Towards Human-Like Automated Test Generation: Perspectives from Cognition and Problem Solving

Eduard Enoiu[1], Robert Feldt[2]
[1]School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
[2]Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden

***Abstract*—Automated testing tools typically create test cases that are different from what human testers create. This often makes the tools less effective, the created tests harder to understand, and thus results in tools providing less support to human testers. Here, we propose a framework based on cognitive science and, in particular, an analysis of approaches to problem solving, for identifying cognitive processes of testers. The framework helps map test design steps and criteria used in human test activities and thus to better understand how effective human testers perform their tasks. Ultimately, our goal is to be able to mimic how humans create test cases and thus to design more human-like automated test generation systems. We posit that such systems can better augment and support testers in a way that is meaningful to them.**

I. Introduction

The creation of test cases in software testing is an intellectual activity in which engineers allocate a variety of cognitive resources when confronted with challenges, as they go along. This is largely a manual activity, dependent on the ingenuity and thoroughness of humans. Automated test generation [1] has been proposed to allow test cases to be created with less effort. The goal is to automatically find a small set of test cases that check the correctness of the system and guard against (previous as well as future) faults. While a lot of progress has been made, it remains a challenge to create strong as well as small test suites that are also relevant to developers. Consequently, industry still mainly rely on manual testing.

Given the emerging evidence (e.g., [2], [3]) suggesting that *automated test generation cannot match the fault detection effectiveness of manually created tests*, the purpose herein is to propose a framework to help understand and, ultimately, to mimic/simulate the problem-solving behaviors of testers. Recently, Gay et al. [4], [5] proposed some strategies for improving automated test data generation using adaptation and context-based characteristics. However, context and domain-specific information is just some of the dimensions used by human testers. While many of these test generation approaches are AI/Machine Learning (ML) oriented, we are concerned with grounding this framework in high-level cognitive theories and psychological data. For example, we should certainly not treat search-based test generation as a theory of human-based testing, in the same way as AlphaGo is not used as a theory of human Go. While this shows that automated systems can beat humans, we argue that when they do not we have something to gain from modeling human cognition and problem solving and, potentially, mimicking it in our tools.

II. The HAT Framework

The *Human-based Automated Test (HAT)* generation framework assembles cognitive processes used by testers into a problem solving model and cognitive architecture that can be used for improving automated test generation. Rather than directly trying to build more 'intelligent' automated test generation systems, we outline a framework for the study of the human mind when creating test cases. Figure 1 provides a high level summary of the key components of the framework. It uses a problem solving approach [6], [7] to depict the behavior by which testers perform test creation activities. The HAT framework proposes the use of cognitive architectures and simulation to discover the conditions when humans create effective test cases. Below we outline its three main levels.

1) Intermediate Products and Protocol Analysis: If we are interested in how people create test cases, we need to collect information about the various cognitive actions and decisions taken during the process. Whereas the created test cases typically do not convey their underlying creation process, this may be revealed by studying the intermediate products [8]. Testers often create sketches or note useful information in the course of creating test cases. These intermediate products provide much finer constraints (states S(x) of a cognitive process in Figure 1) on possible explanations of certain behaviors. These can be complemented by verbal protocol [9] analysis using a think aloud method that provides information about the course of test creation (verbal reports v(y) in Figure 1). Although protocol analysis is quite old, and it has its limitations, there is evidence [10], [11] supporting the use of verbal reports for providing valid data about what humans are thinking and serve as the basis for theories of testing.

2) Problem Solving: It is clear that humans involved in software testing are basically trying to solve a sequence of test problems/challenges. Therefore, examining test case design as a problem solving process can help us explain how individuals are able to apply their knowledge to existing and novel problems. The results of analyzing intermediate products and verbal protocols should be used to provide finer constraints and possible accounts of how people solve the problem of creating test cases. Problem solving is a very important topic of research in cognitive science but also in applied fields such as artificial intelligence. In software engineering, the cognitive models developed by Hale et al. [12] and Enoiu et al. [13] and instantiated for the specific problem of creating test cases

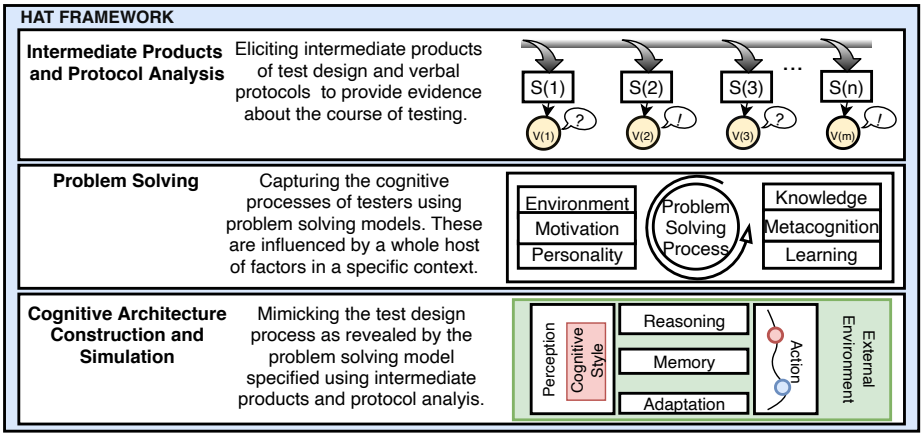

Fig. 1: Outline of the HAT framework and its key features.

and debugging seem to be the closest models to the HAT framework for capturing the cognitive processes of testers. Nevertheless, since the classical models these results were based on are quite limited, they should be complemented with other cognitive theories that can handle different psychological issues of human-based test creation. However, problem solving is not enough and many factors affect humans during testing, e.g. motivation, creativity, emotion, transfer of knowledge, language parsing, expertise, etc. Over time, we will thus need to elaborate the framework to cater also to other psychological as well as, eventually, social factors.

3) Cognitive Architecture Construction and Simulation: Problem solving in software testing, learning and the development of testing expertise are behaviours that require rigorous explanations. Just as a software architect may produce an architectural model that shows how different parts of the software are working together, so are we interested in building a model that incorporates a theory of how testers think and learn. One way to mimic the captured problem solving process is to build a general model of architecture in which a variety of cognitive processes can be specified as runnable computational models (e.g., algorithms guiding the test generation). Simulations are then performed to check if the posited processes are sufficient to lead to the observed protocol behavior. While there is no cognitive architecture for software testing, several general architectural models of cognition exist [14] such as ACT-R, SOAR and CLARION. Such cognitive models could help us reverse engineer the human tester— the only available intelligent test design system. These cognitive architectures could constitute a solid basis for developing human-like automated test generation tools, properly grounded in existing cognitive research.

III. Conclusions

We conclude that the HAT framework is a basis for employing intermediate products and protocol analysis to build cognitive models that can be used for advancing the understanding of humans performing software testing. Therefore, we argue that this framework can help enhance automated test generation as well to better accommodate the cognitive needs of human testers with the testing tools.

Acknowledgments

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957212. This work is partially supported by Software Center and Vinnova (XIVT).

References

[1] S. Anand *et al.*, "An orchestrated survey of methodologies for automated software test case generation," *JSS*, vol. 86, no. 8, pp. 1978–2001, 2013.
[2] G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, "Does automated unit test generation really help software testers? a controlled empirical study," *TOSEM*, vol. 24, no. 4, pp. 1–49, 2015.
[3] E. Enoiu, A. Causevic, D. Sundmark, and P. Pettersson, "A controlled experiment in testing of safety-critical embedded software," in *ICST*. IEEE, 2016, pp. 1–11.
[4] G. Gay, "The fitness function for the job: Search-based generation of test suites that detect real faults," in *ICST*. IEEE, 2017, pp. 345–355.
[5] ——, "One-size-fits-none? improving test generation using context-optimized fitness functions," in *SBST Workshop*. IEEE, 2019, pp. 3–4.
[6] A. Newel and H. A. Simon, "Human problem solving," *Englewood Cliffs, NJ*, 1972.
[7] J. Pretz, A. Naples, and R. Sternberg, "Recognizing, defining, and representing problems," *The psychology of problem solving*, vol. 30, no. 3, 2003.
[8] S. Jaarsveld and C. van Leeuwen, "Sketches from a design process: Creative cognition inferred from intermediate products," *Cognitive Science*, vol. 29, no. 1, pp. 79–101, 2005.
[9] A. Ericsson and H. Simon, "Protocol analysis," *A companion to cognitive science*, vol. 14, pp. 425–432, 1998.
[10] J. Hughes and S. Parkes, "Trends in the use of verbal protocol analysis in software engineering research," *Behaviour & Information Technology*, vol. 22, no. 2, pp. 127–140, 2003.
[11] J. Itkonen, M. V. Mäntylä, and C. Lassenius, "The role of the tester's knowledge in exploratory software testing," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 707–724, 2012.
[12] J. Hale, S. Sharpe, and D. Hale, "An evaluation of the cognitive processes of programmers engaged in software debugging," *Journal of Software Maintenance: Research and Practice*, vol. 11, no. 2, pp. 73–91, 1999.
[13] E. Enoiu, G. Tukseferi, and R. Feldt, "Towards a model of testers' cognitive processes: Software testing as a problem solving approach," in *International Conference on Software Quality, Reliability and Security Companion*. IEEE, 2020, pp. 272–279.
[14] I. Kotseruba and J. K. Tsotsos, "40 years of cognitive architectures: core cognitive abilities and practical applications," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 17–94, 2020.