Developments in Requirements Engineering

Jeffrey C. Carver

THIS ISSUE’S COLUMN reports on a presentation and papers from the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 17). REFSQ aims to foster the creation of a strong European requirements-engineering community across industry and academia. Feedback and suggestions are welcome. In addition, if you try or adopt any of the practices included in the column, please send me and the paper authors a note about your experiences.

How to Support Natural-Language Requirements

In the opening keynote, “Analyzing Natural-Language Requirements: The Not-Too-Sexy and Yet Curiously Difficult Research That Industry Needs,” Lionel Briand overviewed various approaches for working with natural-language requirements. Given the prevalence of natural language for documenting requirements, a great need exists to understand the types of support requirements analysts need and how to scale that support to requirements documents containing hundreds or thousands of requirements.

Briand described several experiences from his research group’s extensive collaboration with industry. Examples included checking conformance with predefined sentence templates, extracting glossary terms and domain models, analyzing change impact, and deriving test cases. You can access the presentation slides at bit.ly/PD_July17_1.

Managing Collaboration

“Patterns of Collaboration Driven by Requirements in Agile Software Development Teams: Findings from a Multiple Case Study,” by Irum Inayat and her colleagues, discusses how distributed agile teams manage the collaboration necessary for requirements activities. The authors collected data from four mid-sized projects that used Scrum. Each project lasted three to five months, consisting of two to three iterations. Each project’s team members were distributed across two or more countries.

On the basis of the authors’ observations, surveys, and interviews, they identified patterns for requirements-driven collaboration, including these:

- Team members assumed different roles than assigned.
- Teams rarely needed to seek information from outside the team.
- Distance didn’t seem to negatively affect communication.
- Communication (even with remote teammates) was an important source of awareness.
These results reinforce the need for managers of distributed, agile teams to ensure they have a well-defined communication infrastructure and associated practices to guarantee that the team is working toward a common goal. You can access this paper at bit.ly/PD_July17_2.

**Semiautomatic Information Extraction**

“Semi-automatic Software Feature-Relevant Information Extraction from Natural Language User Manuals: An Approach and Practical Experience at Roche Diagnostics GmbH,” by Thomas Quirchmayr and his colleagues, addresses the problem that information about software features might not be centrally located but might be spread across several software engineering artifacts, including user manuals and issue trackers. In the authors’ semiautomated approach, team members first ensure that a system user manual is syntactically correct, and then extract a list of domain-specific terms from it. Using those terms, this approach then automatically extracts atomic feature-relevant information (a clause containing at most one subject and one predicate) from the manual.

Quirchmayr and his colleagues applied this approach to a 43-page excerpt from the user manual for an existing product. The approach achieved very high precision and recall compared to a fully manual analysis of the same excerpt. This result indicates that this approach might be useful for teams whose feature-relevant information is spread across multiple documents. You can access this paper at bit.ly/PD_July17_2.

**Detecting Ambiguity**

“On the Ability of Lightweight Checks to Detect Ambiguity in Requirements Documentation,” by Martin Wilmink and Christoph Bockisch, examines the ambiguity that occurs when requirements engineers use natural language to express requirements. Natural language’s inherent complexities, including grammar and context, make automated analysis difficult. Conversely, approaches that ignore these complexities are often inaccurate.

Wilmink and Bockisch’s lightweight approach analyzes natural-language requirements to annotate aspects that are strong (unambiguous) or weak (ambiguous). The solution works on Microsoft Word documents by using Word’s comment feature to annotate requirements. This implementation lets requirements analysts execute the automated detection tool frequently as they document requirements.

The authors applied this approach to two real requirements documents from KLM Engineering & Maintenance. These documents contained 293 requirements resulting in 454 annotations. Comparing these results with an analysis by experts showed a high level of precision and recall, especially in identifying weak phrases. These findings show promise for using lightweight tools to analyze natural-language requirements to help engineers identify and remove ambiguities before they lead to problems.

You can access this paper at bit.ly/PD_July17_3.

**Detecting Requirements Defects**

“Using NLP to Detect Requirements Defects: An Industrial Experience in the Railway Domain,” by Benedetta Rosadini and her colleagues, deals with the dearth of empirical evidence about whether rule-based natural-language-processing techniques can effectively identify defects in natural-language requirements documents. The authors first defined a set of typical defect classes. Then, they gave an expert from the subject company a tool to define defect detection patterns for those classes. Using these patterns, the tool automatically detected those types of defects in a given requirements document.

To evaluate this approach, Rosadini and her colleagues used a set of 1,866 requirements that had been annotated with defects. Domain experts also examined the analysis results to provide insight into the output. Although the results showed a high level of recall, several areas still needed improvement. In particular, company-specific information was necessary to improve the results’ accuracy. In other words, you shouldn’t expect to define an approach that will work well for all requirements documents, without providing tuning.
for the specific environment.

This research also identified practices that requirements engineers can employ to improve automated analysis. You can access this paper at bit.ly/PD_July17_5.

References


JEFFREY C. CARVER is an associate professor in the University of Alabama’s Department of Computer Science. Contact him at carver@cs.ua.edu.