Editor: **Jeffrey C. Carver**
University of Alabama
carver@cs.ua.edu

Extracting Requirements and Modeling Information and Controlling Risk

Jeffrey C. Carver, Silvia Abrahão, and Birgit Penzenstadler

THE "PRACTITIONERS' DIGEST" department in this issue of *IEEE Software* includes papers from the 2020 IEEE Conference on Requirements Engineering and the ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS 2020). Feedback or suggestions are welcome. In addition, if you try or adopt any of the practices included in this article, please send me and the authors of the paper(s) a note about your experiences.

Data-Driven Risk Management

"Data-Driven Risk Management for Requirements Engineering: An Automated Approach Based on Bayesian Networks" by Wiesweg and colleagues[1] addresses one of the most important requirements engineering activities: controlling risk. This activity is particularly difficult for junior team members and for those who are new to requirements engineering. The paper describes an approach that automatically predicts the most likely risks for a given requirements engineering project. The authors provided a web-based prototype to accompany their paper that will present a customized list of Top-N risks relevant for the project based upon user-supplied project characteristics. The tool builds the prediction using a series of Bayesian Networks that model relationships among requirements engineering problems, their causes, and their effects. These models can be used for postmortem analysis, to diagnose probable causes of suboptimal requirements engineering performance, or for predictive analysis, to identify issues that might arise. The tool learns the relationships based on the experiences of almost 500 practitioners with varying backgrounds from across the world. To assess the validity of the tool, the authors 1) compare the outputs of the tool with the assessment of a requirements engineering expert for a specific project, 2) examine how the expert assesses the usefulness of the tool and the style of the presentation, and 3) identify the contexts in which the expert would use the tool. Out of 10 reported problems, the tool agreed with the expert on the causes for seven of them. The expert rated the tool's usefulness to be high because the ranked predictions with probabilities would be useful input for group discussions. There are three primary uses for the tool: 1) in traditional projects, the tool provides input into discussion to improve the development process; 2) in agile teams, the tool provides input to improve the retrospective; and 3) in teams with inexperienced requirements engineers, the tool highlights low-hanging fruit for process improvement. The authors are working to make the tool generally available online with a questionnaire for anonymous user feedback. This paper appears in the 2020 IEEE Conference on Requirements Engineering. Access it at http://bit.ly/PD-2021-May-01.

Digital Object Identifier 10.1109/MS.2021.3056989
Date of current version: 16 April 2021

Human Value Analysis

"Continual Human Value Analysis in Software Development: A Goal Model Based Approach" by Harsha Perera and colleagues[2] addresses the challenges of looking at human values in software. The underlying concept is that software developers embed their values into the systems they create, either consciously or unconsciously. As Grady Booch[3] stated, every line of code has a moral and ethical implication. While it is easy to say that software rules the world, it is much harder to accept the related responsibility that comes with developing software systems. This difficulty occurs not only in systems used to determine actions in warfare or whether someone goes to jail, but also for social network algorithms that influence self-worth (especially of teenagers), or for news sites that influence how people treat others based on underlying biases. This paper examines the challenges of operationalizing human values in software, particularly, the lack of dedicated techniques to integrate human values, mechanisms to trace them, and metrics to measure them. The authors define a framework that includes four components: 1) identifying the values of stakeholders, 2) developing an initial feature model, 3) value-brainstorming iterations, and 4) measuring the values of the levels of satisfaction of stakeholders at each development level by tracking and assessing requirements and design decisions. The framework embodies social science concepts without requiring the users to fully understand them. The authors also developed a tabular representation of the reasoning chains from design choices to human values, which eases the applicability of goal models in an agile context. Finally, the paper describes a case study of an emergency alarm system for the elderly to illustrate the feasibility of their framework. This paper appears in the Proceedings of the International Conference on Requirements Engineering 2020. Access it at http://bit.ly/PD-2021-May-02.

While it is easy to say that software rules the world, it is much harder to accept the related responsibility that comes with developing software systems.

Extracting Acceptance Criteria From Natural Language

"Leveraging Natural-Language Requirements for Deriving Better Acceptance Criteria From Models" by Veizaga and colleagues[4] describes an approach to improving (semi-) formal requirements engineering processes by combining natural language requirement statements, UML models, and the automated generation of acceptance criteria. This new approach processes natural language statements to enrich the UML models to improve the quality of the generated acceptance criteria. The model-based derivation of system test-acceptance criteria is particularly useful for complex systems that have a large number of requirements or for systems that have frequently evolving requirements.

This approach automatically extracts information about acceptance criteria from the natural language requirements and helps modelers enrich their UML models with it. The authors define 13 information-extraction rules for natural language requirements, written in the Rimay language, along with corresponding recommendations on how to enrich the UML models based on those rules. Finally, they use an existing technique for deriving acceptance criteria from the model, which they had enriched with the information extracted from the natural language requirements. The process is fully automated except for the analysts' approval or rejection of the recommendations.

The authors conducted an industrial case study (from the financial domain) to evaluate the new approach. The results show that a group of five domain experts found most of the recommended enrichments relevant to the acceptance criteria but missing from the original UML model (constructed without the new approach). In addition, the experts could not pinpoint any additional information in the natural language requirements, relevant to the acceptance criteria, that the new approach did not identify. This paper appears in the Practice and Innovation Track of MODELS 2020. Access this paper at http://bit.ly/PD-2021-May-03.

Deployment of Edge Computing Applications

"Model-Based Fleet Deployment of Edge Computing Applications" by Song and colleagues[5] describes their joint research with a smart health-care application provider on a model-based approach for automatically assigning multiple software deployments to hundreds of edge computing devices. When using DevOps, software developers need to continuously add new features to software which may be running both on the cloud and at the edge. An edge computing application comprises tens to thousands of distributed and heterogeneous edge devices, which the authors refer to as a *device fleet*. A practical challenge is figuring out how to automatically deploy updated software after each DevOps iteration to the distributed edge devices while satisfying the device's constraints (that is, hardware capacity, user preferences, and network connection) and system requirements.

This paper describes an approach that leverages model-based engineering and constraint-solving techniques to automatically deploy multiple software variants on many edge computing devices in an accurate and efficient way. The approach was implemented, integrated into a DevOps toolchain, and tested in a real-world environment for a health-care company, using state-of-the-art Microsoft technologies (Azure Internet of Things Edge and Z3 solver). The results of the industrial case study show that the approach is able to generate correct deployment assignments, automate key DevOps activities, and increase development productivity. While the approach is specific to the deployment problem faced by the health-care company, the authors are generalizing it to be used across multiple edge application providers. This paper appears in the Practice and Innovation Track of MODELS 2020. Access this paper at http://bit.ly/PD-2021-May-04.

ABOUT THE AUTHORS

JEFFREY C. CARVER is a professor in the University of Alabama's Department of Computer Science, Tuscaloosa, Alabama, 35487, USA. Further information about him can be found at http://carver.cs.ua.edu. Contact him at carver@cs.ua.edu.

SILVIA ABRAHÃO is an associate professor at Universitat Politècnica de València, Valencia, 46022, Spain. Further information about her can be found at http://sabrahao.wixsite.com/dsic-upv. Contact her at sabrahao@dsic.upv.es.

BIRGIT PENZENSTADLER is an assistant professor at Chalmers University of Technology and the University of Gothenburg, Gothenburg, 412 96, Sweden. Further information about her can be found at https://www.chalmers.se/en/Staff/Pages/birgit.aspx. Contact her at birgitp@chalmers.se.

Live Modeling From Natural Language

"From Text to Visual BPMN Process Models: Design and Evaluation" by Ivanchikj and colleagues[6] describes BPMN Sketch Miner, an online tool for quickly creating Business Process Model and Notation (BPMN) models in real time based on notes taken in constrained natural language. The authors identify two contexts in which the tool could be useful: 1) to facilitate communication and knowledge sharing between domain experts and business analysts when performing requirements gathering and 2) to facilitate learning BPMN and process modeling in classrooms.

The main feature of BPMN Sketch Miner is its textual domain-specific language for entering a textual description of a process. The tool then transforms the textual description into a diagram (which is compliant with the BPMN visual syntax) while the user is typing/entering the information. The paper describes the design decisions behind the live modeling environment. First, the domain-specific language supports a large subset of BPMN elements while using a limited number of textual constructs that are easy to learn and remember. Second, because the monolithic structure of the textual syntax makes it difficult to represent control flow graph structures in BPMN models, the authors use a process-mining algorithm to reconstruct a process-control flow graph model from a set of sequential execution traces written in plain text. The resulting sketch can be exported for refinement in other standard-compliant BPMN editors.

To evaluate BPMN Sketch Miner, the authors surveyed industry analysts well-skilled in the BPMN language. They also conducted a user study with M.Sc. students who had no prior knowledge of BPMN. The results show that business analysts appreciated the usability of BPMN Sketch Miner and the expressiveness of the domain-specific language (in terms of supported BPMN constructs) and that BPMN Sketch Miner successfully helps BPMN beginners quickly and accurately model a nontrivial process. This paper appears in the Practice and Innovation Track of MODELS 2020. Access it at http://bit.ly/PD-2021-May-05. The BPMN Sketch Miner tool is available at http://bit.ly/PD-2021-May-06.

Model-Driven Test Schedule Generation

"Automating Test Schedule Generation With Domain-Specific Languages: A Configurable, Model-Driven Approach" by Anjorin and colleagues[7] addresses the optimization of test scheduling tasks, especially those requiring humans, which are often error-prone and time-consuming activities. The paper focuses on the optimization of a highly configurable test schedule process in a company that develops software and hardware solutions for mechatronic control systems. Their challenge is figuring out how to fully automate a test schedule strategy that can be understandable and configured by domain experts without prior knowledge of the solution domain.

To address this problem, this paper describes an approach that combines model-driven engineering and linear programming techniques. The new approach first generates valid candidate schedules with the help of a Triple Graph Grammars tool, which provides a domain-specific language to help test managers configure test schedules in a high-level and rule-based manner. The approach then uses an integer linear programming solver to identify the optimal schedules. This approach has been deployed and is being used at the company. To evaluate the approach, the authors examined the test schedule generator in terms of resource usage, task distribution by priority and risk, and performance. They also gathered qualitative and quantitative data from the use of the approach in a production setting, working closely with a test manager. The authors also compared the test schedules generated by the approach to manually created test schedules. Overall, the results show that the approach produces test schedules of good quality, reduces the effort required to create and maintain the schedules, and can be well-understood and used by the test manager. This paper appears in the Practice and Innovation Track of MODELS 2020. Access it at http://bit.ly/PD-2021-May-07.

References

1. F. Wiesweg, A. Vogelsang, and D. Mendez, "Data-driven risk management for requirements engineering: An automated approach based on Bayesian networks," in *Proc. 2020 IEEE 28th Int. Require. Eng. Conf. (RE)*, Zurich, Switzerland, pp. 125–135. doi: 10.1109/RE48521.202c0.00024.
2. H. Perera, G. Mussbacher, W. Hussain, R. Ara Shams, A. Nurwidyantoro, and J. Whittle, "Continual human value analysis in software development: A goal model based approach," in *Proc. 2020 IEEE 28th Int. Require. Eng. Conf. (RE)*, Zurich, Switzerland, pp. 192–203. doi: 10.1109/RE48521.2020.00030.
3. G. Booch, "The future of software engineering (SEIP Keynote)," in *Proc. 2015 IEEE/ACM 37th Int. Conf. Software Engineering*, Florence, Italy, vol. 2, pp. 3-3, May 16–24, 2015. doi: 10.1109/ICSE.2015.128.
4. A. Veizaga, M. Alferez, D. Torre, M. Sabetzadeh, L. Briand, and E. Pitskhelauri, "Leveraging natural-language requirements for deriving better acceptance criteria from models," in *Proc. 23rd ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. (MODELS '20)*, New York, Oct. 2020, pp. 218–228. doi: 10.1145/3365438.3410953.
5. H. Song, R. Dautov, N. Ferry, A. Solberg, and F. Fleury, "Model-based fleet deployment of edge computing applications," in *Proc. 23rd ACM/IEEE Int. Conf. Model Driven Eng. Languages Syst. (MODELS '20)*, New York, Oct. 2020, pp. 132–142. doi: 10.1145/3365438.3410951.
6. A. Ivanchikj, S. Serbout, and C. Pautasso, "From text to visual BPMN process models: Design and evaluation," in *Proc. 23rd ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. (MODELS '20)*, New York, Oct. 2020, pp. 229–239. doi: 10.1145/3365438.341090.
7. A. Anjorin, N. Weidmann, R. Oppermann, L. Fritsche, and A. Schürr, "Automating test schedule generation with domain-specific languages: A configurable, model-driven approach," in *Proc. 23rd ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. (MODELS '20)*, New York, Oct. 2020, pp. 320–331. doi: 10.1145/3365438.3410991.