Putting Software Testing Terminology to the Test

M.A.Sc. Seminar

Samuel Crawford, B.Eng.

McMaster University
Department of Computing and Software

Fall 2024

Table of Contents

Table of Contents

The Need for Standardized Terminology

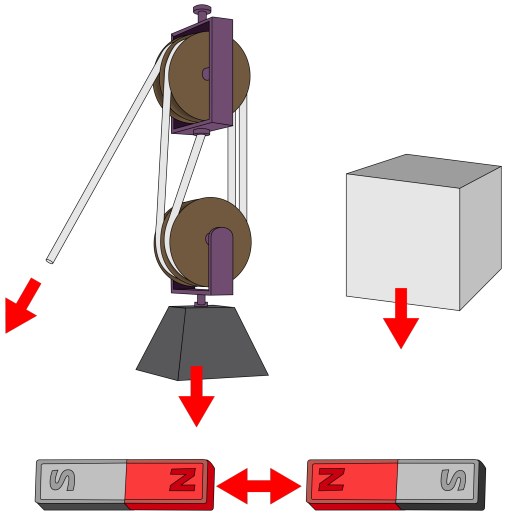
- Engineering is applied science
- Scientific fields use precise terminology

$\Rightarrow$ SOFTWARE ENGINEERING

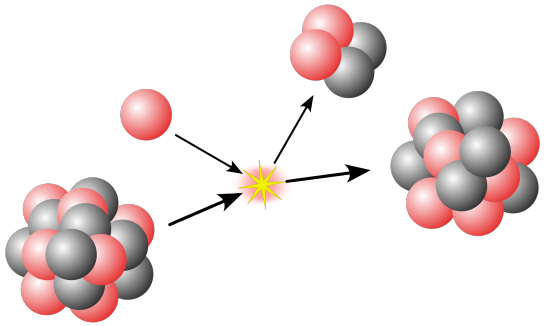
The Need for Standardized Terminology

- Engineering is applied science
- Scientific fields use precise terminology

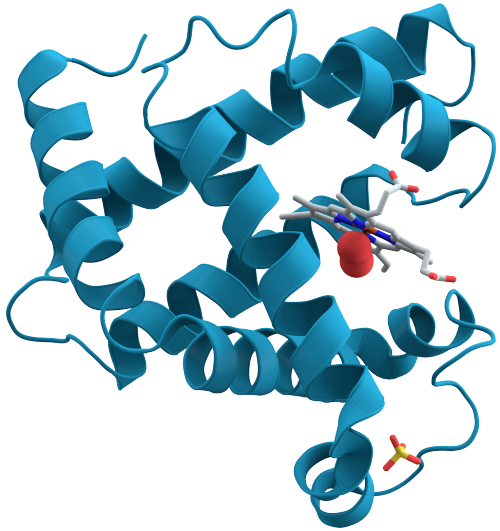
$\Rightarrow$ SOFTWARE ENGINEERING



Penubag and Ramey (2010)



Kjerish (2016)



AzaToth (2008)

The Lack of Standardized Terminology

"The Problem"

- Unfortunately, a search for a systematic, rigorous, and complete taxonomy for software testing revealed that the existing ones are inadequate and mostly focus on the high-level testing process rather than the testing approaches themselves:
  - Tebes et al. (2020) focus on *parts* of the testing process (e.g., test goal, test plan, testing role, testable entity) and how they relate to one another,
  - Souza et al. (2017) prioritize organizing testing approaches over defining them, and
  - Unterkalmsteiner et al. (2014) focus on the "information linkage or transfer" (p. A:6) between requirements engineering and software testing and "do[] not aim at providing a systematic and exhaustive state-of-the-art survey of [either domain]" (p. A:2).

Unstandardized Standards

"The Problem" (cont.)

- Tours "guide[] testers through the paths of an application" by following a structure that is:
  - quite general (ISO/IEC and IEEE, 2022, p. 34)

Unstandardized Standards

"The Problem" (cont.)

- Tours "guide[] testers through the paths of an application" by following a structure that is:
  - quite general (ISO/IEC and IEEE, 2022, p. 34)
  - "organized around a special focus" (Hamburg and Mogyorodi, 2024)

Unstandardized Standards

"The Problem" (cont.)

- Tours "guide[] testers through the paths of an application" by following a structure that is:
  - quite general (ISO/IEC and IEEE, 2022, p. 34)
  - "organized around a special focus" (Hamburg and Mogyorodi, 2024)
- Load testing is "conducted to evaluate the behaviour of a test item under anticipated conditions of varying load" (ISO/IEC and IEEE, 2022, p. 5; 2017, p. 253), such as:
  - loads "between anticipated conditions of low, typical, and peak usage" (2022, p. 5)

Unstandardized Standards

"The Problem" (cont.)

- Tours "guide[] testers through the paths of an application" by following a structure that is:
  - quite general (ISO/IEC and IEEE, 2022, p. 34)
  - "organized around a special focus" (Hamburg and Mogyorodi, 2024)
- Load testing is "conducted to evaluate the behaviour of a test item under anticipated conditions of varying load" (ISO/IEC and IEEE, 2022, p. 5; 2017, p. 253), such as:
  - loads "between anticipated conditions of low, typical, and peak usage" (2022, p. 5)
  - loads that are as large as possible (Patton, 2006, p. 86)

Unstandardized Standards

"The Problem" (cont.)

- Alpha testing is the "first stage of testing before a product is considered ready for commercial or operational use" (ISO/IEC and IEEE, 2017, p. 17) performed by:
  - "users within the organization developing the software" (p. 17)

Unstandardized Standards

"The Problem" (cont.)

- Alpha testing is the "first stage of testing before a product is considered ready for commercial or operational use" (ISO/IEC and IEEE, 2017, p. 17) performed by:
  - "users within the organization developing the software" (p. 17)
  - "a small, selected group of potential users" (Washizaki, 2024, p. 5-8)

Unstandardized Standards

"The Problem" (cont.)

- Alpha testing is the "first stage of testing before a product is considered ready for commercial or operational use" (ISO/IEC and IEEE, 2017, p. 17) performed by:
  - "users within the organization developing the software" (p. 17)
  - "a small, selected group of potential users" (Washizaki, 2024, p. 5-8)
  - "roles outside the development organization" conducted "in the developer's test environment" (Hamburg and Mogyorodi, 2024)

Unstandardized Standards

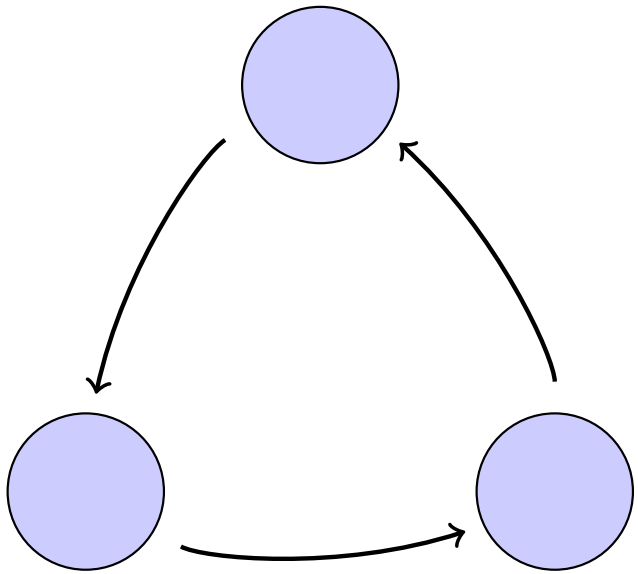
"The Problem" (cont.)

- Alpha testing is the "first stage of testing before a product is considered ready for commercial or operational use" (ISO/IEC and IEEE, 2017, p. 17) performed by:
  - "users within the organization developing the software" (p. 17)
  - "a small, selected group of potential users" (Washizaki, 2024, p. 5-8)
  - "roles outside the development organization" conducted "in the developer's test environment" (Hamburg and Mogyorodi, 2024)

"Okay testing team, we want to conduct alpha testing on our product. What's our timeline? Budget? Sample size?"

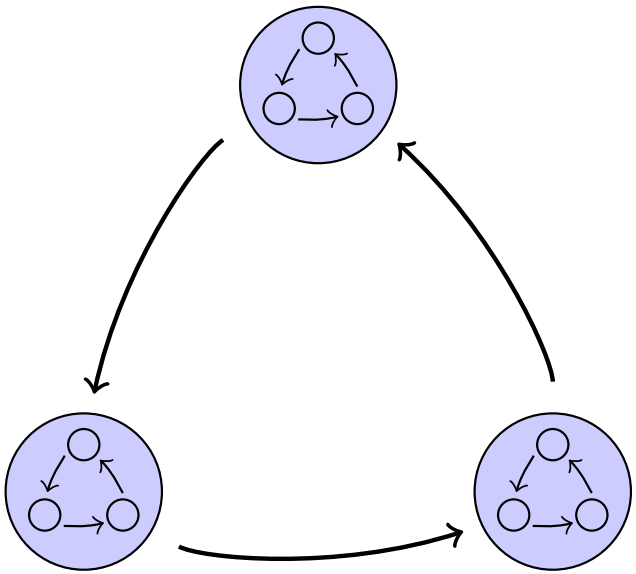
Interorganizational

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)

Table of Contents

Research Questions

Research Question 1

What testing approaches do the literature describe?

Research Question 2

Are these descriptions consistent?

Research Question 3

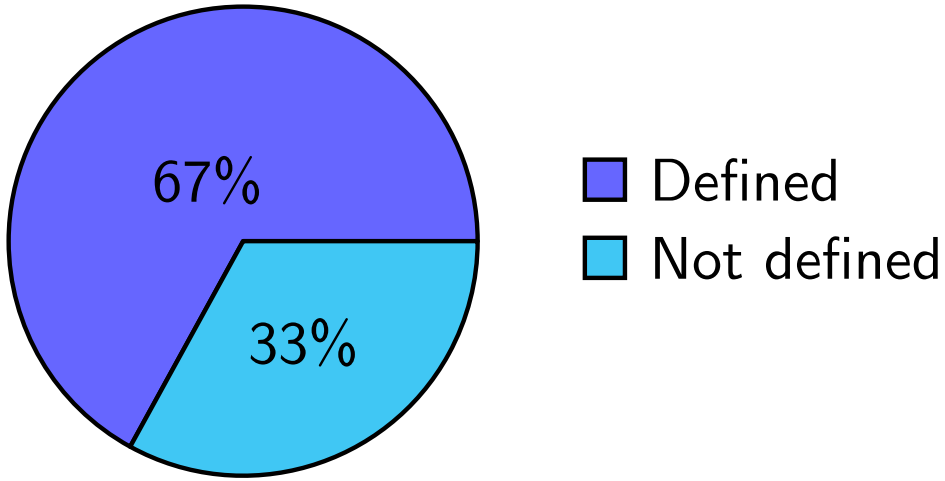
Can we systematically resolve any of these inconsistencies?

Research Questions

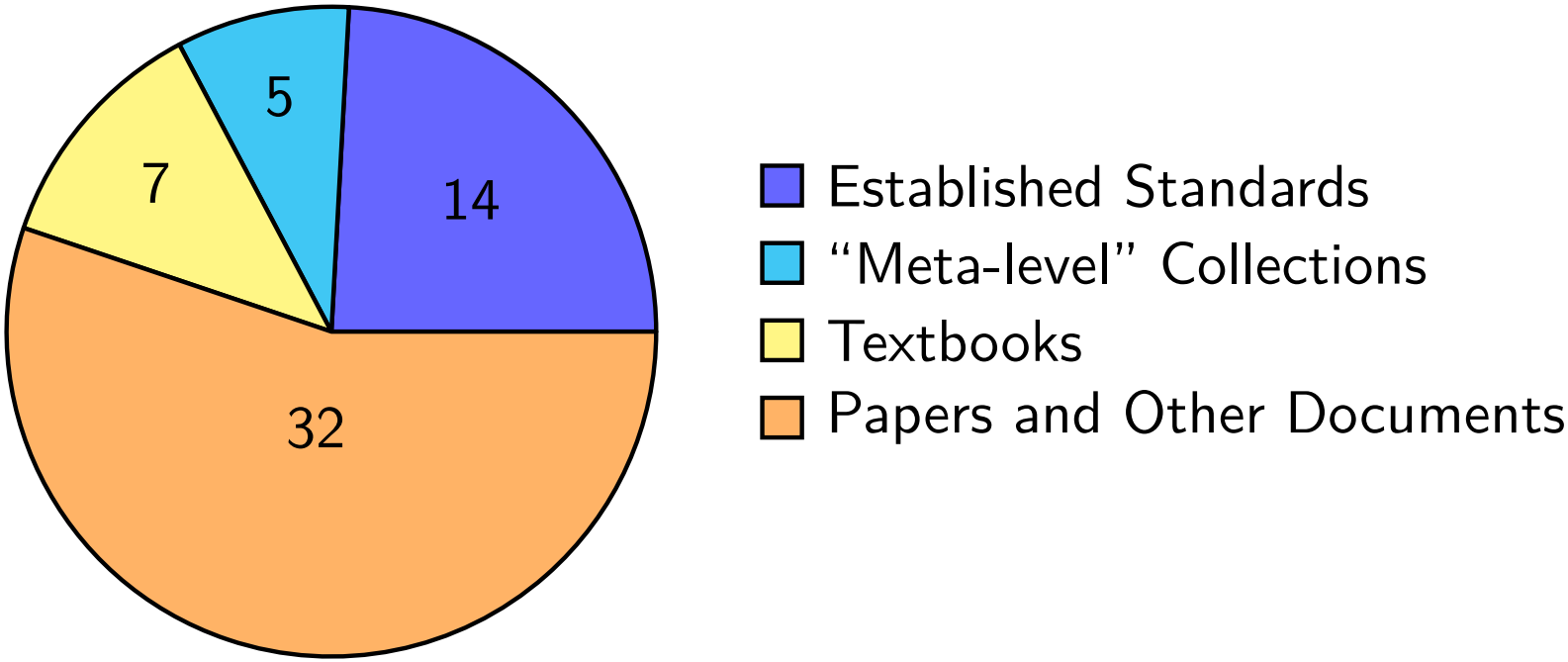
Research Question 1

What testing approaches do the literature describe?

- 527 test approaches $\rightarrow$
- 76 software qualities



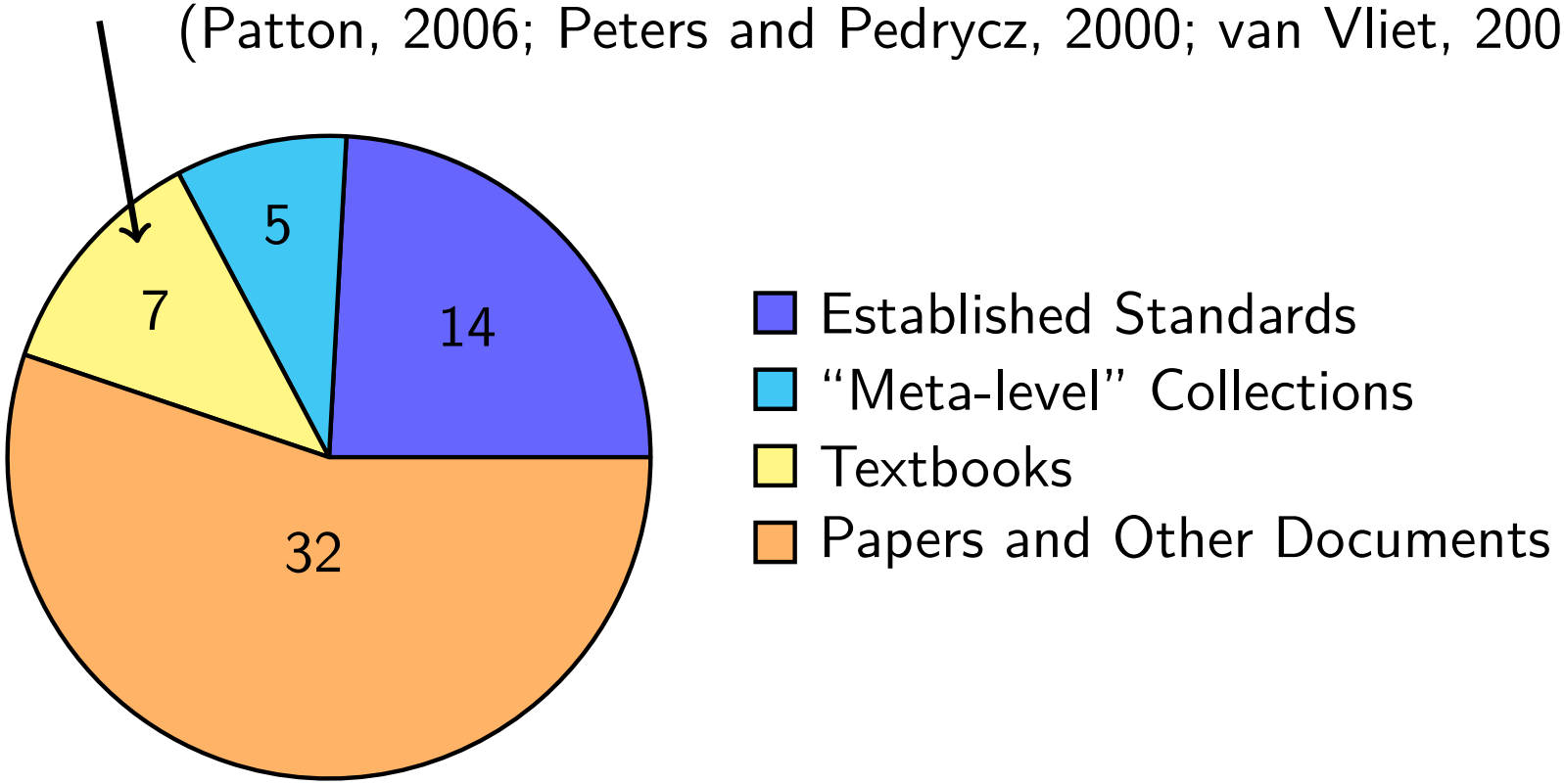
Methodology: Sources



Summary of how many sources comprise each source category.

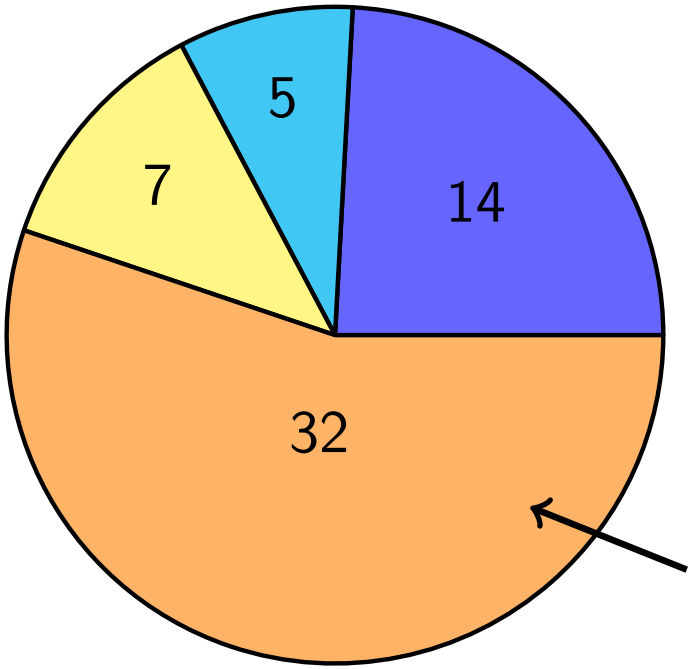
Methodology: Sources

Textbooks trusted at McMaster were our ad hoc starting points (Patton, 2006; Peters and Pedrycz, 2000; van Vliet, 2000)



Summary of how many sources comprise each source category.

Methodology: Sources

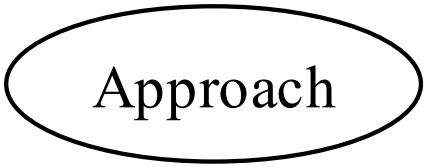


- Established Standards
- "Meta-level" Collections
- Textbooks
- Papers and Other Documents

Includes websites (LambdaTest, 2024; Pandey, 2023) and a booklet (Knüvener Mackert GmbH, 2022)

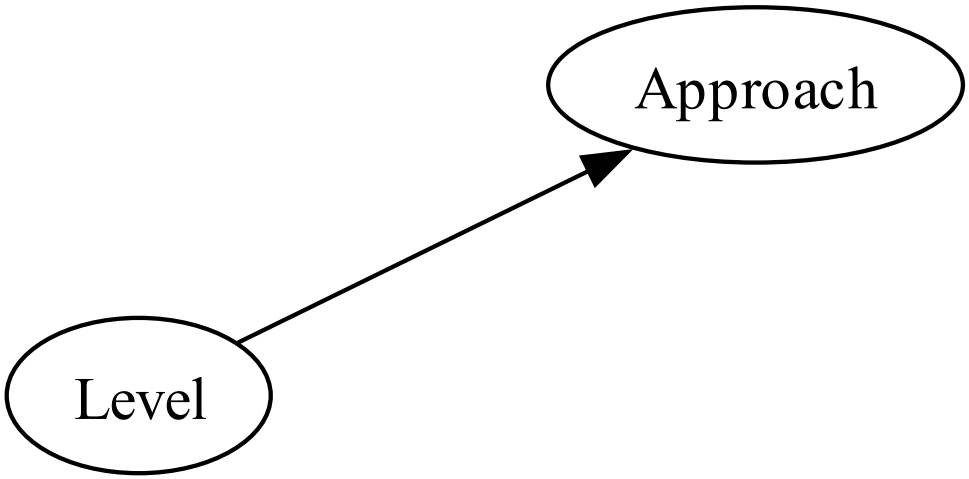
Summary of how many sources comprise each source category.

Methodology: Categories



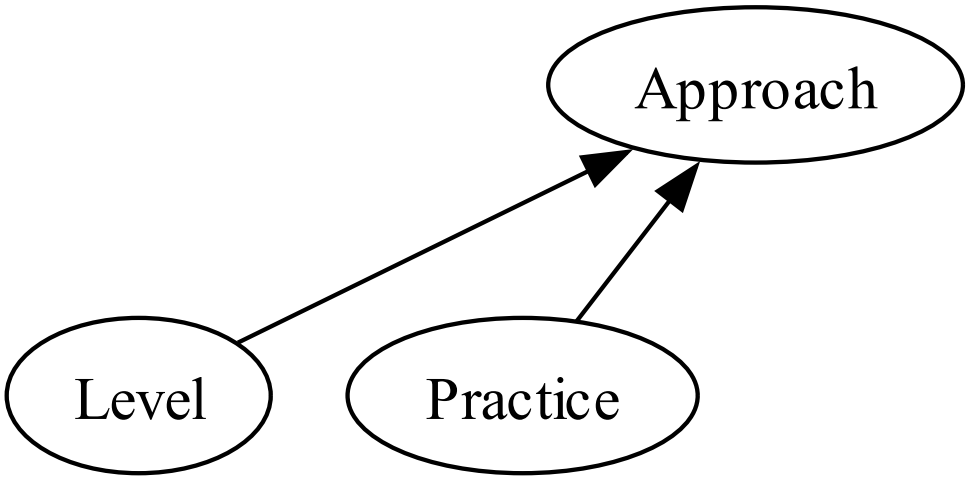
Approach: a "high-level test implementation choice" (ISO/IEC and IEEE, 2022, p. 10) used to "pick the particular test case values" (2017, p. 465)

Methodology: Categories



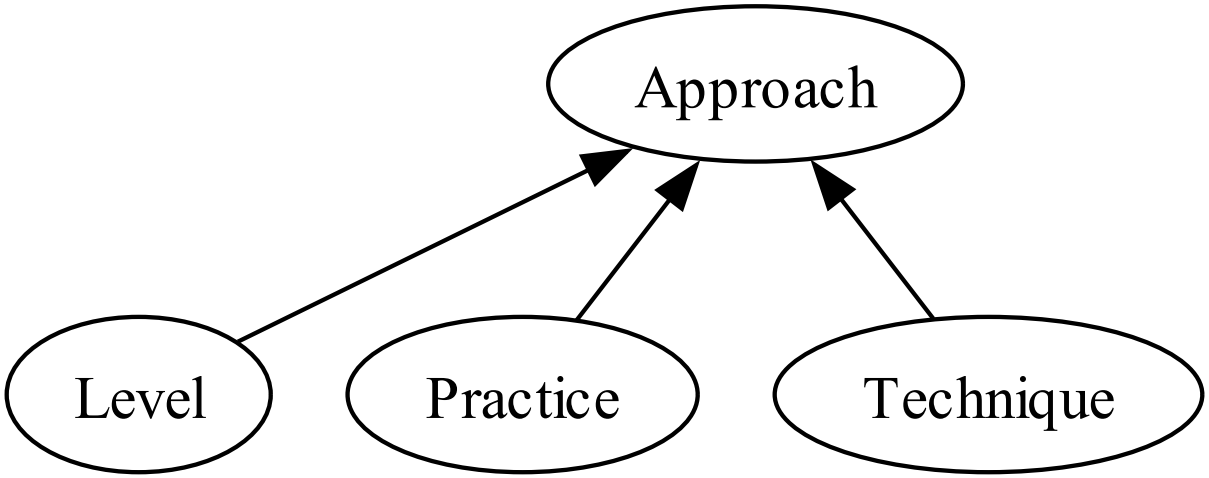
Level: a stage of testing with "particular objectives and . . . risks", each performed in sequence (ISO/IEC and IEEE, 2022, p. 12; 2021, p. 6)

Methodology: Categories



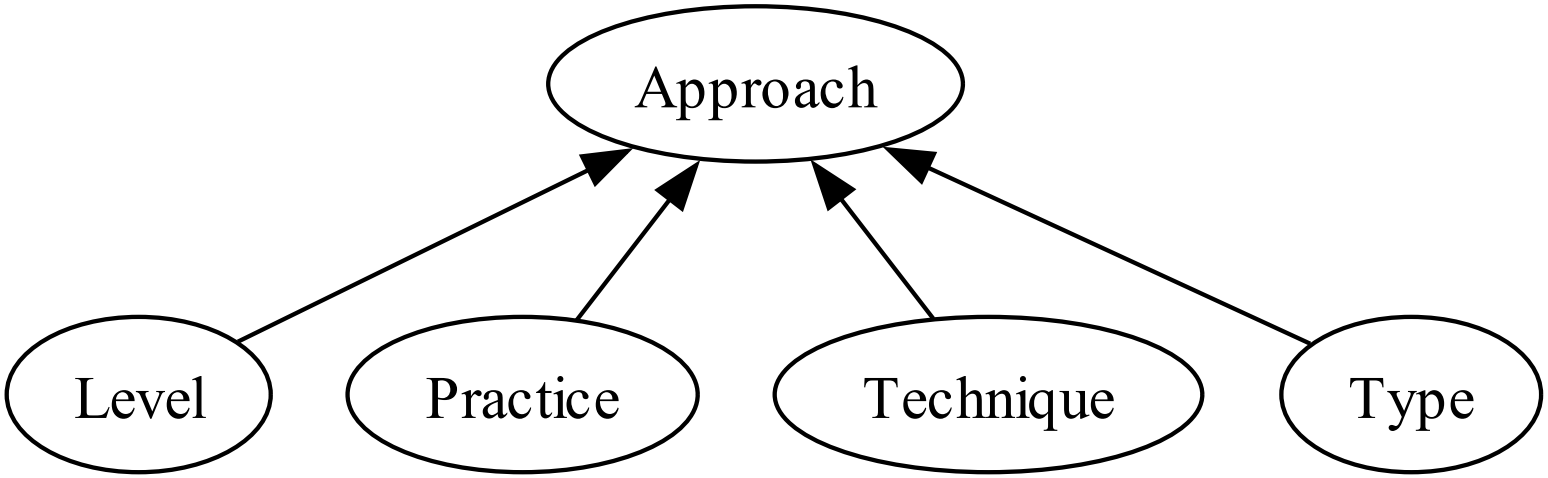
Practice: a "conceptual framework that can be applied to . . . [a] test process to facilitate testing" (ISO/IEC and IEEE, 2022, p. 14; 2017, p. 471)

Methodology: Categories



Technique: a "defined" and "systematic" (ISO/IEC and IEEE, 2017, p. 464) "procedure used to create or select a test model, identify test coverage items, and derive corresponding test cases" (2022, p. 11)

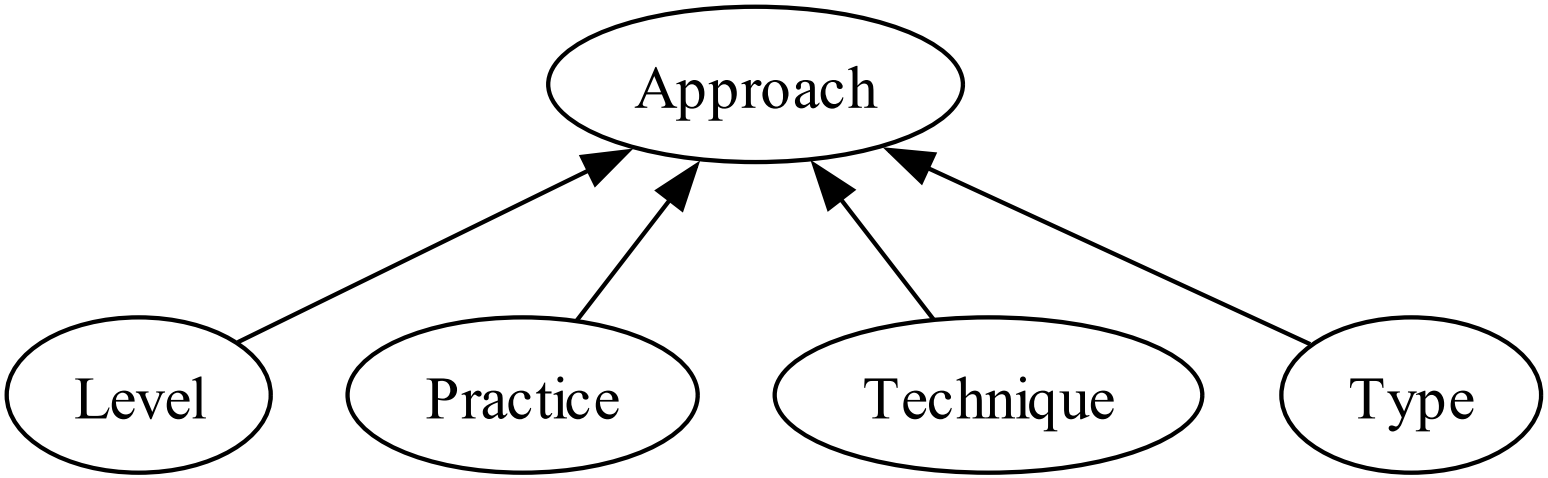
Methodology: Categories



Type: "Testing that is focused on specific quality characteristics" (ISO/IEC and IEEE, 2022, p. 15; 2021, p. 7; 2017, p. 473)

Methodology: Graph Notation

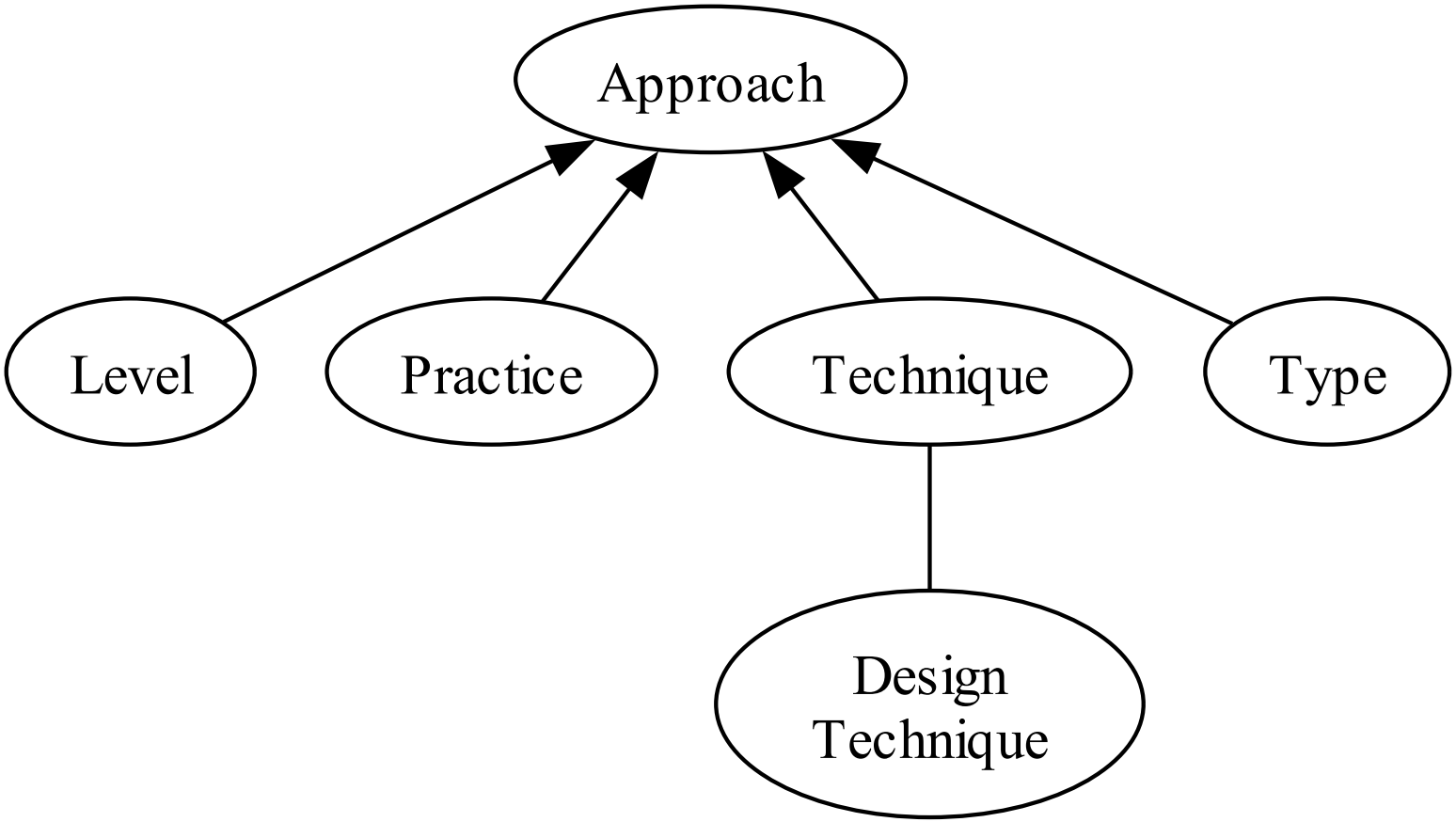
Relations



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

Methodology: Graph Notation

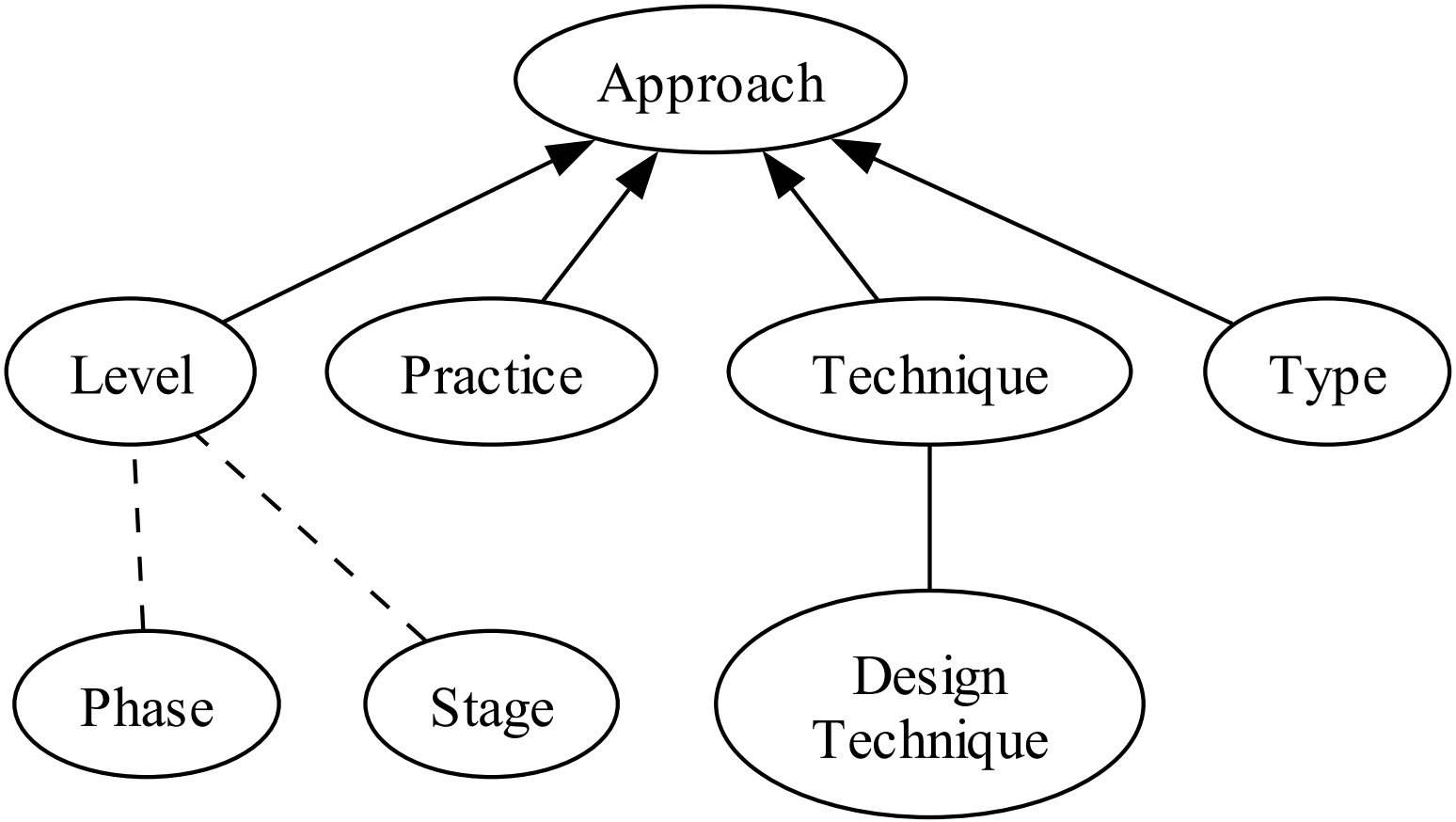
Relations



Lines without arrowheads connect *synonyms*.

Methodology: Graph Notation

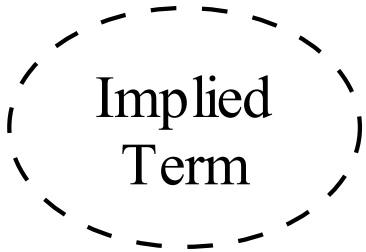
Relations



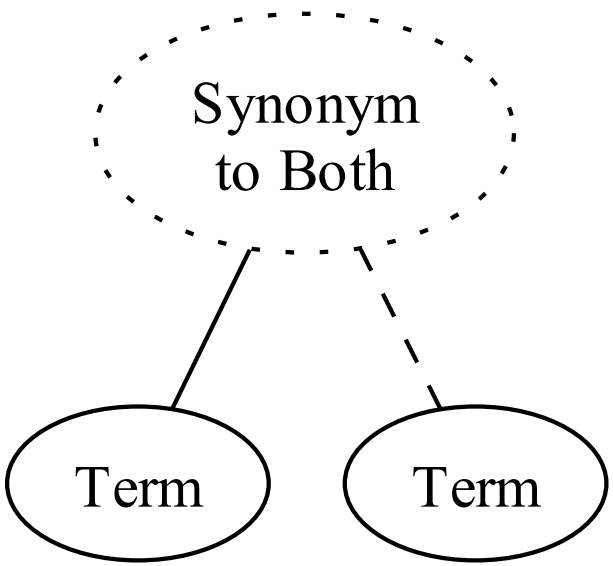
Dashed lines indicate a relationship is *implied*.

Methodology: Graph Notation

Terms

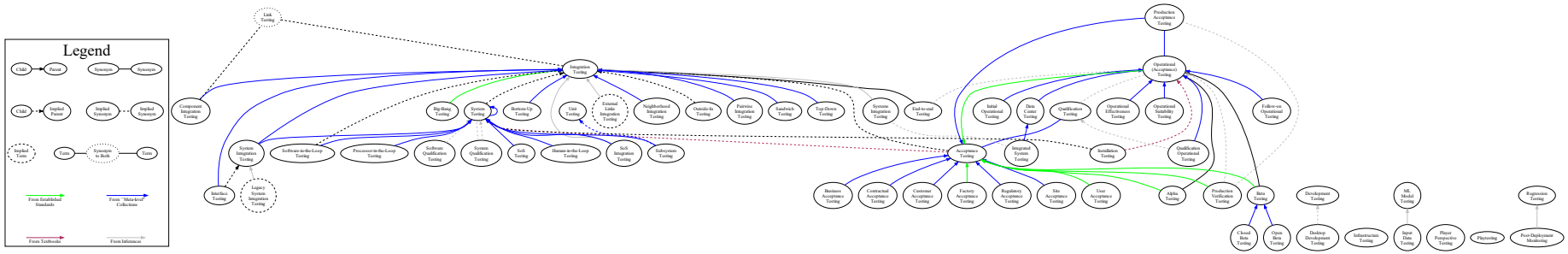


Dashed outlines indicate a term is *implied*.

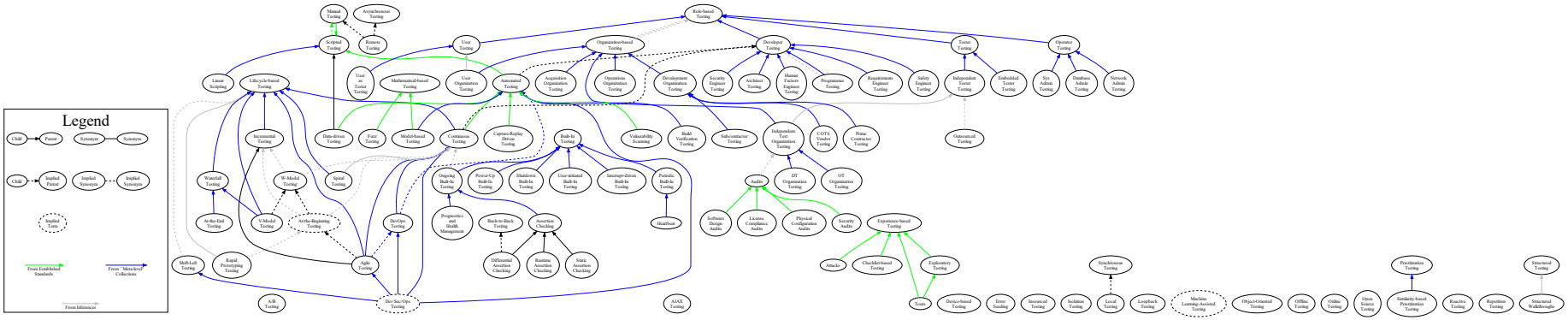


Dotted outlines indicate a term is a *synonym* to more than one term.

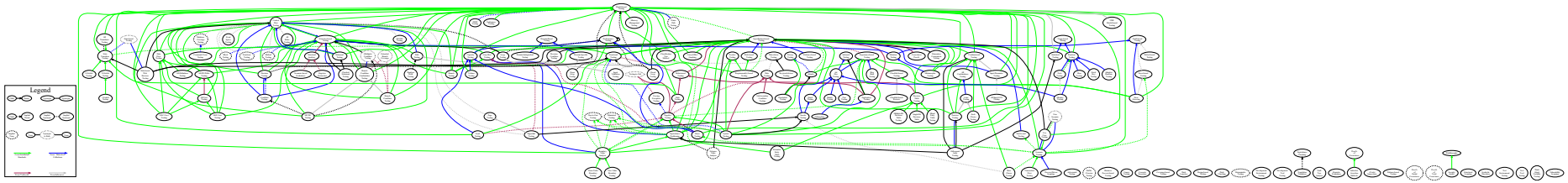
Graph of Test Levels



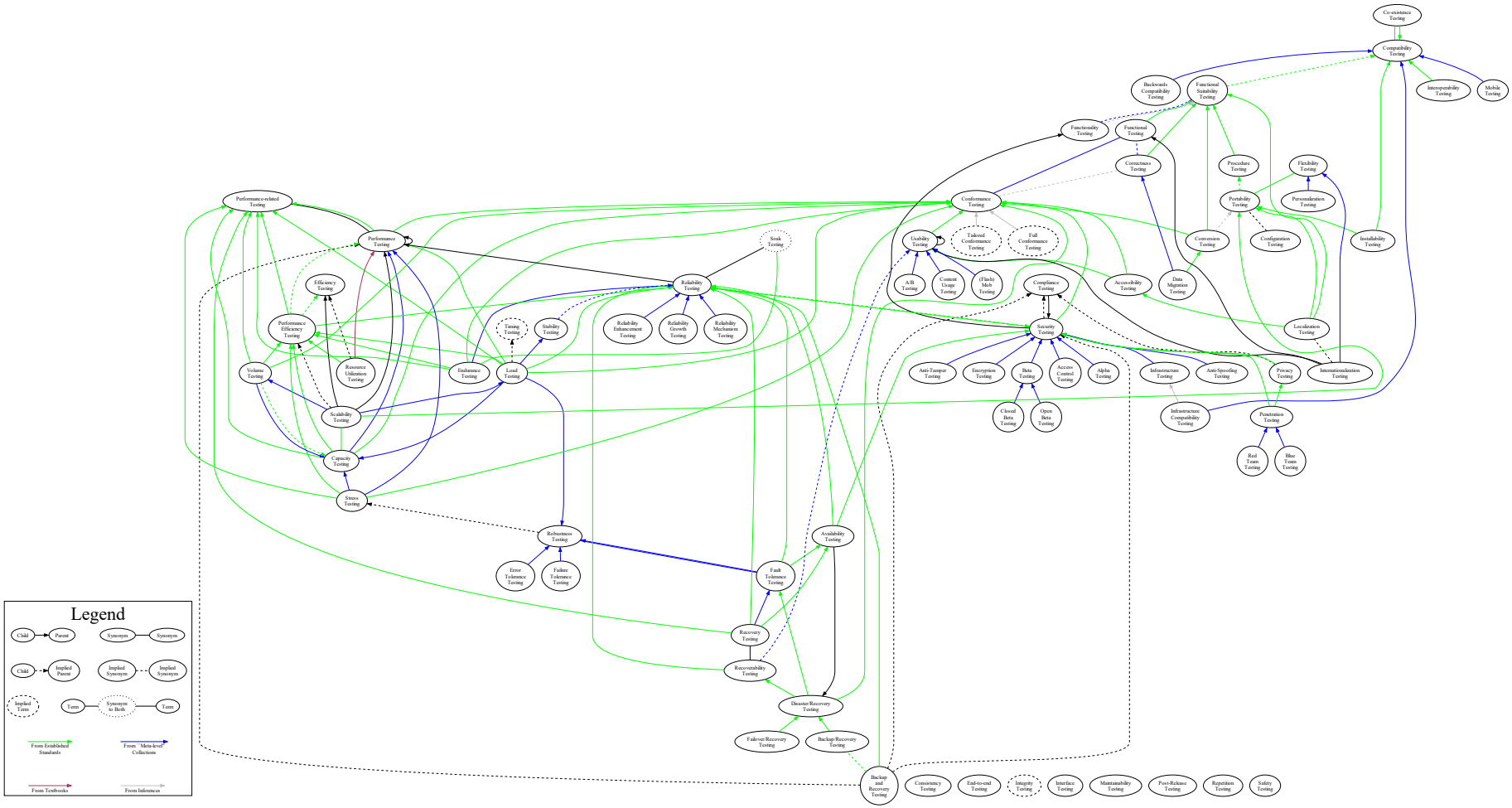
Graph of Test Practices



Graph of Test Techniques

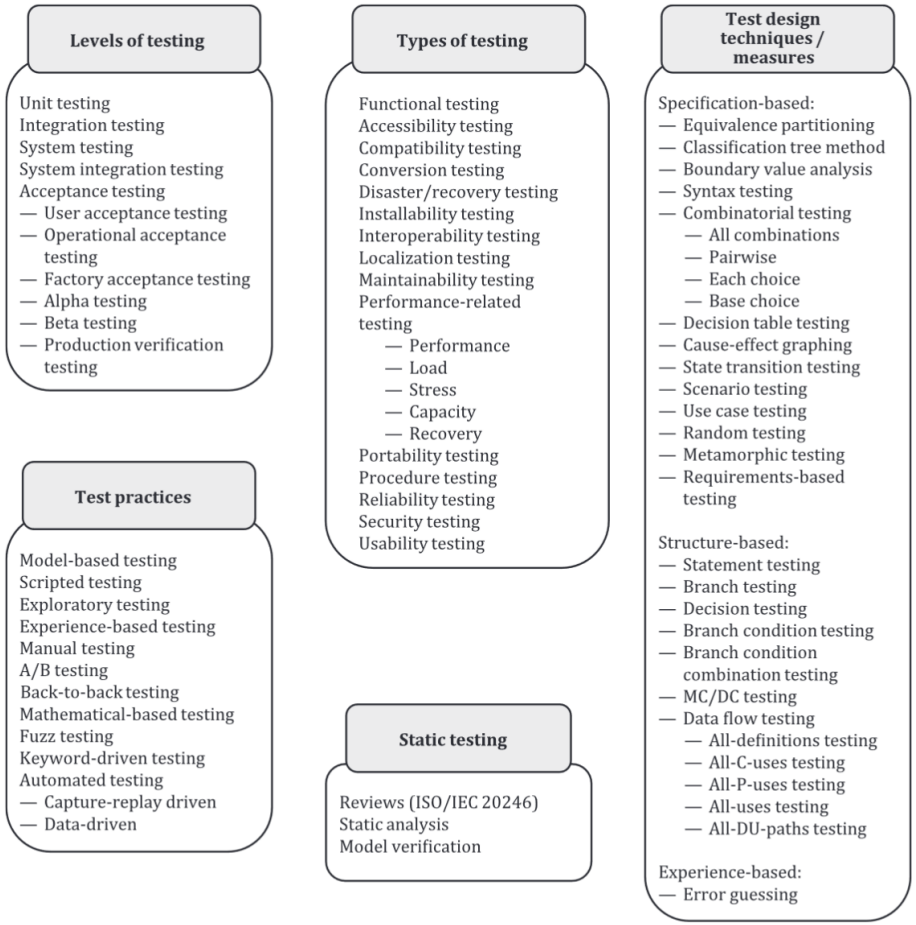


Graph of Test Types



Methodology: Graph Notation

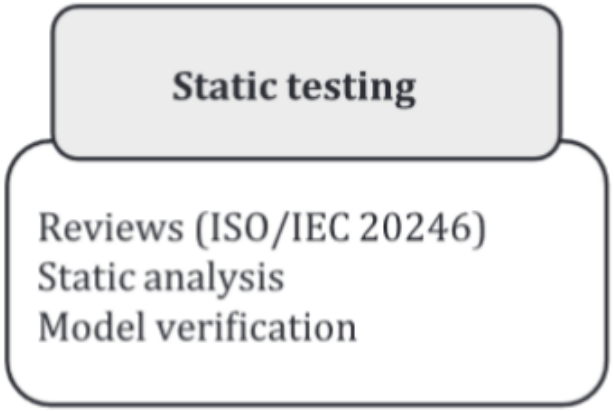
Static Testing



Example test approach choices (ISO/IEC and IEEE, 2022, Fig. 2).

Methodology: Graph Notation

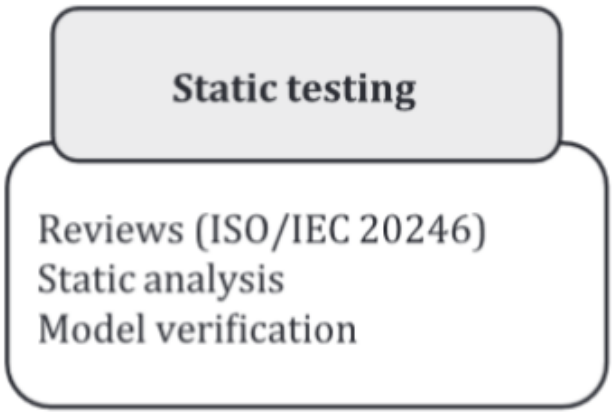
Static Testing



Example test approach choices. Adapted from (ISO/IEC and IEEE, 2022, Fig. 2)

Methodology: Graph Notation

Static Testing

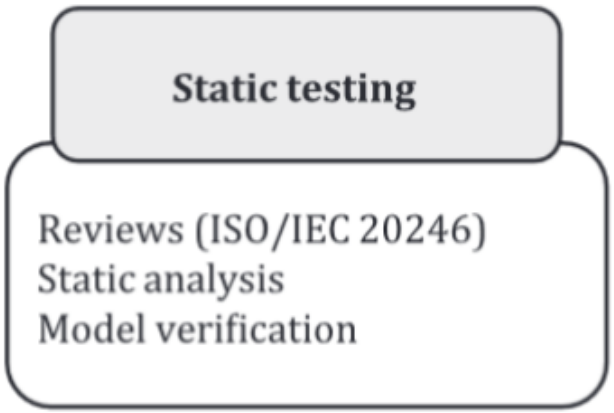


Example test approach choices. Adapted from (ISO/IEC and IEEE, 2022, Fig. 2)

- While our focus is on dynamic testing, we include static testing in our research for completeness

Methodology: Graph Notation

Static Testing

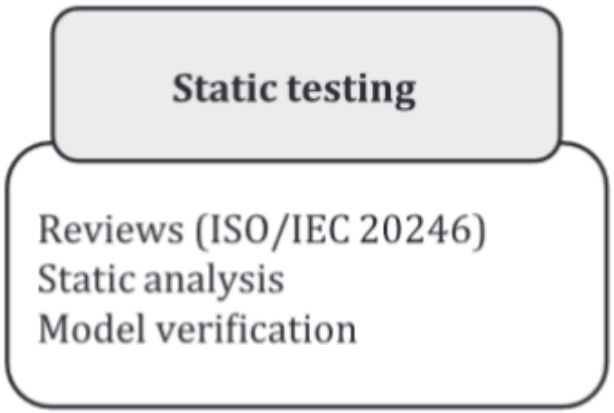


Example test approach choices. Adapted from (ISO/IEC and IEEE, 2022, Fig. 2)

- While our focus is on dynamic testing, we include static testing in our research for completeness
- Static testing *is* quite distinct from dynamic testing, but this does not necessarily make it an orthogonal category

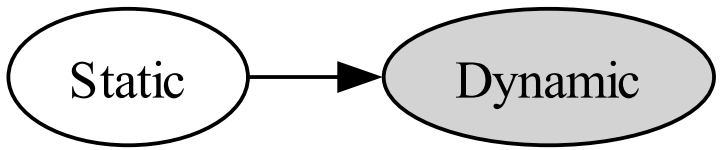
Methodology: Graph Notation

Static Testing

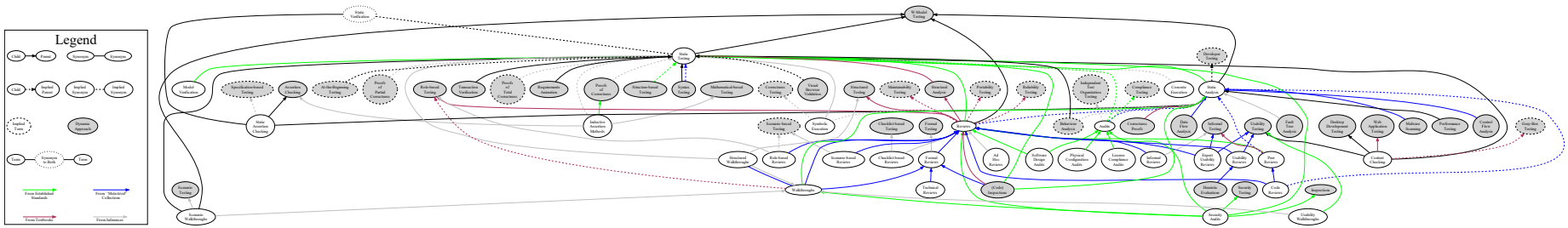


Example test approach choices. Adapted from (ISO/IEC and IEEE, 2022, Fig. 2)

- While our focus is on dynamic testing, we include static testing in our research for completeness
- Static testing *is* quite distinct from dynamic testing, but this does not necessarily make it an orthogonal category
- When considering static testing in isolation, terms with gray backgrounds are related *dynamic approaches*



Graph of *Static* Test Approachs



Methodology: Procedure

Approaches

- A row is created for each test approach, such as the following which is based on (ISO/IEC and IEEE, 2022)

Name	Category	Definition	Parent(s)	Synonym(s)
A/B Testing	Practice (p. 22)	Testing "that allows testers to determine which of two systems or components performs better" (p. 1)	Statistical Testing (pp. 1, 35), ...	Split-Run Testing (pp. 1, 35)

Methodology: Procedure

Approaches

- A row is created for each test approach, such as the following which is based on (ISO/IEC and IEEE, 2022)

Name	Category	Definition	Parent(s)	Synonym(s)
A/B Testing	Practice (p. 22)	Testing "that allows testers to determine which of two systems or components performs better" (p. 1)	Statistical Testing (pp. 1, 35), ...	Split-Run Testing (pp. 1, 35)

- This information is gathered from sources by looking for
  - Glossaries
  - Testing-related terms
  - Terms described *by* other approaches
  - Terms that *imply* other approaches

Methodology: Procedure

Other Information

- It seems that the existence of a software quality implies the existence of a test type associated with it

Methodology: Procedure

Other Information

- It seems that the existence of a software quality implies the existence of a test type associated with it
- Some test approaches use shared or complicated terminology

Methodology: Procedure

Other Information

- It seems that the existence of a software quality implies the existence of a test type associated with it
- Some test approaches use shared or complicated terminology
- For each of these, we record its
  - Name
  - Definition
  - Precedence for a related test type (only for qualities)
  - Synonym(s)

Methodology: Procedure

- Recording these data in a consistent format allows for graphs to be generated according to a certain logic

Methodology: Procedure

- Recording these data in a consistent format allows for graphs to be generated according to a certain logic
- It also allows for subsets of discrepancies to be identified

Methodology: Procedure

- Recording these data in a consistent format allows for graphs to be generated according to a certain logic
- It also allows for subsets of discrepancies to be identified

Research Question 2

Are these descriptions consistent?

Table of Contents

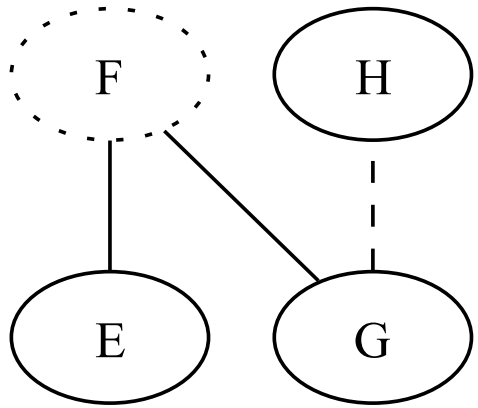
Automated Discrepancies

- Some terms are given as a synonym to two (or more) disjoint, unrelated terms, making the relation between the given synonyms ambiguous

Automated Discrepancies

- Some terms are given as a synonym to two (or more) disjoint, unrelated terms, making the relation between the given synonyms ambiguous
- These are included in generated graphs automatically

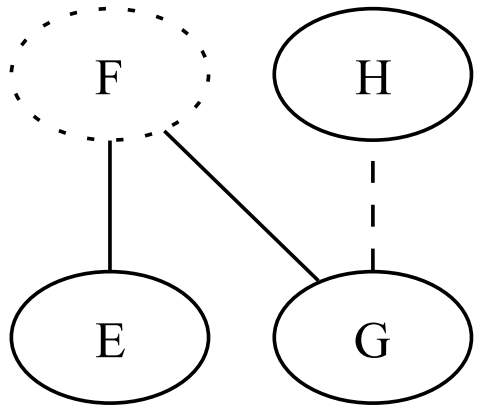
Name	Synonym(s)
E	F (Author, 0000; implied by 0001)
G	F (Author, 0002), H (implied by 0000)
H	X



Automated Discrepancies

- Some terms are given as a synonym to two (or more) disjoint, unrelated terms, making the relation between the given synonyms ambiguous
- These are included in generated graphs automatically

Name	Synonym(s)
E	F (Author, 0000; implied by 0001)
G	F (Author, 0002), H (implied by 0000)
H	X



- The following four are the most prominent examples of the ten identified automatically:

Automated Discrepancies

1. **Invalid Testing:**
   - Error Tolerance Testing (Kam, 2008, p. 45)
   - Negative Testing (Hamburg and Mogyorodi, 2024; implied by ISO/IEC and IEEE, 2021, p. 10)
2. **Soak Testing:**
   - Endurance Testing (ISO/IEC and IEEE, 2021, p. 39)
   - Reliability Testing (Gerrard, 2000a, Tab. 2; 2000b, Tab. 1, p. 26)
3. **User Scenario Testing:**
   - Scenario Testing (Hamburg and Mogyorodi, 2024)
   - Use Case Testing (Kam, 2008, p. 48) (although "an actor can be a user or another system" (ISO/IEC and IEEE, 2021, p. 20))
4. **Link Testing:**
   - Branch Testing (implied by ISO/IEC and IEEE, 2021, p. 24)
   - Component Integration Testing (Kam, 2008, p. 45)
   - Integration Testing (implied by Gerrard, 2000a, p. 13)

Acknowledgment

- Dr. Smith and Dr. Carette have been great supervisors in the past and have, both then and now, provided me with valuable guidance and feedback
  - They have helped me refine the scope of this project
  - Dr. Smith first suggested generating test cases back in 2020!

Acknowledgment

- Dr. Smith and Dr. Carette have been great supervisors in the past and have, both then and now, provided me with valuable guidance and feedback
  - They have helped me refine the scope of this project
  - Dr. Smith first suggested generating test cases back in 2020!
- The format of this presentation was *heavily* based on a previous presentation by Jason Balaci, who also provided a great thesis template

Acknowledgment

- Dr. Smith and Dr. Carette have been great supervisors in the past and have, both then and now, provided me with valuable guidance and feedback
  - They have helped me refine the scope of this project
  - Dr. Smith first suggested generating test cases back in 2020!
- The format of this presentation was *heavily* based on a previous presentation by Jason Balaci, who also provided a great thesis template
- The past and current Drasil team have created a truly amazing framework!

Thank you!

Questions?

References I

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

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.

Matthias Hamburg and Gary Mogyorodi, editors. ISTQB Glossary, v4.3, 2024. URL https://glossary.istqb.org/en_US/search.

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 2021. 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.

Knüvener Mackert GmbH. *Knüvener Mackert SPICE Guide*. Knüvener Mackert GmbH, Reutlingen, Germany, 7th edition, 2022. ISBN 978-3-00-061926-7. URL https://knuevenermackert.com/wp-content/uploads/2021/06/SPICE-BOOKLET-2022-05.pdf.

LambdaTest. What is Operational Testing: Quick Guide With Examples, 2024. URL https://www.lambdatest.com/learning-hub/operational-testing.

References IV

Pranav Pandey. Scalability vs Elasticity, February 2023. URL `https://www.linkedin.com/pulse/scalability-vs-elasticity-pranav-pandey/`.

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`.

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.

References V

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].

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 4.0*. January 2024. URL `https://waseda.app.box.com/v/SWEBOK4-book`.