Requirements Processing Tool

Software project documentation

Viliam Šimko, Jiří Vinárek, Ondřej Fiala, Jan Krajíček, Rudo Tomori

Contents

1 Introduction

1.1 Motivation

When developing a software application, analysts together with end-users negotiate the intended system behaviour. An important part of the process is the elicitation of user requirements ranging from use-cases to non-functional properties. A substantial part of the intended behaviour can be captured by writing textual use-cases. It is then up to the developers to transform the sentences to code and implement all the business parts of the system properly. Developers can identify business entities and actions invoked by these entities. Moreover, when following the iterative development paradigm, it is necessary that the requirements elicitation phase be visited multiple times thus highlighting the importance of traceability links [6].

Obviously, there is a significant gap between the original specification, written in natural language, and the final code. When coded manually there is a high chance that the intended behaviour of a system does not correspond to the behaviour of a running application. One of the possible approaches to minimizing such human errors is the model-driven development paradigm [3] where automatic transformations between models are employed. However, before any transformation can be used, formal models have to be constructed first. In case of use-cases written in natural language, this can be achieved in a semi-automatic way with the support of existing natural language processing (NLP) tools.

1.2 Related work

Based on the simple and uniform sentence structure used in textual use cases [5], a conversion scheme has been proposed in [7][8] and later implemented in the Procasor Environment project [11]. The tool developed provides a GUI that guides users in writing use-case steps as sentences in plain English. With the support of readily available NLP tools [12][13][14], Procasor Environment supports a semi-automatic transformation into formal notation called Procases (Behaviour Protocols) [1]. There has already been an attempt to take procases as an input for transformation to executable code [9].

1.3 Goals

Similarly to the Procasor Environment project, the goal is to develop an interactive environment for specifying requirements. The existing Procasor infrastructure has to be integrated into the Eclipse platform as a set of plugins, particularly the integration with [4] has to be achieved. Requirements will be specified as textual use cases employing the already developed transformation mechanism. The editor will provide two alternative ways of use case specification:

1. The tool will interactively derive formal specification of the system's behavior from the textual use cases as the user enters them.

2. The user will select paragraphs and sentences from an arbitrary text that will be progressively decomposed into elements of the meta-model with the help of the editor.

This way, both ”user-readable” textual specification and precise formal specifications of the system will be developed at the same time, only slightly increasing the effort required to write the use case models. Instead of the Procase-notation, the EMF framework will be employed and a meta-model capturing the system's behaviour will be filled during transformation from natural language. Once the requirements are formally captured by a meta-model, further transformation will be applied resulting into:

- Behavior Protocols [1]
- UML State Machines [2]
- Model of Components [10] – as an optional feature, elements from this model will be transformed to a simple executable prototype

It should be noted that the interface between the linguistic tools and the core application should be clearly defined and each NLP tool should be encapsulated as an Eclipse plugin.

References

[1] Plasil, F., Visnovsky, S.: Behavior Protocols for Software Components, IEEE Transactions on Software Engineering, vol. 28, no. 11, Nov 2002

[2] Object Management Group (OMG): Unified Modeling Language: Superstructure, version 2.3, `http://www.omg.org/spec/UML/2.3/Superstructure/PDF/`

[3] Stahl T., Voelter M., Czarnecki K.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, 2006

[4] Eclipse Modeling Framework Project (EMF), `http://www.eclipse.org/modeling/emf/`

[5] Cockburn, A.: Writing Effective Use Cases, Addison-Wesley Pub Co, ISBN: 0201702258, 1st edition, Jan 2000

[6] Larman C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition). Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004

[7] Mencl V.: Deriving Behavior Specifications from Textual Use Cases, in Proceedings of Workshop on Intelligent Technologies for Software Engineering (WITSE04, Sep 21, 2004, part of ASE 2004), Linz, Austria, ISBN 3-85403-180-7, pp. 331-341, Oesterreichische Computer Gesellschaft, Sep 2004

[8] Drazan J., Mencl V.: Improved Processing of Textual Use Cases: Deriving Behavior Specifications, in Proceedings of SOFSEM 2007, January 20 - 26, 2007, Harrachov, Czech Republic, LNCS 4362, pp.856-868, DOI: 10.1007/978-3-540-69507-3_74, Springer, Jan 2007

[9] Francu J., Hnetynka P.: Automated Code Generation from System Requirements in Natural Language, e-Informatica Software Engineering Journal, Volume 3, Issue 1, 2009

[10] Šimko V., Hnětynka P., Bureš T.: From textual use-cases to component-based applications, In proceedings of SNPD 2010, London, UK, Studies in Computational Intelligence (SCI), Springer, Jun 2010

[11] Procasor Environment: Interactive Environment for Requirement Specification, `http://d3s.mff.cuni.cz/~mencl/procasor-env/`

[12] Michael Collins: A New Statistical Parser Based on Bigram Lexical Dependencies, Proceedings of 34th Annual Meeting of the Association for Computational Linguistics, ACL 1996, 24-27 June 1996, University of California, Santa Cruz, California, USA, Morgan Kaufmann Publishers, 1996, `http://www.cis.upenn.edu/~mcollins/`

[13] Adwait Ratnaparkhi: A Maximum Entropy Part-Of-Speech Tagger, Proceedings of the Empirical Methods in Natural Language Processing Conference, May 17-18, 1996. University of Pennsylvania, 1996, `http://www.cis.upenn.edu/~adwait/statnlp.html`

[14] Minnen, G., Carroll J., Pearce, D.: Applied morphological processing of English, Natural Language Engineering, 7(3), pp. 207-223, 2001, `http://www.informatics.susx.ac.uk/research/nlp/carroll/abs/01mcp.html`