Unified Requirements Modeling for Environmental Systems

Jonas Helming, Maximilian Koegel, Bernd Bruegge
Institut für Informatik
Technische Universität München

{helming, koegel, bruegge}@in.tum.de

Brian Berenbach
Siemens Corporate Research
Princeton, NJ

brian.berenbach@siemens.com

ABSTRACT

Building environmentally friendly software and hardware systems is about understanding and respecting the requirements that are imposed by climate change or regulation. This is especially true, as those systems often require inter-disciplinary collaboration between domain experts who might not have a software or systems engineering background. Existing requirements models lack in three dimensions supporting this objective: First, they do not integrate approaches to explicitly model environmental requirements and possible hazards. Second, they do not provide visualizations that are intuitively understandable by non-system engineers. Third, there is no integrated tool-support for requirements modeling in a globally distributed set-up.

We propose a unified requirements modeling language (URML) to address these problems. URML supports integrated requirements modeling as well as the cross-disciplinary visualization of environmental requirements, regulations and hazards. Furthermore we describe UNICASE, a tool for collaboration on these models allowing the visualization of traceability from climate critical requirements to their implementation in the system.

Categories and Subject Descriptors

D.2.2 [**Requirements/Specifications**]: Languages, Tools

Keywords

Requirements engineering, modeling, software engineering, systems engineering.

1. INTRODUCTION

Understanding the requirements and their interrelations for complex systems has always been a critical success factor for software and system projects [1]. The ongoing trend to focus on environmental sustainability (e.g. carbon neutrality) adds a new dimension of complexity to the problem of requirements analysis. More concrete environmental sustainability and the mitigation of environmental hazards add new requirements or even new types of requirements to a project. These requirements are highly interrelated with existing system requirements, in fact they are often conflicting. As an example, the non-functional requirement "high maximum speed for a car" may conflict with the environmental requirement of low carbon emissions. We claim, it is necessary to enable a dialog and understand the interdependencies between all the requirements. In the GEMS [2] system we already used a high level modeling language [3] to express the requirements in a way that could be understood by domain experts from multiple disciplines such as mechanical engineering, chemistry and computer science. The power of modeling languages has evolved significantly since then, leading to languages such as UML [4] and SysML [5]. In contrast to UML or SysML, URML offers explicit traceability between requirements from different domains. For example it offers traceability between business goals, product features, functional requirements, possible threats and possible hazards and their mitigation. The analysis of climate critical requirements usually involves the collaboration of domain experts from a range of different disciplines such as carbon emission exports and software engineers. Therefore it is necessary to integrate quality assurance for the requirements model. For this purpose URML provides executable validation rules. For example it can be checked whether critical environmental risks have been assigned sufficient mitigations. Furthermore a multi-disciplinary approach of requirements engineering requires a commonly understandable visual syntax. "Historically, SE researchers have ignored or undervalued the role of visual syntax" [6]. "The Details of Conceptual Modeling Notations are generally shrouded in mystery" [7]. URML focuses on visualization, which is intuitively understandable by non-software engineers and therefore fosters inter-disciplinary collaboration.

Although URML defines a language to express, validate and interrelate requirements, it does not provide tooling or an implementation. UNICASE is a platform for building modeling applications with tooling for traceability, visualization, validation and collaboration. Therefore we propose UNICASE as a platform for the implementation of URML.

2. UNIFIED REQUIREMENT MODELING

The Unified Requirements Modeling Language (URML) has been initially proposed in [8] and refined in [9]. The basic idea is to create a single, traceable and consistent requirements model instead of relying on thousands of pages of text with manually created traceability links. Requirements are captured in a visual language following the principles for cross-disciplinary use [6]. URML contains not only diagrams but also semantics defining the relationships between requirements and rules for creating diagrams and textual artifacts. It incorporates elements of:

- Goal Modeling
- Use cases (including UML related artifacts)
- Feature Modeling
- Hazard Analysis
- Threat Modeling
- Other requirements related visual modeling techniques

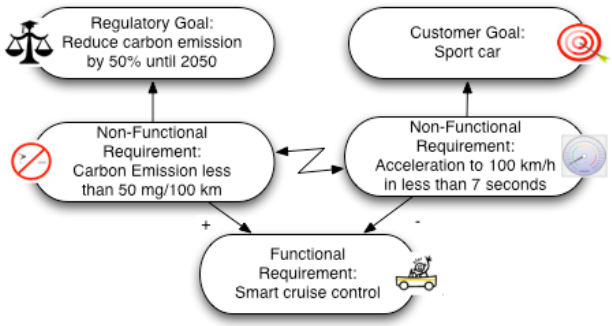

Figure 1: Example for a requirements model in URML

A unified requirements model is a model that has been created using the URML. In such a model, all related requirements are intrinsically visible to all the stakeholders without manual tracing. The requirements created and their relationships are inherently unambiguous, and can be reviewed by all stakeholders with minimal language or cultural issues. The models scale nicely and manage crosscutting requirements, such as climate critical requirements. Furthermore, project plans, requirement specifications and tests can be generated directly from the model.

3. UNICASE

UNICASE is a platform based on Eclipse to support the implementation of model-based tools focused on research in that area [10]. As all artifacts in UNICASE are part of a unified model, we will refer to them as model elements. UNICASE offers support for collaboration on model elements and versioning [11]. This supports multi-disciplinary and distributed projects. Model elements can be associated to each other by model links offering explicit traceability as required by URML. As an example a model element representing a hazard can be linked to a model element representing an actor, modeling that the hazard threads that actor. UNICASE provides an initial set of model elements from two domains, the system model and the project model. Model elements from the system model describe the system under construction, such as functional requirements, sustainability requirements, and potential hazards. Model elements from the project model describe the on-going project, such as tasks, bug reports, the organizational structure, iterations or meetings. This helps to coordinate a distributed project and offer traceability, e.g. from requirements driven decisions made in meeting to the affected model elements. The underlying model of UNICASE is easily extensible by defining new model elements or links between them. Therefore our proposed model could be extended, e.g. to implement new elements of URML. UNICASE offers generic editors for browsing and modifying model elements: (1) A tabular view, showing a filtered and sort able list of model elements, (2) A tree-based view, showing the containment structure of the model, (3), A form-based editor to visualize and modify the content of one model element and its references (4) A diagram view, showing graphical representations such as UML diagrams. Additionally UNICASE can be extended by model specific views, e.g. a task view showing all tasks of a certain developer. Furthermore UNICASE offers support for tool instrumentation and project analysis based on history data [12].

4. CONCLUSION

In this paper we presented the ongoing project of creating a Unified Requirements Modeling Language for modeling requirements for complex systems including environmentally critical requirements. We are currently mining existing approaches of requirements engineering to include and integrate their ideas and concepts in URML. The goal is a taxonomy for a unified requirements model. Based on this taxonomy we want to enhance URML with missing concepts, such as product line modeling. Further we want to research in detail, how environmentally critical requirements can be better supported by the model and how decisions and conflicts can be better supported. We plan to demonstrate those concepts with a prototype based on UNICASE. Evaluated concepts shall have influence on commercial requirements tool vendors to better support multi-disciplinary projects.

5. REFERENCES

[1] R.N. Charette, Why software fails, IEEE spectrum, vol. 42, 2005, pp. 36.

[2] B. Bruegge, E. Riedel, G. McRae. T. Russel, Developing GEMS: An Environmental Modeling System, Journal for Computational Science and Engineering, IEEE, pp.55-68, September 1995.

[3] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy,W. Lorensen, Object-Oriented, Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1991.

[4] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 2005.

[5] OMG, The system modeling language SysML, http://www.omg.org/spec/SysML/1.1/, 2008.

[6] D.L. Moody, The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering, IEEE Trans. on Software Engineering, 2009, pp. 756–779.

[7] S. Hitchman, The Details of Conceptual Modelling Notations are Important-A Comparison of Relationship Normative Language, Comm. of the Association for Information Systems, vol. 9, 2002, pp. 10.

[8] B. Berenbach, M. Gall, Toward a Unified Model for Requirements Engineering, ICGSE, 2006, pp. 237-238.

[9] B. Berenbach, T. Wolf, A unified requirements model; integrating features, use cases, requirements, requirements analysis and hazard analysis, ICGSE, 2007, pp. 197-203.

[10] B. Bruegge, O. Creighton, J. Helming, M. Koegel, "Unicase – an Ecosystem for Unified Software Engineering Research Tools," Workshop Distributed Software Development - Methods and Tools for Risk Management, Bangalore, India: 2008, pp. 12-17.

[11] M. Koegel, J. Helming, S. Seyboth, Operation-based conflict detection and resolution, 2009 ICSE Workshop on Comparison and Versioning of Software Models, Vancouver, BC, Canada: 2009, pp. 43–48.

[12] M. Koegel, Y. Li, H. Naughton, J. Helming, Towards a Framework for Empirical Project Analysis for Software Engineering Models, ASE VASE Workshop, Auckland, New Zealand: 2009.