A Textual Domain Specific Language for Requirement Modelling

Oyindamola Olajubu
University of Northampton,UK
oyindamola.olajubu@northampton.ac.uk

ABSTRACT

Requirement specification is usually done with a combination of Natural Language (NL) and informal diagrams. Modelling approaches to support requirement engineering activities have involved a combination of text and graphical models. In this work, a textual domain specific modelling notation for requirement specification is presented. How certain requirement attributes are addressed using this notation is also described.

Categories and Subject Descriptors

D.2.1 [**Software**]: Software Engineering

General Terms

Languages

Keywords

Requirement specification, Domain specific languages

1. PROBLEM AND MOTIVATION

Model Based Software Development (MBSD) is an approach to software development which focuses on the use of models in all the phases. It has led to increase in productivity through automation of various development activities including code generation and testing [6]. The development lifecycle of a software starts with the specification of what the software is to do or achieve. NL is often used for specification due to its expressiveness and ease of understanding by stakeholders. NL specifications can however be ambiguous and imprecise. Controlled Natural Languages (CNL), which are subsets of NL aim to increase the conciseness of specifications. However to support a model based approach to development these NL or CNL specifications would have to be converted into models. In [1], NL requirements specifications are first translated to CNL . Natural language processing techniques are then used to generate Software Cost Reduction (SCR) textual models for model based testing. Formal notations have also been used for specification especially in safety critical domains.

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
ACM. 978-1-4503-3675-8/15/08
http://dx.doi.org/10.1145/2786805.2807562

Formal notations are mathematically based languages which allow for concise specifications for test case generation. These languages however require a high learning curve rendering them impractical for all initial requirement specifications.

Modeling notations such as UML and SysML can also be used to support specifications. Models in these notations, however are geared towards refinement of requirements or design activities rather than initial requirement specification. A model based approach to requirement specification is presented in this paper using domain specific modelling concepts. Domain specific languages (DSLs) are languages tailored to a particular domain or built for a specific purpose. They can be described as modeling languages whose constructs are based on domain related concepts. DSLs developed for a particular purpose increase expressiveness of specifications by experts in a domain [4].A textual DSL is developed for requirement specification.

2. RELATED WORK

There are several existing approaches for requirement specifications using models. Techne [3] is an abstract requirement modelling language for the development of new requirement modelling languages. It allows for development of graphical modelling languages based on stakeholder goals or desires of the system to be modelled. This differs as a textual modelling language is presented in this work.

The authors of [2] propose a model driven methodology for requirements engineering that integrate the use of requirement models with CNL specifications. The methodology aims to combine the advantages of NL-based and model based documentation formats through bidirectional model transformations. Models are used for defining the context, goals, scenarios and functional hierarchy specification. The function hierarchy models are then transformed into CNL representation of the requirement specification. We take a more direct approach in that the requirement specifications are textual models. This also eliminates the need to perform further transformations to a text based format.

Another approach where models are used in requirements representation is presented in [5]. The aim of this approach is also to integrate models and textual specifications to improve the quality of requirement specification. The approach requires the creation of a textual specification chapter for each requirement element. Block diagrams and state-charts are then used to represent the architectural and behavioral

aspects of the specified element. This approach differs from our approach in that textual specifications are generated from graphical models (block diagrams and state-charts) while in our approach, the specifications are textual models.

The tool Cucumber[1] also uses a natural based DSL for automated test derivation. It allows for scenario-based behaviour specification. An iterative process of writing implementation code and running acceptance tests is done for the different parts of defined scenarios. The DSL in this work supports behaviour and non-behaviour requirements. In this work, the focus is on requirement specification a higher level of abstraction without execution of implementation code.

3. UNIQUENESS OF APPROACH

Existing methods of requirement modelling have involved natural text, graphical models and transformations between them. In this work, an approach where text and modelling are melded is presented. GE Aviation, our industry partner is used as a case study. The software development cycle at GE starts with NL requirements in MSWord. These requirements are then manually imported into the DOORs requirement management system. The requirements are also manually translated to Simulink Models in partial adoption of a model based approach to development.

3.1 DSL Implementation

The first phase involved analysis of sample requirements provided by GE. This was done to identify the structure of specification documents and domain concepts. The modelling language is then implemented using XText[2], a language development tool. The domain specificity of the approach lies in the definition of the grammar using identified domain concepts. The grammar in this context is equivalent to the metamodel of the language. The XText tool also generates a dedicated editor used for the requirement specifications/requirement modelling.

The DSL allows for specification of internal and external input/output signals, software components /elements with attributes. Different types of requirements can be represented using the language. The requirement types are templates used as guides for concise specifications. **Descriptive requirements** allow for optional NL descriptions of elements in the domain and the assignment of features or states to the elements. **Range based requirements** allow for the specification of acceptable upper and lower boundaries for defined elements. **Logic based requirements** are for specification of requirements where decisions are made based on a combination of boolean conditions. In cases where pseudo code is integrated with the specifications, **Pseudo requirements** can be used. **Transition requirements** are for behavioral descriptions of the implications of element state transitions. **Event based requirements** are for specifying requirements where consequences of events or combinations of events can be defined.

4. RESULTS AND CONTRIBUTIONS

The language development was followed by translation of sample NL requirements provided by GE into DSL specifications. This was done by manually interpreting the NL requirements and specifying the requirements using the editor generated by XText. The observations reported in this section are based on how DSL-specified requirements addresses some requirement quality attributes.

[1]https://cucumber.io/

[2]http://www.eclipse.org/Xtext/index.html

REQ_255 The Lateral/Vertical 'ERC' and 'WRD' labels shall be displayed in green.

Figure 1: Sample Natural Language specification

```
REQ13: The Lateral_ERC_label shall be displayed in green.
REQ14: The Lateral_WRD_label shall be displayed in green.
REQ15: The Vertical_ERC_label shall be displayed in green.
REQ16: The Vertical_WRD_label shall be displayed in green.
```

Figure 2: DSL representation of requirement in Figure 1

4.1 Atomicity

The grammar of the language is designed such that every requirement is unique and traceable with an ID attribute. The DSL approach addresses atomicity in that it allows compound NL requirement to be decomposed into individual requirements. This is done by the definition of one element for per requirement.

4.2 Completeness

This approach supports completeness in that it prevents hanging references. All the variables, input and output signals used for behavior requirement specifications have to be predefined before use. Foreign or undefined elements used within specifications are flagged as errors in the editor.

4.3 Unambiguity

Requirement specifications should be unambiguous and concise. It should be such that each requirement has no more than one interpretation. It was observed that the DSL approach addresses this quality attribute to some extent. The requirement in Figure 1 is a modified representation of a sample requirement from GE. These specifications can however be interpreted in multiple ways. The equivalent DSL specification in Figure 2 shows a more concise representation of the NL requirement.

5. CONCLUSIONS

In this work, a model based domain specific approach to requirement specification is presented. With the requirements represented as models, software development artefacts such as requirement based tests could be generated. How this approach addresses Atomicity, Completeness and Unambiguity requirement quality attributes is also described. Work is currently being done to extend the DSL prototype, develop a tool integrating a verification module for the requirements and automatic test case generation. This developed tool would also be evaluated by engineers at GE.

6. ACKNOWLEDGMENTS

Special thanks to GE Aviation UK and University of Northampton, UK for the continued support and joint sponsorship of this research.

7. REFERENCES

[1] G. Carvalho, D. Falcão, F. Barros, A. Sampaio, A. Mota, L. Motta, and M. Blackburn. Nat2testscr: Test case generation from natural language requirements based on scr specifications. *Science of Computer Programming*, 95:275–297, 2014.

[2] M. Fockel and J. Holtmann. A requirements engineering methodology combining models and controlled natural language. In *Model-Driven Requirements Engineering Workshop (MoDRE), 2014 IEEE 4th International*, pages 67–76. IEEE, 2014.

[3] I. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In *RE*, pages 115–124, 2010.

[4] A. Raja and D. Lakshmanan. Domain specific languages. *International Journal of Computer Applications*, 1(21):99–105, 2010.

[5] C. L. Robinson-Mallett. An approach on integrating models and textual specifications. In *Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE*, pages 92–96. IEEE, 2012.

[6] M. Volter, T. Stahl, J. Bettin, A. Haase, and S. Helsen. *Model-driven software development: technology, engineering, management*. John Wiley and Sons, New York, 2013.