ReqPat: Efficient Documentation of High-Quality Requirements using Controlled Natural Language

Markus Fockel and Jörg Holtmann
Software Engineering, Project Group Mechatronic Systems Design, Fraunhofer Institute for Production Technology IPT
Zukunftsmeile 1, 33102 Paderborn, Germany
Email: [markus.fockel | joerg.holtmann]@ipt.fraunhofer.de

***Abstract*—The growing complexity of today's software-intensive systems results in an increased size of requirements specifications, which are typically documented by means of natural language (NL). Large NL requirements specifications are prone to contain defects (e.g., contradictions), and the inherent ambiguity of NL impedes automatic techniques to support the requirements engineer. In order to cope with this problem, we conceived a requirements documentation approach implemented in the tool REQPAT. Using a controlled NL, it supports an efficient requirements documentation, an automatic requirements validation, and an automatic transition to models—while still keeping the requirements understandable for all stakeholders.**

I. INTRODUCTION

The growing functionality and complexity of today's software-intensive systems results in an increased size of requirements specifications, which are typically documented by means of natural language (NL)—particularly in the embedded sectors (e.g., automotive) [9]. For instance, the sum of all requirements specifications of an overall vehicle easily reaches the category "Very Large-Scale Requirements Engineering" of Regnell et al. [8] with an order of magnitude of ten thousands of requirements, according to [7]. NL requirements specifications are prone to contain ambiguous, contradictory, or incomplete requirements. Furthermore, the inherent ambiguity of NL impedes automatic techniques to support the requirements engineer. Thus, requirements documentation, validation, and the transition to models have to be performed manually, which is tedious, time-consuming, and error-prone.

In order to cope with this problem, we conceived in previous work a requirements documentation approach that applies so-called *Requirement Patterns* as controlled NL (CNL) (e.g., [10]) for the specification of functional requirements. This CNL restricts the expressiveness of NL to enable automatic processing of the requirements while still keeping them understandable for all stakeholders. The approach is a part of a requirements engineering methodology combining models and CNL [3], [1] and integrated into a design methodology for automotive systems [2], [4], [5].

In this extended abstract, we sketch the tool support REQPAT for this requirements documentation approach. We illustrate our approach with the example of an automotive electronic control unit called Body Control Module, which centrally controls distributed vehicle body functions like turn signals, brake light, and central locking.

II. PATTERN-BASED REQUIREMENTS DOCUMENTATION

By means of the requirement patterns presented in this abstract, we describe and decompose the overall functionality of the system under development using so-called *analysis functions*. Furthermore, the inputs and outputs required and provided by the different analysis functions are specified and decomposed along with the analysis functions.

Two requirement patterns that we use in our example are listed below. They consist of static parts, variable parts (contained in '<', '>'), and alternative parts ('|').

1) The analysis function <*function*> is a subfunction of the analysis function <*parent*>.
2) The following information is (used | created) by the analysis function <*function*>: <*signal list*>.

Requirement patterns have to be conceived along with a domain-specific requirements documentation methodology. The presented methodology is used to systematically decompose functionality. However, the general approach is transferable to other methodologies and requirement patterns. A more comprehensive presentation of the methodology and its catalog of requirement patterns can be found in [1].

III. REQPAT: TOOL SUPPORT

Our editor REQPAT seamlessly integrates into the Eclipse Requirements Management Framework (RMF) and its user interface ProR (cf. Fig. 1). We also provide an integration into IBM Rational DOORS. In the following, we present the main features of REQPAT.

A. Constructive Support for Requirements Documentation

REQPAT provides an editor that supports the user while specifying requirements by making suggestions for requirement patterns and variables (cf. the popup in the lower part of Fig. 1). These suggestions base on the current location in the requirements specification (e.g., the chapter) and other requirements that were written using requirement patterns.

In Fig. 1, a new pattern-based requirement is inserted within the chapter 1.4.1.2 Inputs & Outputs of the parent chapter 1.4.1 Lighting. Thus, REQPAT only proposes patterns describing the inputs and outputs of the analysis function Lighting. In the figure, the user has chosen to specify Lighting's inputs via the requirement pattern 2 (cf. Sect. II). For the variable part <*signal list*> of the pattern, REQPAT suggests inputs that were defined in other chapters of the document. Here, the user is about to select the input information indicationRequest.

978-1-4673-6905-3/15/$31.00 © 2015 IEEE

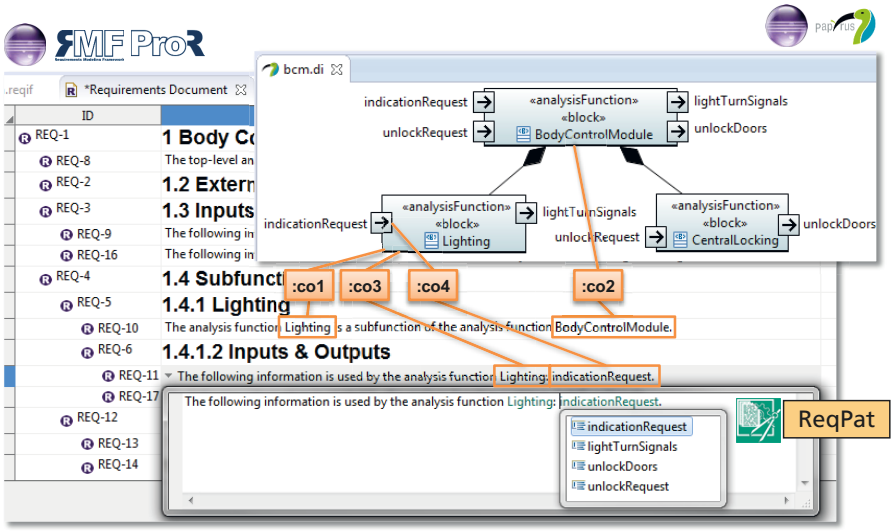

Fig. 1. Integration of ReqPat into Eclipse RMF/ProR and Papyrus ("Eclipse", "RMF", "ProR", and "Papyrus" are trademarks of Eclipse Foundation, Inc.)

B. Automatic Analyses for Requirements Validation

Exploiting the formality of our CNL, REQPAT supports automatic requirements validation. Syntax checks automatically highlight typos or an incorrect use of a requirement pattern directly while writing in the editor. More complex checks on the whole function hierarchy verify the fulfillment of well-formedness rules by the press of a button [6]. For instance, it is checked whether an input information like indicationRequest is consistently used by all analysis functions along a decomposition path from the root function down to a leaf function of the function hierarchy.

C. Transition to Models

Graphical models are often easier to understand than long texts, and models are heavily applied in the subsequent design phases. By means of bidirectional, synchronizing model transformations, REQPAT automatically derives an analysis model from pattern-based requirements or vice versa and keeps them consistent afterwards [4], [3], [1]. We use a function hierarchy analysis model in the UML/SysML modeling tool Eclipse Papyrus (cf. upper right of Fig. 1).

In Fig. 1, out of REQ-10 an analysis function block Lighting is derived that is a subfunction (i.e., composition association) of the overall function BodyControlModule. Additionally, the transformation adds the input port indicationRequest to Lighting based on the information in REQ-11.

During the transformation, REQPAT automatically establishes and maintains the traceability between CNL requirements and the analysis model by creating and updating correspondence objects (i.e., :co1 - :co4 in Fig. 1) between a part of a CNL requirement and an analysis model element.

IV. Conclusion and Future Work

NL requirements are easy to understand, but have the drawbacks of being prone to defects like contradictions and of being unamenable for automatic processing. Our CNL, called requirement patterns, aims at bridging this gap by providing an understandable and formalized notation at the same time. REQPAT exploits the formality to enable an efficient documentation of high-quality requirements by means of constructive and analytical tool support as well as the automatic transition to model-based development. In future work, we aim for a more flexible framework approach to rapidly integrate new requirement patterns to address other processes and domains.

Acknowledgment: This research is partially funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster "Intelligent Technical Systems OstWestfalenLippe" (it's OWL) and is managed by the Project Management Agency Karlsruhe (PTKA).

References

[1] M. Daun, M. Fockel, J. Holtmann, and B. Tenbergen. Goal-scenario-oriented requirements engineering for functional decomposition with bidirectional transformation to controlled natural language: Case study body control module. Technical Report ICB-Research Report No. 55, University of Duisburg-Essen, 2013.

[2] M. Fockel, P. Heidl, J. Holtmann, W. Horn, J. Höfflinger, H. Hönninger, J. Meyer, M. Meyer, and J. Schäuffele. Application and evaluation in the automotive domain. In *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*, chapter 12. Springer, 2012.

[3] M. Fockel and J. Holtmann. A requirements engineering methodology combining models and controlled natural language. In 4^{th} *Int. Model-Driven Requirements Engineering Workshop (MoDRE)*. IEEE, 2014.

[4] M. Fockel, J. Holtmann, and J. Meyer. Semi-automatic establishment and maintenance of valid traceability in automotive development processes. In 2^{nd} *Int. Workshop on Software Engineering for Embedded Systems (SEES)*. IEEE, 2012.

[5] J. Holtmann, J. Meyer, and M. Meyer. A seamless model-based development process for automotive systems. In 2^{nd} *Workshop ENVISION2020*, volume P-184 of *LNI*. Bonner Köllen Verlag, 2011.

[6] J. Holtmann, J. Meyer, and M. von Detten. Automatic validation and correction of formalized, textual requirements. In *ICST Workshop on Requirements and Validation, Verification & Testing (ReVVerT)*. IEEE, 2011.

[7] J. Leuser and D. Ott. Tackling semi-automatic trace recovery for large specifications. In *Requirements Engineering: Foundation for Software Quality (REFSQ)*, volume 6182 of *LNCS*. Springer, 2010.

[8] B. Regnell, R. B. Svensson, and K. Wnuk. Can we beat the complexity of very large-scale requirements engineering? In *Requirements Engineering: Foundation for Software Quality (REFSQ)*, volume 5025 of *LNCS*. Springer, 2008.

[9] E. Sikora, B. Tenbergen, and K. Pohl. Industry needs and research directions in requirements engineering for embedded systems. *Requirements Engineering*, 17(1), 2012.

[10] Wyner et al. On controlled natural languages: Properties and prospects. In *Controlled Natural Language*, volume 5972 of *LNCS*. Springer, 2010.