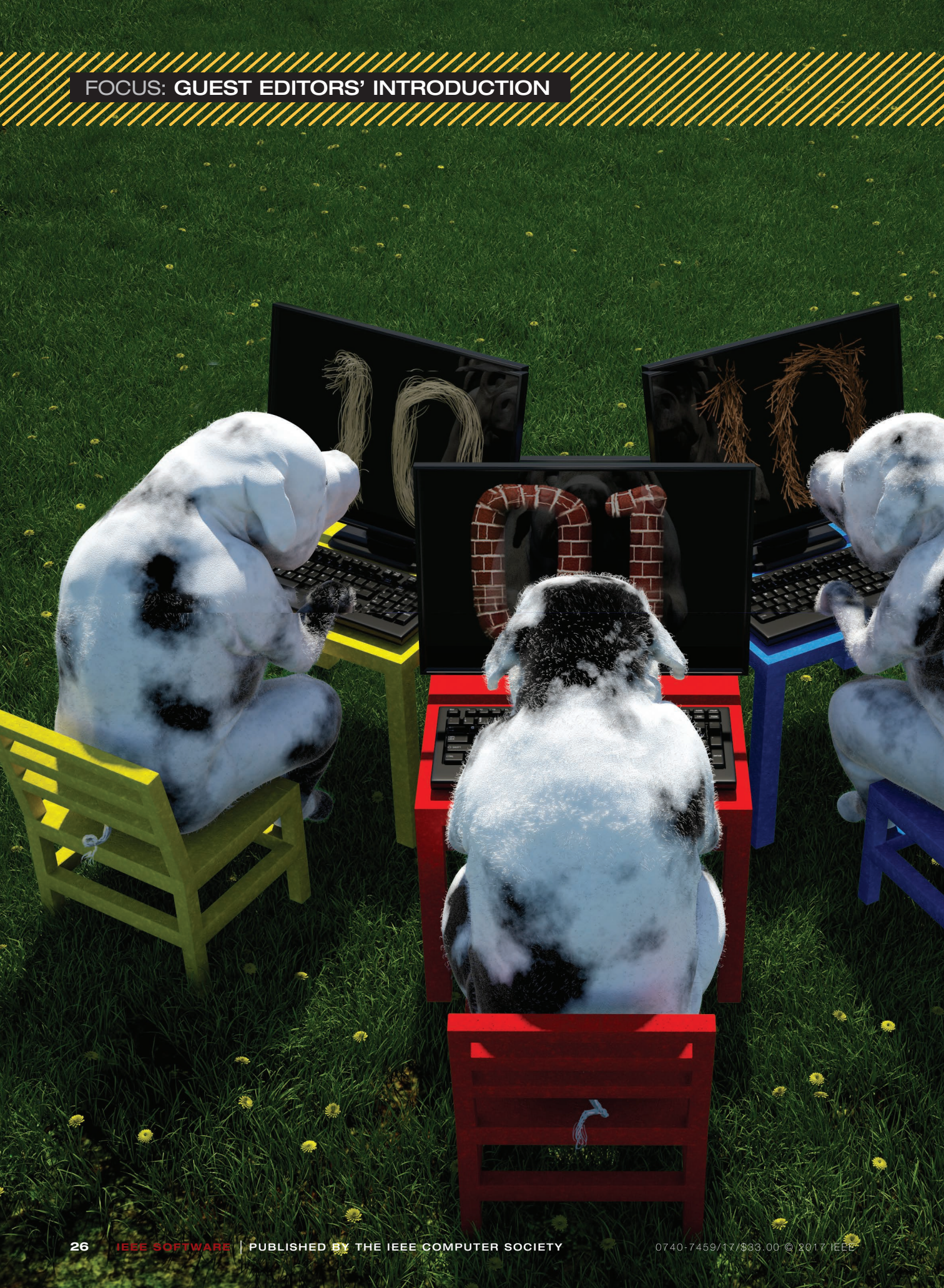


0740-7459/17/$33.00 © 2017 IEEE

Reliability Engineering

Xabier Larrucea, Tecnalia

Fabien Belmonte, Alstom Transport

Adam Welc, Huawei

Tao Xie, University of Illinois at Urbana-Champaign

THE EMERGENCE OF new technologies and architectures such as wearables, autonomous cars, the Internet of Things, and smart cities is providing more information about our surroundings and about us. In fact, our lives indirectly rely to some extent on these new technologies and advances, and reliability engineering is becoming more relevant than ever. Software is an active part of these devices' behaviors, and software engineering should play a crucial role.

For example, in the driverless-transportation domain, the development of safety-critical software (such as automatic train control systems) takes into account various hazard scenarios and the potential for system failure. Some hazard scenarios at the global system level involve a software-based system whose reliability contributes to safety (for example, an audiovisual service that communicates with passengers in emergency scenarios). The lack of specialized software for this area and economic pressure lead operators to use commercial-off-the-shelf software. A failure in these systems can aggravate an already bad situation. Avoiding such failures requires managing reliability throughout the production chain when developing commercial-off-the-shelf software, including components created by third parties.

Reliability engineering is neither a new nor hot topic per se. It dates back to reliability studies in the mid-20th century;[1] since then, various models have been defined and used (see the sidebar). Software engineering plays a key role from several viewpoints, but the main concern is that we're moving toward a more connected world,[2] including enterprises[3] and mobile devices.[4]

In the intersecting paths of software engineering and reliability, software reliability growth models are used for fault prevention, fault removal, fault tolerance, and fault or failure forecasting.[5] Researchers have investigated reliability in relation to software architectures, off-the-shelf components, testing, and metrics. They've also devised innovative applications of technologies such as artificial neural networks to predict software reliability.[6,7]

However, there have been few published experience reports and lessons learned regarding the

RELATED WORK IN SOFTWARE RELIABILITY ENGINEERING

John Musa and William Everett defined software reliability engineering as "the applied science of predicting, measuring, and managing the reliability of software-based systems to maximize customer satisfaction."[1] Nowadays, the number of software-based systems is steadily increasing, and we're surrounded by thousands of devices and systems whose operations rely on their appropriate functioning. This is true for not only safety-critical applications but also the increasing number of platforms related to the Internet of Things and smart cities.

Software engineering is a cornerstone for conferring reliability to such systems. These scenarios involve a myriad of aspects, such as debugging, early error detection, fast recovery, long-term support, dynamic and static analyses, and evolution.

Musa's reliability theory[2] was the precursor of a wide set of approaches such as the Musa–Okumoto basic-execution-time model.[3] In this context, there are two types of *software reliability growth models* (SRGMs).[4] *Black-box SRGMs* include

- the Jelinski–Moranda model,[5]
- the Goel–Okumoto model,[6]
- the Musa–Okumoto basic-execution-time model,
- the Musa–Okumoto logarithmic Poisson model,[3]
- the enhanced nonhomogeneous Poisson process model, and[7]
- the Littlewood–Verrall Bayesian model.[8]

White-box SRGMs include

- Saileshwar Krishnamurthi and Aditya Mathur's path-based model[9] and
- Swapna Gokhale and Kishor Trivedi's state-based model.[10]

References

1. J.D. Musa and W.W. Everett, "Software-Reliability Engineering: Technology for the 1990s," *IEEE Software*, vol. 7, no. 6, 1990, pp. 36–43.
2. J.D. Musa, "A Theory of Software Reliability and Its Application," *IEEE Trans. Software Eng.*, vol. SE-1, no. 3, 1975, pp. 312–327.
3. J.D. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, 1987.
4. A.K. Verma, S. Ajit, and D.R. Karanki, "Software Reliability," *Reliability and Safety Engineering*, Springer, 2010, pp. 193–228.
5. Z. Jelinski and P. Moranda, "Software Reliability Research," *Statistical Computer Performance Evaluation*, W. Freiberger, ed., Elsevier, 1972, pp. 465–484.
6. A.L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Trans. Reliability*, vol. R-28, no. 3, 1979, pp. 206–211.
7. S.S. Gokhale et al., "An Analytical Approach to Architecture-Based Software Reliability Prediction," *Proc. 1998 IEEE Int'l Computer Performance and Dependability Symp.* (IPDS 98), 1998, pp. 13–22.
8. B. Littlewood and J.L. Verrall, "A Bayesian Reliability Growth Model for Computer Software," *J. Royal Statistical Soc., Series C (Applied Statistics)*, vol. 22, no. 3, 1973, pp. 332–346.
9. S. Krishnamurthy and A.P. Mathur, "On the Estimation of Reliability of a Software System Using Reliabilities of Its Components," *Proc. 8th Int'l Symp. Software Reliability Eng.*, 1997, pp. 146–155.
10. S.S. Gokhale and K.S. Trivedi, "Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework," *IEEE Trans. Reliability*, vol. 55, no. 4, 2006, pp. 578–590.

practical long-term application of software reliability engineering. Such reports would inform the software engineering community about the state of the practice. In addition, software engineers could identify improvements in this area and even identify potential approaches to their application domains.

In This Issue

The three articles in this special issue illustrate current trends in this domain.

In "Automated System-Level Regression Test Prioritization in a Nutshell," Per Strandberg and his colleagues report their experience using an automated tool to determine the effective ordering of regression test cases. They evaluated their tool in real-world settings and identified interesting challenges.

In "Safety Analysis of Safety-Critical Systems Using State-Space Models," Vinay Kumar and his colleagues explain how they use UML statechart diagrams and Petri nets

to represent unobservable dynamic components. They validated their technique on a nuclear power plant's emergency core cooling system.

In "Requirements Engineering for Safety-Critical Systems: Overview and Challenges," Luiz Martins and Tony Gorschek discuss a systematic literature review of the most-cited approaches for capturing and handling safety requirements. They discovered that practitioners largely preferred traditional approaches such as fault tree analysis and failure mode and effects analysis.

Reliability isn't just a desirable characteristic of software or non-software-based systems. It's a property or an ability of all systems, especially when our lives rely on them. Engineers should consider reliability a cornerstone of their development process. This theme issue presents three articles discussing different development phases: requirements, design, and testing. As engineers or managers, our responsibility is to apply or facilitate the means to ensure that reliability is a critical feature of the final product.

References

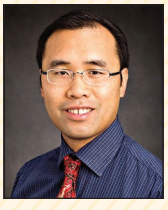
1. E.D. Cook, "Reliability in Industrial Electronic Equipment," *Trans. Am. Inst. Electrical Engineers, Part 1: Communication and Electronics*, vol. 72, no. 4, 1953, pp. 351–360.
2. M. Barth, "Living in a Connected World," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, 2014, pp. 4–7.
3. "Engineering Management Great Books List," *IEEE Eng. Management Rev.*, vol. 31, no. 4, 2003, p. 136.
4. L. Gorlenko and R. Merrick, "No Wires Attached: Usability Challenges in the Connected Mobile World," *IBM Systems J.*, vol. 42, no. 4, 2003, pp. 639–651.
5. A.K. Verma, S. Ajit, and D.R. Karanki, "Software Reliability," *Reliability and Safety Engineering*, Springer, 2010, pp. 193–228.
6. M.R. Lyu, "Software Reliability Engineering: A Roadmap," *Proc. 2007 Future of Software Eng.* (FOSE 07), 2007, pp. 153–170.
7. M.K. Saley and S. Sreedharan, "A Survey of Software Reliability Growth Models Using Non-parametric Methods," *Proc. 2014 IEEE Int'l Conf. Computational Intelligence and Computing Research* (ICCIC 14), 2014.

ABOUT THE AUTHORS

XABIER LARRUCEA is a senior project leader and research scientist at Tecnalia and a part-time lecturer at the University of the Basque Country. His research interests include safety-critical software systems, software quality assurance, software process improvement, empirical software engineering, and metamodeling technology strategy. Larrucea received a PhD in software engineering from Universidad del País Vasco. Contact him at xabier.larrucea@tecnalia.com.

FABIEN BELMONTE is the mainline projects safety assurance manager at Alstom Transport. His research interests include using model-driven engineering to provide continuous improvement of industrial practices, particularly railway safety. Belmonte received a PhD in information sciences and techniques from Université de Technologie de Compiègne. Contact him at fabien.belmonte@transport.alstom.com.

ADAM WELC is a principal architect at the Huawei America Research Center, where he optimizes programming-language implementations for the cloud. Welc received a PhD in computer science from Purdue University. Contact him at adamwwelc@gmail.com.



TAO XIE is an associate professor and a Willett Faculty Scholar in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His research interests include software testing, program analysis, software analytics, software security, and educational software engineering. Xie received a PhD in computer science from the University of Washington. Contact him at taoxie@illinois.edu.