Formal Specifications and Software Testing, a Fruitful Convergence

Marie-Claude Gaudel(✉)

LRI, Université Paris-Sud, Orsay, France
marieclaude.gaudel@gmail.com

Abstract. This paper gives some account of the evolution of ideas and the main advances in the domain of software testing based on formal specifications and reports some personal anecdotes on my activity in this field. Going back to the seventies, being slightly caricatural, software testing was perceived, on the one hand, by its actors as an empirical activity that had nothing to gain from formal methods, on the other hand, by the advocates of these methods as doomed to disappear based on the belief that in the long run programs will be correct by construction. Currently, these two communities haven't yet reached a complete consensus. But fortunately there have been some significant moves from both sides and various success stories that allow saying that there is a fruitful convergence toward testing methods based on formal specifications.

Keywords: Formal methods · Software testing · History

1 Introduction

Software testing based on formal specifications has a slightly troubled history. In the seventies, most actors of both fields considered that they had nothing to bring to each other. Even worse, some influential scientists from both sides emitted mutual doubts on the pertinence of these fields: from the side of the software-testing research community, one can cite De Millo et al. [13]; and from the side on the formal-approaches-to-software-engineering community, one can cite the famous Dijkstra curse against testing [15], which definitely deserves to be quoted:

> Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence. The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness

Another Dijkstra's quotation, more directly relevant to the topic of this paper, is:

> A common approach to get a program correct is . . . by subjecting the program to numerous test cases. From the failures around us we can derive ample evidence that this approach is inadequate. To remedy the situation

E. Sekerinski et al. (Eds.): FM 2019 Workshops, LNCS 12233, pp. 87–94, 2020.
https://doi.org/10.1007/978-3-030-54997-8_5

> it has been suggested that what we really need are "automatic test case generators" by which the pieces of program to be validated can be exercised still more extensively. But will this really help? I don't think so. (EWD303, year unknown[1])

Fifty years later, the idea of software testing methods based on formal specifications is accepted as respectable, among the numerous existing approaches to software testing [12], and some powerful tools exist. Most formal specification methods come with some notion of test derivation, submission, verdict and there is a number of success stories. Moreover, the complementarity of tests and proofs in software validation and verification is the subject of active research activities, attested by numerous publications and the fact that an international conference "Tests and Proofs" has taken place annually since 2007. There is an excellent survey by Hierons et al., titled "Using Formal Specifications to Support Testing", published in 2009 in ACM Computing Surveys [22].

Therefore, it is of interest to look back at this evolution towards convergence of two research fields that were originally so distant.

2 Software Engineering, Formal Methods, and Testing in the Seventies and the Beginning of the Eighties

A possible subtitle of this section could be: Why formal? Why testing?.

In the seventies, formal approaches to software development, validation, and verification were not considered as credible by the majority of software engineers and software testers. Symmetrically as mentioned above, many supporters of formal approaches considered that software testing was an ineffective activity doomed to disappearance after the generalisation of formal methods.

My interest in testing based on formal specifications arose during my Ph.D. whose topic was compiler generation based on formal semantics of the source and target languages. I needed examples of source programs to test the generated compilers and my attention was drawn to a work by Houssais on the verification of an Algol 68 implementation [24] and on a little-known subsequent research report of the Université de Rennes 1, titled "Un générateur de tests commandé par les grammaires". Since testing compilers was not the main focus of my Ph.D., I didn't spend much time on fully exploiting Houssais ideas. However, I used them as guidelines, which turned out to be useful.

I had noticed some similarities between programming language definitions and algebraic data types. In both cases, there is a syntax (for algebraic data types, the signature), and some constraints (contextual rules for programming languages, axioms for algebraic data types). This led me to propose as Ph.D. subject to Luc Bougé the generation of test from algebraic specifications. The thesis was defended in 1982 [3], followed by four other ones at the end of the

[1] This note can be seen at https://www.cs.utexas.edu/users/EWD/ where it has no date. But one can bracket it between two dated EWDs: EWD 292, 31 Aug 1970, and EWD 306, 16th March 1971 (thanks to Jeremy Gibbons for the hint).

eighties. It was the origin of successful developments, including industrial applications, that are summarised in [20]. At that time, it was not easy to publish on this subject. Dijkstra's curse was very present in the mind of researchers in formal methods. Our first publication in an international conference was in 1985 [4]. However, these ideas turned out to be a rather good selling point of formal methods to industry: in October 1981, I was hired to create a research group on the use of such methods for software engineering in the research laboratory of the Compagnie Générale d'Electricité.[2]

At the same time, i.e. 1981–1983, several research works were led on exactly the same topic, namely testing based on algebraic abstract data types: Gannon, McMullin, and Hamlet in the U.S. [19], and Jalote in India [26]. A lot of technical questions were still open, but the fundamental ideas were well stated. Among the open issues at that time, there were:

- the determination of equality between two test results, the difficulty coming from the difference of levels of abstraction between the specification and the implementation under test[3].
- the occurrence of partial operations in the specification, with two causes of partiality: either the result is mathematically undefined, or it is not specified, the specification allowing some freedom to the implementation.
- the possibility of non-determinism in the specification or in the implementation.

Note that these issues are not due to the use of formal specifications. They arise in any testing method that is specification-based. For some hints on the way they have been formally treated later on, one can see [20] and [22].

Meanwhile, another research community had been very active in the domain with very strong motivations: the researchers in telecommunication protocols.

3 The Area of Telecommunication Protocols at the Beginning of the Eighties

Communicating systems use well-defined formats for exchanging messages. Each message has an exact role and corresponds to a range of possible answers.

[2] During my stay there (1981–1983) I interacted with a group of engineers who developed the software of a telephone switching system. I was impressed by their professionalism. They motivated me to pursue this line of research.

A (not so funny) anecdote is that some years later, being back as a professor in a university, I visited the same place with a Ph.D. student, searching for challenging case studies. Mood and people had changed and the head of the group explained that their goal was to be first on the market and their development strategy was "quick an dirty". To that the Ph.D. student replied that "dirty we can do, but quick I am not sure". The meeting was unproductive...

[3] This is similar to the issue of lifting computational types and values to the logical level in Hoare's logic.

The specification of the protocol is independent of how it may be implemented and exploited by the telecommunication operators. Besides, testing can be based on the specifications only (i.e. it's black-box testing), as manufacturers generally don't disclose implementation details.

Since communication protocols must be agreed upon by various entities such as international agencies, operators, or developers, technical standards, formal or not, have been developed very early for their definitions (see for instance [9]) as well as for conformance testing methodologies [25]. Certification is performed via standard abstract test sets.

In such a context, research projects and publications on specification-based testing of such protocols had flourished (see [33] among the early ones). Note that this paper was published in 1984, at the 2nd IFIP International Workshop on Protocol Specification, Verification and Testing (IWPTS), which has occurred yearly since, modulo a few changes of names[4]. Since 2007, under the name TESTCOM this conference merged with the FATES workshop[5] and has gathered together researchers working on specification-based testing, independently of the kind of considered software: communication protocols or others.

The characteristics of the approaches developed for communication protocols was: the formal specification languages, such as SDL, ESTELLE, LOTOS, were standardised, with some semantics based on Finite State Machines or Labelled Transition Systems that are well-suited for the description of non-terminating processes; the notions of conformance, abstract test specifications, certification, were standardised as well.

In contrast, the approaches developed in the community of formal software engineering, or those that were on the verge to be developed, were based on specification languages designed by research groups, with notations inspired from logic, and semantics based on axioms satisfaction (algebraic data types) or predicate transformers (Z, VDM).

4 What Happened in the Nineties and Later

The end of the eighties and the beginning of the nineties were a turning point for the topic of testing based on formal specifications. Within five years a lot of well-founded testing methods were established and validated for the main formal specification methods. It's impossible to cite all of them. Let us just mention three sets of works[6].

Tests derivation from formal specifications with semantics based on labelled transition systems, such as LOTOS or SDL, started to be abundantly studied at

[4] IWPTS: 1983–1996; IWTCS: 1997–1999, TESTCOM: 2000–2009, ICTSS: 2010-now.

[5] This series of workshops (Formal Approaches to Testing of Software) took place in 2001–2007.

[6] A notable omission here is the corpus of research on testing based on FSM (Finite State Machines), which has been considerably influential since the sixties both in hardware and in software testing. For an excellent survey with some historical indications see [28].

the end of the eighties [5,32,38]. Later, in 1996, Tretmans introduced the *ioco* conformance relation [36], which turned out to be quite pertinent for testing those systems that are input-enabled, i.e. they are always ready to accept some input. It has been at the origin of several tools such as TGV [27], TorX [10] and more recently TorXakis [37].

For the VDM specification method, Dick and Faivre presented in 1993 a testing method [14] at the Formal Method Europe Conference (FME'93). It must be observed that this work was performed in an industrial context, namely Bull Information Systems in the UK and its Corporate Research Centre in France, and that it has strongly influenced the subsequent researches on testing based on model-based formal specifications. At the same time, Stocks and Carrington proposed Test Templates, a test method based on Z specifications [35], which was published at the International Conference on Software Engineering (ICSE 15). Both software formalists and software engineers recognised the interest of testing methods based on model-based formal specifications.

In my group, we pursued the work on testing based on algebraic specification. In 1991, with Bernot and Marre we published a case study on the test of binary search trees [2], and in 1993 Dauchy and Marre successfully applied the method and the tool to critical parts of the software of an automatic subway [11]. Since then, our approach has been generalised to other formal methods: full LOTOS [21], Lustre and Esterel [30] with applications to nuclear control systems certification, CSP [7] and *Circus* [8].

One observes that since the mid-nineties, test derivation from formal specifications has become a popular research topic, well accepted both in conferences and journals devoted to formal methods and in conferences and journals devoted to software engineering and testing. New formal specification methods are almost systematically enriched by some test derivation methods: for instance, it was the case for the B method [29,34], and ASMl, the Abstract State Machine Language [1] and its companion testing environment SPEC EXPLORER [39] developed at Microsoft Research.

Moreover, the cultural gap between researchers from the telecommunication world and from the formal software engineering community is much less pronounced. Most theories, methods, and tools are of common use or unified, even if the telecommunication universe remains much more dependent on standards.

5 Conclusion

Despite Dijkstra's curse, the convergence of formal methods and software testing has happened. Definitely, one of the antidotes to Dijksta's curse has been an invited conference and paper by Tony Hoare at the FME Symposium in 1996 titled "How Did Software Get So Reliable Without Proof?" [23]. There, he underlined the effectiveness of software engineering techniques, among which testing, and the fact that formal methods

> provide a conceptual framework and basic understanding to promote the best of current practice, and point directions for future improvement.

As mentioned above, nowadays, most formal specification methods come with some notion of test derivation, submission, verdict. There is a number of significant success stories. A majority of them are in link with academia, but a number of them took place in industry. Testing based on formal specification is now one of the recognised resources in the arsenal of software testing methods.

At present, theorem provers and model checkers make use of tests [16,31], and testing tools make use of theorem provers [6,17] and model checkers [18]. The Test and Proof (TAP) series of conferences and the Verified Software: Theories, Tools, Experiments (VSTTE) series of workshops are well established. They attract both communities, provide a forum for meetings and talks, ensuring the development of research activities integrating formal methods and testing.

Acknowledgment. I am grateful to Burkhart Wolff and the members of the LRI Test Club who gave me the idea to talk and write on this topic.

References

1. Barnett, M., Grieskamp, W., Nachmanson, L., Schulte, W., Tillmann, N., Veanes, M.: Towards a tool environment for model-based testing with AsmL. In: Petrenko, A., Ulrich, A. (eds.) FATES 2003. LNCS, vol. 2931, pp. 252–266. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24617-6_18
2. Bernot, G., Gaudel, M.C., Marre, B.: Software testing based on formal specifications: a theory and a tool. Softw. Eng. J. **6**(6), 387–405 (1991)
3. Bougé, L.: Modeling the notion of program testing; application to test set generation. Theses, Université Pierre et Marie Curie - Paris VI, October 1982. https://tel.archives-ouvertes.fr/tel-00416558
4. Bougé, L., Choquet, N., Fribourg, L., Gaudel, M.C.: Application of PROLOG to test sets generation from algebraic specifications. In: Ehrig, H., Floyd, C., Nivat, M., Thatcher, J. (eds.) TAPSOFT 1985. LNCS, vol. 186, pp. 261–275. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-15199-0_17
5. Brinksma, E.: A theory for the derivation of tests. In: Proceedings of 8th International Conference on Protocol Specification, Testing and Verification, pp. 63–74. North-Holland (1988)
6. Brucker, A.D., Wolff, B.: On theorem prover-based testing. Formal Asp. Comput. **25**(5), 683–721 (2013)
7. Cavalcanti, A., Gaudel, M.-C.: Testing for refinement in CSP. In: Butler, M., Hinchey, M.G., Larrondo-Petrie, M.M. (eds.) ICFEM 2007. LNCS, vol. 4789, pp. 151–170. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76650-6_10
8. Cavalcanti, A., Gaudel, M.C.: Testing for refinement in *Circus*. Acta Inf. **48**(2), 97–147 (2011)
9. CCITT: Functional specification and description language (SDL), Recommendation Z.100–Z.104 (1984)
10. Chaudron, M.R.V., Tretmans, J., Wijbrans, K.: Lessons from the application of formal methods to the design of a storm surge barrier control system. In: Wing, J.M., Woodcock, J., Davies, J. (eds.) FM 1999. LNCS, vol. 1709, pp. 1511–1526. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48118-4_30

11. Dauchy, P., Gaudel, M.C., Marre, B.: Using algebraic specifications in software testing: a case study on the software of an automatic subway. J. Syst. Softw. **21**(3), 229–244 (1993)
12. DeMillo, R.A.: Software testing. In: Encyclopedia of Computer Science, pp. 1645–1649. John Wiley and Sons Ltd., GBR (2003)
13. DeMillo, R.A., Upton, R.J., Perlis, A.J.: Social processes and proofs of theorems and programs. Math. Intell. **3**(1), 31–40 (1980). https://doi.org/10.1007/BF03023394
14. Dick, J., Faivre, A.: Automating the generation and sequencing of test cases from model-based specifications. In: Woodcock, J.C.P., Larsen, P.G. (eds.) FME 1993. LNCS, vol. 670, pp. 268–284. Springer, Heidelberg (1993). https://doi.org/10.1007/BFb0024651
15. Dijkstra, E.W.: The humble programmer. Commun. ACM **15**(10), 859–866 (1972)
16. Dubois, C., Giorgetti, A.: Tests and proofs for custom data generators. Formal Aspects Comput. **30**(6), 659–684 (2018). https://doi.org/10.1007/s00165-018-0459-1
17. Feliachi, A., Gaudel, M.-C., Wenzel, M., Wolff, B.: The *Circus* testing theory revisited in Isabelle/HOL. In: Groves, L., Sun, J. (eds.) ICFEM 2013. LNCS, vol. 8144, pp. 131–147. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41202-8_10
18. Fraser, G., Wotawa, F., Ammann, P.: Issues in using model checkers for test case generation. J. Syst. Softw. **82**(9), 1403–1418 (2009)
19. Gannon, J.D., McMullin, P.R., Hamlet, R.G.: Data-abstraction implementation, specification, and testing. ACM Trans. Program. Lang. Syst. **3**(3), 211–223 (1981)
20. Gaudel, M.-C., Le Gall, P.: Testing data types implementations from algebraic specifications. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) Formal Methods and Testing. LNCS, vol. 4949, pp. 209–239. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78917-8_7
21. Gaudel, M.C., James, P.R.: Testing algebraic data types and processes: a unifying theory. Formal Asp. Comput. **10**(5–6), 436–451 (1998)
22. Hierons, R.M., et al.: Using formal specifications to support testing. ACM Comput. Surv. **41**(2), 9:1–9:76 (2009)
23. Hoare, C.A.R.: How did software get so reliable without proof? In: Gaudel, M.-C., Woodcock, J. (eds.) FME 1996. LNCS, vol. 1051, pp. 1–17. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-60973-3_77
24. Houssais, B.: Verification of an Algol 68 implementation. ACM SIGPLAN Not. **12**(6), 117–128 (1977)
25. ISO: Conformance testing methodology and framework. International Standard IS-9646 (1991)
26. Jalote, P.: Specification and testing of abstract data types. In: IEEE International Computer Software and Applications Conference COMSAC, pp. 508–511 (1983)
27. Jard, C., Jéron, T.: TGV: theory, principles and algorithms. Int. J. Softw. Tools Technol. Transf. **7**(4), 297–315 (2005)
28. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines-a survey. Proc. IEEE **84**(8), 1090–1123 (1996)
29. Legeard, B., Peureux, F.: Generation of functional test sequences from B formal specifications-presentation and industrial case study. In: 16th IEEE International Conference on Automated Software Engineering (ASE 2001), Coronado Island, San Diego, CA, USA, 26–29 November 2001, pp. 377–381. IEEE Computer Society (2001)

30. Marre, B., Blanc, B.: Test selection strategies for Lustre descriptions in GATeL. Electr. Notes Theor. Comput. Sci. **111**, 93–111 (2005)
31. Petiot, G., Kosmatov, N., Botella, B., Giorgetti, A., Julliand, J.: How testing helps to diagnose proof failures. Formal Aspects Comput. **30**(6), 629–657 (2018). https://doi.org/10.1007/s00165-018-0456-4
32. Pitt, D.H., Freestone, D.: The derivation of conformance tests from LOTOS specifications. IEEE Trans. Software Eng. **16**(12), 1337–1343 (1990)
33. Sarikaya, B., von Bochmann, G.: Some experience with test sequence generation for protocols. In: Protocol Specification, Testing and Verification, Proceedings of the IFIP WG6.1 Second International Workshop on Protocol Specification, Testing and Verification, Idyllwild, CA, USA, 17–20 May 1982, pp. 555–567. North-Holland (1982)
34. Satpathy, M., Butler, M., Leuschel, M., Ramesh, S.: Automatic testing from formal specifications. In: Gurevich, Y., Meyer, B. (eds.) TAP 2007. LNCS, vol. 4454, pp. 95–113. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73770-4_6
35. Stocks, P., Carrington, D.A.: Test templates: a specification-based testing framework. In: Proceedings of the 15th International Conference on Software Engineering, Baltimore, Maryland, USA, 17–21 May 1993, pp. 405–414. IEEE Computer Society/ACM Press (1993)
36. Tretmans, J.: Test generation with inputs, outputs and repetitive quiescence. Softw. Concepts Tools **17**(3), 103–120 (1996)
37. Tretmans, J., van de Laar, P.: Model-based testing with TorXakis. In: Proceedings of 30th CECIIS, the Central European Conference on Information and Intelligent Systems, Varaždin, Croatia, 2–4 October 2019, pp. 247–258 (1987)
38. Ural, H.: A test derivation method for protocol conformance testing. In: Protocol Specification, Testing and Verification VII, Proceedings of the IFIP WG6.1 Seventh International Conference on Protocol Specification, Testing and Verification, Zurich, Switzerland, 5–8 May 1987, pp. 347–358. North-Holland (1987)
39. Veanes, M., Campbell, C., Grieskamp, W., Schulte, W., Tillmann, N., Nachmanson, L.: Model-based testing of object-oriented reactive systems with Spec Explorer. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) Formal Methods and Testing. LNCS, vol. 4949, pp. 39–76. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78917-8_2