Putting Software Testing Terminology to the Test

M.A.Sc. Defense

Samuel Crawford, B.Eng.

McMaster University
Department of Computing and Software

Fall 2025

Table of Contents

1. Introduction
2. Project
   - Research Questions
   - Methodology
3. Results
4. Next Steps
5. Conclusion

Table of Contents

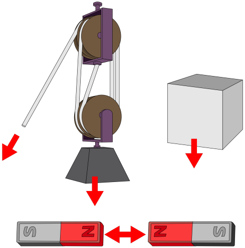
1. Introduction
2. Project
   - Research Questions
   - Methodology
3. Results
4. Next Steps
5. Conclusion

Introduction

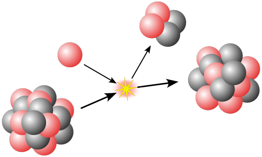
The Need for Standardized Terminology

- Engineering is applied science
- Scientific fields use precise terminology

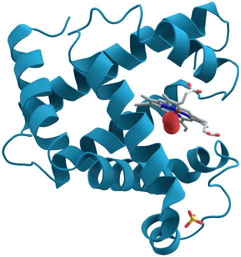
⇒ SOFTWARE ENGINEERING



Penubag and Ramey (2010)



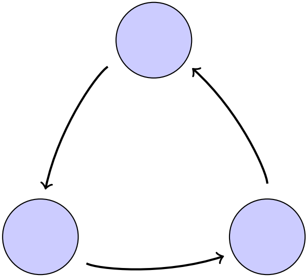
Kjerish (2016)



AzaToth (2008)

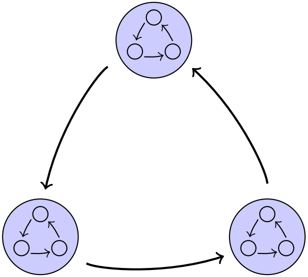
Interorganizational

Schools, companies, etc.



Interorganizational

Schools, companies, etc.



Intraorganizational

“Complete testing” could require the tester to:

- discover every bug,
- exhaust the time allocated,
- implement every planned test,
- . . . (Kaner et al., 2011, p. 7)

Introduction

Taxonomies to the Rescue?

- Existing software testing taxonomies:
    - Tebes et al. (2020)
    - Souza et al. (2017)
    - Firesmith (2015)
    - Unterkalmsteiner et al. (2014)

Focus on:

The Testing Process
Organizing Terminology
Relations between Approaches
Traceability between Stages

Table of Contents

1. Introduction
2. Project
   - Research Questions
   - Methodology
3. Results
4. Next Steps
5. Conclusion

Research Questions

Research Question 1

What test approaches do the literature describe?

Research Question 2

How consistent are these descriptions?

Research Question 3

Can we systematically resolve any of these inconsistencies?

Methodology

Procedure

Research Question 1

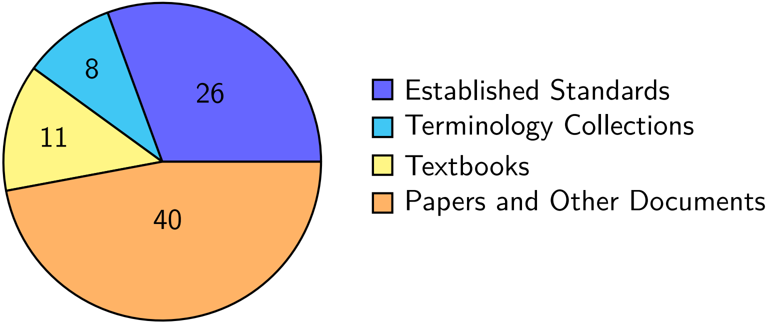
What test approaches do the literature describe?

1. Identify authoritative sources on software testing
2. Identify all test approaches and testing-related terms
3. Record data for these terms; test approach data are comprised of:
    1. Names
    2. Categories
    3. Definitions
    4. Synonyms
    5. Parents
    6. Flaws
4. Repeat steps 1 to 3 for any missing or unclear terms

Methodology

Sources

In total, we investigated 85 sources

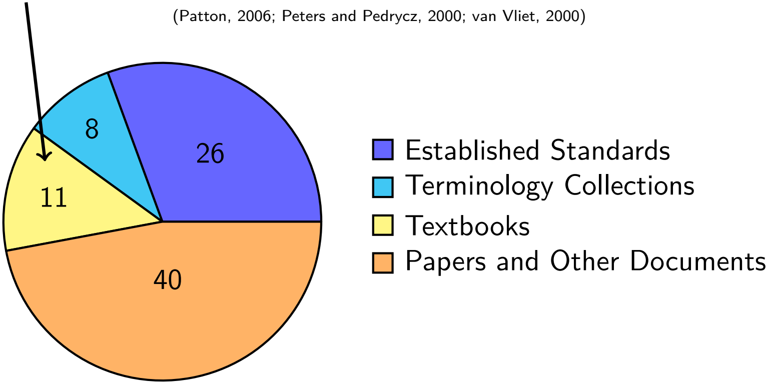


Methodology

Sources

Textbooks used at McMaster were our ad hoc starting points

(Patton, 2006; Peters and Pedrycz, 2000; van Vliet, 2000)



Methodology

Terms

- We built a glossary with a row for each test approach

Name	Category	Definition	Parent(s)	Synonym(s)
A/B Testing	Practice (Fig. 2)	Testing "that allows testers to determine which of two systems or components performs better" (pp. 1, 36)	Statistical Testing (pp. 1, 36), . . .	Split-Run Testing (pp. 1, 36)

Information from (ISO/IEC and IEEE, 2022)

- We gathered this information from sources by looking for:
  - Glossaries, taxonomies, hierarchies, etc.
  - Testing-related terms
  - Terms described *by* other approaches
  - Terms that *imply* other approaches

Methodology

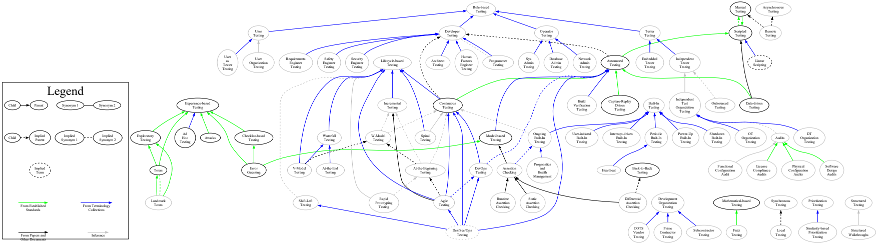
Procedure

Research Question 2

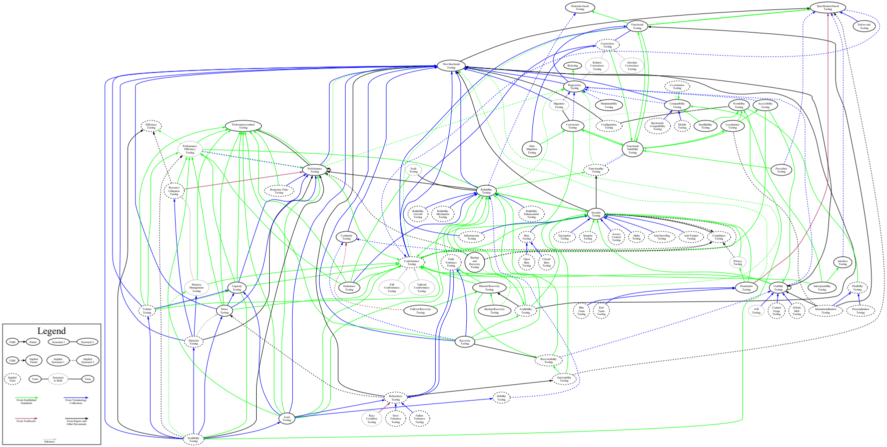
How consistent are these descriptions?

5. Automatically analyze recorded test approach data
   1. Visualize approach relations
   2. Detect certain classes of flaws
   3. Analyze manually recorded flaws from step 3.6
6. Report results of flaw analysis

Visualization of Test Practices



Visualization of Test Types



Visualization of Performance-related Testing

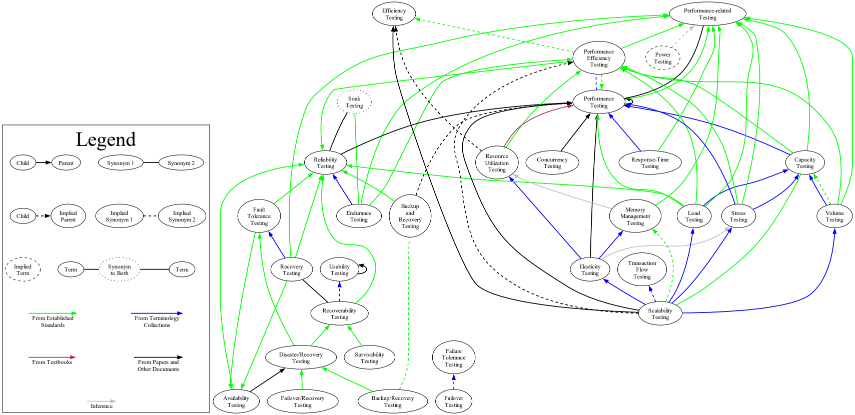


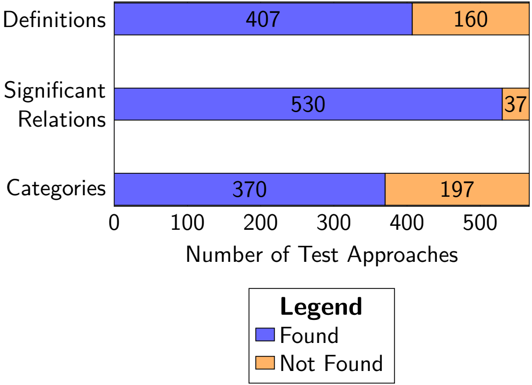
Table of Contents

1. Introduction
2. Project
   - Research Questions
   - Methodology
3. Results
4. Next Steps
5. Conclusion

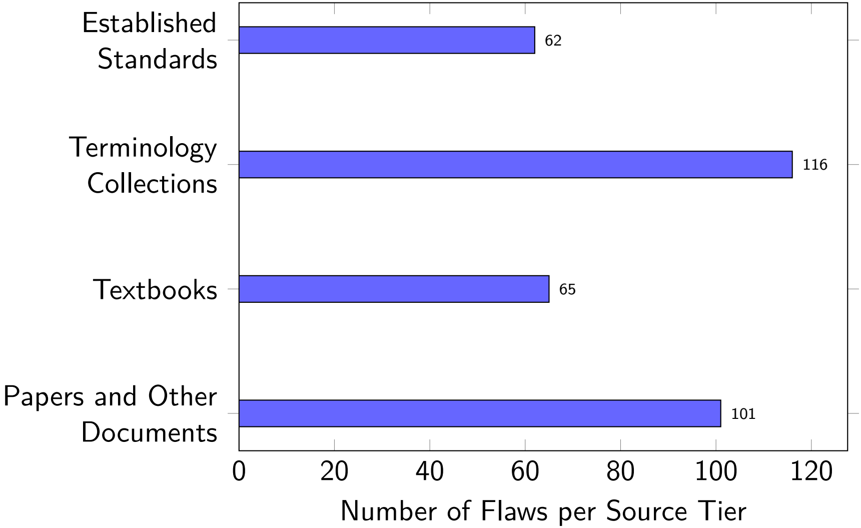
Overview

- 567 test approaches →
- 75 software qualities

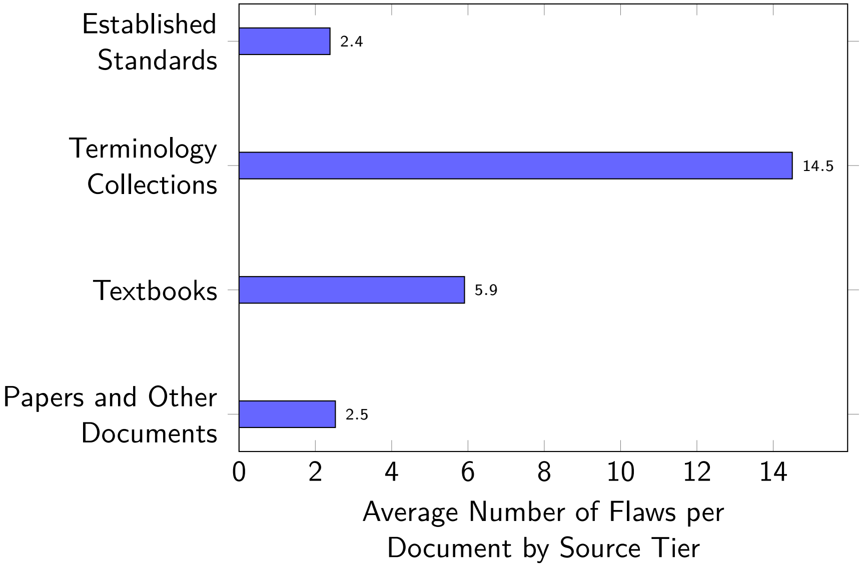
  (may imply test approaches)
- 344 flaws in the software testing literature



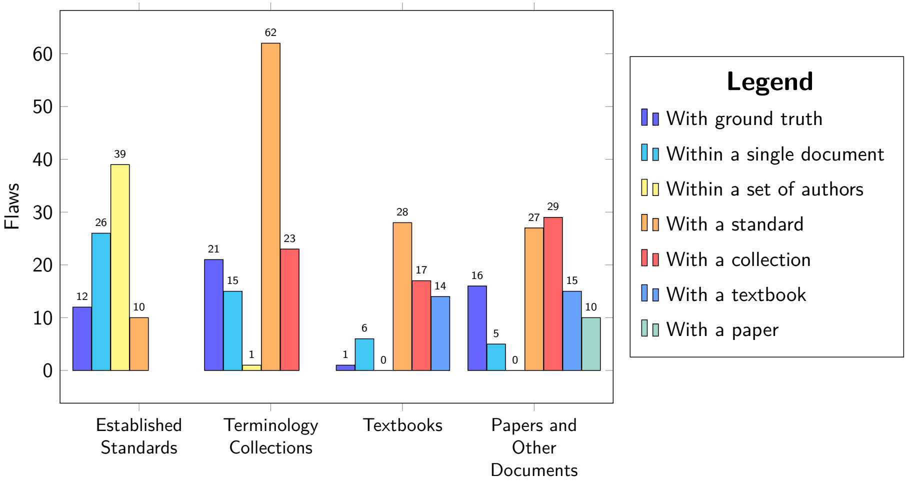
Flaw Summary by Source Tier



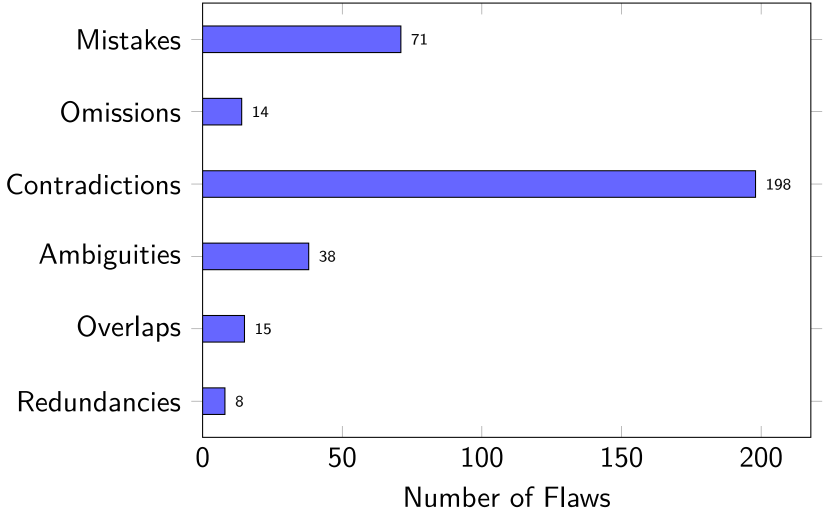
Normalized Flaw Summary



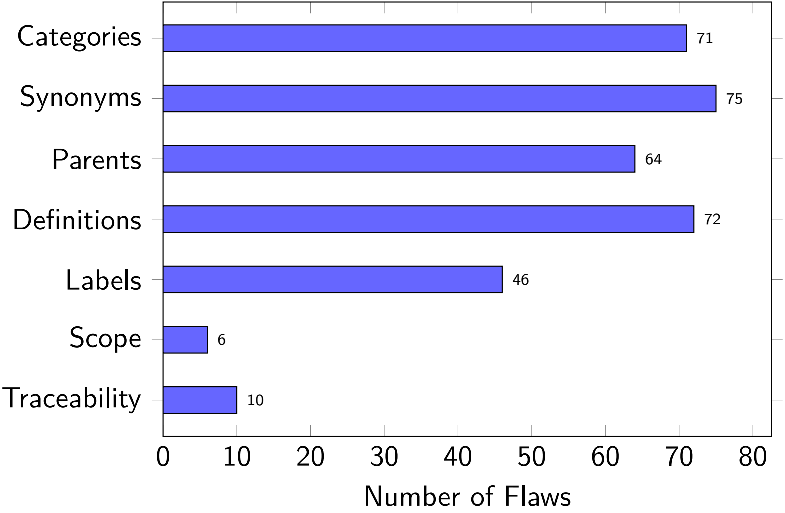
More Detailed Flaw Summary by Source Tier



Flaw Summary by Manifestation



Flaw Summary by Domain



General Flaw Observations

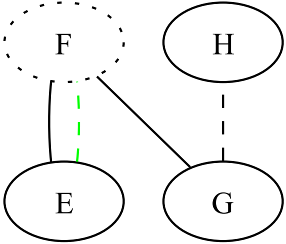
1. Contradictions are the most common manifestation, likely due to our automation and inconsistencies between (sets of) authors
2. Approach categories are the most subjective and one of the most common domains
3. Semantic flaws are more common than syntatic ones

Automated Flaws

Intransitive Synonyms

Some terms are given as a synonym to two (or more) disjoint terms, making their relations ambiguous

Name	Synonym(s)
E	F (Author, 2022; implied by StdAuthor, 2021)
G	F (Author, 2017), H (implied by 2022)
H	X (StdAuthor, 2021)



Automated Flaws

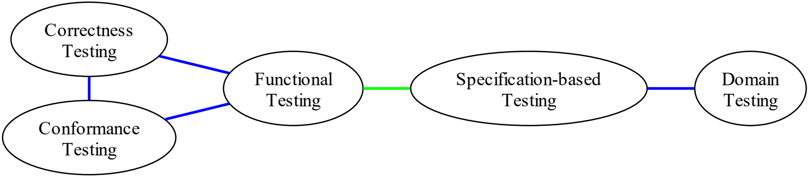
Intransitive Synonyms

Functional testing:

- Specification-based testing (ISO/IEC and IEEE, 2017, p. 196; van Vliet, 2000, p. 399; . . .)
- Conformance testing *and* correctness testing (Washizaki, 2025, p. 5-7)

Specification-based testing:

- Functional testing (ISO/IEC and IEEE, 2017, p. 196; van Vliet, 2000, p. 399; Kam, 2008, pp. 44–45, 48)
- Domain testing (Washizaki, 2025, p. 5-10)



Automated Flaws

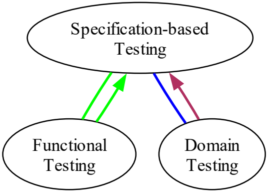
Irreflexive Parents

We also found some test approaches that are given as parents of themselves:

1. Performance testing (Gerrard, 2000a, Tab. 2; 2000b, Tab. 1)
2. System testing (Firesmith, 2015, p. 23)
3. Usability testing (Gerrard, 2000a, Tab. 2; 2000b, Tab. 1)

Automated Flaws

Synonym and Parent-Child Overlaps



- Functional testing is a:
  - Synonym (ISO/IEC and IEEE, 2017, p. 196; van Vliet, 2000, p. 399; Kam, 2008, pp. 44–45, 48)
  - Child (ISO/IEC and IEEE, 2021c, p. 38; Kam, 2008, p. 42)
- Domain testing is a:
  - Synonym (Washizaki, 2025, p. 5-10)
  - Child (Peters and Pedrycz, 2000, Tab. 12.1)

Table of Contents

1. Introduction
2. Project
   - Research Questions
   - Methodology
3. Results
4. Next Steps
5. Conclusion

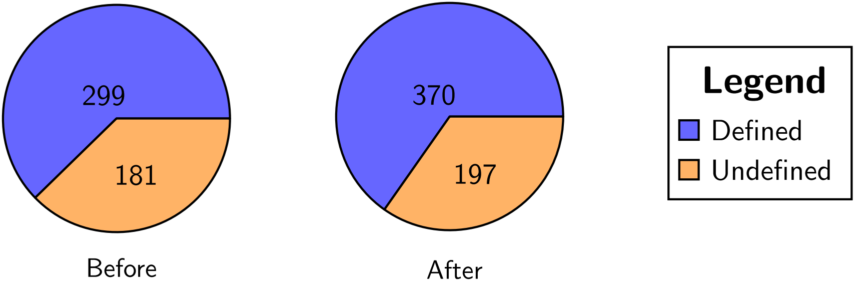
Next Steps

Threats to Validity

1. Performed by a single researcher over a period of time
2. Terms we defined: categories, flaws, . . .
3. Derived approaches: test types from qualities, test techniques from coverage metrics, . . .
4. Limits and enforcement of standards

Next Steps

Future Work



1. Iterate over undefined test approaches
2. Investigate missing approaches
3. Fill in other approach data
4. Improve approach relation visualizations
5. Identify and detect more flaws

Next Steps

Future Work

Research Question 3

Can we systematically resolve any of these inconsistencies?

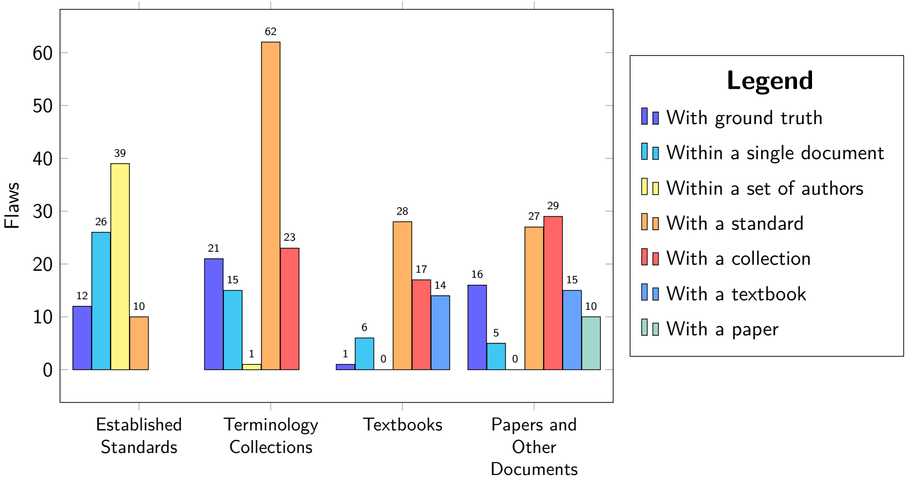
- It will be more effective to do this more systematically once the previous tasks are finished
- Our glossaries and tools for analyzing them provide a solid foundation for this task on which future researchers can build

Table of Contents

1. Introduction
2. Project
   - Research Questions
   - Methodology
3. Results
4. Next Steps
5. Conclusion

Conclusion

The software testing literature is flawed, so don’t assume everyone is on the same page!



Acknowledgment

- Dr. Spencer Smith and Dr. Jacques Carette have been great supervisors and valuable sources of guidance and feedback
- The format of this presentation was *heavily* based on a previous presentation by Jason Balaci, who also provided a great thesis template
- My family has also supported me in more ways than I can count, and I cannot thank them enough

- ChatGPT was used to help generate supplementary Python code for constructing visualizations and generating LaTeX code, including regex
- ChatGPT and GitHub Copilot were both used for assistance with LaTeX formatting

References I

AzaToth. Myoglobin 3D structure, February 2008. URL https://commons.wikimedia.org/wiki/File:Myoglobin.png.

Donald G. Firesmith. A Taxonomy of Testing Types, 2015. URL https://apps.dtic.mil/sti/pdfs/AD1147163.pdf.

Paul Gerrard. Risk-based E-business Testing - Part 1: Risks and Test Strategy. Technical report, Systeme Evolutif, London, UK, 2000a. URL https://www.agileconnection.com/sites/default/files/article/file/2013/XUS129342file1_0.pdf.

Paul Gerrard. Risk-based E-business Testing - Part 2: Test Techniques and Tools. Technical report, Systeme Evolutif, London, UK, 2000b. URL wenku.uml.com.cn/document/test/EBTestingPart2.pdf.

ISO/IEC and IEEE. ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary. *ISO/IEC/IEEE 24765:2017(E)*, September 2017. doi: 10.1109/IEEESTD.2017.8016712.

References II

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)*, October 2021c. doi: 10.1109/IEEESTD.2021.9591574.

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)*, January 2022. doi: 10.1109/IEEESTD.2022.9698145.

Ben Kam. Web Applications Testing. Technical Report 2008-550, Queen's University, Kingston, ON, Canada, October 2008. URL `https://research.cs.queensu.ca/TechReports/Reports/2008-550.pdf`.

References III

Cem Kaner, James Bach, and Bret Pettichord. *Lessons Learned in Software Testing: A Context-Driven Approach*. John Wiley & Sons, December 2011. ISBN 978-0-471-08112-8. URL https://www.wiley.com/en-ca/Lessons+Learned+in+Software+Testing%3A+A+Context-Driven+Approach-p-9780471081128.

Kjerish. Part of CNO cycle diagram, made just to be illustrative for nuclear reactions in general, December 2016. URL https://commons.wikimedia.org/wiki/File:NuclearReaction.svg.

Ron Patton. *Software Testing*. Sams Publishing, Indianapolis, IN, USA, 2nd edition, 2006. ISBN 0-672-32798-8.

Penubag and Arnaud Ramey. A few images illustrating forces, August 2010. URL https://commons.wikimedia.org/wiki/File:Force_examples.svg.

References IV

J.F. Peters and W. Pedrycz. *Software Engineering: An Engineering Approach*. Worldwide series in computer science. John Wiley & Sons, Ltd., 2000. ISBN 978-0-471-18964-0.

Erica Souza, Ricardo Falbo, and Nandamudi Vijaykumar. ROoST: Reference Ontology on Software Testing. *Applied Ontology*, 12:1–32, March 2017. doi: 10.3233/AO-170177.

Guido Tebes, Luis Olsina, Denis Peppino, and Pablo Becker. TestTDO: A Top-Domain Software Testing Ontology. pages 364–377, Curitiba, Brazil, May 2020. ISBN 978-1-71381-853-3.

Michael Unterkalmsteiner, Robert Feldt, and Tony Gorschek. A Taxonomy for Requirements Engineering and Software Test Alignment. *ACM Transactions on Software Engineering and Methodology*, 23(2):1–38, March 2014. ISSN 1049-331X, 1557-7392. doi: 10.1145/2523088. URL http://arxiv.org/abs/2307.12477. arXiv:2307.12477 [cs].

References V

Hans van Vliet. *Software Engineering: Principles and Practice*. John Wiley & Sons, Ltd., Chichester, England, 2nd edition, 2000. ISBN 0-471-97508-7.

Hironori Washizaki, editor. *Guide to the Software Engineering Body of Knowledge, Version 4.0a*. May 2025. URL https://ieeecs-media.computer.org/media/education/swebok/swebok-v4.pdf.