Putting Software Testing Terminology to the Test

Samuel J. Crawford*, Spencer Smith*, Jacques Carette*
*Department of Computing and Software
McMaster University
Hamilton, Canada
{crawfs1, smiths, carette}@mcmaster.ca

Abstract—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.

Index Terms—component, formatting, style, styling, insert.

I. Methodology

A. Source Order

0) Textbooks trusted at McMaster [1, 2, 3]
   - Ad hoc and arbitrary; not systematic
   - Colored maroon
1) Established standards (such as IEEE, ISO/IEC, and SWEBOK) [4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
   - Standards organizations colored green
   - "Meta-level" commentaries or collections of terminology (often based on these standards), such as [14], colored blue
2) Other resources: less-formal classifications of terminology (such as [15]), sources investigated to "fill in" missing definitions (see Undefined Terms), and testing-related resources that emerged for unrelated reasons
   - Colored black, along with any "surface-level" analysis that followed trivially

By looking at a variety of sources, discrepancies both within and between them can be uncovered, and the various classifications of terminology can be analyzed.

B. Procedure

Except for some sources in Undefined Terms, all sources were looked through in their entirely to "extract" as much terminology as possible. Heuristics were used to guide this process, by investigating…

- …glossaries and lists of terms
- …testing-related terms
  e.g., terms that included "test(ing)", "validation", "verification", "review(s)", or "audit(s)"
- …terms that had emerged as part of already-discovered testing approaches, especially those that were ambiguous or prompted further discussion
  e.g., terms that included "performance", "recovery", "component", "bottom-up", "boundary", or "configuration"
- …terms that implied testing approaches
  - A kind of "coverage" implies a test approach aimed to maximize this coverage; for example, "path testing" is testing that "aims to execute all entry-to-exit control flow paths in a SUT's control flow graph" [5, p. 5013], thus maximizing the path coverage (see also #63, [16, Fig. 1])
  - TODO: qualities imply types

If a term's definition had already been recorded, either the "new" one replaced it if the "old" one wasn't as clear/concise or parts of both were merged to paint a more complete picture. If any discrepancies or ambiguities arose, they were investigated to a reasonable extent and documented. If a testing approach was mentioned but not defined, it was still added to the glossary to indicate it should be investigated further. A similar methodology was used for tracking software qualities, albeit in a separate document.

During this investigation, some terms came up that seemed to be relevant to testing but were so vague, they didn't provide any new information. These were decided to be not worth tracking (see #39, #44, #28) and are listed below:

- Evaluation: the "systematic determination of the extent to which an entity meets its specified criteria" [7, p. 167]
- Product Analysis: the "process of evaluating a product by manual or automated means to determine if the product has certain characteristics" [7, p. 343]
- Quality Audit: "a structured, independent process to determine if project activities comply with organizational and project policies, processes, and procedures" [7, p. 361]
- Software Product Evaluation: a "technical operation that consists of producing an assessment of one or more characteristics of a software product according to a specified procedure" [7, p. 424]

However, over the course of this research, our scope was adjusted to include some terms for our initial list of test approaches to be filtered out later, such as types of attacks (see #55), meaning that some entries were missed during the first pass(es) of these resources. While reiterating over

Identify applicable funding agency here. If none, delete this.

these resources would be ideal, this may not be possible due to time constraints.

C. Undefined Terms

This process also led to some testing approaches without definitions; [4] and [14] in particular introduced many. Once more "standard" sources had been exhausted, a strategy was proposed to look for sources that explicitly defined these terms, with the added benefit of uncovering more terms to explore, potentially in different domains (see #57). This also uncovered some out-of-scope testing approaches, including EMSEC testing, HTML testing, and aspects of loop testing and orthogonal array testing; since these are out of scope, relevant sources were not investigated fully.

The following terms (and their respective related terms) were explored, bringing the number of testing approaches from 431 to 514 and the number of undefined terms from 152 to 170 (the assumption can be made that about 78% of added terms also included a definition):

- Assertion Checking
- Loop Testing
- EMSEC Testing
- Asynchronous Testing
- Performance(-related) Testing
- Web Application Testing
  - HTML Testing
  - DOM Testing
- Sandwich Testing
- Orthogonal Array Testing
- Backup Testing

Different sources categorized software testing approaches in different ways; while it is useful to record and think about these categorizations, following one (or more) during the research stage could lead to bias and a prescriptive categorization, instead of letting one emerge descriptively during the analysis stage. Since these categorizations are not mutually exclusive, it also means that more than one could be useful (both in general and to this specific project); more careful thought should be given to which are "best", and this should happen during the analysis stage.

II. Observations

A. Categories of Testing Approaches

For classifying different kinds of tests, (author?) [4] provide some terminology (see ??). However, other sources [17, 18] provide alternate categories (see ??) which may be beneficial to investigate to determine if this categorization is sufficient. A "metric" categorization was considered at one point, but was decided to be out of the scope of this project , #21, and #22. Related testing approaches may be grouped into a "class" or "family" to group those with "commonalities and well-identified variabilities that can be instantiated", where "the commonalities are large and the variabilities smaller" (see #64). Examples of these are the classes of combinatorial [13, p. 15] and data flow testing (p. 3) and the family of performance-related testing [19, p. 1187][1], and may also be implied for security testing, a test type that consists of "a number of techniques[2]" [13, p. 40].

It also seems that these categories are orthogonal. For example, "a test type can be performed at a single test level or across several test levels" (4, p. 15; (year?)). Due to this, a specific test approach can be derived by combining test approaches from different categories; for some examples of this. However, the boundaries between items within a category may be unclear: "although each technique is defined independently of all others, in practice [sic] some can be used in combination with other techniques" [13, p. 8]. For example, "the test coverage items derived by applying equivalence partitioning can be used to identify the input parameters of test cases derived for scenario testing" (p. 8). Even the categories themselves are not consistently defined, as some approaches are categorized differently by different sources; these differences will be tracked noted so that they can be analyzed more systematically (see #21). There are also several instances of inconsistencies between parent and child test approach categorizations (which may indicate they aren't necessarily the same, or that more thought must be given to classification/organization). Examples of discrepancies in test-approach categorization:

1) Experience-based testing is categorized as both a test design technique and a test practice on the same page [4, pp. 22, 34]!
   - These authors previously say "experience-based testing practices like exploratory testing … are not … techniques for designing test cases", although they "can use … test techniques" [(year?), p. viii]. This implies that "experience-based test design techniques" are techniques used by the practice of experience-based testing, not that experience-based testing is itself a test technique. If this is the case, it is not always clearly articulated (4, pp. 4, 22; (year?); 5, p. 5-13; [12]) and is sometimes contradicted [14, p. 46]. However, this conflates the distinction between "practice" and "technique", making these terms less useful, so this may just be a mistake (see #64).
   - This also causes confusion about its children, such as error guessing and exploratory testing; again, on the same page, (author?) say error guessing is an "experience-based test design technique" and "experience-based test practices

[1] The original source describes "performance testing … as a family of performance-related testing techniques", but it makes more sense to consider "performance-related testing" as the "family" with "performance testing" being one of the variabilities.

[2] This may or may not be distinct from the notion of "test technique" described in ??.

include … exploratory testing, tours, attacks, and checklist-based testing" [(year?), p. 34]. Other sources also do not agree whether error guessing is a technique (pp. 20, 22; (year?)) or a practice [5, p. 5-14].

2) The following are test approaches that are categorized as test techniques in [13, p. 38], followed by sources that categorize them as test types:
   a) Capacity testing (4, p. 22; (year?); implied by its quality (11; 13, Tab. A.1) and by [14, p. 53])
   b) Endurance testing (8, p. 2; implied by [14, p. 55])
   c) Load testing (4, pp. 5, 20, 22; (year?) ; [12]; implied by [14, p. 54])
   d) Performance testing (4, pp. 7, 22, 26-27; (year?); implied by [14, p. 53])
   e) Stress testing (4, pp. 9, 22; (year?); implied by [14, p. 54])
3) Model-based testing is categorized as both a test practice (4, p. 22; (year?)) and a test technique (20, p. 4; implied by 13, p. 7; (year?)).
4) Data-driven testing is categorized as both a test practice [4, p. 22] and a test technique [20, p. 43] .

Since test techniques are able to "identify test coverage items… and derive corresponding test cases" (4, p. 11; similar in (year?)) in a "systematic" way [(year?), p. 464], "the coverage achieved by a specific test design technique" can be calculated as "the number of test coverage items covered by executed test cases" divided by "the total number of test coverage items identified" [(year?), p. 30]. "Coverage levels can range from 0% to 100%" and may or may not include "infeasible" test coverage items, which are "not … executable or [are] impossible to be covered by a test case" (p. 30).

B. Categorizations

Software testing approaches can be divided into the following categories. Note that "there is a lot of overlap between different classes of testing" [14, p. 8], meaning that "one category [of test techniques] might deal with combining two or more techniques" [5, p. 5-10]. For example, "performance, load and stress testing might considerably overlap in many areas" [19, p. 1187]. A side effect of this is that it is difficult to "untangle" these classes; for example, take the following sentence: "whitebox fuzzing extends dynamic test generation based on symbolic execution and constraint solving from unit testing to whole-application security testing" [21, p. 23]!

Despite its challenges, it is useful to understand the differences between testing classes because tests from multiple subsets within the same category, such as functional and structural, "use different sources of information and have been shown to highlight different problems" [5, p. 5-16]. However, some subsets, such as deterministic and random, may have "conditions that make one approach more effective than the other" [5, p. 5-16].

- Visibility of code: black-, white-, or grey-box (specificational/functional, structural, or a mix of the two) (13, p. 8; 5, pp. 5-10, 5-16; 16, p. 601, called "testing approaches" and (stepwise) code reading replaced "grey-box testing"; 22, pp. 57-58; 15, p. 213; 1, pp. 53, 218; 23, p. 69; 20, pp. 4-5, called "testing methods")
- Level/stage of testing: unit, integration, system, or acceptance (5, pp. 5-6 to 5-7; [12]; 15, p. 218 ; 1; 23; 2; 24, pp. 9, 13) (sometimes includes installation [3, p. 439] or regression [17, p. 3])
- Source of information for design: specification, structure, or experience [13, p. 8]
  - Source of test data: specification-, implementation-, or error-oriented [2, p. 440]
- Test case selection process: deterministic or random [5, p. 5-16]
- Coverage criteria: input space partitioning, graph coverage, logic coverage, or syntax-based testing [22, pp. 18-19]
- Question: what-, when-, where-, who-, why-, how-, and how-well-based testing; these are then divided into a total of "16 categories of testing types"[3] [14, p. 17]
- Execution of code: static or dynamic (15, p. 214; 24, p. 12; 1, p. 53)
- Goal of testing: verification or validation (15, p. 214; 23, pp. 69-70)
- Property of code [15, p. 213] or test target [20, pp. 4-5]: functional or non-functional
- Human involvement: manual or automated [15, p. 214]
- Structuredness: scripted or exploratory [15, p. 214]
- Coverage requirement: data or control flow [20, pp. 4-5]
- Adequacy criterion: coverage-, fault-, or error-based ("based on knowledge of the typical errors that people make") [3, pp. 398-399]
- Priority[4]: smoke, usability, performance, or functionality testing [24, p. 12]
- Category of test "type"[5]: static testing, test browsing, functional testing, non-functional testing, or large scale integration (testing) [24, p. 12]
- Purpose: correctness, performance, reliability, or security [25]

Tests can also be tailored to "test factors" (also called "quality factors" or "quality attributes"): "attributes of the software that, if they are wanted, pose a risk to the success of the software" [23, p. 40]. These include correctness, file integrity, authorization, audit trail, continuity of processing, service levels (e.g., response time), access

[3]Not formally defined, but distinct from the notion of "test type" described in ??.

[4]In the context of testing e-business projects.

[5]"Each type of test addresses a different risk area" [24, p. 12], which is distinct from the notion of "test type" described in ??.

control, compliance, reliability, ease of use, maintainability, portability, coupling (e.g., with other applications in a given environment), performance, and ease of operation (e.g., documentation, training) [23, pp. 40-41]. These may overlap with ?? and/or the "Results of Testing (Area of Confidence)" column in the summary spreadsheet.

Engström "investigated classifications of research" [26, p. 1] on the following four testing techniques. These four categories seem like comparing apples to oranges to me.

- Combinatorial testing: how the system under test is modelled, "which combination strategies are used to generate test suites and how test cases are prioritized" [26, pp. 1-2]
- Model-based testing: the information represented and described by the test model [26, p. 2]
- Search-based testing: "how techniques [6] had been empirically evaluated (i.e. objective and context)" [26, p. 2]
- Unit testing: "source of information (e.g. code, specifications or testers intuition)" [26, p. 2]

C. Existing Taxonomies, Ontologies, and the State of Practice

One thing we may want to consider when building a taxonomy/ontology is the semantic difference between related terms. For example, one ontology found that the term " 'IntegrationTest' is a kind of Context (with semantic of stage, but not a kind of Activity)" while " 'IntegrationTesting' has semantic of Level-based Testing that is a kind of Testing Activity [or] … of Test strategy" [27, p. 157].

A note on testing artifacts is that they are "produced and used throughout the testing process" and include test plans, test procedures, test cases, and test results [18, p. 3]. The role of testing artifacts is not specified in [17]; requirements, drivers, and source code are all treated the same with no distinction [17, p. 3].

In [18], the ontology (ROoST) is made to answer a series of questions, including "What is the test level of a testing activity?" and "What are the artifacts used by a testing activity?" [18, pp. 8-9]. The question "How do testing artifacts relate to each other?" [18, p. 8] is later broken down into multiple questions, such as "What are the test case inputs of a given test case?" and "What are the expected results of a given test case?" [18, p. 21]. These questions seem to overlap with the questions we were trying to ask about different testing techniques.

Most ontologies I can find seem to focus on the high-level testing process rather than the testing approaches themselves. For example, the terms and definitions [28] from TestTDO [29] provide some definitions of testing approaches, but mainly focus on parts of the testing process (e.g., test goal, test plan, testing role, testable entity) and how they relate to one another. (author?) [27, pp. 152-153] may provide some sources for software testing terminology and definitions (this seems to include the ones suggested by Dr. Carette) in addition to a list of ontologies (some of which have been investigated).

One software testing model developed by the Quality Assurance Institute (QAI) includes the test environment ("conditions …that both enable and constrain how testing is performed", including mission, goals, strategy, "management support, resources, work processes, tools, motivation"), test process (testing "standards and procedures"), and tester competency ("skill sets needed to test software in a test environment") [23, pp. 5-6].

(author?) [30] provide a foundation to allow one "to classify and characterize alignment research and solutions that focus on the boundary between [requirements engineering and software testing]" but "do[] not aim at providing a systematic and exhaustive state-of-the-art survey of [either domain]" (p. A:2).

Another source introduced the notion of an "intervention": "an act performed (e.g. use of a technique[7] or a process change) to adapt testing to a specific context, to solve a test issue, to diagnose testing or to improve testing" [26, p. 1] and noted that "academia tend[s] to focus on characteristics of the intervention [while] industrial standards categorize the area from a process perspective" [26, p. 2]. It provides a structure to "capture both a problem perspective and a solution perspective with respect to software testing" [26, pp. 3-4], but this seems to focus more on test interventions and challenges rather than approaches [26, Fig. 5].

[6] Not formally defined, but distinct from the notion of "test technique" described in ??.

[7] Not formally defined, but distinct from the notion of "test technique" described in ??.

III. Introduction

This document is a model and instructions for LaTeX. Please observe the conference page limits. For more information about how to become an IEEE Conference author or how to write your paper, please visit IEEE Conference Author Center website: https://conferences.ieeeauthorcenter.ieee.org/.

A. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

IV. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections IV-A to IV-H below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., \eqref{Eq}) cross references instead of "hard" references (e.g., (1)). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the {eqnarray} equation environment. Use {align} or {IEEEeqnarray} instead. The {eqnarray} environment leaves unsightly spaces around relation symbols.

Please note that the {subequations} environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a \label command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a \label command should not go before the caption of a figure or a table.

Do not use \nonumber inside the {array} environment. It will not stop equation numbers inside {array} (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within

quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)

- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is The Technical Writer's Handbook.

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
Table Type Styles

Table Head	Table Column Head		
	Table column subhead	Subhead	Subhead
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including

units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

Please number citations consecutively within brackets. The sentence punctuation follows the bracket. Refer simply to the reference number; do not use "Ref. [1]" or "reference [1]" except at the beginning of a sentence: "Reference [1] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished". Papers that have been accepted for publication should be cited as "in press". Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.

References

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

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

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

[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] H. Washizaki, Ed., Guide to the Software Engineering Body of Knowledge, Version 4.0, Jan. 2024. [Online]. Available: https://waseda.app.box.com/v/SWEBOK4-book

[6] P. Bourque and R. E. Fairley, Eds., Guide to the Software Engineering Body of Knowledge, Version 3.0. Washington, DC, USA: IEEE Computer Society Press, 2014. [Online]. Available: www.swebok.org

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

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

[9] ISO/IEC, "ISO/IEC 25019:2023 - Systems and software engineering –Systems and software Quality Requirements and Evaluation (SQuaRE) –Quality-in-use model," ISO/IEC 25019:2023, Nov. 2023. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:25019:ed-1:v1:en

[10] IEEE, "IEEE Standard for System and Software Verification and Validation," IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004), 2012.

[11] ISO/IEC, "ISO/IEC 25010:2023 - Systems and software engineering –Systems and software Quality Requirements and Evaluation (SQuaRE) –Product quality model," ISO/IEC 25010:2023, Nov. 2023. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en

[12] M. Hamburg and G. Mogyorodi, editors, "ISTQB Glossary, v4.3," 2024. [Online]. Available: https://glossary.istqb.org/en_US/search

[13] 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.

[14] D. G. Firesmith, "A Taxonomy of Testing Types," Pittsburgh, PA, USA, 2015. [Online]. Available: https://apps.dtic.mil/sti/pdfs/AD1147163.pdf

[15] I. Kuļešovs, V. Arnicane, G. Arnicans, and J. Borzovs, "Inventory of Testing Ideas and Structuring of Testing Terms," vol. 1, pp. 210–227, Jan. 2013.

[16] S. Sharma, K. Panwar, and R. Garg, "Decision Making Approach for Ranking of Software Testing Techniques Using Euclidean Distance Based Approach," International Journal of Advanced Research in Engineering and Technology, vol. 12, no. 2, pp. 599–608, Feb. 2021. [Online]. Available: https://iaeme.com/Home/issue/IJARET?Volume=12&Issue=2

[17] E. F. Barbosa, E. Y. Nakagawa, and J. C. Maldonado, "Towards the Establishment of an Ontology of Software Testing," vol. 6, San Francisco, CA, USA, Jan. 2006, pp. 522–525.

[18] E. Souza, R. Falbo, and N. Vijaykumar, "ROoST: Reference Ontology on Software Testing," Applied

Ontology, vol. 12, pp. 1–32, Mar. 2017.
[19] M. H. Moghadam, "Machine Learning-Assisted Performance Testing," in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1187–1189. [Online]. Available: https://doi.org/10.1145/3338906.3342484
[20] B. Kam, "Web Applications Testing," Queen's University, Kingston, ON, Canada, Technical Report 2008-550, Oct. 2008. [Online]. Available: https://research.cs.queensu.ca/TechReports/Reports/2008-550.pdf
[21] P. Godefroid and D. Luchaup, "Automatic Partial Loop Summarization in Dynamic Test Generation," in Proceedings of the 2011 International Symposium on Software Testing and Analysis, ser. ISSTA '11. New York, NY, USA: Association for Computing Machinery, Jul. 2011, pp. 23–33. [Online]. Available: https://dl.acm.org/doi/10.1145/2001420.2001424
[22] P. Ammann and J. Offutt, Introduction to Software Testing, 2nd ed. Cambridge, United Kingdom: Cambridge University Press, 2017. [Online]. Available: https://eopcw.com/find/downloadFiles/11
[23] W. E. Perry, Effective Methods for Software Testing, 3rd ed. Indianapolis, IN, USA: Wiley Publishing, Inc., 2006.
[24] P. Gerrard, "Risk-based E-business Testing - Part 1: Risks and Test Strategy," Systeme Evolutif, London, UK, Tech. Rep., 2000. [Online]. Available: https://www.agileconnection.com/sites/default/files/article/file/2013/XUS129342file1_0.pdf
[25] J. Pan, "Software Testing," 1999. [Online]. Available: http://users.ece.cmu.edu/~koopman/des_s99/sw_testing/
[26] E. Engström and K. Petersen, "Mapping software testing practice with software testing research — serp-test taxonomy," in 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2015, pp. 1–4.
[27] G. Tebes, D. Peppino, P. Becker, G. Matturro, M. Solari, and L. Olsina, "A Systematic Review on Software Testing Ontologies," Aug. 2019, pp. 144–160.
[28] G. Tebes, L. Olsina, D. Peppino, and P. Becker, "TestTDO_terms_definitions_vfinal.pdf," Feb. 2020. [Online]. Available: https://drive.google.com/file/d/19TWHd50HF04K6PPyVixQzR6c7HjW2kED/view
[29] ——, "TestTDO: A Top-Domain Software Testing Ontology," Curitiba, Brazil, May 2020, pp. 364–377.
[30] 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, no. 2, pp. 1–38, Mar. 2014, arXiv:2307.12477 [cs]. [Online]. Available: http://arxiv.org/abs/2307.12477