML standard. This was decided because it was considered that the existing UML constraints would make the meta-model more complex than an essential meta-model should be. An example of this complexity is presented in [26], where a use case meta-model focusing on compliance with the UML standard is proposed. Therefore, in future works this meta-model can be improved by making it compliant with the UML. Another limitation is in the proposed semantics: they were described informally using natural language. In future works more precise semantics may be proposed. Finally, the most important limitation in this study pertains to the survey carried out. Some arbitrary decisions were involved, mainly regarding the research databases used, the keywords used to find studies, and the number of books analyzed. As a result, other proposals were not considered, such as [12], [13], and [26]. Still, we believe these criteria have led to a representative group of studies to be analyzed, thus resulting in a representative meta-model.

The motivation for this study was ongoing research that aims to transform a requirements model, represented as an enterprise model, into a functional specification model, represented as use cases. In that context, this meta-model was used as an output for an automatic transformation, and XML was used as its concrete syntax (defined in XSD). Nevertheless, this meta-model can be used by other MDE approaches that require an essential use case meta-model.

Acknowledgments

This study was partly supported by *Fundação de Amparo à Pesquisa do Estado de São Paulo* (FAPESP), grant no. 2007/58489-4.

References

1. Alexander, I., Maiden, N.: Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle. Wiley (2004)
2. Anda, B., Sjøberg, D., Jørgensen, B.: Quality and Understandability of Use Case Models. In: ECOOP, LNCS 2072, pp. 402-428 (2001)
3. Armour, F., Miller, G.: Advanced Use Case Modeling: Software Systems. Addison-Wesley (2001)
4. Barrett, S., Sinnig, D., Chalin, P., Butler, G.; Merging of Use Case Models: Semantic Foundations. In: TASE, pp.182-189 (2009)
5. Bézivin, J. Model Driven Engineering: An Emerging Technical Space. In: GTTSE 2005, LNCS 4143, pp. 36-64 (2006)
6. Bittner, K., Spence, I.: Use Case Modeling. Addison-Wesley (2002)
7. Brunet, G., Chechik, M., Easterbrook, S., Nejati, S., Niu, N., Sabetzadeh, M.: A Manifesto for Model Merging. In: GaMMa, pp. 5-12 (2006)

8. Cabral, G., Sampaio, A.: Formal Specification Generation from Requirement Documents. Electronic Notes in Theoretical Computer Science, vol. 195, pp. 171-188 (2008)
9. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley (2000)
10. Denney, R.: Succeeding with Use Cases: Working Smart to Deliver Quality. Addison Wesley (2005)
11. Durán, A., Bernárdez, B., Genero, M., Piattini, M.: Empirically Driven Use Case Metamodel Evolution. In: UML, LNCS 3273, pp. 1-11 (2004)
12. Jacobson, I.: Object-oriented software engineering: a use case driven approach. Addison-Wesley (1992)
13. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley (1999)
14. Kulak, D., Guiney, E.: Use Cases: Requirements in Context. Addison-Wesley, 2nd edition (2003)
15. Leffingwell, D., Widrig, D.: Managing Software Requirements: A Use Case Approach. 2nd edition. Addison-Wesley (2003)
16. Lei, M., Jiang, W. C.: Research on Activity Based Use Case Meta-Model. In: ICACTE, pp.843-846 (2008)
17. Lu, C., Song, I.: A Comprehensive Aspect-Oriented Use Case Method for Modeling Complex Business Requirements. In: ER Workshops, LNCS 5232, pp. 133-143 (2008)
18. Nakatani, T., Urai, T., Ohmura, S., Tamai, T.: A requirements description metamodel for use cases. In: ASPEC, pp.251-258 (2001)
19. OMG: Meta Object Facility (MOF) Core Specification. Version 2.0, formal/06-01-01 (2006)
20. OMG: OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.2, formal/2009-02-02 (2009)
21. Pettersson, F., Ivarsson, M., Öhman, P.: Automotive use case standard for embedded systems. In: Workshop on Software Engineering for Automotive Systems, pp.1-6 (2005)
22. Rosenberg, D., Stephens, M.: Use Case Driven Object Modeling with UML: Theory and Practice. Apress (2007)
23. Rui, K., Butler, G.: Refactoring Use Case Models: The Metamodel. In: ACSC, pp. 484-491 (2003)
24. Schneider, G., Winters, J. P.: Applying Use Cases: A Practical Guide. Addison-Wesley, 2nd edition (2001)
25. Smialek, M., Bojarski, J., Nowakowski, W., Ambroziewicz, A., Straszak, T.: Complementary Use Case Scenario Representations Based on Domain Vocabularies. In: MODELS, LNCS 4735, pp. 544-588 (2007)
26. Somé, S.S.: A Meta-Model for Textual Use Case Description. Journal of Object Technology, v. 8, n. 7, pp. 87-106 (2009)
27. Subramaniam, K., Far, B.H., Eberlein, A.: Automating the transition from stakeholders' requests to use cases in OOAD. In: CCECE, vol.1, pp. 515-518 (2004)
28. Wiegers, K.: Software Requirements. Microsoft Press, 2nd edition (2003)
29. Yue, T., Briand, L.C., Labiche, Y.: A Use Case Modeling Approach to Facilitate the Transition towards Analysis Models: Concepts and Empirical Evaluation. In: MODELS, LNCS 5795, pp. 484-498 (2009)
30. Yue, T., Briand, L.C., Labiche, Y.: A Systematic Review of Transformation Methodologies between User Requirements and Analysis Models. Technical Report SCE-09-03, Carleton University (2009)
31. Zelinka, L., Vranic, V.: A Configurable UML Based Use Case Modeling Metamodel. In: ECBS-EERC, pp.1-8 (2009)