A New Taxonomy of Software Testing Approaches

Seeking More Standardized Standards

Samuel Joseph Crawford, Jacques Carette, & Spencer Smith
{crawfs1, carette, smiths}@mcmaster.ca

Department of Computing and Software, McMaster University

April 22, 2023

Goal

The first step to any formal process is **understanding the underlying domain**. Therefore, a systematic and rigorous understanding of software testing approaches is needed to develop formal tools to test software. In my specific case, my motivation was seeing **which kinds of testing can be generated automatically by Drasil**, "a framework for generating all of the software artifacts for (well understood) research software" [1].

Problem

Most software testing ontologies seem to focus on the high-level testing process rather than the testing techniques themselves. For example:

- [2] mainly focuses on parts of the testing process (e.g., test goal, testable entity)
- [3] provides a foundation for classification but "do[] not aim at providing a systematic and exhaustive state-of-the-art survey of [either domain]" (p. A:2)

Methodology

- If a taxonomy doesn't already exist, I should create one!
  - I started with an ad hoc approach, focusing on textbooks trusted at McMaster [4, 5, 6]
  - We then realized that this was too arbitrary, so I then started from more established sources [7, 8, 9, 10, 11, 12]
  - The goal of this approach is to iterate, eventually revisiting the original textbooks, until I build up enough knowledge to encounter diminishing returns (ideally no returns!)
- Since there are many standardized documents about software testing (or software in general), this should be trivial, no?

In My Experience

NO.

Examples

[7] is a standard for general concepts related to software testing. However, it is not comprehensive. For example, as shown in Figure 1, most (55 out of 99) testing approaches mentioned in this standard do not have an accompanying definition! Eight of these were present in the previous version of this standard [13], and nine were present in another IEEE standard [10] that would have been available upon publication of this standard. However, the presence of a definition does not guarantee that it is useful! See Figure 1 for some examples.

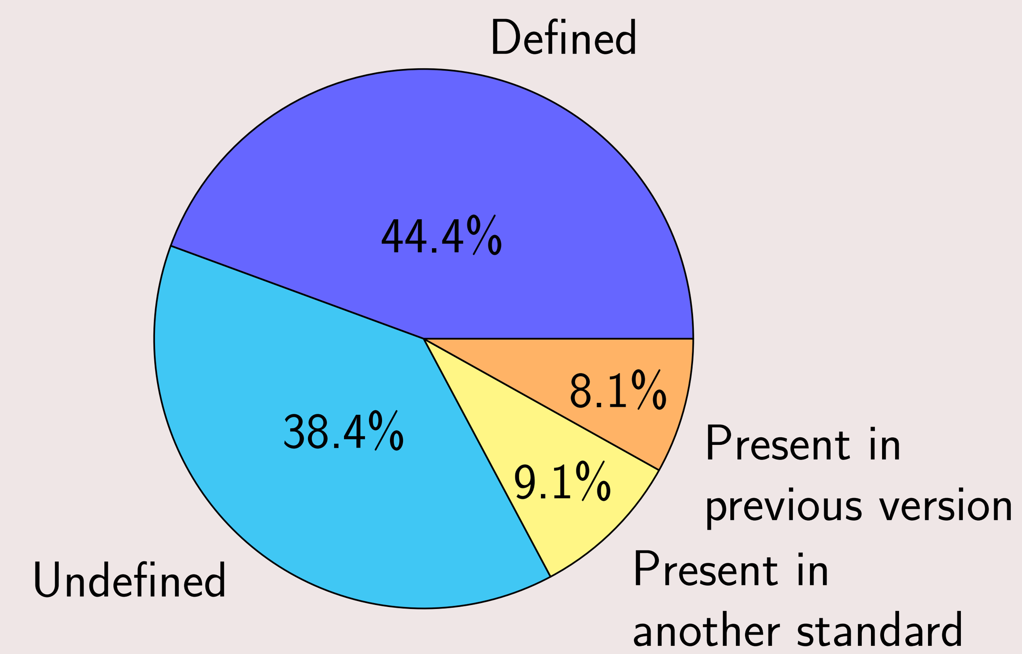
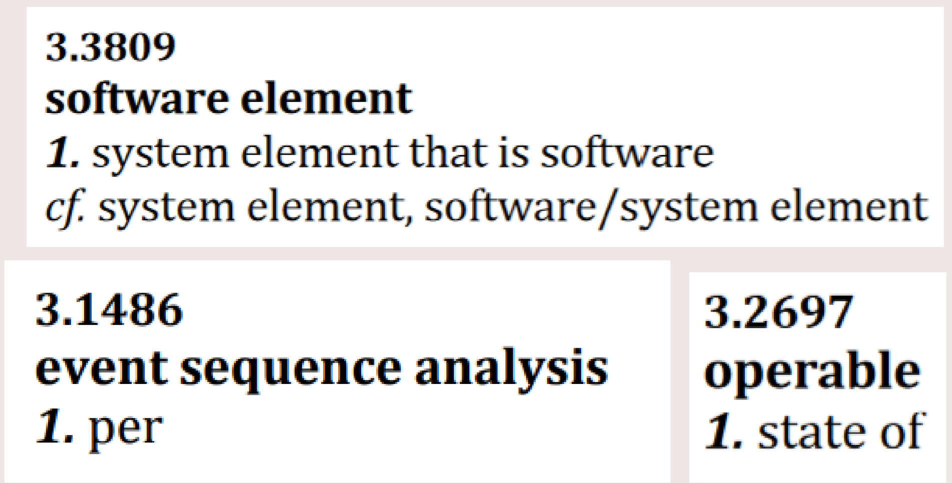


Figure 1: Breakdown of testing approach definitions from [7].



3.3809
software element
1. system element that is software
cf. system element, software/system element

3.1486
event sequence analysis
1. per

3.2697
operable
1. state of

Figure 2: Some less-than-helpful definitions from [10].

Conclusions & Future Work

Leveraging the strictly-typed nature of Elm and its model-view-update architecture, we were able to create a simplified multi-user framework, requiring the programmer to write no server-side code. In addition to the upcoming pedagogical study, future work includes a data modelling extension allowing persistent, structured data, an authentication/authorization scheme, a binary data format to reduce network communication, and curriculum development for a TEASync-based summer camp.

References

[1] J. Carette, S. Smith, J. Balaci, T.-Y. Wu, S. Crawford, D. Chen, D. Szymczak, B. MacLachlan, D. Scime, and M. Niazi, "Drasil," Feb. 2021.

[2] G. Tebes, L. Olsina, D. Peppino, and P. Becker, "TestTDO: A Top-Domain Software Testing Ontology," (Curitiba, Brazil), pp. 364–377, May 2020.

[3] M. Unterkalmsteiner, R. Feldt, and T. Gorschek, "A Taxonomy for Requirements Engineering and Software Test Alignment," *ACM Transactions on Software Engineering and Methodology*, vol. 23, pp. 1–38, Mar. 2014.
arXiv:2307.12477 [cs].

[4] R. Patton, *Software Testing*.
Indianapolis, IN, USA: Sams Publishing, 2nd ed., 2006.

[5] J. Peters and W. Pedrycz, *Software Engineering: An Engineering Approach*.
Worldwide series in computer science, John Wiley & Sons, Ltd., 2000.

[6] H. van Vliet, *Software Engineering: Principles and Practice*.
Chichester, England: John Wiley & Sons, Ltd., 2nd ed., 2000.

[7] ISO/IEC and IEEE, "ISO/IEC/IEEE International Standard - Systems and software engineering –Software testing –Part 1: General concepts," *ISO/IEC/IEEE 29119-1:2022(E)*, Jan. 2022.

[8] H. Washizaki, ed., *Guide to the Software Engineering Body of Knowledge, Version 4.0*.
Jan. 2024.