

# Putting Software Testing Terminology to the Test

## M.A.Sc. Seminar

Samuel Crawford, B.Eng.

McMaster University  
Department of Computing and Software

Fall 2025

# Table of Contents

## 1 Introduction

- The Need for Standardized Terminology
- The Lack of Standardized Terminology

## 2 Project

- Research Questions
- Methodology

## 3 Results

## 4 Future Work

## 5 Conclusion

# Table of Contents

## 1 Introduction

- The Need for Standardized Terminology
- The Lack of Standardized Terminology

## 2 Project

- Research Questions
- Methodology

## 3 Results

## 4 Future Work

## 5 Conclusion

# The Need for Standardized Terminology

- Engineering is applied science
- Scientific fields use precise terminology



SOFTWARE  
ENGINEERING

# The Need for Standardized Terminology

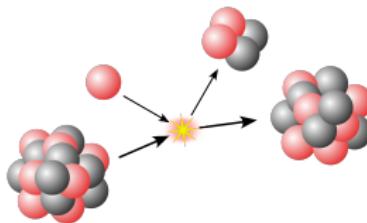
- Engineering is applied science
- Scientific fields use precise terminology



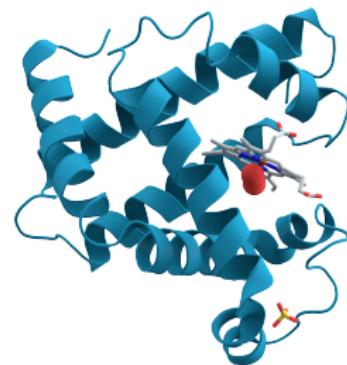
SOFTWARE  
ENGINEERING



Penubag and Ramey (2010)



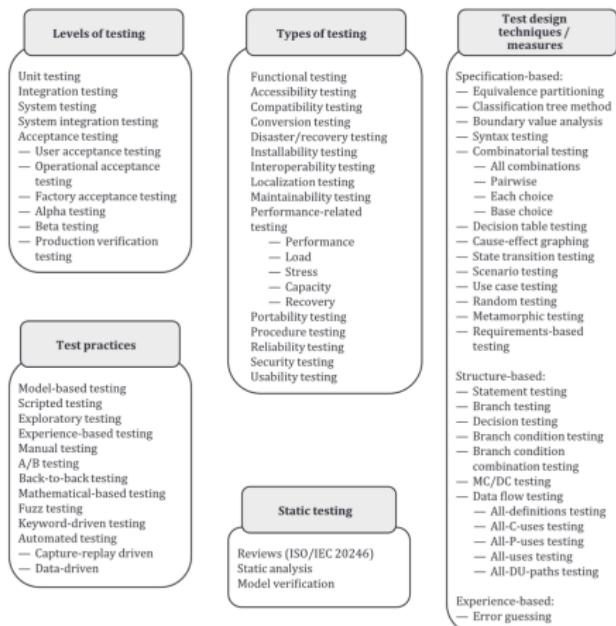
Kjerish (2016)



AzaToth (2008)

# The Lack of Standardized Terminology

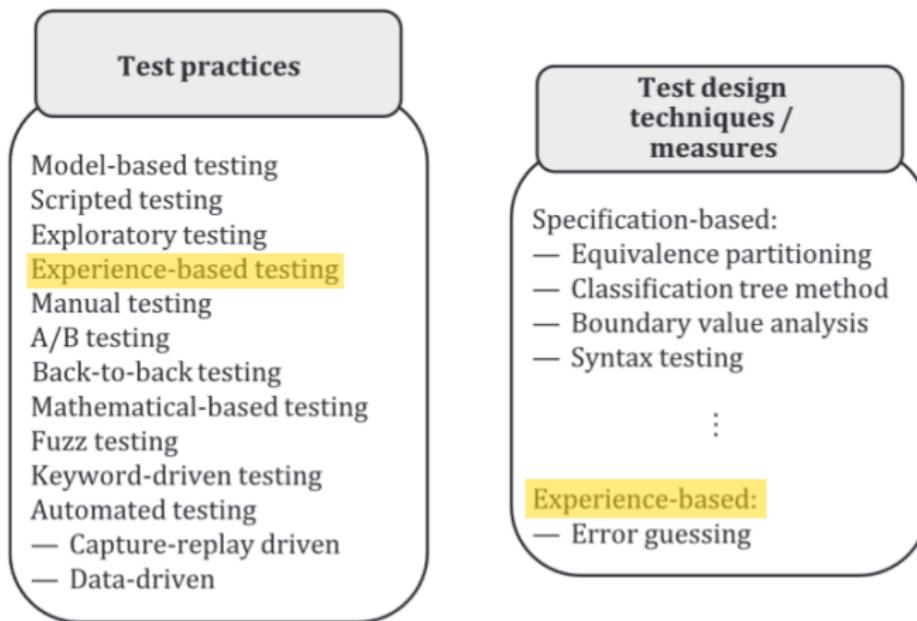
## "The Problem"



(ISO/IEC and IEEE, 2022, Fig. 2)

# The Lack of Standardized Terminology

## "The Problem"

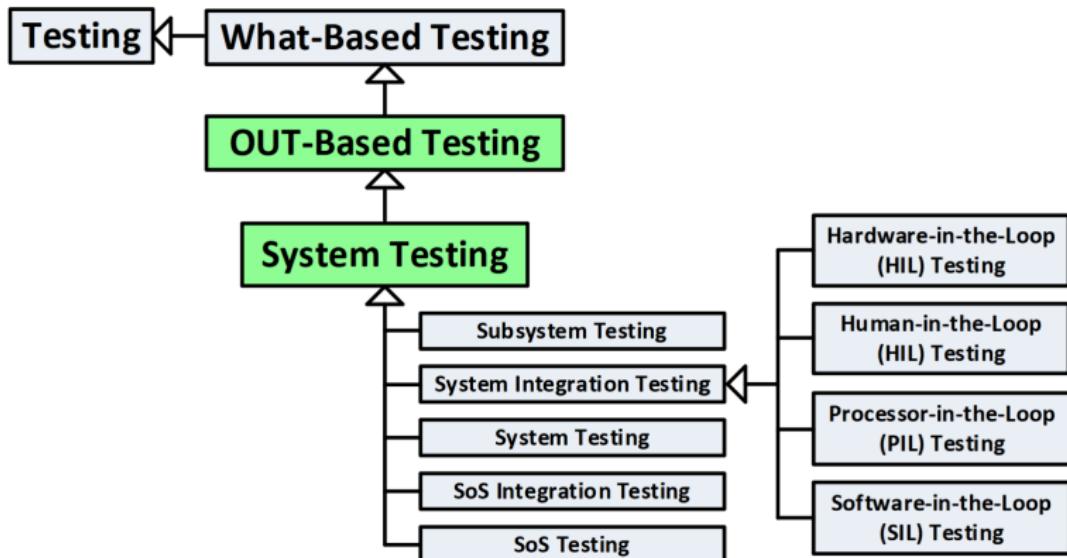


Adapted from (ISO/IEC and IEEE, 2022, Fig. 2)

# The Lack of Standardized Terminology

“The Problem” (cont.)

## What: by Object Under Test (OUT) – System Testing

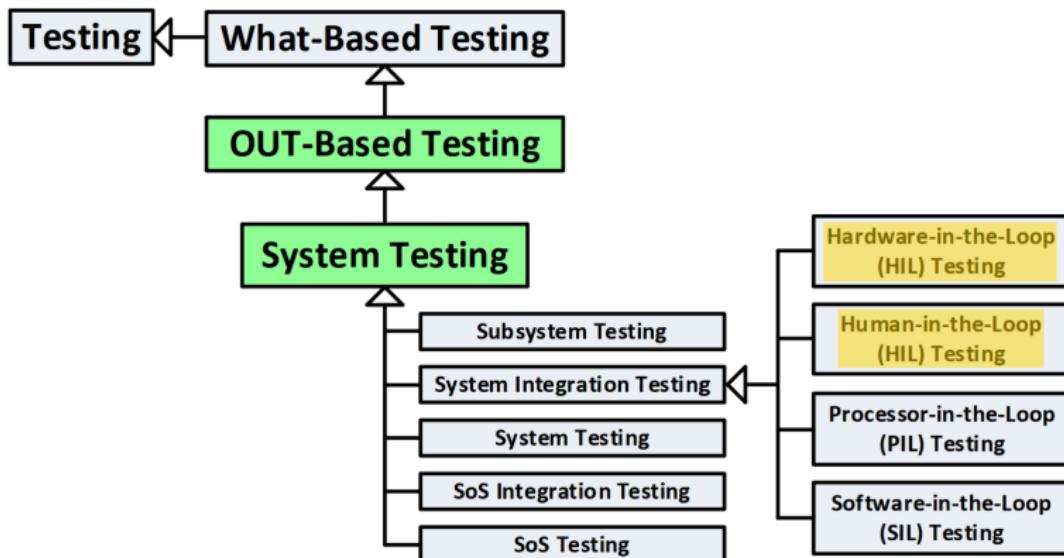


(Firesmith, 2015, p. 23)

# The Lack of Standardized Terminology

“The Problem” (cont.)

## What: by Object Under Test (OUT) – System Testing

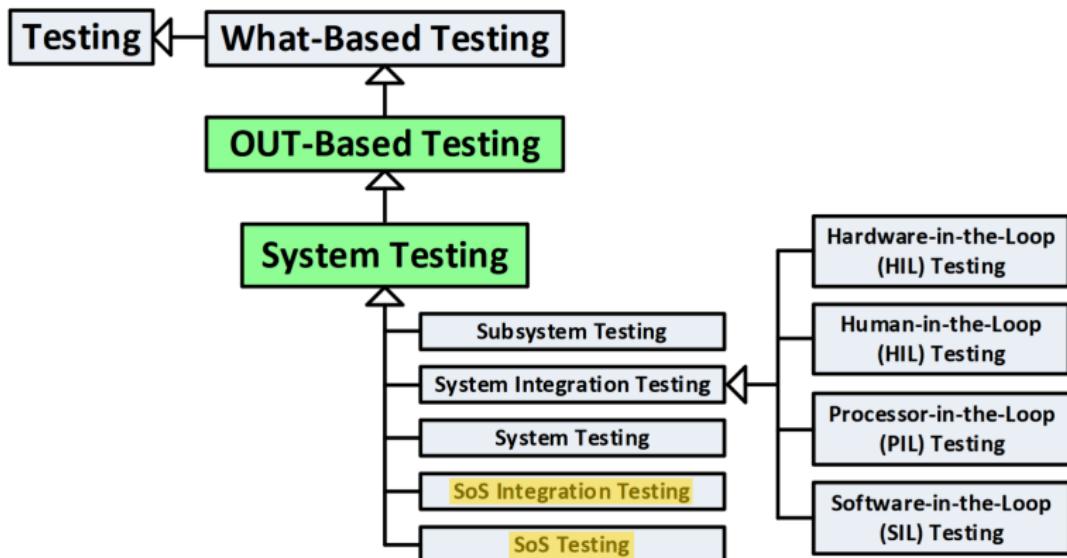


Adapted from (Firesmith, 2015, p. 23)

# The Lack of Standardized Terminology

“The Problem” (cont.)

## What: by Object Under Test (OUT) – System Testing

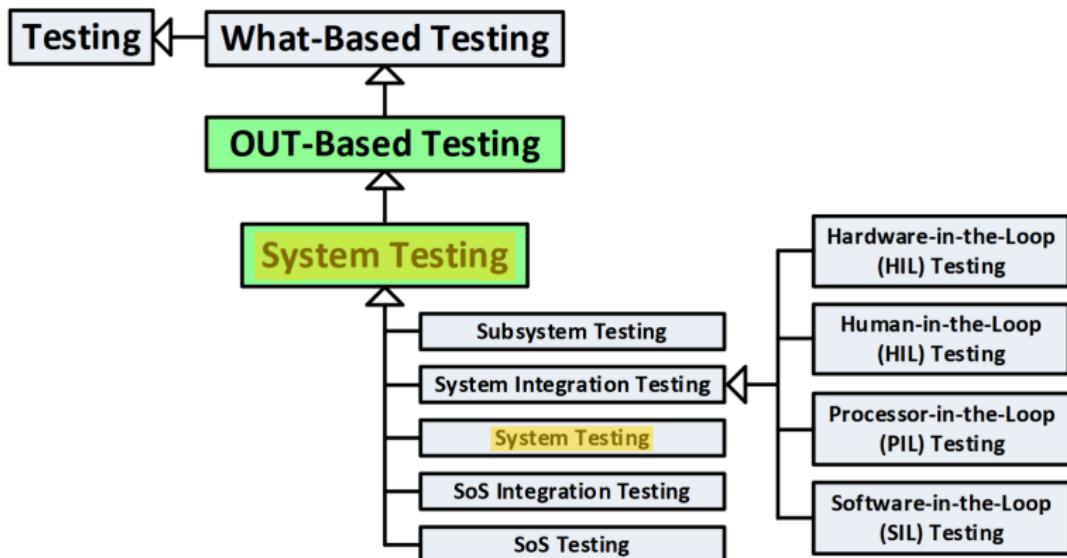


Adapted from (Firesmith, 2015, p. 23)

# The Lack of Standardized Terminology

“The Problem” (cont.)

## What: by Object Under Test (OUT) – System Testing



Adapted from (Firesmith, 2015, p. 23)

# The Lack of Standardized Terminology

“The Problem” (cont.)

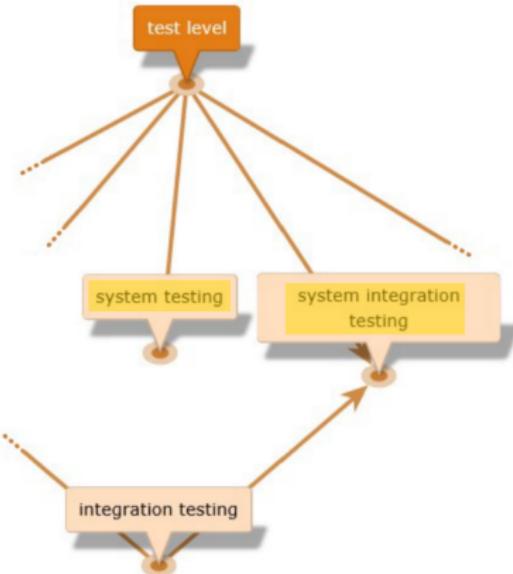
## What: by Object Under Test (OUT) – System Testing



Adapted from (Firesmith, 2015, p. 23)

# The Lack of Standardized Terminology

“The Problem” (cont.)

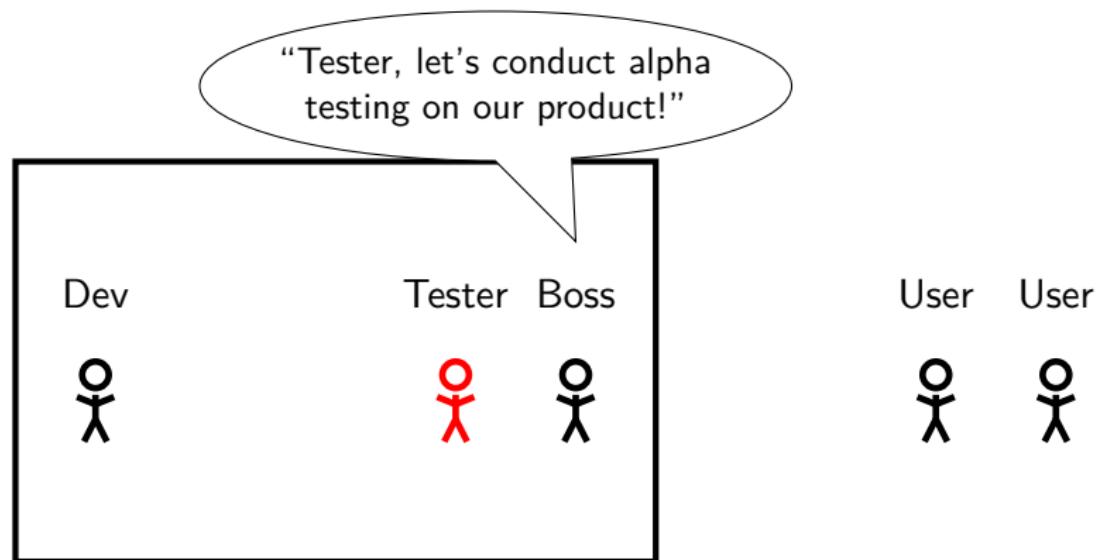


Adapted from (Hamburg and Mogyorodi, 2024)

Adapted from (Firesmith, 2015, p. 23)

# The Lack of Standardized Terminology

“The Problem” (cont.)

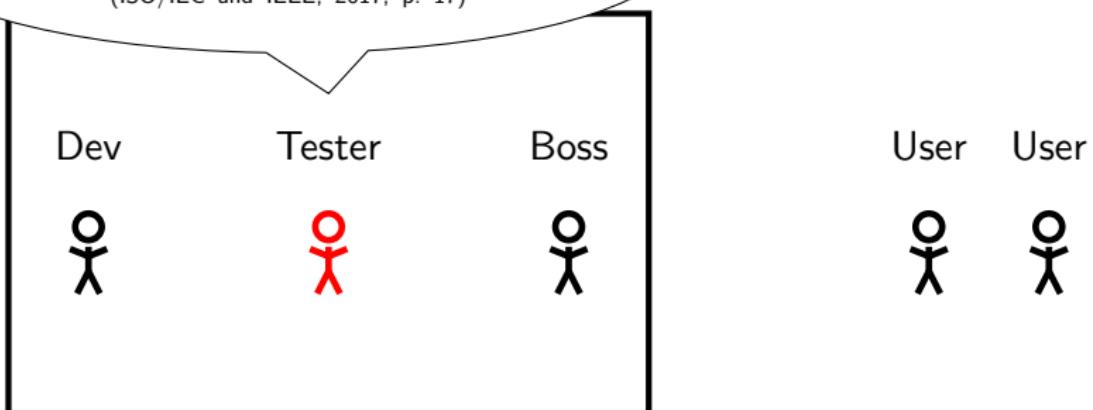


# The Lack of Standardized Terminology

## "The Problem" (cont.)

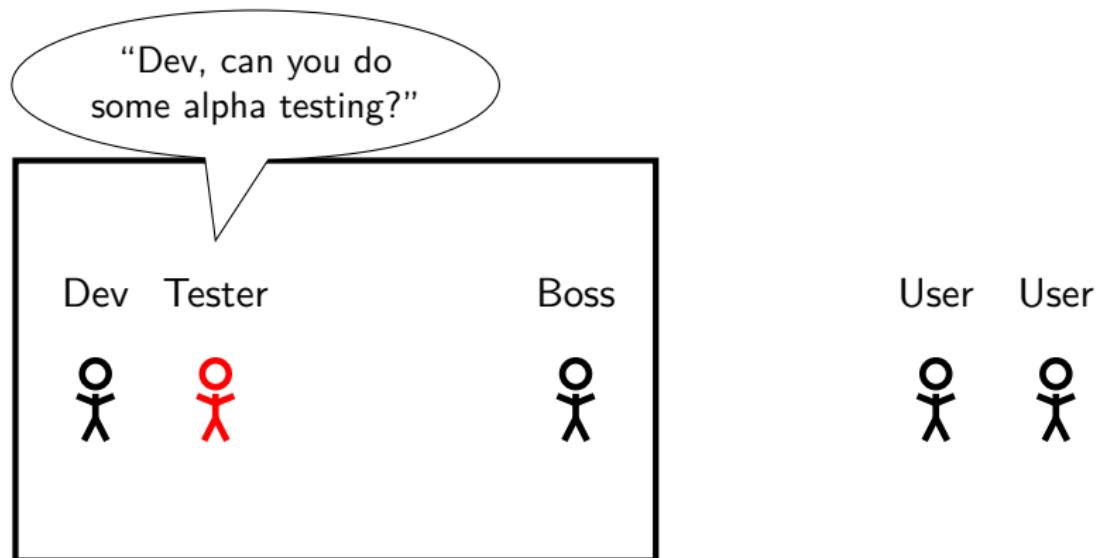
"Alpha testing is done by 'users within the organization developing the software'."

(ISO/IEC and IEEE, 2017, p. 17)



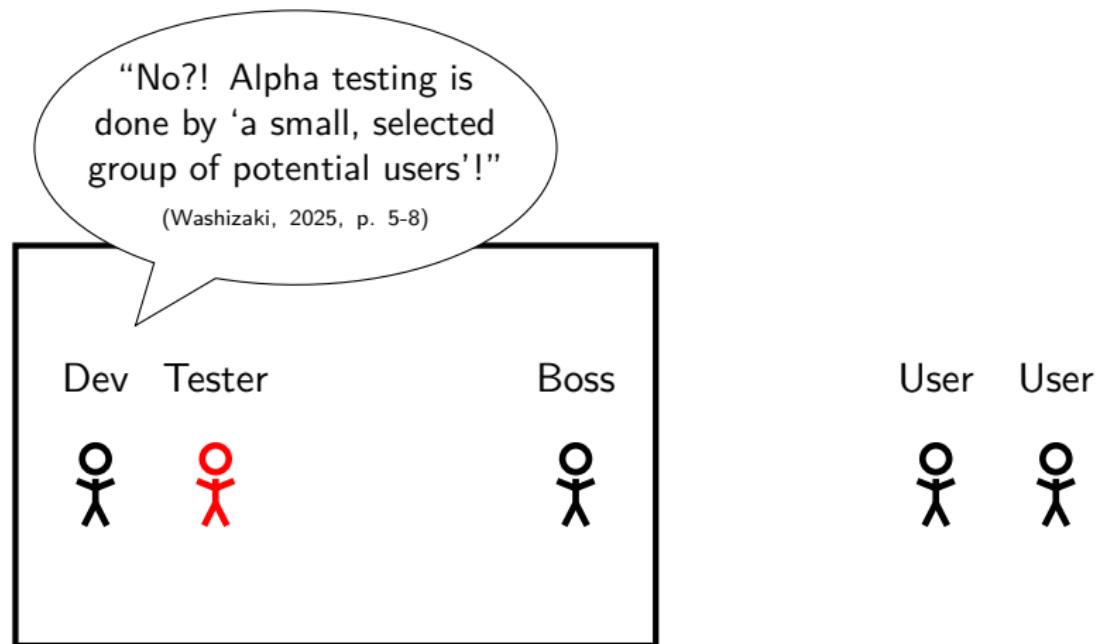
# The Lack of Standardized Terminology

## "The Problem" (cont.)



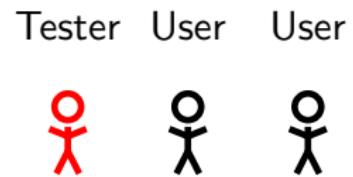
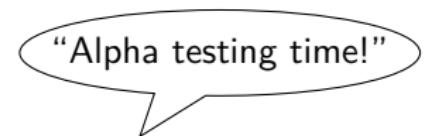
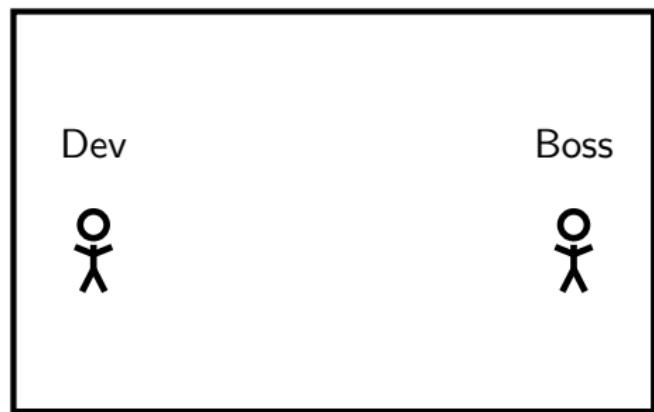
# The Lack of Standardized Terminology

## "The Problem" (cont.)



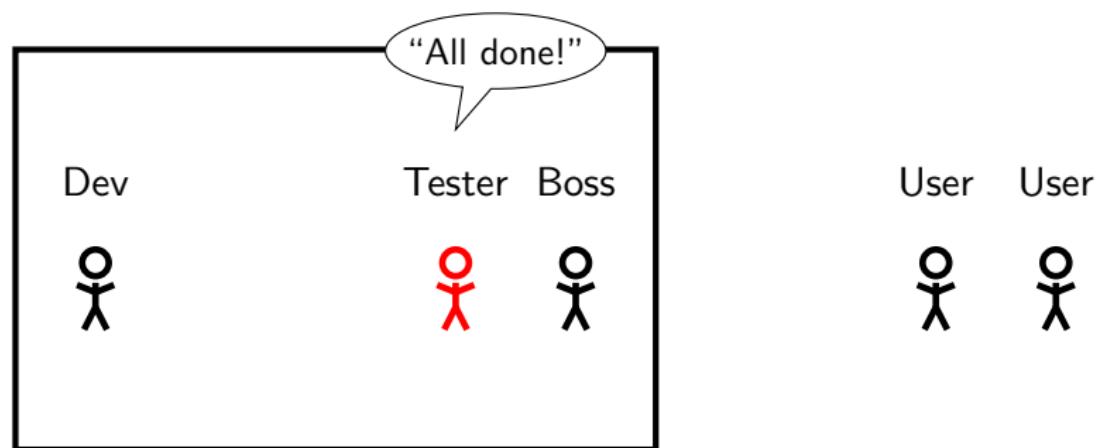
# The Lack of Standardized Terminology

“The Problem” (cont.)



# The Lack of Standardized Terminology

“The Problem” (cont.)

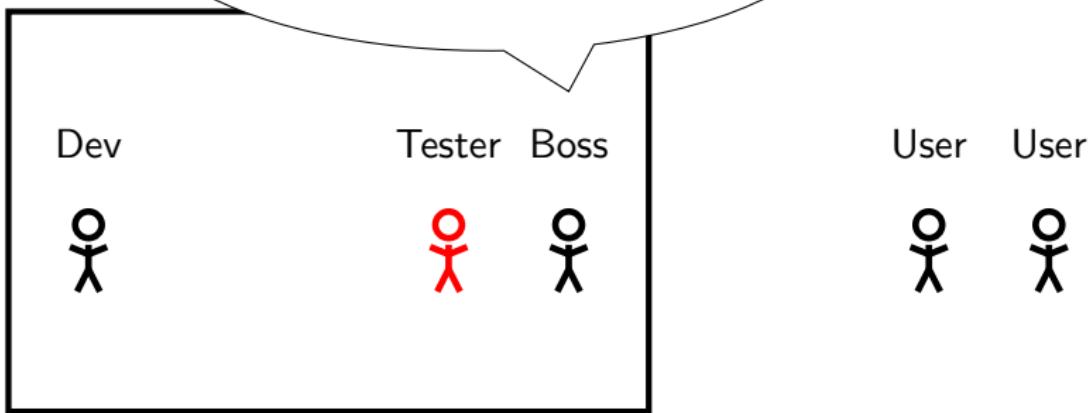


# The Lack of Standardized Terminology

"The Problem" (cont.)

"How? Alpha testing is performed  
'in the developer's test environment',  
but you didn't bring anyone in."

(Hamburg and Mogyorodi, 2024)



# Barriers to Effective Communication

“The Problem” (cont.)

## Interorganizational

Schools, companies, etc.

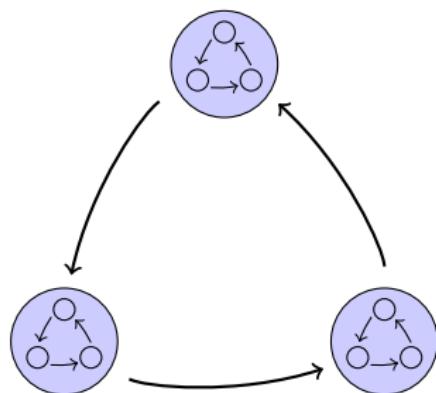


# Barriers to Effective Communication

“The Problem” (cont.)

## 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)

# Taxonomies to the Rescue?

## "The Problem" (cont.)

- 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

- The Need for Standardized Terminology
- The Lack of Standardized Terminology

## 2 Project

- Research Questions
- Methodology

## 3 Results

## 4 Future Work

## 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?

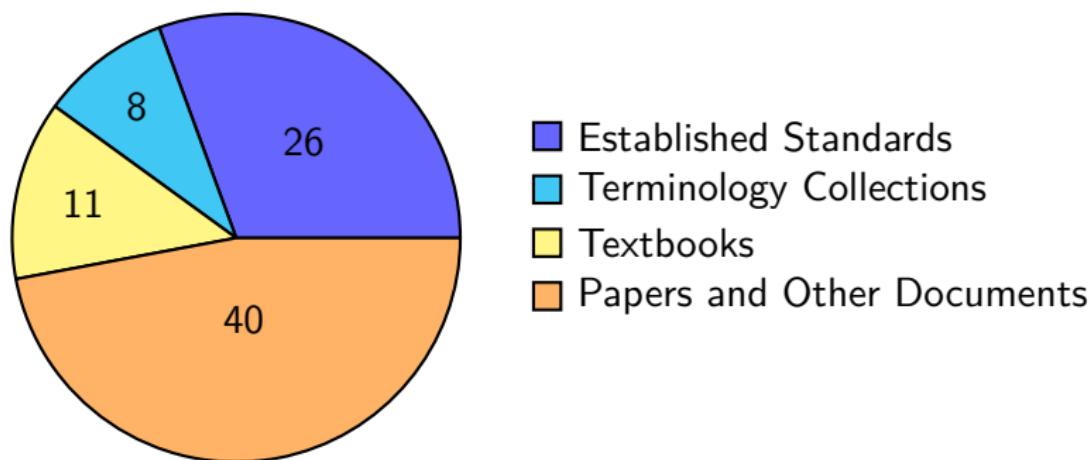
- ① Identify authoritative sources on software testing
- ② Identify all test approaches and testing-related terms
- ③ Record data for these terms; test approach data are comprised of:

① Names	③ Definitions	⑤ Parents
② Categories	④ Synonyms	⑥ Flaws
- ④ Repeat steps 1 to 3 for any missing or unclear terms

# Methodology

## Sources

In total, we investigate 85 sources

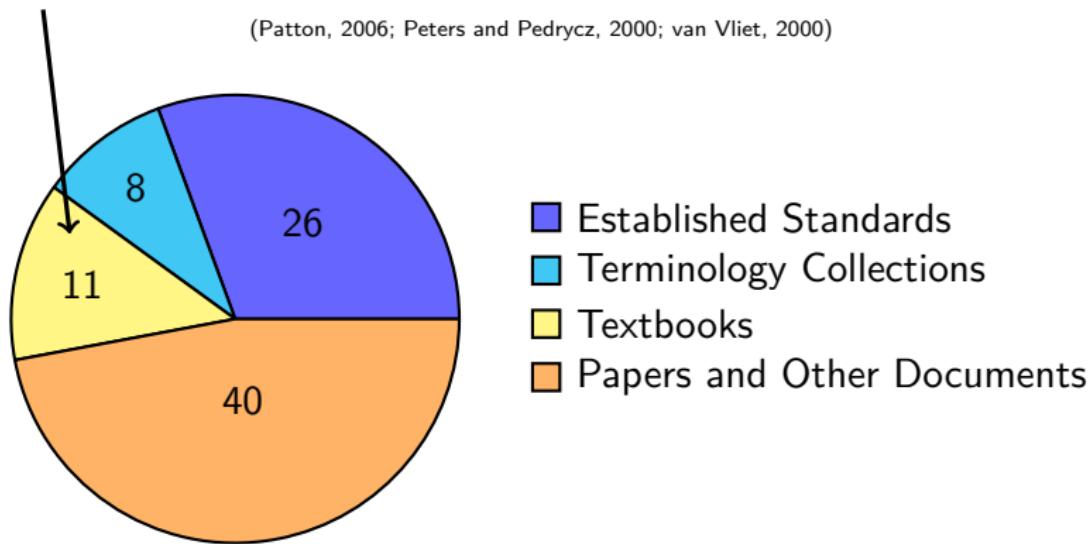


# Methodology

## Sources

Textbooks used at McMaster were our ad hoc starting points

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



# Methodology

## Terms

- We build 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 gather 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

### Research Question 3

Can we systematically resolve any of these inconsistencies?

- ⑦ Provide examples of how to resolve these flaws

# Methodology

## Categories

Approach

**Approach:** a “high-level test implementation choice” (ISO/IEC and IEEE, 2022, p. 10) used to “pick the particular test case values” (2017, p. 465)

# Methodology

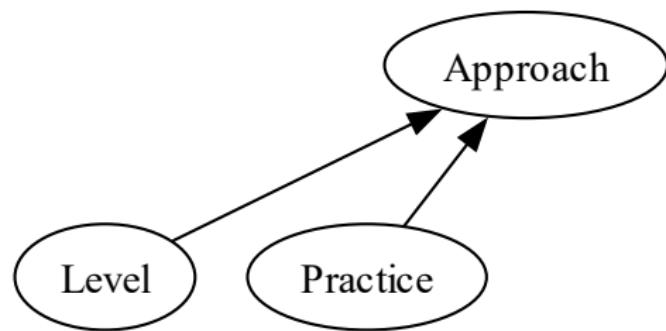
## Categories



**Level:** a stage of testing with “particular objectives and . . . risks”, each performed in sequence (ISO/IEC and IEEE, 2022, p. 12; 2021a, p. 6; 2021c, p. 6)

# Methodology

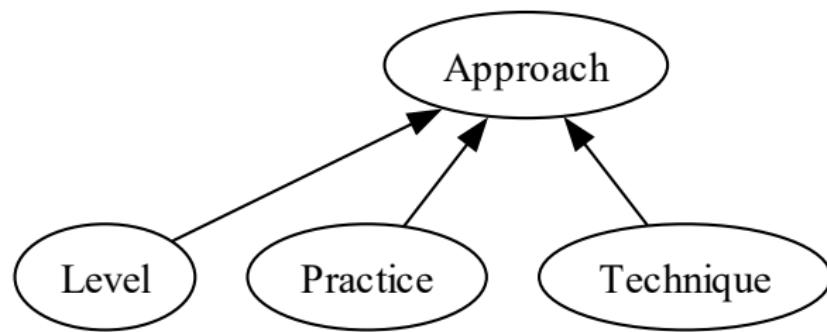
## Categories



**Practice:** a “conceptual framework that can be applied to . . . [a] test process to facilitate testing” (ISO/IEC and IEEE, 2022, p. 14; 2017, p. 471)

# Methodology

## Categories



**Technique:** a “procedure used to create or select a test model, identify test coverage items, and derive corresponding test cases” (2022, p. 11; 2021a, p. 5; similar in 2017, p. 467)

# Methodology

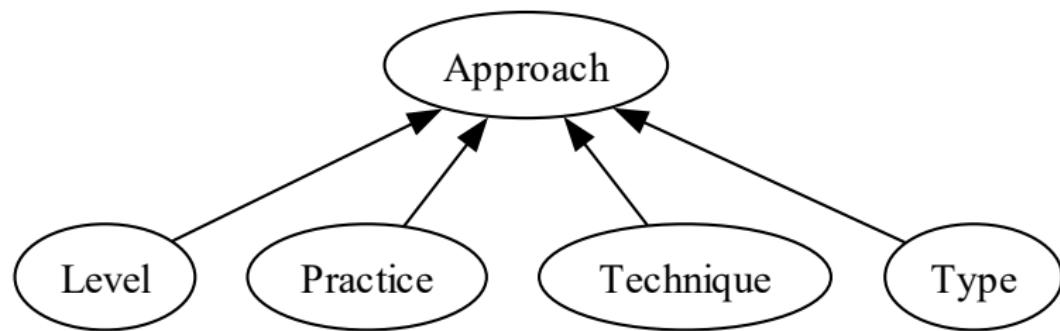
## Categories



**Type:** “testing that is focused on specific quality characteristics”  
(ISO/IEC and IEEE, 2022, p. 15; 2021c, p. 7; 2017, p. 473)

# Methodology

## Visualization Notation



Arrows point from a *child* node to a *parent* node.

# Methodology

## Visualization Notation



Lines without arrowheads connect *synonyms*.

# Methodology

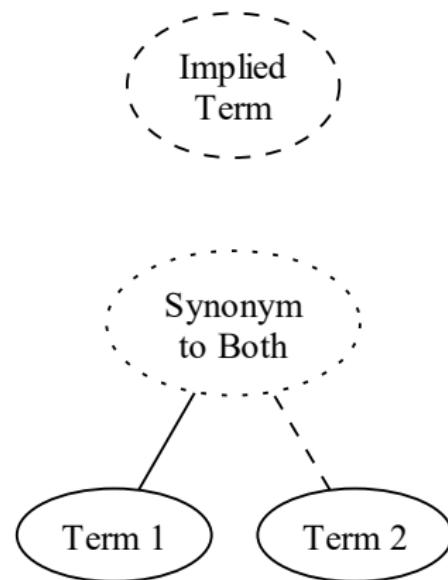
## Visualization Notation



Dashed lines indicate a relationship is *implicit*.

# Methodology

## Visualization Notation



Dashed outlines indicate a term is *implicit*.

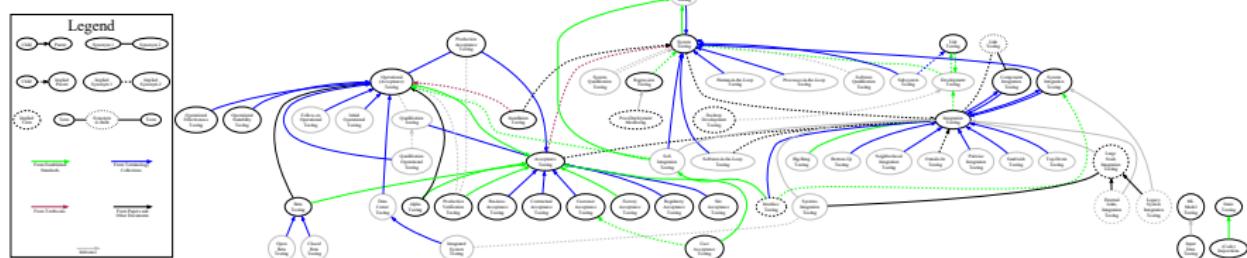
Dotted outlines indicate a term is a *synonym* to more than one term.

# Visualization of Test Approaches

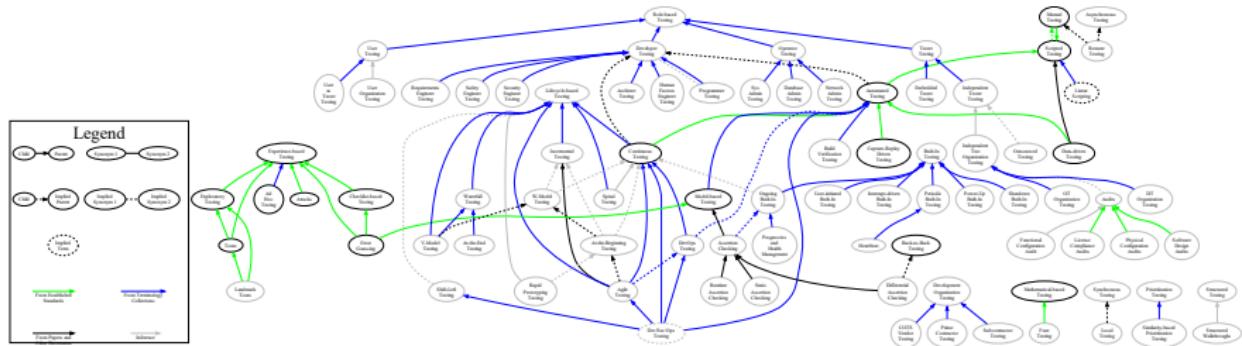
# Visualization of Test Approaches

! Dimension too large.

# Visualization of Test Levels



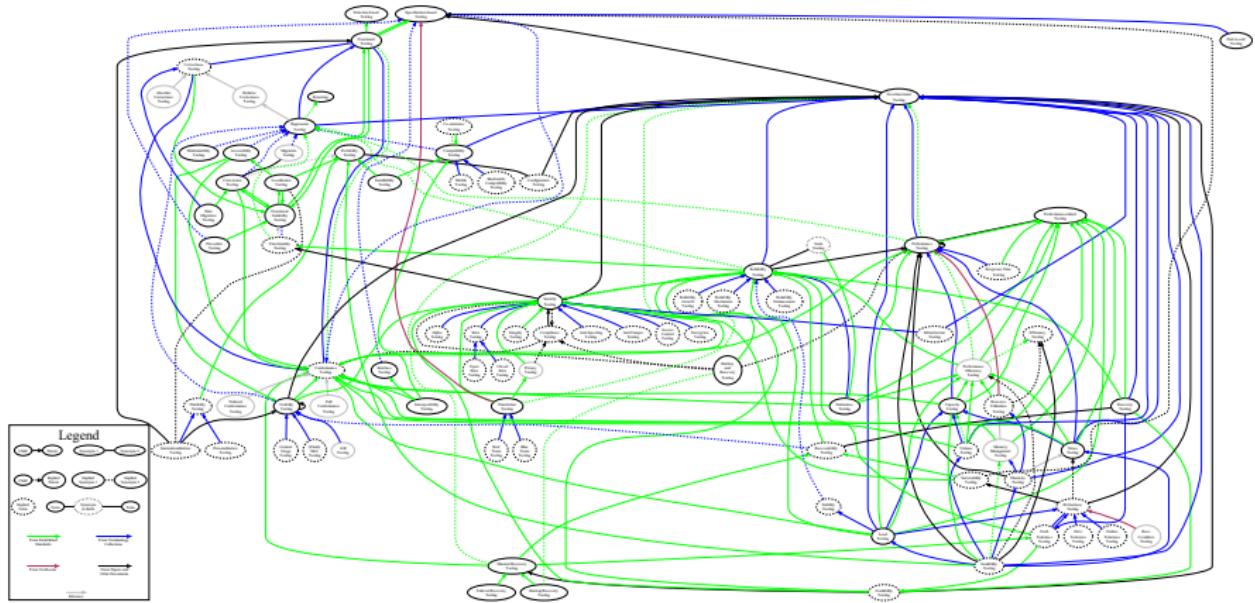
# Visualization of Test Practices



# Visualization of Test Techniques



# Visualization of Test Types



# Table of Contents

## 1 Introduction

- The Need for Standardized Terminology
- The Lack of Standardized Terminology

## 2 Project

- Research Questions
- Methodology

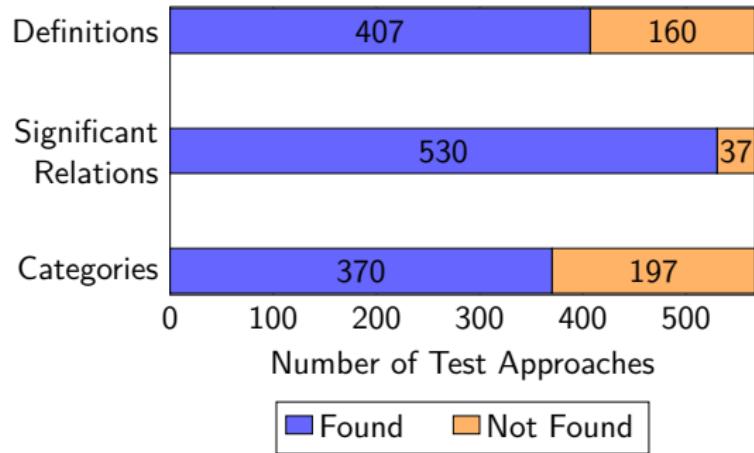
## 3 Results

## 4 Future Work

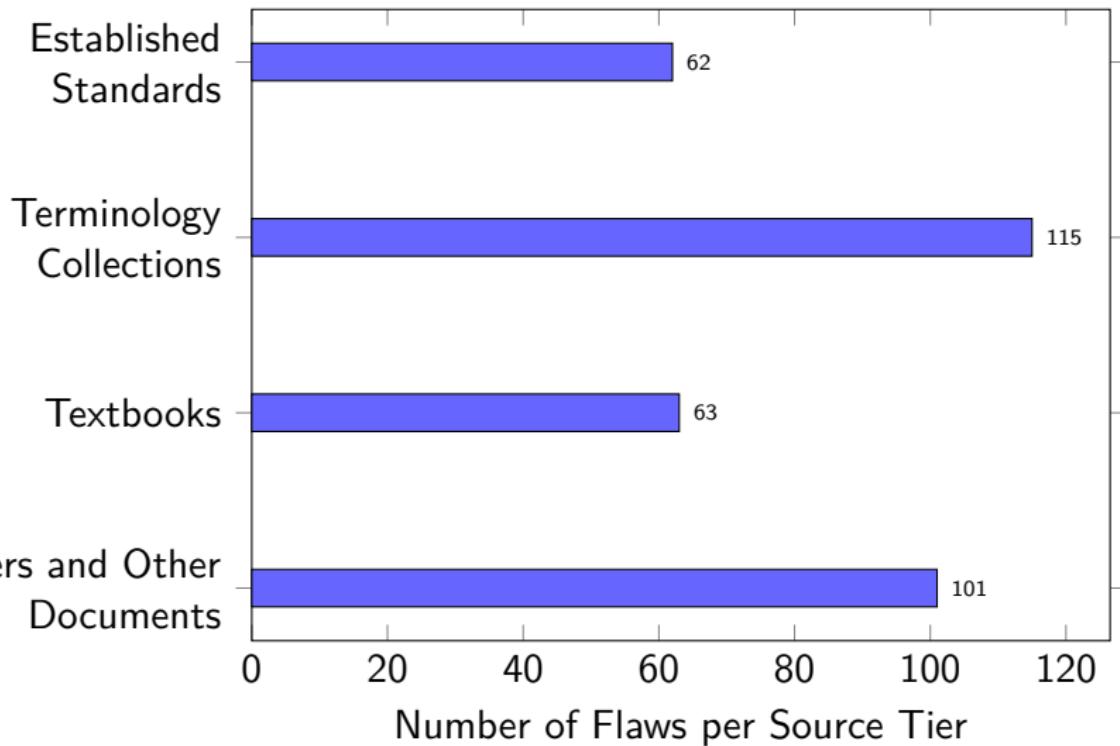
## 5 Conclusion

# Overview

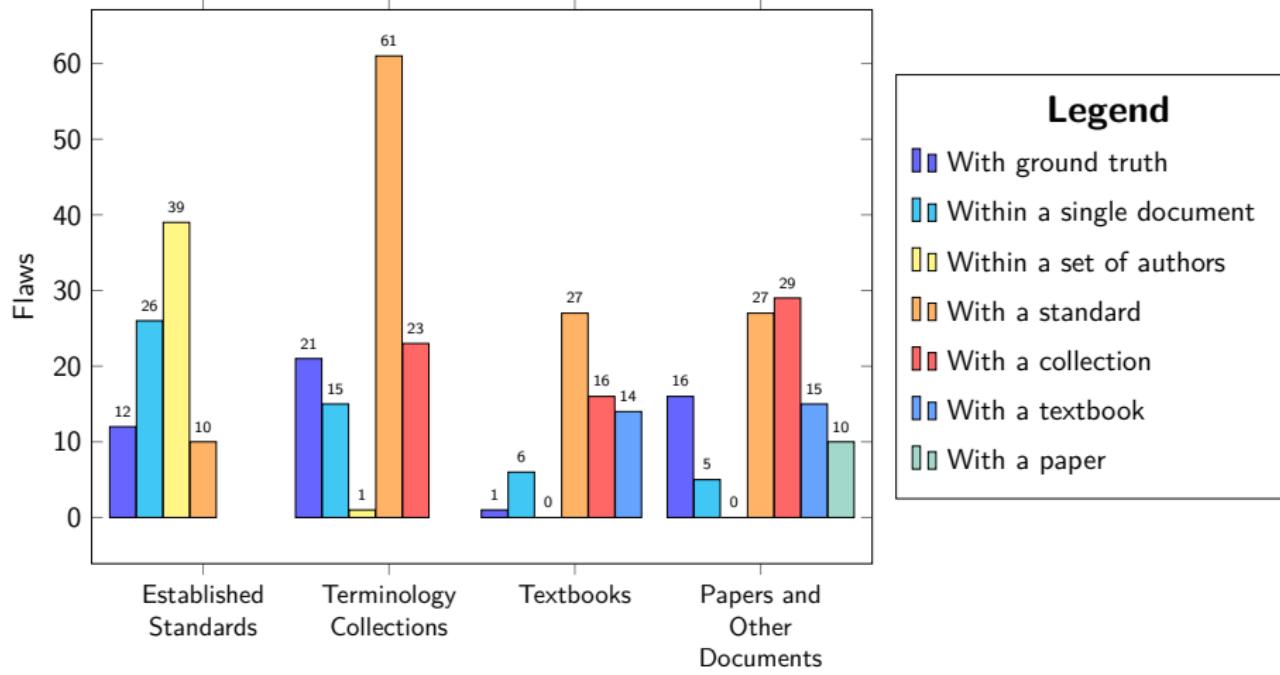
- 567 test approaches →
- 75 software qualities  
(may imply test approaches)
- 341 flaws in the  
software testing  
literature



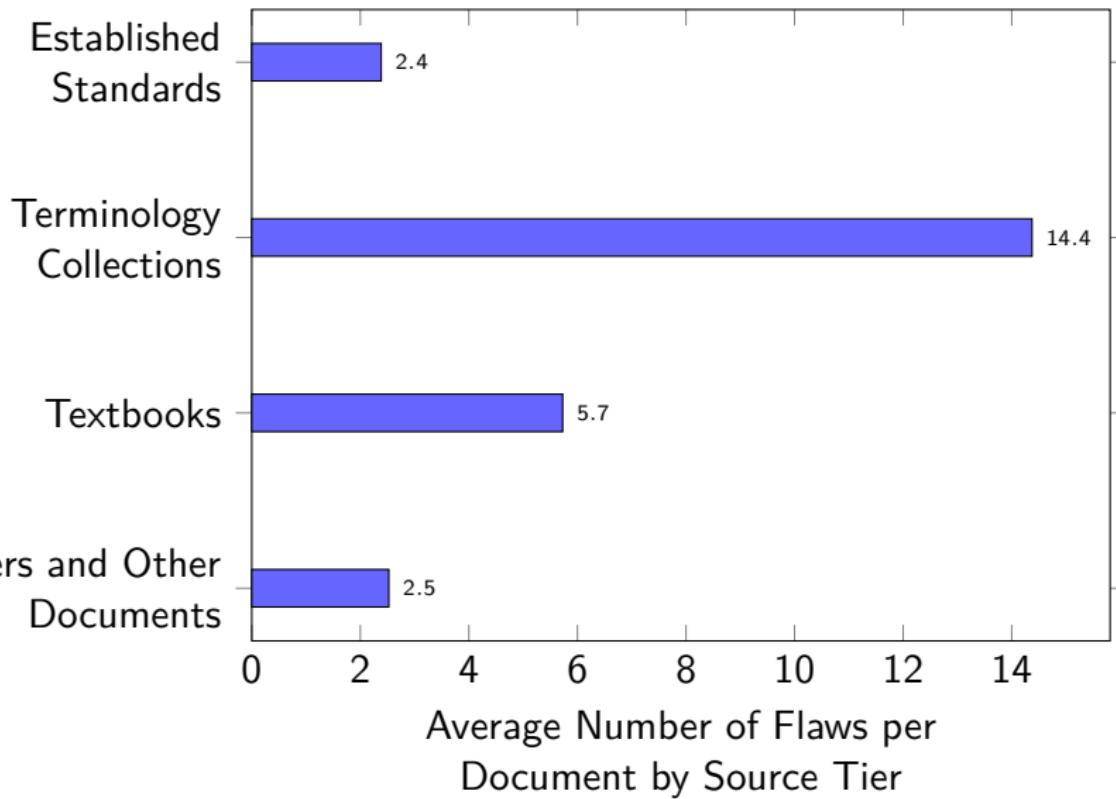
## Flaw Summary by Source Tier



# Flaw Summary by Source Tier



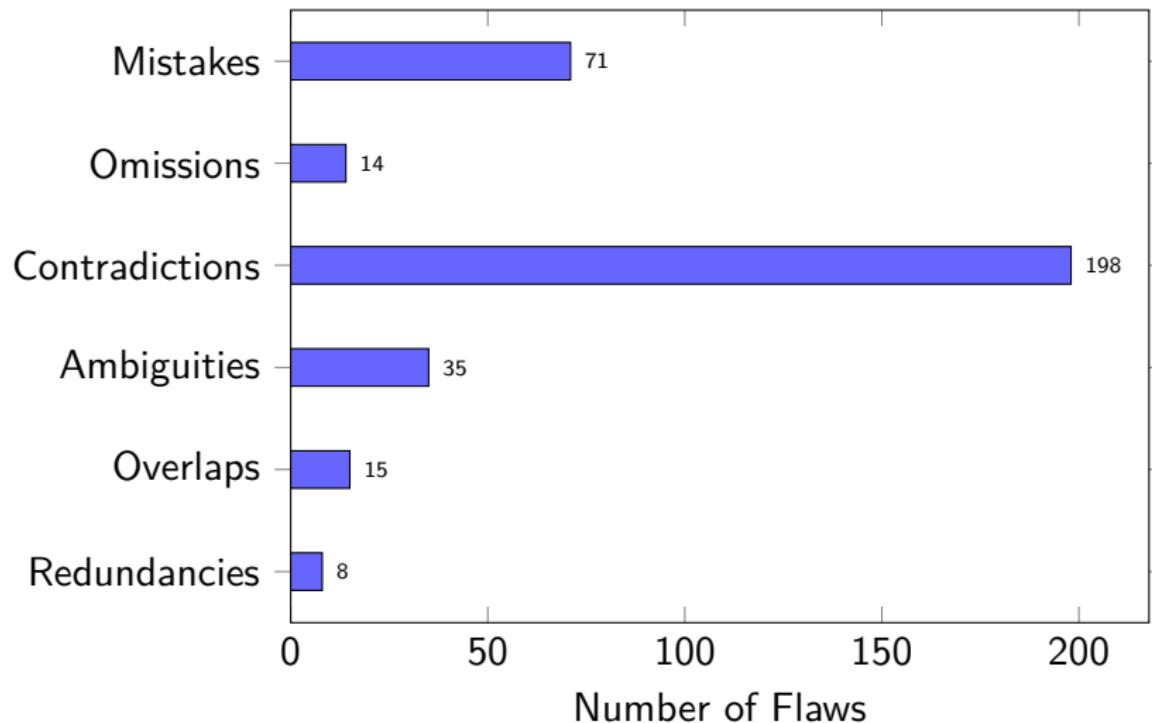
# Normalized Flaw Summary



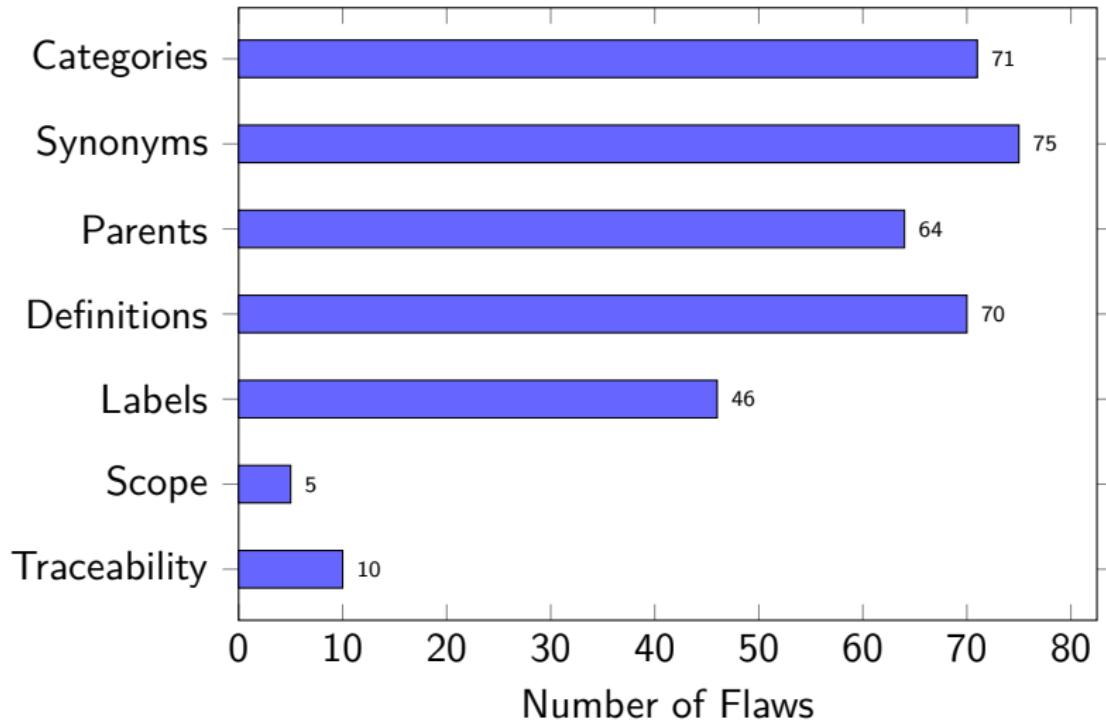
# Flaws by Source Tier Observations

- ① Established standards are not actually standardized, since:
  - ① other documents frequently disagree with them and
  - ② they are the most internally inconsistent source tier!
- ② Less standardized documents, such as terminology collections and textbooks, are also not followed to the extent they should be.
- ③ Documents across the board have flaws within the same document, between documents with the same author(s), or even with assertions of ground truth!

# Flaw Summary by Manifestation



# Flaw Summary by Domain



# General Flaw Observations

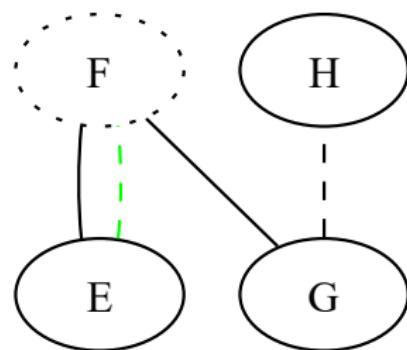
- ① Contradictions are by far the most common manifestation. This is likely because these are the most obvious flaws to detect automatically and because two (sets of) authors using different resources and not communicating have a high chance of disagreeing.
- ② Approach categorizations are the most subjective and one of the most common flaw domains, likely due to the lack of standardization about what categories to use.
- ③ In general, semantic flaws are more common than syntactic ones. The number of category flaws is comparable to the numbers of flaws with the relations and definitions of test approaches.

# 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

Some prominent examples:

### ① Functional Testing:

- Specification-based Testing
- *Conformance Testing*
- *Correctness Testing*

### Source(s)

(ISO/IEC and IEEE, 2017, p. 196; ...)

(Washizaki, 2025, p. 5-7)

(Washizaki, 2025, p. 5-7)

# Automated Flaws

## Intransitive Synonyms

Some prominent examples:

### ① Functional Testing:

- Specification-based Testing
- *Conformance Testing*
- *Correctness Testing*

### Source(s)

- (ISO/IEC and IEEE, 2017, p. 196; ... )  
(Washizaki, 2025, p. 5-7)  
(Washizaki, 2025, p. 5-7)

### ② Portability Testing:

- Flexibility Testing
- Configuration Testing

- (ISO/IEC, 2023)  
(Kam, 2008, p. 43)

### ③ Soak Testing:

- Endurance Testing
- Reliability Testing

- (ISO/IEC and IEEE, 2021c, p. 39)  
(Gerrard, 2000a, Tab. 2; 2000b, Tab. 1, p. 26)

# Automated Flaws

## Irreflexive Parents

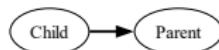
We also find some test approaches that are given as parents of themselves:

- 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

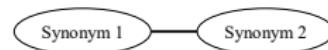
## Synonym and Parent-Child Overlaps

### Legend



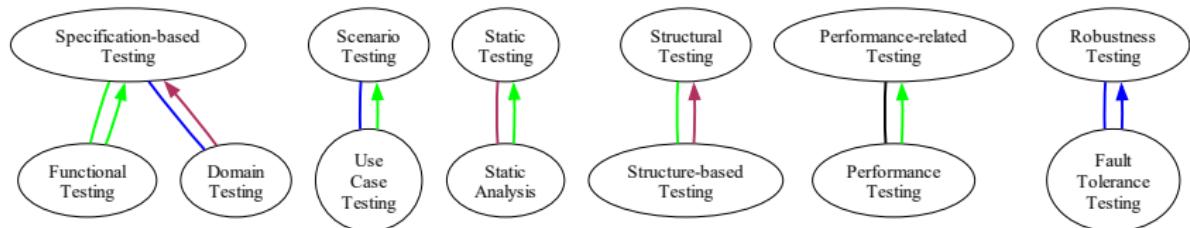
From Established Standards

From Terminology Collections



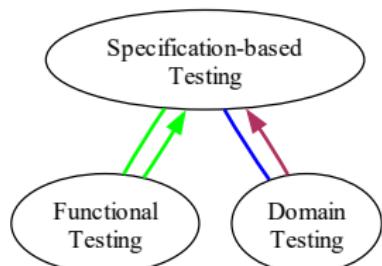
From Textbooks

From Papers and Other Documents



# 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

- The Need for Standardized Terminology
- The Lack of Standardized Terminology

## 2 Project

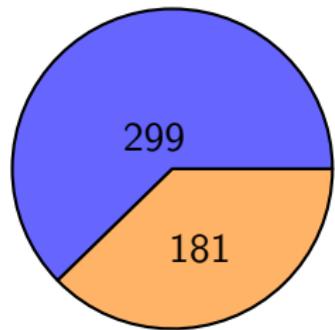
- Research Questions
- Methodology

## 3 Results

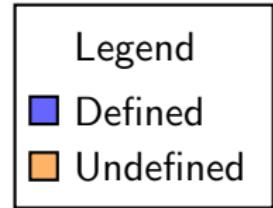
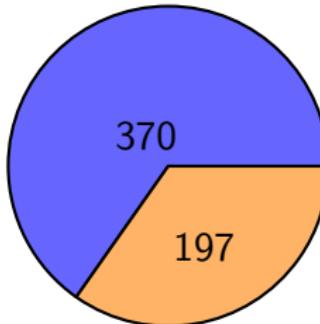
## 4 Future Work

## 5 Conclusion

# Future Work



Before



- 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

# Table of Contents

## 1 Introduction

- The Need for Standardized Terminology
- The Lack of Standardized Terminology

## 2 Project

- Research Questions
- Methodology

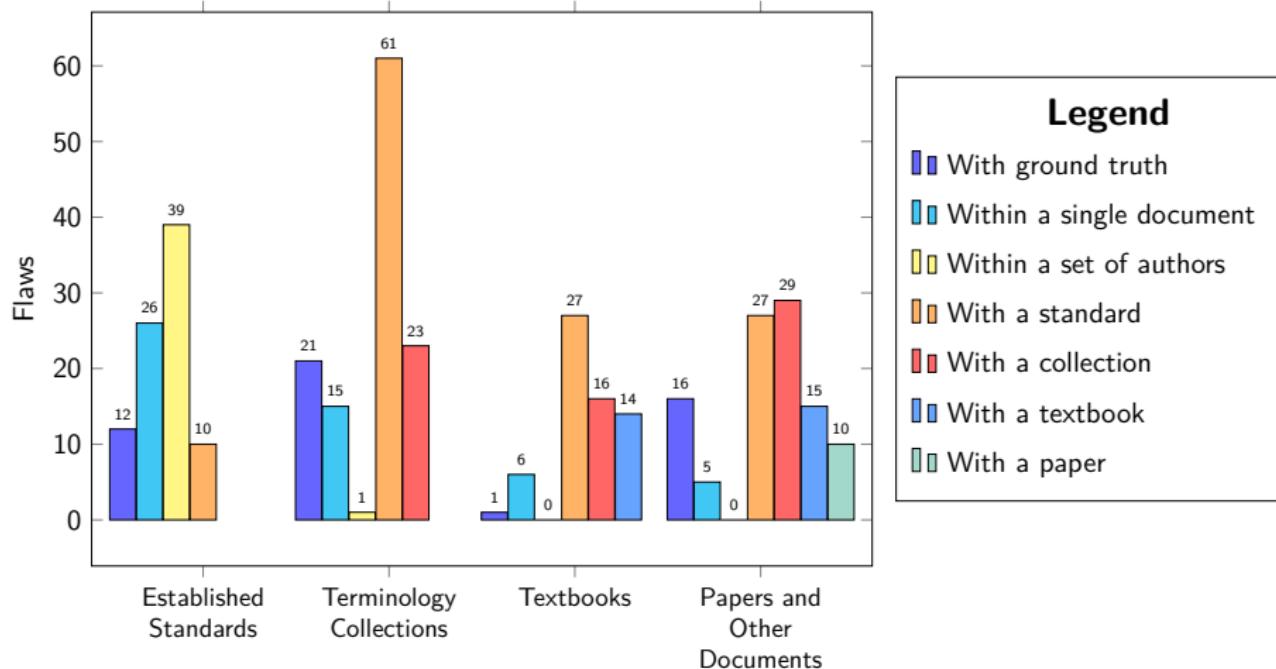
## 3 Results

## 4 Future Work

## 5 Conclusion

# Conclusion

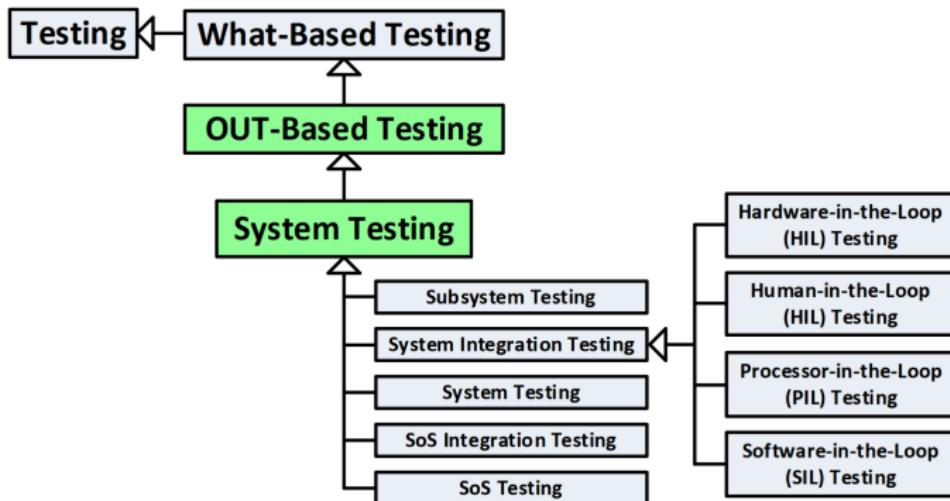
- The software testing literature is flawed, so don't assume everyone is on the same page



# Conclusion

- The software testing literature is flawed, so don't assume everyone is on the same page
- Even if they are, there can still be issues!

## What: by Object Under Test (OUT) – System Testing



(Firesmith, 2015, p. 23)

# 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  $\text{\LaTeX}$  code, including regex
  - ChatGPT and GitHub Copilot were both used for assistance with  $\text{\LaTeX}$  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](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/document/test/EBTestingPart2.pdf](http://wenku.uml.com/document/test/EBTestingPart2.pdf).
- Matthias Hamburg and Gary Mogyorodi, editors. ISTQB Glossary, v4.3, 2024. URL [https://glossary.istqb.org/en\\_US/search](https://glossary.istqb.org/en_US/search).

## References II

- 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*, November 2023. URL <https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en>.
- 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.
- ISO/IEC and IEEE. ISO/IEC/IEEE International Standard - Software and systems engineering –Software testing –Part 2: Test processes. *ISO/IEC/IEEE 29119-2:2021(E)*, October 2021a. doi: 10.1109/IEEESTD.2021.9591508.

## References III

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 IV

- 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](https://commons.wikimedia.org/wiki/File:Force_examples.svg).

## References V

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 VI

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>.