Statistical Science
2016, Vol. 31, No. 4, 541–544
DOI: 10.1214/16-STS583
Main article DOI: 10.1214/16-STS565
© Institute of Mathematical Statistics, 2016

On Software and System Reliability Growth and Testing

Frank P. A. Coolen

Abstract. Singpurwalla presents an insightful proposal on foundations of reliability [*Statist. Sci.* **31** (2016) 521–540], suggesting to consider reliability not as a probability but as a propensity, in particular as the unobservable parameter in De Finetti's famous representation theorem. One specific issue considered is reliability growth, with example scenario the performance of software as it evolves over time. We briefly discuss some related aspects, mainly based on applied research on statistical methods to support software testing and insights from our research on system reliability.

Key words and phrases: Reliability growth, software testing, system reliability.

1. INTRODUCTION

Singpurwalla [8] presents an interesting contribution to the foundations of reliability theory, proposing that reliability is not a probability but a propensity, also referred to as a chance. The main issue is that reliability is considered to be "an objective, albeit unobservable, physical quantity" [8]. The essential proposal by Singpurwalla is that this reliability is the parameter appearing in De Finetti's famous representation theorem; see Theorem 1 in [8]. This theorem involves an infinite sequence of exchangeable observable binary random quantities and plays a central role in (subjective) Bayesian statistics. The proposal to consider reliability as a propensity suggests that it is a property of the system or unit under consideration. As reliability of a system typically changes over time, this should be reflected through a time-evolving propensity, for which Singpurwalla presents a competing risk model.

In Section 2, we discuss some aspects of reliability of software systems based mainly on experience in supporting software testers. While Singpurwalla's proposal is of theoretical interest, our experiences point to possible difficulties, or at least major challenges, for its implementation. In Section 3, we briefly discuss some further related issues on general aspects of system reliability including competing risk models.

Frank P. A. Coolen is Professor of Statistics, Department of Mathematical Sciences, Durham University, Durham, DH1 3LE, United Kingdom (e-mail: frank.coolen@durham.ac.uk).

2. SOFTWARE TESTING AND RELIABILITY

Singpurwalla's [8] proposal to consider reliability as a propensity is linked to a possible interpretation of the parameter appearing through De Finetti's representation theorem. This is an interesting proposal, yet its practical relevance for modelling reliability requires careful consideration. Typically, a system may have to perform many different tasks, possibly under a wide range of circumstances, and these may not all be carefully described; this is certainly the case for software systems. About two decades ago, we started considering reliability of software systems in collaboration with an industrial partner, and soon it became clear that the main challenges, apart from the development of the software, were in testing it. From a background in statistics, we were surprised at the lack of suitable methods in the literature with regard to careful modelling of the tasks software testers had, indeed it seemed that many mathematical and statistical methods for software reliability, presented in the literature, were of little practical use. One approach that had received much attention in the literature was so-called partition testing; one defines a partition of the input space, so of all inputs the software system has to deal with, in such a way that all inputs in one set of the partition are "identical". This was surprisingly often interpreted in a "one-test-tests-all" sense, but also the more general exchangeability setting where each input in one set of the partition has the same probability of leading to a fault, was considered. Generally, it seemed to be assumed that the software's success on dealing with inputs in different sets of the partition was completely independent, which led us to present a quite basic method to overcome this unrealistic assumption [4]. In order to guide software testing and assess the reliability of most substantial practical software systems, it is crucial to model the inputs it has to deal with. One cannot assume that the system's success or failure to deal correctly with an input is exchangeable over all inputs, although we have encountered this assumption frequently in the literature. A clear consequence of that assumption would be that it is irrelevant which specific input one tests, reducing the test effort mostly to deciding on when enough tests have been performed. Software testers know that their job is more complicated, and mainly consists of deciding which specific inputs to use for testing a system, and ideally also in which specific order.

In a long-running project with an industrial partner, we developed a Bayesian graphical model (BGM) approach for support of testing a substantial software system close to release. The core method, with example of its application, has been presented by Wooff et al. [9], some further aspects of the project were discussed by Coolen et al. [5]. A discussion of challenges for statistical methods related to software testing was presented by Coolen [1]. The actual nature of the task was so-called integration testing, where new functionality had been added to existing systems and some minor corrections had also been applied. These were mainly database activities, and communication between databases. For example, some functions needed to be performed on numbers of different length, where the testers distinguished between short and long numbers. They also expected that some functionality might be affected by whether the starting digit of a number was a zero or not. Functionalities considered included adding or deleting a customer to one or more databases, and also changing a customer number, which was expected to be done by a combination of adding and deleting the relevant numbers with appropriate transfer of the information related to the number. The software system consisted of different subsystems, some programmed in house and some bought in, the latter being mainly "black-box". On existing functionality there was quite some experience, including knowledge of what had gone wrong before and which actions had been taken to resolve the problems. The newly added functionality had been tested on its own but not as part of the fully integrated system. After several discussions with the software testers, and detailed study of the literature, we found that adequate statistical methodology was available for the required support, which was mainly on developing test suites to be used under great time pressure, but we did not find it being used in the literature on software engineering. Questions of interest included not only which inputs to test but, crucially, also the sequencing of testing, where re-testing in case failures appeared needed to be taken into account, and possible also guidance on further tests in such cases to get a better idea on the location of the fault in the software system which caused the failure. This re-testing was needed to check if a previous failure on an input had been resolved, a corresponding major problem for the testers was to assess if earlier tests would also require re-testing. An even more challenging question for test design was with regard to checking the model and (partial) exchangeability assumptions. In our specific applications, there was very little time to add such tests, but we noticed from previous testing that even in extreme cases where testers believed that testing one input would reveal the software's reliability on the specific functionality for all inputs in the same set in the partition of the input space, they would sometimes test two such inputs reflecting that they were not entirely sure about the one-test-tests-all assumption. In an ideal scenario, one would want to create a test approach where checking of the model assumptions, on which the test design is based, is implicit, and in case of any test results which lead to doubt about the assumptions one would want a level of robustness which would allow information from previous tests still to be useful. We have not come across work addressing this challenging aspect of software test design in the literature.

We were quite surprised that there was no detailed list reporting which specific functioning would be tested when a specific input was used, creating this took a considerable effort but the testers did acknowledge that it was a useful exercise. The BGMs we created were on functionality, where for each function of the software (partial) exchangeability beliefs of the testers were represented. For example, a node "problems due to number length" had two child nodes, "problems due to short length" and "problems due to long length", and similarly for other characteristics such as starting digit (zero or nonzero). Such child nodes each had a further parent node, to represent a problem specifically due to the stated length in the child node, so not a common problem due to length. At the bottom of these BGMs were the observation nodes, so the test results for specific combinations of the aspects considered in the combined parent nodes, for example, test results for a long number starting with zero in the BGM for the add function. The standard Bayesian methodology was used for the inverse inference to update the BGM given test information, for quite many functionalities combined, and this provided a tool for support of test design.

The main point of summarizing this earlier work here is how the proposition by Singpurwalla [8], to consider reliability to be a propensity, would fit with such practical work. It is clear that the realistic software systems we considered did not have a single reliability propensity such that all possible observations in the testing process were exchangeable. Indeed the main task was to model the partial exchangeability which we did using the BGMs for each functionality. One could argue that, for the whole system, there are many such propensities, one each for every node at the very bottom of a BGM (which indeed has observable test results as child nodes); the main issue is the modelling of the interconnections of these propensities. Then, when a failure is observed and action is taken to remove the fault, which may be successful or not and may even introduce other faults, the information about the propensities changes but also the exchangeability assumptions themselves may change: the observation of a specific failure could lead to more detailed specification of an input partition, for example, testers may become aware of a feature affecting numbers starting with one or long numbers ending with zero. This is just mentioned to emphasize the dynamic nature of such testing processes, which are at the heart of software reliability growth. It will be interesting to see whether or not the new foundational approach proposed by Singpurwalla will open new insights and methods for modelling for support of software testing.

3. SYSTEM RELIABILITY

In recent years, we have been considering many aspects of system reliability and proposed several new (statistical) methods in the literature. For example, in order to enable reliability quantification for large systems consisting of different component types, where at least some of the components have exchangeable failure times, we proposed the survival signature [2], which enables substantially larger systems to be considered than previously available methods. The survival signature generalizes Samaniego's system signature [7], which has become a popular tool in the literature on mathematical methods for reliability, to systems with multiple types of components, which most practical systems are and which includes networks. In presenting such work and discussing it with practitioners, we repeatedly had to consider what "system reliability" really is, due to a variety of issues not really addressed by existing methods in the literature. This has led us to two propositions [3]. First, to mainly focus on one or more future tasks, so to consider the ability of the system to deal with these correctly, instead of defining reliability as an underlying property of the system. This can also take into account a range of such tasks, and even include the possibility to take unknown or undefined tasks into account. Second, we proposed to generalize the classical concept of a deterministic structure function, which models whether or not the system functions given the states (functioning or not) of all its components, by allowing it to be probabilistic. So, for specific states of the system components, one models a probability that the system will function for its next task. This has several advantages, for example, it can take into account if successful functioning depends on external aspects and it can also be used if only part of the full system is being modelled, for example, to focus on major components in a system. Singpurwalla [8] proposes further to model evolving propensities via competing risks methods. This is an interesting view which corresponds quite well with our experiences with support of practical software testing as discussed above. We would like to point out that competing risks methods bring many challenges for statistics, in particular where some of the risks may be unobserved or unknown, or where faults or failure modes have been removed or re-defined; see Coolen-Maturi and Coolen [6] for further discussion and some proposals for dealing with such cases. It will be interesting to see if these issues and proposals can be linked to Singpurwalla's view of reliability as a propensity, and if there is a benefit for the modelling and related statistical inferences resulting from changing the foundations of reliability along Singpurwalla's proposal.

REFERENCES

[1] COOLEN, F. P. A. (2012). On some statistical aspects of software testing and reliability. In *Complex Systems and Dependability* (W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak and J. Kacprzyk, eds.) 103–113. Springer, Berlin.

[2] COOLEN, F. P. A. and COOLEN-MATURI, T. (2012). On generalizing the signature to systems with multiple types of components. In *Complex Systems and Dependability* (W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak and J. Kacprzyk, eds.) 115–130. Springer, Berlin.

[3] COOLEN, F. P. A. and COOLEN-MATURI, T. (2016). The structure function for system reliability as predictive (imprecise) probability. *Reliab. Eng. Syst. Saf.* **154** 180–187.

[4] COOLEN, F. P. A., GOLDSTEIN, M. and MUNRO, M. (2001). Generalized partition testing via Bayes linear methods. *Inf. Softw. Technol.* **43** 783–793.

[5] COOLEN, F. P. A., GOLDSTEIN, M. and WOOFF, D. A. (2007). Using Bayesian statistics to support testing of software systems. *J. Risk Reliab.* **221** 85–93.

[6] COOLEN-MATURI, T. and COOLEN, F. P. A. (2011). Unobserved, re-defined, unknown or removed failure modes in competing risks. *J. Risk Reliab.* **225** 461–474.

[7] SAMANIEGO, F. J. (2007). *System Signatures and Their Applications in Engineering Reliability*. *International Series in Operations Research & Management Science* **110**. Springer, New York. MR2380178

[8] SINGPURWALLA, N. D. (2016). Filtering and tracking survival propensity (reconsidering the foundations of reliability). *Statist. Sci.* **31** 521–540.

[9] WOOFF, D. A., GOLDSTEIN, M. and COOLEN, F. P. A. (2002). Bayesian graphical models for software testing. *IEEE Trans. Softw. Eng.* **28** 510–525.