Software Testing: Test-Driven Development

Engineering Software as a Service – Chapter 8

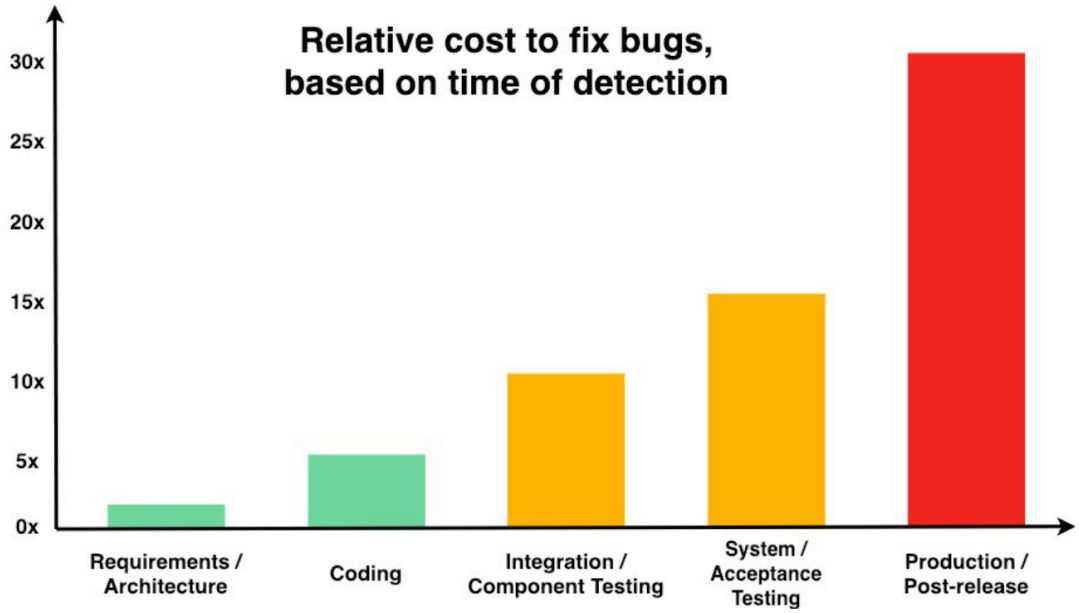
Software Testing

Test-Driven Development:

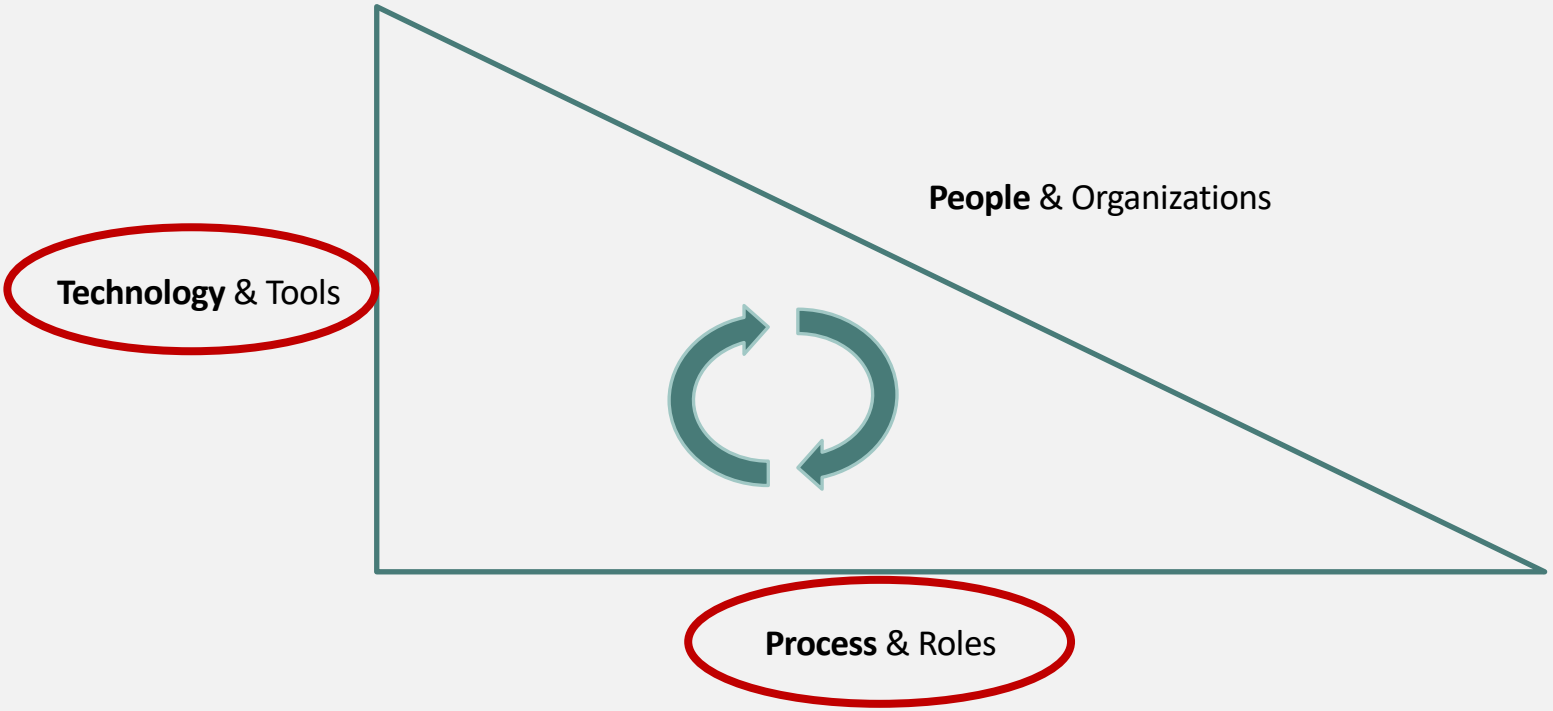
- 8.1 Verification and Validation

The Cost of Fixing a Defect Increases Exponentially

The following graph courtesy the NIST helps in visualizing how the effort in detecting and fixing defects increases as the software moves through the five broad phases of software development.



The **Righteous** Triangle of Software Development



Waterfall vs Iterative vs Agile Testing

	Waterfall	Iterative	Agile
References	United States Department of Defense: DOD-STD-2167A (1985)	Rational Unified Process (RUP) Open Unified Process	Scrum Kanban Scaled Agile Framework (SAFe)
Priorities	Planning and predictability	Architecture, modeling, and efficiency through early detection & fixing of issues (verification)	Responsiveness to feedback, efficiency through engineering practices, early detection & fixing of issues, and validation
Principles	Execute phases sequentially: 1. Requirements 2. Analysis 3. Design 4. Coding 5. Testing 6. and Operations Define and commit to Scope, Cost, and Timeline "early" Implement strict Change Control	Develop and test iteratively Manage requirements Use components Model visually Verify quality Control changes	Develop, test, deploy, and release iteratively Capture lightweight near term requirements Empower teams Allow requirements to evolve but maintain fixed timelines Apply engineering practices and systems thinking (e.g. TDD) Integrate early user feedback into remaining plan Maintain a collaborative approach between all stakeholders

Software Testing

Consider:

The goal should not be better testing. It should be avoiding defects in the first place, and then finding and fixing those unavoidable defects sooner.

You can’t afford to test in quality. Developers must be responsible for the technical product quality and defects.

Software Testing

Test-Driven Development:

- 8.2 FIRST, TDD, and Red-Green-Refactor

Software Testing

Test-Driven Development:

- 8.3 Seams, Doubles, and the Code You Wish You Had

Software Testing

Test-Driven Development:

- 8.4 Expectations, Mocks, Stubs, and Example Setup & Teardown
- 8.5 Fixtures and Factories

Software Testing

Test-Driven Development:

- 8.6 Implicit Requirements and Stubbing the Internet
- 8.7 Coverage Concepts and Unit vs. Integration Testing

Software Testing

Test-Driven Development:

- 8.6 Implicit Requirements and Stubbing the Internet
- 8.7 Coverage Concepts and Unit vs. Integration Testing

Software Testing

Test-Driven Development:

- 8.8 Other Testing Approaches and Terminology
- 8.9 The Plan-And-Document Perspective

Software Testing

Final Thoughts:

1. Web Services and APIs
2. Test Data
3. Automated Testing & False Positives
4. Logging
5. Internal Testing and Test Cases as Product Documentation
6. Maybe Test-Driven Development is more of a spectrum and evolution

Software Testing: Test-Driven Development

Engineering Software as a Service – Chapter 8