2014 9th International Conference on the Quality of Information and Communications Technology

Preliminary experience using JetBrains MPS to implement a requirements specification language

Dušan Savić
Software Engineering Department
Faculty of Organizational Sciences, University of Belgrade
Belgrade, Serbia
dules@fon.bg.ac.rs

Alberto Rodrigues da Silva
Department of Computer Science and Engineering
IST Universidade de Lisboa && INESC-ID
Lisboa, Portugal
alberto.silva@acm.org

Siniša Vlajić, Saša Lazarević, Ilija Antović,Vojislav Stanojević, Miloš Milić
Software Engineering Department
Faculty of Organizational Sciences, University of Belgrade
Belgrade, Serbia
{vlajic,slazar,ilijaa,vojkans,mmilic}@fon.bg.ac.rs

Abstract— **People prefer to use textual specification of requirements, but their representations are not suitable for automatic transformation and reuse. Use case modeling is commonly used to structure and document requirements. The integration of use cases within the Model Driven Development paradigm requires a rigorous definition of the use case specification. In this paper we describe the key part of SilabReq language for requirements specification based on use case and present the main result from our preliminary experience with implementation of the SilabReq language with JetBrains Meta Programming System.**

Keywords—requirements; requirements specification; use case; requirements specification tools;language workbench

I. INTRODUCTION

The development of information systems is a complex and social process that it involves many interactions among different stakeholders. In order to make this process successful, it is necessary to understand the system requirements and document them in a suitable manner. Requirements Engineering (RE) involves two main processes [1]: (1) requirements development (with elicit, analyze, specify, and validate software requirements) and (2) requirements management.

Use case modeling (including use case diagrams and use case textual specifications) is commonly used to structure and document requirements. Use cases are narrative descriptions of interactions between actors and a software system under study. Despite the fact that use cases are narratives, there is no single standard that helps to specify what textual specification of use case should be [2] [3]. The integration of use cases within the MDD paradigm [4] requires a rigorous definition of the use case specification, particularly description of sequences of actions, pre- and post-conditions, and relationships between elements from both use case models and domain models [5]. In this paper we introduce the second version of SilabReq, which is a use cases specification language, and present how SilabReq is supported by JetBrains Meta Programming System (MPS[1]). SilabReq language is part of the Silab project that has the goal to provide a complete software development workbench to be used by requirements engineers, developers, as well as by other non-technical stakeholders.

In short, the contributions of this article are twofold. First, the introduction of SilabReq specification language which can be used for requirements specification. Second, the summary of the preliminary experience implementing SilabReq with JetBrains MPS. Achieving this goal i.e. extending and supporting use cases within the MDD approach, use cases become a central artifact in software development and can be used at different levels of abstraction.

This paper is organized as follows. Section II describes the background of this work. Section III presents SilabReq requirements language. Section IV presents relevant implementation issues of the SilabReq language using JetBrains MPS. Section V discusses other alternative tools that could be used. Finally, Section VI concludes the paper and outlines future work.

II. BACKGROUND

Requirements are mostly documented using natural language, as paragraphs of text. However, natural language requirements specification tends to be ambiguous, unclear, and inconsistent. In fact, it is difficult to clearly define data, function and behavior perspectives of these requirements because they overlap. On the other hand, documenting requirements using semi-formal models require using specific modeling language for each particular perspective. Three types of requirements proposed by Klaus Pohl [1] are: (1) goals, which document intentions of stakeholders; (2) scenarios, that describe concrete example of system usage; and (3) solution-oriented requirements, that can be used as complement each other. Different requirements modeling

[1] http://www.jetbrains.com/mps/

languages can be used for modeling different requirements artifacts for different perspective. For example, i* [1] and KAOS[4] goal models can be used for specification of the goals; UML use cases, sequence and activity diagrams can be used for modeling scenarios; while UML state machine or data flow diagrams can be used for modeling behavior. In spite people prefer to use textual specification of requirements, their representations are not suitable for automatic transformation and also for reusing.

UML has become a standard language for modeling software systems and many people have used it for requirements specification. There is other Requirements Specification Language (RSL) that uses natural language in a controlled way. RSL [6] is a semiformal natural language that employs use case for specifying requirements. The goal of ProjectIT [7] [8] is to provide a complete software development workbench, with support for project management, requirements engineering, analysis, design and code generation activities. ProjectIT-Requirement is the component of the ProjectIT architecture that deals with requirements issues. The main goal of the ProjectIT - Requirement is to develop a model for the definition and documentation of requirements, which, by raising their specification rigor, facilitates the reuse and faster the integration with development environments driven by models. Taking into account the different types of requirements, this project uses software requirements, those that can more easily be "converted" in software design models by MDD approaches [9]. RSLingo [10] is a linguistic approach for improving the quality of requirements specification, which is based on two languages and mapping between them. The first language is RSL-PL (Pattern Language) extensible language for defining linguistic patterns dealing with information extraction from requirements written in natural language; and the second one is RSL-IL (Intermediate Language), formal language with a fixed set of constructs for representing and conveying RE-specific concerns. UML tools like IBM Rational Software Architect or EclipseUML do not support the description of the internal structure of use cases. On the other hand, tools focusing on textual use cases descriptions like CaseComplete[2] leave out the use case constructs defined by the UML specification. NaUTiluS [11] present an extensible, Eclipse-based toolkit, which offers integrated UML use case modeling support, as well as editing capabilities for their textual descriptions. NaUTiluS consists of a set of plug-ins that is embedded in the ViPER platform.

III. THE SILABREQ LANGUAGE

SilabReq language is a key result of Silab Project which was initiated in 2007 in the Software Engineering Laboratory at Faculty of Organizational Sciences, University of Belgrade. The main goal of this project was to enable automated analysis and processing of software requirements in order to achieve automatic generation of different parts of a software system. The first version of SilabReq was designed as a textual DSL [12], and was developed using Xtext framework.

To increase the readability and understandability of user requirements specification, we introduce the second version of SilabReq DSL[13][14] which proposes the idea of use cases specifications at different levels of abstraction and which focus our interest on exploring notations that allows a non-technical stakeholders to quickly read and understand use cases descriptions. This version of SilabReq DSL focuses on use cases specifications at different levels of abstractions according to the different roles that use cases play in the software development [13]. There are three different abstraction levels: (1) use case interaction level (high-level), (2) use case behavior level (medium-level), and (3) use case UI-based level (lower-level). each abstraction level extends and semantically enriches the previous level.

IV. IMPLEMENTING SILABREQ WITH JETBRAINS MPS

Language Oriented Programming [15] offers a style of development based on the idea of building software around a set of DSLs, while Language Workbenches is a generic term for tools that use this style of programming. Meta Programming System (MPS) from JetBrains is one of the most popular meta-programming systems that enable language oriented programming with a projection editor in persistent abstract representation [15].

A. *The supported tool: JetBrains MPS*

MPS provides its own language named BaseLanguage. MPS allows extending the BaseLanguage to create new custom languages, extend existing languages, and use them to develop software applications. During the process of creating a new language it is needed to derive concepts from the BaseLanguage as a reference for new languages. The major goal of MPS is to allow languages definitions through extension, which means that language's designer, can use concepts from a new extended language as well as combine concepts from different languages.

B. *The architecture of SilaReq language*

The goal of our approach is to provide a software development workbench based on MPS to better integrate and support requirements specification and software development activates. The current architecture view of this workbench is represented in Fig.1.

The SilabReq Core Language component defines the main concepts of the language such as Actor, SystemEntity, BusinessEntity, BusinessEntityAttribute or BusinessAttributeState. Also, in this component of the language it is defined concepts for Business Rule and Glossary definition. The SilabReq Language component defines the use case meta-model which is linked with concepts defined in SilabReq Core Language component and BaseLanguage while SilabReq UI Language component is used to specify UI lower-level aspects of use case. Finally, the SilabReq Transformation component is responsible for transforming requirements specified in

[2] http://casecomplete.com/

SilabReq into different models. Currently, we have developed transformations that transform from SilabReq model into UML use case and class models.

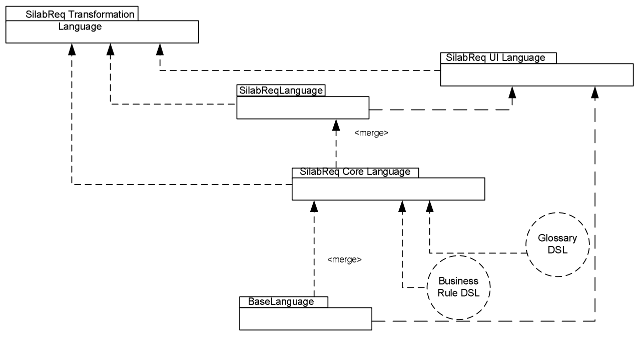


Figure 1 Architecture of SilabReq language

From a developer perspective, this programming approach seems very promising. Developers have two ways to implement software. First, they can use requirements specification to manually create source code. Second, they can use or create different transformation to generate source code from these models. Both alternatives can be used and integrated with MPS because there is already a plug-in for IntelliJ IDEA which allows including MPS concept models in Java project. So, programmer can use MPS for writing Java application and integrate relationship between source code and requirements specification.

C. Some aspects of SilabReq implementation

MPS comes with some sets of DSLs which are useful to define the structure of language, editor, type systems, and generators. All of these DSLs are built using MPS itself. The language definition starts by defining its abstract syntax (Fig.2). The concept is one element of language in MPS which describes how the respective element looks like, behave and generate[3]. So, each concept can have a definition in one or more aspects of the language such as structure, editor or generator. Fig. 3 shows the Concept Declaration Editor for some part of SilabReqUseCase specification language, while Fig.4 depicts the using of Aspect Editor for the UseCase concept.

MPS generate Java source code from the requirements specification model. Java is embedded into MPS, so Java source code generation is a simple transformation based on the template mechanism[4]. This transformation has two main important building blocks: mapping rules (define which concepts are processed with which templates), reduction rules (define transformations which removes source node and replace it with associated template) and templates.

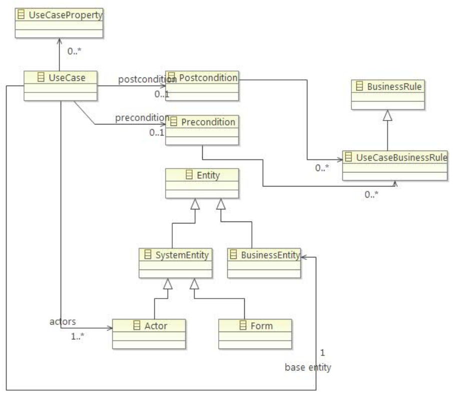
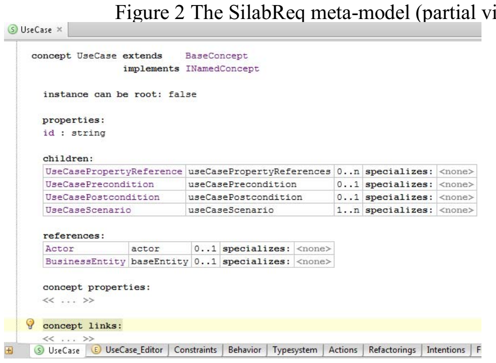
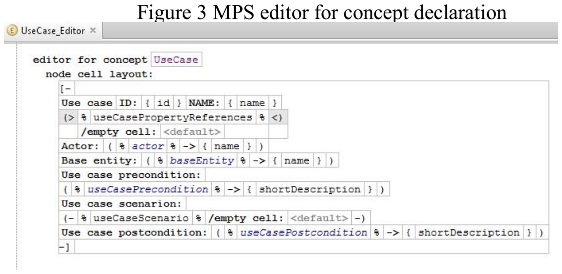


Figure 2 The SilabReq meta-model (partial view)



```
concept UseCase extends    BaseConcept
                implements INamedConcept

  instance can be root: false

  properties:
  id : string

  children:
  UseCasePropertyReference useCasePropertyReferences 0..n specializes: <none>
  UseCasePrecondition      useCasePrecondition       0..1 specializes: <none>
  UseCasePostcondition     useCasePostcondition      0..1 specializes: <none>
  UseCaseScenario          useCaseScenario           1..n specializes: <none>

  references:
  Actor          actor      0..1 specializes: <none>
  BusinessEntity baseEntity 0..1 specializes: <none>

  concept properties:
  << ... >>

  concept links:
  << ... >>
```

Figure 3 MPS editor for concept declaration



```
editor for concept UseCase
  node cell layout:
    [-
    Use case ID: { id } NAME: { name }
    (> % useCasePropertyReferences % <)
       /empty cell: <default>
    Actor: ( % actor % -> { name } )
    Base entity: ( % baseEntity % -> { name } )
    Use case precondition:
    ( % useCasePrecondition % -> { shortDescription } )
    Use case scenarion:
    (- % useCaseScenario % /empty cell: <default> -)
    Use case postcondition: ( % useCasePostcondition % -> { shortDescription } )
    -]
```

Figure 4Aspect Editor for Use Case concept

V. DISCUSSION

MPS as well as other Language Workbenches Tools (see Table I) offer capabilities for extending an existing language. MPS is not a classic grammar-based IDE for creating DSL. The main strength of MPS is that it is not parser-based, giving consequently much freedom to a language designer and allowing combinations of languages. Instead of text editors it provides a "projection editor" which means that MPS works with abstract syntax tree instead of parsing a concrete syntax. Also, MPS has a module support that enables us to include existing languages into projects and can use text or table as a textual representation of concrete syntax. On the other hand, its main weaknesses are that it is a standalone tool and, currently, can be only integrated with IntelliJ IDE and also it does not support graphical concrete syntax. In contrast, Spoofax[5] is a text-based language workbench that uses high-level SDF grammar formalism for DSL definitions. The main strength of Spoofax tool is that it is an Eclipse integrated tool which enables exporting the

[3] http://dslbook.squarespace.com

[4] http://dslbook.squarespace.com

[5] http://strategoxt.org/Spoofax

created DSL as an Eclipse plug-in. Obeo Designer[6] is another workbench focuses on graphical modeling languages. It is based on Eclipse and works with Ecore metamodel. Like MetaEdit+, it provides a symbol editor to create your own graphical language. MetaEdit+[7] is an example of a pure graphical language workbench based on defining languages using meta-models, constraints and graphical symbols associated with the meta-model. In MetaEdit+ languages can be defined in either a graphical or form-based manner using the GOPPRR[8].

TABLE I. LANGUAGE WORKBRANCHES TOOLS

	MPS	**Spoofax**	**Obeo**	**MetaEdit+**	**Xtext**	**Papyrus**	**EMFText**
Which representation is used for the abstract syntax	meta-model	grammar	meta-model	meta-model	grammar	meta-model	grammar
Which representation can be used for concrete syntax	text, table, (graphical soon)	text	graphical	graphical, matrix	text	graphical	text
Is it integrated with IDE?	InteliJ	Eclipse based	Eclipse based	Eclipse plugin, Visaual Studio	Eclipse based	Eclipse based	Eclipse based
Is it parser based?	No	Yes		No	Yes	No	Yes
Is it commercial?	No	No	Yes	Yes	No	No	No

Creating a DSL can be achieved using the Eclipse platform with Xtext[9], EMFText[10] or Papyrus[11] . Xtext supports the creation of extremely powerful text editors with code completion, error checking and syntax coloring. The framework uses the BNF grammar for the description of concrete syntax of the language. Papyrus provides diagram editors for EMF-based modeling languages; amongst them UML 2 and SysML. It also offers an advanced support of UML profiles definition that enables to define editors for DSL based on the UML 2 standard and its extension mechanisms.

Comparing with these alternatives, we decide to use MPS mainly for two reasons. First, it uses meta-model approach to represent abstract syntax. Second, it provides a projection editor, even if it does not support integration with IDE such as Eclipse or NetBeans, but instead of full integration with Eclipse, it is still possible to use EMF meta-model.

VI. CONCLUSION

Use cases are graphical and narrative approaches to requirements specification because they describe the context and requirements in natural language. Therefore, use cases are primary a textual representation technique. The key point in describing use cases lies in their readability and understanding for the majority of participants in the software development project.

In this paper we describe the key part of SilabReq language for requirements specification based on use case and present the main result from our preliminary experience implementing the SilabReq language with MPS. Currently, this approach supports the specification of software requirements based on use cases and do not care how these use cases are discovered (from textual requirements specification or from goals or business models). In the near future, we intend to integrate this approach with RSLingo approach [10]. Also, SilabReq language leave out of use case constructs defined by the UML specification and in the future it will be also defined as a UML profile. The goal of this approach is to provide a complete software development workbench based on MPS to integrate requirements specification and software development activates. This will allow us to use languages for requirements specification integrated with implementation languages for full support model driven development approach.

[6] http://www.obeodesigner.com/
[7] http://www.metacase.com/
[8] https://www.metacase.com/support/45/manuals/mwb/Mw-1_1_1.html
[9] www.eclipse.org/Xtext
[10] http://www.emftext.org/
[11] www.eclipse.org/papyrus/

REFERENCES

[1] K.Pohl, Requirements Engineering - Fundamentals, Principles, and Techniques. Springer 2010

[2] S. Som´e, "A Meta-model for textual use case description", Journal of Object Techology, Zurich, 2009.

[3] F.L.Siqueira, and P.S.M.Silva, "An Essential Textual Use Case Meta-model Based on an Analysis of Existing Proposals". WER 2011

[4] P.Nguyen, and R.Chun, "Model Driven Development with Interactive Use Cases and UML Models", Software Engineering Research and Practice 2006:534-540

[5] H. Astudillo, G. Génova, M. Smialek, J. L. Morillo, P. Metz, and R. Prieto-Díaz, "Use cases in Model-Driven Software Engineering", In Proceedings of MoDELS Satellite Events,2005. pp.272-279

[6] M. Smiałek, J.Bojarski, W.Nowakowski and T. Straszak, "Scenario construction tool based on extended UML metamodel". Lecture Notes in Computer Science, 3713:414–429, 2005.

[7] A.Silva, C.Videira, J.Saraiva, D.Ferreira and R.Silva, "The ProjectIT-Studio, an integrated environment for the development of information systems", In Proc. of the 2nd Int. Conference of Innovative Views of .NET Technologies (IVNET'06), pages 85–103. Sociedade Brasileira de Computação and Microsoft.

[8] A. R. d. Silva, J. Saraiva, D. Ferreira, R. Silva, and C. Videira, "Integration of RE and MDE Paradigms: The ProjectIT Approach and Tools", IET Software: On the Interplay of .NET and Contemporary Development Techniques, 2007

[9] D. A. Ferreira and A.R.Silva,"A Controlled Natural Language Approach for Integrating Requirements and Model-Driven Engineering", ICSEA 2009: 518-523

[10] D. A. Ferreira and A.R.Silva,"RSLingo: An information extraction approach toward formal requirements specifications", MoDRE 2012: 39-48

[11] V.Hoffmann, H.Lichter, A. Nyßen and A.Walter, "Towards the Integration of UML- and textual Use Case Modeling", Journal of Object Technology 8(3): 85-100 (2009)

[12] D.Savic, I. Antovic, S. Vlajic, V. Stanojevic and M. Milic, "Language for Use Case Specification ", Conference Publications of 34th Annual IEEE Software Engineering Workshop, IEEE Computer Society, 2011, Pages: 19-26, ISSN : 1550-6215, ISBN: 978-1-4673-0245-6, Limerick, Ireland, 20-21 June 2011, DOI: 10.1109/SEW.2011.9

[13] Savić, A.R. Silva, S.Vlajić, S.D.Lazarević, V.Stanojevic, M.Milić, M.Milic, "Use Case Specification at Different Abstraction Level", 8th International Conference on the Quality of Information and Communications Technology, Lisbon, Portugal, 03-06.09. 2012

[14] I.Antović, S.Vlajić, M.Milić,D.Savić and V.Stanojević, "Model and software tool for automatic generation of user interface based on use case and data model," Software, IET , vol.6, no.6, pp.559-573, Dec. 2012 doi: 10.1049/iet-sen.2011.0060D.

F, Martin. Language Workbenches: The Killer-App for Domain Specifc Languagess [online]. Available on: http://martinfowler.com/articles/languageWorkbench.html