A Semantic Formalization for Use Case Modeling

Marinos G. Georgiades and Andreas S. Andreou
Department of Electrical Engineering and Information Technologies, Cyprus University of Technology,
30 Archbishop Kyprianos Str., Limassol, Cyprus

Keywords: Use Cases, Formalization, Software Requirements, Semantics.

Abstract: It has been recognized that a certain level of formalization is needed to produce precise and well-defined use case models, thus tackling problematic issues such as the lack of a specific, semantic classification of use cases, the vagueness of the use case specifications written in free natural language (NL), and the limited tool support that makes use case driven analysis a time-consuming and error-prone activity. This paper presents a formal semantics for the basic use case model elements, including specific semantic types of use cases, specific types of basic and alternative flow actions, and specific functional roles actors can play. To maintain a high-degree of readability and understandability and to minimize ambiguity, the proposed approach provides a semi-formal, NL-based specification syntax, tailored to each semantic use case type, with a specific sequence of basic and alternative flow actions. The formalization is facilitated by utilizing elements of a novel methodology named *Natural Language Syntax and Semantics Requirements Engineering*.

1 INTRODUCTION

Use case driven analysis (UCDA) is very popular among the many methods in requirements engineering, due to the fact that it allows functional requirements to be represented in an easy-to-use and attractive style for both users and analysts (Dias et al., 2008). UCDA helps to cope with the complexity of the requirements analysis process; by identifying and then independently analysing different use cases, the analysts may focus on one narrow aspect of the system usage at a time (Kim et al., 2004).

It has been admitted that a certain level of formalization is needed to produce precise and understandable use case models (Somé, 2005). This paper presents a formalization approach, designed for information systems (ISs) in general and with particular focus on transactional business systems (TBSs) which are ISs that maintain data, have reporting capabilities, and use business rules to carry the everyday transactions of a business/organisation. A *Hospital IS* or a *Library IS* are examples of ISs that could contain TBSs such as a billing IS, a salary payment IS, and others. The proposed approach is intended to provide use case formalization with (i) a formal semantics for the basic use case model elements, including specific semantic types of use cases, specific types of basic and alternative flow actions, and specific functional roles actors can play; and (ii) a semi-formal, NL-based specification syntax tailored to each semantic use case type, including a specific sequence of basic and alternative flow actions.

The rest of this paper is structured as follows: Section 2 outlines related work, while section 3 describes the proposed formalization of the use case model. Section 4 provides conclusions and recommendations for future work.

2 RELATED WORK

A number of approaches and guidelines have been proposed to provide some degree of formalization to textual use cases, such as the CREWS Guidelines (Salinesi, 2004) and the CS Rules (Cox and Phalp, 2003). However, the focus of these approaches is on providing some NL semantic and syntactic guidelines on specifying actions, but their guidelines are too general and not linked to any semantic use case types at a higher level, contrary to our approach that provides a specific semantic classification of use case types, each of which contains specific types and sequence of actions.

G. Georgiades M. and S. Andreou A..
A Semantic Formalization for Use Case Modeling.
DOI: 10.5220/0004001301720175
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 172-175
ISBN: 978-989-8565-11-2

A number of structured techniques for the description of use cases have been proposed. In Eriksson et al.'s work (2004), a tabular representation is used, and in Leite et al.'s (1997), a structured natural language is presented to describe the use cases. These structured representations provide a generic formalization of the use case (UC) specification template, hence not a clear formalism of the use case specification elements, and especially the basic and alternative flow actions. Ochodek and Nawrocki (2007) provide a semi-formal NL representation of use case actions, however this formalism is still generic, lacks several types of flow actions as well as the use case elements (e.g., actors) involved in each action.

Within the context of transactional business systems, Chalin et al. (2008) make a general reference to three categories of use cases, including transactional use cases – those associated with the realization of the core business events/ business rules; support use cases – those usually associated with data maintenance (typical CRUD functions) or system configuration; data extraction use cases – those that do not modify the state of the system, and are usually associated with reporting capabilities.

In contrast, our approach provides detailed formal semantics for use case modeling of transactional business systems, through a semantic classification of use cases for creating, processing and presenting/ reporting information, through specific types of actors to identify business roles for each type of use case, and through the definition of actions for each use case type. The proposed formalization builds upon work done by Georgiades and Andreou (2010).

3 USE CASE FORMALIZATION

To provide formalization of the TBS use case model, we utilize specific elements from the NLSSRE methodology (Georgiades and Andreou, 2005, 2010). NLSSRE offers the means to engineer user requirements concerned with the operational aspect of an IS, that is a TBS, and building these requirements with the use of the following IS elements: people (usually end-users, clients and trusted external users), processes related to the creation, modification, transmission, storage and presentation of information along with the circumstances within which these processes are performed, as well as data, constraints, and business rules. In particular, NLSSRE focuses on formalizing and automating the discovery, analysis and specification of user requirements for the development of TBSs. NLSSRE is designed so that the analyst is guided in advance, through a step-by-step approach, what specific types of data, functions, business rules and functional conditions to use and search for, what questions to ask, in what specific way to analyse the answers to the questions, and how to write them using formalized sentential requirement patterns. The formalized requirements are then easily transformed, with the use of specific rules, into diagrammatic notations, including class diagrams, data flow diagrams and use-case diagrams. The formalization of NLSSRE is achieved with the aid of NL elements such as verbs, nouns, genitive case, adjectives and adverbials.

Two of the major elements of NLSSRE, which we utilize in our use case approach, are the *Information Object* (IO) and the *CAREN* functions. According to NLSSRE, an IO is a digital representation of a tangible or intangible entity — described by a set of attributes — which the users need to manage through *Creating*, *Altering*, *Reading*, and *Erasing* its instances, and be *Notified* by the messages each instance (IOi) can trigger (an IO is conceived and processed at an abstraction level, while an IOi is conceived and processed at a factual level; instances of the same IO differ only in the values of their attributes). In NLSSRE, the *Create*, *Alter*, *Read*, *Erase* and *Notify* functions are called *CAREN* functions.

In the proposed approach, use cases are derived from the CAREN functions, therefore we call them *CAREN Use Cases* (CUCs). CUCs are system use cases, not business use cases. The formalization concept is more easily applicable to the system use cases, because they are applied on electronic information, while it is hardly applicable to the business level use cases, due to the complexity of the business environment, in both size and terminology. For example, *Enroll in Seminar* may be represented and implemented as a business or a system use case by conventional approaches (e.g., Cockburn, 2000), while in the proposed approach it is represented through the CUCs under the IOs *Enrolment* and *Seminar*. For the IO *Enrolment*, we have the system use cases *Create*, *Alter*, *Cancel*, *Erase* and *Read Enrolment*, and for the IO *Seminar*, we have the system use cases *Create*, *Alter*, *Cancel*, *Erase* and *Read Seminar* (often we consider that *Notify* is contained as a set of main flow actions in each of the rest CUCs–we will discuss this point later). For an enrolment to be created, a seminar needs to be already created, therefore the CUC *Create Enrolment* is extended by the CUC *Create*

Table 1: Part of the use case specification template for the CUC *Create IO.*

Basic flow	1. <**Creator**> selects create <**IO**>. 2. System displays new <**IO**> creation form, including required and optional fields. 3. <**Creator**>, <**Accompaniment**> enter(s) <**IO**><**IO.attribute.value**>. (Repeat 3,4) 4. System must check <**IO**><**IO.attribute.value**>. 5. <**Creator**> selects to submit the new <**IO**>. 6. System saves the new <**IO**> in the database. 7. System notifies <**Creator**>, <**Accompaniment(s)**>, <**Intended Recipient(s)**> that <**IO**> is created *via UC <**UC$_{in}$.ID**>*.
Alternative flow	If <**IO**> <**IO.attribute**> is incorrect, then system returns: "Invalid <**IO**><**IO.attribute**>. <**IO**> cannot be saved".

Table 2: Part of the use case specification for the CUC *Create Prescription.*

Basic flow	1. Doctor selects create Prescription by clicking on 'create prescription' button. 2. System displays new prescription creation form, including required and optional fields. 3. Doctor, Patient enter(s) Patient ID. 3.1. The System checks Patient ID. 12. The System notifies the Doctor, Pharmacist, and Patient that Prescription is created *via UC 15.*
Alternative flow	3.1. If patient ID is incorrect, then system returns: "Invalid Patient ID. Prescription cannot be saved".
Includes	UC 15: Send Notification

Seminar, and it also includes the CUC *Read Seminar* (fig. 1).

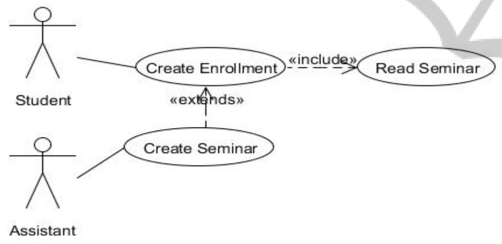

Figure 1: Conceptual representation of use cases through the proposed approach.

A typical use case is essentially described as a sequence of actions. To maintain a high-degree of readability and understandability and to minimize ambiguity, our approach formalizes the use case actions by providing specific types of actions, written in a structured form of NL. Apart from making expression of requirements more disciplined, understandable and organized, a structured form of specification also facilitates the automation procedures for building the use case specification per se, and also for later transformations into diagrammatic notations. The formalization is facilitated by utilizing elements of NLSSRE, such as the sub-functions of each CAREN function (Georgiades and Andreou, 2010).

Table 1 presents a part of the template that formalizes the actions of the CUC *Create IO*, and table 2 gives a relevant example.

For all CUC types, actions are either of request or response type. Normally, request actions are executed by an actor, and respond actions are executed by the system. Usually an actor's action is followed by a system's action. Exception conditions, which are part of the alternative flow of actions, are easily defined by the use of data constraints.

To identify the actors involved in each use case, we utilize the functional roles actor can play provided by the NLSSRE methodology. By making questions regarding the functional roles, we can identify the actors. Indicatively, a *Create* use case involves the functional roles *Creator*, *Accompaniment*, *Intended Recipient*, and *Notifiee*. The following are indicative question patterns for identifying (i) the *Creator*: Who should create an <IO> ?; and (ii) the *Accompaniment*: Who should assist the <Creator> to create an <IO>?

4 CONCLUSIONS

This paper presented an approach to formalize the use case model. The main motivation behind this endeavour is that existing use case driven analysis (UCDA) approaches often result in poorly defined use case models due to: (i) lack of specific support in identifying the basic use case elements, including use cases, actors, and use actions (ii) use of generic use case specification templates that do not guide the analyst clearly how to identify each element of the template; (iii) use of free natural language to describe the UC specifications, which, often results to inconsistencies and ambiguities in the use case model.

To address these weaknesses, we proposed (i) a formal semantics of the use case model, including the CAREN use cases, specific semantic/functional roles actors can play, specific types of basic and alternative flow actions for each CAREN use case, and specific question patterns to identify the actors; (ii) a semi-formal, NL-based syntax of the UC specification including a specific sequence of actions, for the basic and alternative flows of the CAREN use case specifications.

To evaluate the effectiveness and efficiency of the proposed formalization, we performed a short-scale empirical study through which we compared it to the classical UCDA approach, as described by Cockburn (2000), by applying both of them during the development of a subsystem of the Library Information System (LIS) of the University of Cyprus. Our evaluation tested the quality of the use case model (that is, use case specifications and use case diagrams) produced by the application of both approaches. The results showed that in general the proposed formalization performed much better than the classical approach in the various objective quality assessment metrics used, such as completeness, correctness and consistency. A detailed description of this comparative empirical study may be found in Georgiades and Andreou's work (2011).

Future work will involve the extension of the approach and the CASE tool in order to support the requirements design phase, with the creation of sequence, collaboration and state diagrams. The construction of such diagrams may be facilitated with the application of specific rules on the use case elements, such as the use case actions. Furthermore, we will work towards the enhancement of current types of actions (e.g., study of the circumstances within an action is performed) and identification and formalization of new ones. Additionally, alternative flow types of actions will be thoroughly explored, in addition to the exception condition types which are currently formalized. Moreover, although the proposed approach produced encouraging empirical results, it remains to be tested on real-world projects of a larger scale.

REFERENCES

Chalin, P., Sinnig, D., Torkzadeh, K. 2008. Capturing Business Transaction Requirements in Use Case Models. In *Proceedings of ACM Symposium on Applied Computing*, pp. 602-606.

Cockburn, A. 2000. *Writing Effective Use Cases*. Reading, Massachusetts: Addison Wesley.

Cox, K., Phalp, K. 2003. Exploiting Use Case Descriptions for Specifications and Design. In *Proceedings of EASE*, UK.

Dias, G., Schmitz, A., Campos, M. Correa, A., Alencar, A. 2008. Elaboration of use case specifications: an approach based on use case fragments. In *ACM Symposium on Applied Computing (SAC)*, Fortaleza, Ceara, Brazil, pp. 614-618.

Eriksson, M., Börstler, K., Borg, K. 2004. Marrying Features and Use Cases for Product Line Requirements Modeling of Embedded Systems. In *Proceedings of the Fourth Conference on Software Engineering Research and Practice (SERPS'04)*, Sweden, pp.73-82.

Georgiades, M., Andreou A., Pattichis, C. 2005. A Requirements Engineering Methodology Based On Natural Language Syntax and Semantics. In P*roceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05),* Paris, France. *IEEE Computer Society*, Washington, pp.73-74.

Georgiades, M., Andreou, A. 2010. A Novel Methodology to Formalize the Requirements Engineering Process with the Use of Natural Language. In *Proceedings of the IADIS Conference on Applied Computing*, Timisoara, Romania. IADIS Digital Library, pp.11-18.

Georgiades, M., Andreou, A. 2011. *Formalizing and Automating Use Case Model Development*, The Open Software Engineering Journal. Accepted.

Kim, J., Sooyong, P., Vijayan, S. 2004. A Linguistics-Based Approach for Use Case Driven Analysis Using Goal and Scenario Authoring. In *Proceedings of Applications of Natural Language to Data Bases*, pp. 159-170.

Leite, J., Rossi, G., Balaguer, M., Kaplan, G., Hadad, G., Oliveros, A. 1997. Enhancing a Requirements Baseline with Scenarios. In *Proceedings of Requirements Engineering*, Annapolis, USA.

Ochodek, M., Nawrocki, J. 2007. Automatic Transactions Identification in Use Cases. In *Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques, CEE-SET 2007*, Poznan, Poland, October 2007, pp. 55-68.

Pooley, R., Stevens, P. 1999. *Using UML - Software Engineering with Objects and Components*, Harlow: Addison Wesley Longman.

Salinesi, C. 2004. Authoring Use Cases. Chapter in *Scenarios, Stories, Use Cases*: I. Alexander and N. Maiden, John Wiley.

Somé, S. 2007. *Petri Nets Based Formalization of Textual Use Cases*, Tech. Report in SITE, TR2007-11, Uni. of Ottawa.

Sybase, 2002. *PowerDesigner. Object Oriented User's Guide*. Sybase.