ENHANCING TEST MANAGEMENT AND QUALITY

THE ROLE OF POWER DYNAMICS IN SOFTWARE QUALITY

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
August Diamond

October 27, 2023

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Rider Foley, Department of Engineering and Society

Rosanne Vrugtman, Department of Computer Science

Introduction

In recent decades, the demand for computer software has risen massively. Humans now turn to computers for things that would have been inconceivable a mere generation ago – from hosting millions of concurrent real-time video conversations during a global pandemic (Karl et al., 2021), to large language models being used to write articles (Hosseini et al., 2023). The percentage of United States residents who own a computer has expanded from approximately 20% three decades ago to 92% as of 2018 (Martin, 2021). However, low quality software threatens to bring about a set of new problems. A 2020 survey of office workers indicated that 46 minutes per day may be wasted on slow software (Priestly, 2020). Given our increasing reliance on computing, ensuring software quality has become an important focus of computer scientists. Central to discussions of software quality is the topic of software testing, which remains the primary technique through which computer scientists research the functionality of software (Salahirad et al., 2023) and top industry engineers measure product reliability (Alshahwan et al., 2023).

I was brought into a private Charlottesville-based software startup to take part in the software testing process as a quality assurance (QA) intern. The leading product of this company was a highly dynamic, real-time financial data monitoring application used to inform the decisions of stock market investors. It was important to clients of this company that all features of this application worked as expected, giving the project strong functional requirements, e.g., that pressing the button to view a stock would correctly open a chart window, and that the data in that chart would be correct. The project also had performance-based requirements, particularly relating to the speeds at which updates to market data would be delivered, as delays in information transmission could result in financial losses to clients (Kenton, 2021). Requirements

relating to performance, security, and other domains that are not themselves specific functionalities of an application are broadly categorized as “non-functional requirements” in the software testing field (Hooda & Singh Chhillar, 2015). Over the years spent creating and maintaining this product, the company developed a large suite of functional and non-functional tests addressing these requirements, but many of these tests were not yet automated due to the opportunity costs of the time-consuming test automation process (Ramler & Wolfmaier, 2006).

The primary goal of the internship was to increase the efficiency of the product’s software tests. Engineers felt too much of their time was being spent on manually executing test cases, and higher-up employees disliked the expenses incurred by lengthy manual testing, in addition to problems of test suite scalability as the functionality and adoption of their live-service application grew. Clients, though not directly involved in these decisions, would also enjoy updates being deployed faster and with less errors if testing was more efficient. Though reasons often differ between stakeholders, working towards improved quality assurance in software should be to the benefit of all parties involved.

REVAMPING THE TEST SUITE

There were four technical objectives in the internship. First, and most importantly, the quality assurance team wanted to move their large test suite from an older platform to a more feature-rich test management tool. The new tool supported modular separation of tests into steps and phases, each of which could be marked as passing or failing. This allowed for more precision in determining the point of failure. It also supported separating test results by system (e.g., operating system version, monitor configuration, etc.) to ensure the product functioned as expected with the different architectures our clients may be using.

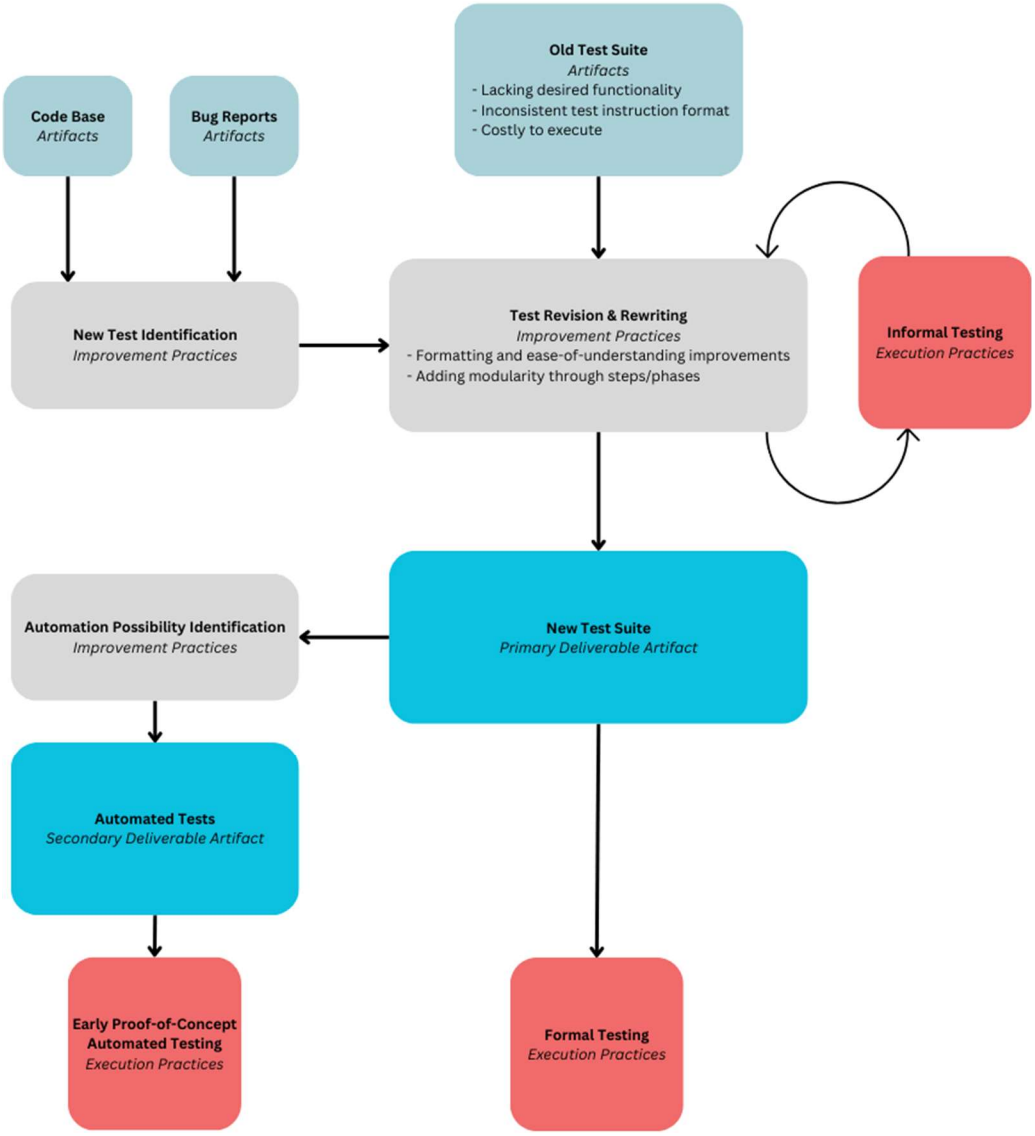

Figure 1. A relational overview of relevant artifacts and practices (Source: Diamond 2023).

Second, the old manual tests were to be rewritten in a standardized format, such that distinct steps were separated where possible, instructions were updated to reflect the current structural flow of the software, and tests were generally easier to parse. It should be noted that a substantial portion of the test cases we dealt with as interns used human test oracles, i.e., tests requiring humans for both execution and analysis. This tends to be the most expensive form of software testing, as was the case at this company. But for the tests that involved manipulating the

complex user interface (UI) of this application, there was often no immediate alternative. Instead of replacing these tests, we focused on quantitative and qualitative human oracle cost reduction, terms which broadly describe techniques used to reduce the amount of labor required to execute and analyze manual tests (Barr et al., 2015). Third, we analyzed the product's bug report database and the application's source code to determine the need for new test cases, which we would then create. The final, longer-term objective was to assist the senior QA engineers in automating tests with Selenium, an automation framework that can be used to test UIs.

There were three categories of testing performed during the internship: informal testing during the revision process, formal testing for correctness, and early automated testing. In the first and most common form, members of the QA team would informally run tests ourselves while in the process of (re)writing instructions. We would use observations made during test execution to further improve the instructions. After a group of tests had been added to the new management service, other software engineers outside of the QA department would run them and formally log their results using the test management system. This was performed less frequently due to the higher cost, and was done to test the system itself rather than to test the quality of our test instructions. Finally, the senior QA staff would occasionally demonstrate the Selenium-based automated tests they had been working on to us and other employees. These would often be accompanied by metrics about time efficiency and pass/failure rate.

The latter two test execution methods were not only parts of the internal QA process, but were means for interacting with other, non-engineer stakeholders. Formal test results would be reviewed by company higher-ups to ensure that the product was developing in a timely manner and would be delivered on schedule to paying users, who themselves value and expect frequent

updates. The metrics given during automated test demonstrations were similarly well-received by higher-ups as indicators of progress towards reducing operational costs.

STAKEHOLDERS AND SOFTWARE QUALITY

The significance of software testers is often poorly understood, even within the software industry itself (Florea et al., 2023). But neglecting the quality assurance process can have significant consequences. In the U.S. alone, it was estimated that nationally, issues with poor-quality software have grown to cost at least $2.41 trillion as of 2022 (Krasner, 2022). In the worst cases, improperly tested software costs not money, but lives. Four years ago, two Boeing-737 Max crashes resulted in 346 deaths after a software component on the planes failed (Ethiopian Civil Aviation Authority, 2022). While software itself was most directly responsible for the accident, there were many social factors at play that led to its failure. The company shareholders and higher-ups, in this case the Boeing executives, desire for growth at minimal cost led to inexperienced, underpaid coders developing crucial parts of their system (Robison, 2019). The users, in this case the pilots, desired a smoother flying experience, which Boeing's new software was intended to provide. And indirect users affected by the system without directly using it, i.e., passengers, want flights to be quick, cheap, safe, and perhaps above all, frequent. Boeing's own analysis of the growing demand for commercial airplanes (Bergman, 2018) likely contributed to their decision to rush out more products. I find this case study to exemplify some of the common stakeholder archetypes that others have identified in software systems (owners, higher-ups, engineers, and users), the oft-conflicting desires between them (Kroeger et al., 2014), and the disastrous results of failing to properly mediate their relationships with and influence over software artifacts.

Social construction of technology, or SCOT, (Pinch & Bijker, 1984) provides a social constructivist framework for analyzing how human actions shape the creation and development of technology. SCOT can be employed to consider how the quality of software is influenced by the relationships between stakeholders. When polled, QA teams themselves unsurprisingly tend to prioritize quality over speed and cost (Katalon et al., 2023), while executives may find the latter factors more important. This difference in perspective is the result of *interpretive flexibility*, through which the artifact of software takes on a different meaning to different stakeholders. From the perspective of owners and shareholders, the end-goal of software is to make the company money. To developers, creating software is a job that delivering high-quality code in an ethical manner helps them maintain (Gotternbarn et al., 1997). Through the concept of symmetry, differences in stakeholder values have real-world implications not only for the artifact, but for other stakeholders. For example, the value of rapid development and update cycles in software, imposed primarily by managers and users, has led to research showing that higher-quality code is produced under constantly tight deadlines as compared to occasional time pressure (Basten et al., 2021). Implicit in this research is that through software as a technological artifact, values are imposed by other stakeholders onto engineers themselves.

RESEARCH QUESTION AND METHODS

With computers influencing more human lives than ever before, it becomes pertinent to ask: How does the balance of power between shareholders, executives, and engineers correlate with software quality? To answer this question, I will compile a list of publicly traded companies that have software products with at least 10 entries in the U.S. National Cybersecurity FFRDC's open-access Common Vulnerabilities and Exposures (CVE) database. The CVE database lists

over 215,000 computer security flaws identified in various software applications, and each entry is accompanied by a Common Vulnerability Scoring System (CVSS) Base Score, which is a ten-point standardized indicator of vulnerability severity. For each company, I will take the average CVSS score of all submitted entries as an approximate measure of software quality. I will then gather the average ratings from and salaries of engineers at these companies on the employer review website Glassdoor to quantify how these companies value engineers. Next, I will examine the Securities and Exchange Commission filings from these companies to determine the pay, or perceived value, of executive staff. And finally, I will gather CVE and Glassdoor data from a sample of private companies to attempt to isolate the effects of shareholder involvement (with consideration to be made in my final report for the lack of executive compensation data for this group). I will plot this data and perform correlational analysis to determine the relationships between the different metrics. Drawing on the relationship defined by SCOT between stakeholder influence and the course of a given technology's development, the primary focus of my analysis will be on the relationship between the three stakeholder power metrics and the software quality metric.

CONCLUSION

When developing real-time financial applications, the properties of accuracy, timeliness, and consistently high software quality are extremely valuable. Through the standardization, expansion, and gradual automation of their older test suite, the company I completed my internship with sought to optimize their software testing process to ensure these properties at lower operational costs.

In our increasingly computerized world, software quality can be the difference between material gain and loss, occupational productivity and wastefulness, or even human life and death. It is more crucial than ever that we seek to understand not only the technical, but the social circumstances that surround the creation of high-quality software. To this end, I hope to determine the correlation between stakeholder power balance and software quality through my research.

REFERENCES

Alshahwan, N., Harman, M., & Marginean, A. (2023). Software Testing Research Challenges: An Industrial Perspective. *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*. https://doi.org/10.1109/icst57152.2023.00008

Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., & Yoo, S. (2015). The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering, 41*(5), 507–525. https://doi.org/10.1109/tse.2014.2372785

Basten, D., Müller, M., Ott, M., Pankratz, O., & Rosenkranz, C. (2021). Impact of time pressure on software quality: A laboratory experiment on a game-theoretical model. *PLOS ONE, 16*(1), e0245599. https://doi.org/10.1371/journal.pone.0245599

Bergman, P. (2018, July 17). *Boeing Forecasts $15 Trillion Commercial Airplanes and Services Market - Jul 17, 2018*. MediaRoom. https://boeing.mediaroom.com/2018-07-17-Boeing-Forecasts-15-Trillion-Commercial-Airplanes-and-Services-Market

Ethiopian Civil Aviation Authority. (2022). Investigation report on accident to the B737-MAX8 Reg. ET-AVJ operated by Ethiopian Airlines. In *Bureau of Enquiry and Analysis for Civil Aviation Safety*. https://bea.aero/fileadmin/user_upload/ET_302__B737-8MAX_ACCIDENT_FINAL_REPORT.pdf

Florea, R., Stray, V., & Sjoberg, D. (2023). On the roles of software testers: An exploratory study. *Journal of Systems and Software*, *204*, 111742–111742. https://doi.org/10.1016/j.jss.2023.111742

Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. *Communications of the ACM*, *40*(11), 110–118. https://doi.org/10.1145/265684.265699

Hooda, I., & Singh Chhillar, R. (2015). Software Test Process, Testing Types and Techniques. *International Journal of Computer Applications*, *111*(13), 10–14. https://doi.org/10.5120/19597-1433

Hosseini, M., Rasmussen, L. M., & Resnik, D. B. (2023). Using AI to write scholarly publications. *Accountability in Research*, 1–9. https://doi.org/10.1080/08989621.2023.2168535

Karl, K. A., Peluchette, J. V., & Aghakhani, N. (2021). Virtual Work Meetings During the COVID-19 Pandemic: The Good, Bad, and Ugly. *Small Group Research*, *53*(3), 104649642110152. https://doi.org/10.1177/10464964211015286

Katalon, Cigniti, & Deloitte. (2023). *State of Software Quality 2023*. Katalon. https://katalon.info/hubfs/download-content/report/2023/state-of-software-quality-2023.pdf

Kenton, W. (2021, August 13). *What Is Real Time?* Investopedia. https://www.investopedia.com/terms/r/real_time.asp

Krasner, H. (2022). The Cost of Poor Software Quality in the US: A 2022 Report. In *CISQ* (p. 3). Consortium for Information & Software Quality. https://www.it-cisq.org/wp-content/uploads/sites/6/2022/11/CPSQ-Report-Nov-22-2.pdf

Kroeger, T. A., Davidson, N. J., & Cook, S. C. (2014). Understanding the characteristics of quality for software engineering processes: A Grounded Theory investigation. *Information and Software Technology, 56*(2), 252–271. https://doi.org/10.1016/j.infsof.2013.10.003

Martin, M. (2021). *Computer and internet use in the United States: 2018 American Community Survey Reports*. https://www.census.gov/content/dam/Census/library/publications/2021/acs/acs-49.pdf

Pinch, T. J., & Bijker, W. E. (1984). The Social Construction of Facts and Artefacts: Or How the Sociology of Science and the Sociology of Technology Might Benefit Each Other. *Social Studies of Science, 14*(3), 399–441. http://www.jstor.org/stable/285355

Priestly, T. (2020). *Tech Troubles: The true cost of outdated workplace tech*. Techtalk.currys.co.uk; Currys, AMD. https://web.archive.org/web/20210210093723/https://techtalk.currys.co.uk/computing/workplace-productivity/

Ramler, R., & Wolfmaier, K. (2006). Proceedings of the 2006 international workshop on Automation of software test. *AST '06: Proceedings of the 2006 International Workshop on Automation of Software Test*. https://doi.org/10.1145/1138929

Robison, P. (2019, June 29). *Boeing's 737 Max Software Outsourced to $9-an-Hour Engineers*. Bloomberg.com. https://www.bloomberg.com/news/articles/2019-06-28/boeing-s-737-max-software-outsourced-to-9-an-hour-engineers

Salahirad, A., Gay, G., & Mohammadi, E. (2023). Mapping the structure and evolution of software testing research over the past three decades. *Journal of Systems and Software, 195*, 111518. https://doi.org/10.1016/j.jss.2022.111518