A Tool Implementation of the Unified Requirements Modeling Language as Enterprise Architect Add-In

Florian Schneider, Bernd Bruegge
Institut für Informatik
Technische Universität München, Germany
florian.schneider, bernd.bruegge@in.tum.de

Brian Berenbach
Siemens Corporate Research
Princeton, NJ, USA
brian.berenbach@siemens.com

***Abstract*—Early modeling languages have focused on the analysis and the design of systems under construction, but not on requirements elicitation. Consequently, a wide variety of approaches exists for the early phases of requirements engineering, modeling various concepts as stakeholders, goals, features, product lines, systems, processes, risks, and requirements. The purpose of Unified Requirements Modeling Language (URML™) is to combine these concepts in a single modeling language. In addition, it strengthens support for danger modeling. URML is implemented as an add-in to the Enterprise Architect CASE tool. This tool demo will showcase the URML implementation.**

I. Introduction

The Unified Modeling Language (UML) [1] does not explicitly support requirements. Use cases can be considered as a transformation of requirements during systems analysis, but do not help finding requirements. SysML [2] is a first step to notationally support requirements so they can be described independently from use cases. With its focus on modeling workflows, BPMN [3] was also not designed for requirements elicitation. The ReqIF standard of the Object Management Group [4] only supports the exchange of textual requirements. In terms of modeling languages for requirements engineering, a wide variety of domain-specific languages has been published, each of them focusing on specific aspects of requirements. For example, KAOS [5] and URN [6] treat stakeholder goals as first-level citizens, but they do not address the modeling of product lines and the differentiation between physical hazards and threats. URML is a new language that connects the important concepts needed for early requirements elicitation and thus enables traceability between them (see details in next section). URML can act as a front-end to the aforementioned languages. URML is implemented as an add-in to the Enterprise Architect (EA) CASE tool, leveraging the UML profile mechanism for extension. The goal of demonstrating the add-in is to get feedback about the feasibility of our approach. Through discussing the language implementation with experienced requirements modelers, we hope to get feedback with respect to the scope and the usability of the tool. Furthermore we want to provide a basis for future discussion of the language among requirements engineering experts.

II. URML

The idea of unifying various requirements modeling concepts in one modeling language has been proposed by Berenbach et al. since 2004 [7]. Since September 2009, the language was actively developed in a collaboration of TUM and Siemens Corporate Research. The progress of the language was reported to the systems and requirements engineering communities (in e.g. [8] or [9]). A preliminary version of the URML specification can be obtained by request from the authors. URML unifies concepts from goal-oriented, feature-oriented, process-oriented, and risk-oriented approaches and integrates them with the classical view of functional and nonfunctional requirements. That integration enables traceability between the following concepts that are central to the URML: dangers, processes, goals, stakeholders, features, and requirements. An example model is shown in Fig. 1. A major

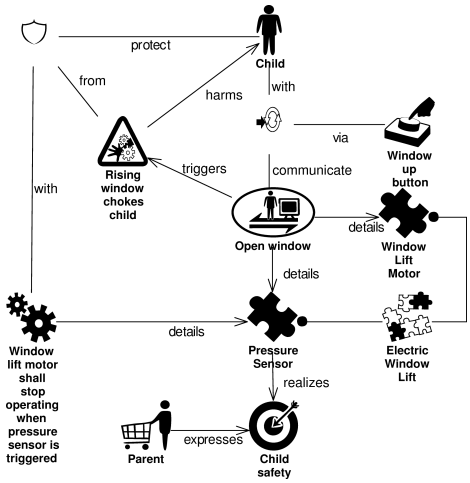

Fig. 1. An exemplary URML model created with the Add-In

feature of URML is its icon-based notation. In contrast to other modeling languages, the graphical notation does not have generic shapes such as rectangles and ovals. Instead, URML offers easily recognizable and distinguishable icons. Every concept of the language that can appear on a diagram is represented by an icon. With the notation of the language being icon-based, every unique concept is represented by a unique icon. Furthermore, attribute values can lead to variations of

978-1-4673-5765-4/13/$31.00 © 2013 IEEE

the base icon of a concept. The notations of UML and i* had been criticized by Moody [10] for a lack of perceptual discriminability and semiotic clarity. When creating the URML, we thus decided not to re-use the symbols used in other languages we incorporated the concepts from.

III. URML Enterprise Architect Add-In

URML is currently implemented as an add-in to the Enterprise Architect (EA) CASE tool [11]. The add-in is largely defined in terms of models, more precisely UML profile diagrams. The UML profile diagrams not only introduce the new concepts and their notation, but also a custom diagram type and toolbox extensions to hold icons representing URML concepts. The URML Add-In extends EA in the following ways: 1) It adds a new language by defining a UML profile with some EA specific additions 2) It provides language-specific toolbars that contain the concepts and relationships of the language. Each concept and relationship is signified by a miniature icon in the toolbar. 3) It defines which concepts or relationships can be created through a contextual quick linker menu. Further additions to the add-in could include validation rules to check correct syntax or heuristics for model quality. Apart from the custom diagram types that come with the UML profiles defining the language, no custom user interface elements can be added to the tool through this mechanism.

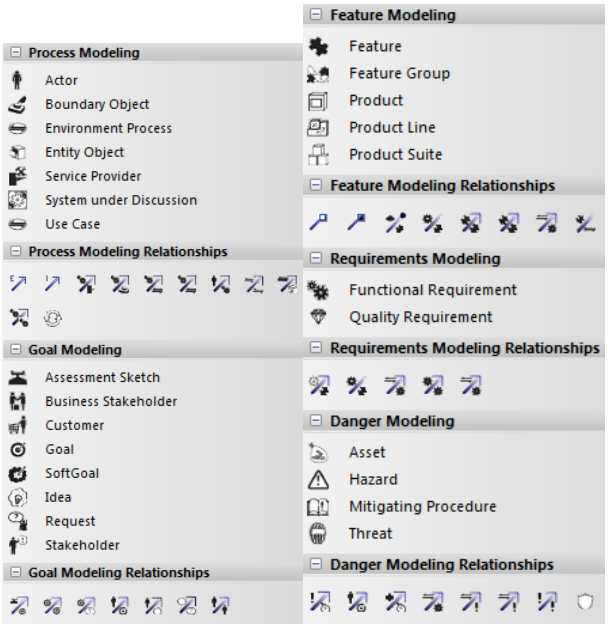

Fig. 2. Toolbar of the URML Add-In

IV. Related Modeling Approaches

VLML described by Glinz and Wüest [12] proposes to use only very few constructs to model early requirements. The language provides means for structuring text. We assume some aspects of it have been realized with the FlexiSketch tool by Wüest [13] and some aspects were showcased earlier with the implementation of the ADORA language [14]. KAOS is a goal-oriented approach that was proposed in the early nineties. The book by Lamsweerde can be seen as a reference guide to the latest state of the language [5]. KAOS is implemented within the Objectiver tool [15]. The User Requirements Notation (URN) [6] combines intentional (i.e. goal-oriented) concepts with use case maps. It is implemented as an Eclipse plugin-in, jUCMNav [16].

V. Future Work

URML in its current state was implemented as an add-in to Enterprise Architect. This was due to the wide use of EA within Siemens. As a result, the tool already has been used in two commercial Siemens projects and one innovation case study. For further dissemination of the language, we plan implementations for other CASE tools. Usually the extensibility of standard CASE tools is limited to some extent. To explore language-specific user interface paradigms, that deviate from the UI elements offered by standard CASE tools, an open source platform like Eclipse could be a viable alternative. The focus of the current implementation of URML has been on the iconic notation of the concepts themselves. We think however that the visualization of relationships between concepts is equally important. Our plan is to do more research on the visualization of relationships.

References

[1] Object Management Group (OMG), "OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1," Aug. 2011.

[2] "OMG Systems Modeling Language (OMG SysML), *Version 1.3*," Object Management Group (OMG), Jun. 2012.

[3] "Business Process Model And Notation (BPMN) Version 2.0," Object Management Group (OMG), Jan. 2011.

[4] "Requirements Interchange Format (ReqIF) *Version 1.0.1*," Object Management Group (OMG), Apr. 2011.

[5] A. van Lamsweerde, *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[6] "ITU-T Z.151: User requirements notation (URN) – Language definition," International Telecommunication Union, 2008.

[7] B. Berenbach, "Toward a unified model for requirements engineering," in *"Fourth International Workshop on Adoption-Centric Software Engineering (ACSE 2004)" co-located with the "26th International Conference on Software Engineering"*. IET, 2004, pp. 26–29.

[8] F. Schneider, H. Naughton, and B. Berenbach, "A modeling language to support early lifecycle requirements modeling for systems engineering," *Procedia Computer Science*, vol. 8, no. 0, pp. 201–206, 2012.

[9] B. Berenbach, F. Schneider, and H. Naughton, "The use of a requirements modeling language for industrial applications," in *Requirements Engineering Conference (RE), 2012 20th IEEE International*, 2012.

[10] D. L. Moody, "The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 756–779, 2009.

[11] Enterprise Architect. [Online]. Available: http://www.sparxsystems.com/products/ea/index.html

[12] M. Glinz and D. Wüest, "A Vision of an Ultralightweight Requirements Modeling Language," Zürich, Switzerland, Tech. Rep. 2010.IFI-2010.0006, Jul. 2010.

[13] D. Wüest, N. Seyff, and M. Glinz, "Flexible, lightweight requirements modeling with Flexisketch," in *Requirements Engineering Conference (RE), 2012 20th IEEE International*, 2012.

[14] ADORA Homepage. [Online]. Available: http://www.ifi.uzh.ch/rerg/research/adora.html

[15] Objectiver. [Online]. Available: http://www.objectiver.com/index.php?id=4

[16] jUCMNav. [Online]. Available: http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome