

# 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 our specific case, our 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 approaches 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 "does not aim at providing a systematic and exhaustive state-of-the-art survey of [either domain]" (p. A:2)

# Methodology

Since a taxonomy doesn't already exist, we should create one!

- We started with an ad hoc approach, focusing on textbooks trusted at McMaster
- We then realized that this was too arbitrary, so we started from more established sources, such as IEEE and SWEBOK
- The goal of this approach is to iterate, eventually revisiting the original textbooks, until enough knowledge is built up 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 Our Experience

#### Levels of testing

Unit testing
Integration testing
System testing
System integration testing
Acceptance testing

- User acceptance testingOperational acceptance
- testing
- Factory acceptance testing
- Alpha testing— Beta testing
- Production verification testing

#### Test practices

Model-based testing
Scripted testing
Exploratory testing
Experience-based testing
Manual testing
A/B testing
Back-to-back testing
Mathematical-based testing
Fuzz testing
Keyword-driven testing
Automated testing
— Capture-replay driven
— Data-driven

#### Types of testing

Functional testing

Accessibility testing
Compatibility testing
Conversion testing
Disaster/recovery testing
Installability testing
Interoperability testing
Localization testing
Maintainability testing
Performance-related

- testing
   Performance
   Load
- StressCapacity
- Recovery
   Portability testing
   Procedure testing
   Reliability testing
   Security testing

Usability testing

## Static testing

Reviews (ISO/IEC 20246) Static analysis Model verification

#### Test design techniques / measures

Specification-based:

— Equivalence partitioning

— Classification tree metho

- Classification tree method
   Boundary value analysis
   Syntax testing
- Syntax testing— Combinatorial testing
- Combinatorial testing
   All combinations
- All combinations— Pairwise
- Each choiceBase choice
- Decision table testing
- Cause-effect graphingState transition testing
- Scenario testing— Use case testing
- Random testingMetamorphic testing
- Metamorphic testing
   Requirements-based
- testing
- Structure-based:

   Statement testing
- Branch testingDecision testing
- Decision testing
  Branch condition testing
  Branch condition
- Branch conditioncombination testing— MC/DC testing
- Data flow testing
  All-definitions testing
  All-C-uses testing
- All-P-uses testing— All-uses testing— All-DU-paths testing
- Experience-based:
   Error guessing

# and a test practice What distinguishes the following pairs is unclear: Disaster/recovery testing and recovery testing Branch condition

testing and

combination

testing

branch condition

Information often

example, the

ambiguities:

appears logical, but this

often breaks down. For

classification of test

reveals the following

approaches in Figure 1

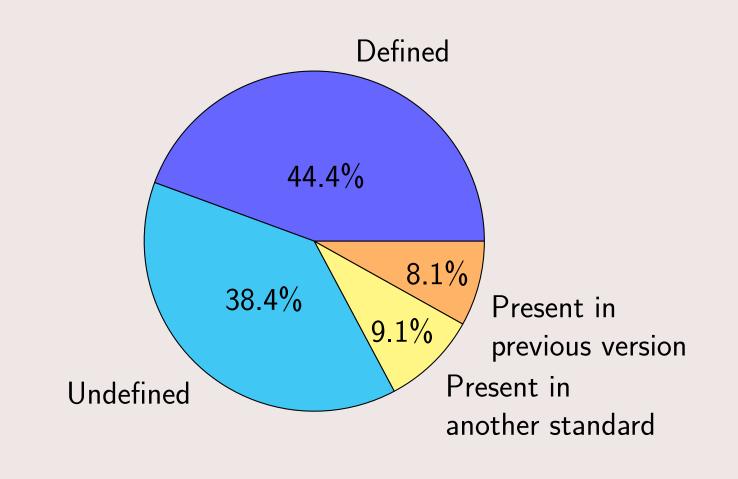
Experience-based

design technique

testing is both a test

# More Examples

A big contributor to the ambiguities in Figure 1 is the number of definitions that are not given. Despite its source [4] being a standard for general concepts related to software testing, it leaves much unstandardized. For example, as shown in Figure 1, most (55 out of 99) testing approaches mentioned do not have a definition! Eight of these were at the very least described in the previous version of this standard [5], and nine were present in the same way in another IEEE standard [6] that would have been available upon publication of this one. However, the presence of a definition does not guarantee that it is useful! See Figure 1 for some good (bad?) examples.



3.3809
software element
1. system element that is software
cf. system element, software/system element
3.1486
3.2697

operable

1. state of

Figure 3: Some less-than-helpful definitions from [6].

event sequence analysis

**1.** per

Figure 2: Breakdown of testing approach definitions from [4].

# Conclusions & Future Work

- Current software testing taxonomies are incomplete, inconsistent, and/or incorrect
- For one to be useful, it needs to be built systematically from a large body of established sources
- We will continue investigating how the literature defines and categorizes software testing approaches to analyze any discrepancies and structure these ideas coherently

#### 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] 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.
- [5] ISO/IEC and IEEE, "ISO/IEC/IEEE International Standard Systems and software engineering –Software testing –Part 1: General concepts," ISO/IEC/IEEE 29119-1:2013, Sept. 2013.
- [6] ISO/IEC and IEEE, "ISO/IEC/IEEE International Standard Systems and software engineering–Vocabulary," ISO/IEC/IEEE 24765:2017(E), Sept. 2017.

### Acknowledgments

We thank the Government of Ontario for OGS funding.

#### Figure 1: A classification of some "test approach choices" [4, p. 22].