A New Taxonomy of Software Testing Approaches

Seeking More Standardized Standards

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

November 1, 2024

Goal

Taxonomy of software testing approaches

- Should be **systematic, rigorous, and "complete"**
- Application: **automatically generating test cases** in Drasil
- The **underlying domain** should drive the scope and prerequisites for generated test cases

Problem

Existing software testing taxonomies are inadequate

- Tebes et al. (2020): focuses on parts of the testing process (e.g., test goal, testable entity)
- Souza et al. (2017): prioritizes organizing testing approaches over defining them
- Unterkalmsteiner et al. (2014): focuses on knowledge transfer between development phases

Methodology

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

- Start from **"standard" resources** (e.g., IEEE [1], [2], [3], [4]; the SWEBOK Guide [5])
- **Collect** relevant information (over 500 testing approaches and 70 software qualities, along with their definitions) and **organize** it into spreadsheets
- *Note: static testing approaches are included, since they are sometimes included in "software testing" [1, p. 17], [3, p. 440], [5, p. 5-2]*
- Iterate this process until there are diminishing returns, implying that something approaching a **complete taxonomy** has emerged!
- Since there are many standardized documents about software testing (or software in general), **this should be trivial, no?**

In Our Experience...

NO.

Levels of testing

Unit testing
Integration testing
System testing
System integration testing
Acceptance testing
— User acceptance testing
— Operational acceptance testing
— Factory acceptance testing
— Alpha testing
— Beta testing
— Production verification testing

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
— Stress
— Capacity
— Recovery
Portability testing
Procedure testing
Reliability testing
Security testing
Usability testing

Test design techniques / measures

Specification-based:
— Equivalence partitioning
— Classification tree method
— Boundary value analysis
— Syntax testing
— Combinatorial testing
— All combinations
— Pairwise
— Each choice
— Base choice
— Decision table testing
— Cause-effect graphing
— State transition testing
— Scenario testing
— Use case testing
— Random testing
— Metamorphic testing
— Requirements-based testing

Structure-based:
— Statement testing
— Branch testing
— Decision testing
— Branch condition testing
— Branch condition combination 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

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

Static testing

Reviews (ISO/IEC 20246)
Static analysis
Model verification

Figure 1: Classification of some "test approach choices" [1, p. 22].

The classification of testing approaches in Figure 1 *appears* logical but contains the following ambiguities:

- Experience-based testing is both a test design technique *and* a test practice
- Pairs of terms are not distinguished:
  - Disaster/recovery testing and recovery testing
  - Branch condition testing and branch condition combination testing
  - Operational acceptance testing and operational testing [3, p. 303]

More Examples

[1] and [2] are software testing standards that leave much unstandardized (see Figure 2)

- About 20% (23 out of 114) of testing approaches from these standards **do not have a definition**!
- Five of these were (at the very least) described in the previous version of this standard [4]
- Four were present in the same way in another IEEE standard [3] before this one was published

Having definitions does not mean they are useful; see Figure 3 for some good (bad?) examples

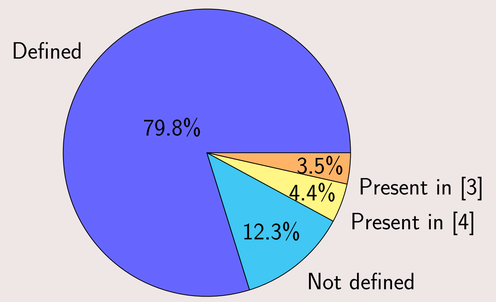

Figure 2: Breakdown of testing approach definitions in [1] and [2].

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

event sequence analysis
1. per

operable
1. state of

device
1. mechanism or piece of equipment designed to serve a purpose or perform a function
cf. platform

Figure 3: Less-than-helpful definitions [3, pp. 421, 170, 136, 301 (counterclockwise from top)]. Note: "equipment" is not defined, and "mechanism" is only defined as how "a function … transform[s] input into output" [p. 270].

The SWEBOK Guide's Definition of "Scalability Testing"

"Scalability testing evaluates the capability to use and learn the system and the user documentation. It also focuses on the system's effectiveness in supporting user tasks and the ability to recover from user errors" [5, p. 5-9]

- This seems to define "usability testing" with elements of functional and recovery testing
- The SWEBOK Guide's definition of elasticity testing [5, p. 5-9] only cites a single source **that doesn't contain the words "elasticity" or "elastic"**!

Alpha testing is quite common, but there is disagreement on who performs it:

1. "users within the organization developing the software" [3, p. 17],
2. "a small, selected group of potential users" [5, p. 5-8], or
3. "roles outside the development organization" [6]

Conclusions & Future Work

- Current software testing taxonomies are **incomplete, inconsistent, and/or incorrect**
- Ideally, one will be built systematically from a large body of established sources
- We will continue investigating, analyzing, and structuring how the literature defines and categorizes software testing approaches
- This **broad and consistent taxonomy** will hopefully grow as the field of testing advances

References

[1] 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.

[2] ISO/IEC and IEEE, "ISO/IEC/IEEE International Standard - Software and systems engineering –Software testing –Part 4: Test techniques," *ISO/IEC/IEEE 29119-4:2021(E)*, Oct. 2021.

[3] ISO/IEC and IEEE, "ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, Sept. 2017.

[4] ISO/IEC and IEEE, "ISO/IEC/IEEE International Standard - Systems and software engineering –Software testing –Part 1: General concepts," *ISO/IEC/IEEE 29119-1:2013*, Sept. 2013.

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

[6] M. Hamburg and G. Mogyorodi, eds., "ISTQB Glossary, v4.3," 2024.

Acknowledgments

We thank the Government of Ontario for OGS funding and Chris Schankula for this template.