Automated Testing of Electronic Health Records

Determining the Impact of Testing Practices on Healthcare Technologies

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
Vance Elliott

November 8, 2024

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
Brianna Morrison, Department of Computer Science

Introduction

As the world shifts to be increasingly reliant on computers, the need to ensure software quality is spotlighted. While testing strategies continue to develop, and the number of quality assurance employees grows at a steady rate, now almost at 200,000 employees, software is still plagued by defects and vulnerabilities (Zippia, 2024). When a piece of software plays a crucial role in society, this can be a very dangerous problem (Joseph et al., 2024). One example of this is the most recent Microsoft outage, caused by a defect in a security system called CrowdStrike. Due to a lack of sufficient testing architecture, a faulty update was allowed to be sent out globally, resulting in the crashing of many windows systems (Cohen, 2024). This error is estimated to have cost businesses billions of dollars (Davis, 2024). Thus, software has a large impact on the world, but a lack of a significant testing infrastructure that allows this technology to succeed.

Particularly when it comes to electronic health records (EHRs), poor testing frameworks can even lead to the loss of life (Bowman, 2013). The National Institute of Standards and Technology reported 18,738 defects that they alone discovered in 2021, and as the amount of software increases, the number of vulnerabilities will only trend upwards (National Institute of Standards and Technology, 2022). Software defects are often reported as the cause of an accident in hospital incident reports (Howe et al., 2018). Thus, there is an urgent need for healthy testing practices.

When it comes to testing software, there are many important principles and procedures that lead to quality software. One of these terms is Test Driven Development (TDD), which refers to a method of developing software that incorporates testing at every stage of the

development process (Baresi et al., 2006). When using TDD, one will write a suite of test cases, essentially the requirements that the developer wants their code to complete, and then writes code until all of those tests pass. Rather than just testing after the software is finished, a technique called Unit Testing, TDD ensures that software is working from the very beginning of the development process. This process creates more efficient and effective software with one study from North Carolina State reporting that 92% of developers believe that TDD yields better code, 79% thought TDD results in simpler design, and 71% thought that it was noticeably effective (George & Williams, 2003).

While most developers recognize that this is a more effective strategy, few actually put it into practice. One survey conducted in 2020 by Vanson Bourne, a research agency, found that only 41% of developers practice TDD. When asked whether they write test cases before they write code, the very definition of Test Driven Development, only 8% of developers said yes (Diffblue, 2020). This is due to TDD being a timely and difficult process (Parsa et al., 2025). Software developers are either too lazy or not educated enough about the benefits of TDD to implement this testing strategy, causing their code to have a significantly larger number of defects (Makinen & Munch, 2014). Due to a lack of effective testing strategies and architecture, crucial software to society, specifically healthcare software, continues to fail, harming the people who rely on these technologies (Howe et al., 2018). Through analyzing software through the framework of the Social Construction of Technology, the impact that ineffective testing practices have on the quality of healthcare software will be studied.

Automated Testing

One technology that has been developed to combat testing boredom and laziness is test automation. This is a process in which routine tests, that don't involve complex nuances that would need to be analyzed by a human, are conducted automatically by a computer. Test automation converts test scenarios that would need to be conducted by a human into a test suite that can run independently of the developers (Kumar & Mishra, 2016). Thus, when there are any updates to the code base, one can just run the suite, and all of the simple, rudimentary tests are conducted automatically to ensure that the software is working properly. I was able to experience the benefits of test automation firsthand when I implemented the technology into a company's testing workflow.

From May to August 2023, I worked in-person as a quality assurance intern for the Physician's Computer Company (PCC) located in Burlington, Vermont. Founded in 1983, the company designs EHRs for pediatric practices (PCC, 2024). Their software not only includes medical records, but an appointment scheduler, billing management, prescription tools, and many more useful features that pediatricians need for their businesses. However, their software is very large and complex, and thus, contains many bugs that go unnoticed until reported by hospitals. I was assigned the task of "quality assurance intern", instructed to learn many of their old testing scenarios, conduct them on the EHR, and then post them on a project management tool called Trello that would allow others to test them once I was gone. After studying their testing procedures, I realized how inefficient parts of their system were and spearheaded a test automation project. This allowed some of the more menial tasks to be done by a computer, while the more important, in-depth tasks could be looked over more carefully by actual employees at PCC.

In order to conduct this task, I used an application named Squish. I would code a set of instructions into the program about what to do in PCC's EHR, such as "click this button" or "hover mouse in this position", and Squish would carry out the instructions much faster than a human could. I was able to create a whole suite of these instructions, so that when an update was rolled out by a developer, one could just press play and run each of these menial tasks automatically. PCC had a "testing day" where every employee of the company, whether they were a developer, in sales, in human resources, etc., would come together and do these tests manually after an update. This testing automation project reduced a significant number of tests that had to be done by hand, giving back time to employees, while also assuring quality of the tests, as a number of the employees conducting tests were not trained in the software, and thus were more prone to errors while testing. When it comes to an EHR, defects could cause serious harm to patients, so it was important that the testing process be as effective as possible (Pew Charitable Trusts, 2018). Through this internship, I was able to learn the importance of software testing and come alongside PCC to discover solutions to poor testing strategies.

Sociotechnical Analysis

Analyzing the testing processes of a company, one can see how the influences of different social groups guided the creation of different tests and the extent to which they were conducted (Baresi & Pezzè, 2006). For example, with PCC's EHR, the most tested feature was the appointment scheduler, even though it was largely simpler than the other aspects of the EHR. While this was a relatively small application, it was the most visible tool used by doctors and patients. This means that if there were any bugs or defects, they would be discovered more often than the more hidden parts of the EHR, resulting in these bugs being reported more often and

inflated the amount the appointment scheduler was tested. This idea of social groups informing technology was proposed first by Wiebe Bjiker, who alongside Trevor Pinch, named this framework "The Social Construction of Technologies" (Bijker et al., 1987). Using the invention of the bicycle, Bjiker proposed that technologies can be described by the influence of the social groups surrounding them. One can look at the forces of different stakeholders and see how their interests and desires molded the technology. This is a useful framework to keep in mind as more testing processes are analyzed. Poor testing strategies are often a result of a lack of social pressure to test, causing companies to cut corners as they do not have ample incentives to test well. Because of deficient motivation for quality assurance employees, and the growth in complexity of testing software, there is an alarming rise in software defects and vulnerabilities that negatively impact people, a problem that requires an urgent solution (Howe et al., 2018).

Bjiker's "Social Construction of Technology" will allow an in-depth, thorough analysis of the effectiveness of a testing system. By looking at the surrounding stakeholders of a company, one will be able to determine the quality of a company's software. For example, at PCC, one of the surrounding social groups of the business was the United States government. This is because PCC's software handles sensitive medical information, and the government desires to protect that data, passing many laws and regulations to promote information privacy such as the Health Insurance Portability and Assurance Act in 1996, otherwise known as HIPAA (U.S. Department). Because of the stricter policies surrounding PCC's EHR, the company tests more rigorously and thoroughly than most companies. Thus, using the social construction of technology, one can look at PCC's invested stakeholders, determine whether testing is an important value amongst any of the groups, and then accurately predict the emphasis PCC places on testing.

Biker's theory will help with the correct analysis of other companies' testing systems, particularly when the exact internal structure of a company is unknown. One can look at the surrounding pressures of the company, such as government regulation, importance of quality to consumers, standards enforced by software vendors, etc., and then can determine if there is sufficient interest in testing from stakeholders that will allow the company to succeed. Thus, when engineering a solution to bad testing architecture, one can look at the root of the problem, the stakeholders, rather than the symptom of the problem, poor testing practices. One can seek to implement good testing strategies such as Test Driven Development, but without the necessary motivation from social groups, employees will be slow to execute these strategies, shown by the previously described hesitation to TDD from software engineers. Therefore, by using Bjiker's Social Construction of Technology, we can analyze a testing system in a helpful way, not ignoring the outside stakeholders, but rather incorporating them into the analysis to create more robust, effective solutions.

Research Question and Methods

As the relation between healthcare software and patient care is analyzed, one question will serve as the focal point for the paper: How do current testing regulations impact the quality of medical software? In order to answer this question, a content analysis will be conducted on the University of Virginia Hospital's public incident report available to students, determining how many incidents are related to defects in EHR's. This will help to determine the severity of the problem. Then, using SCOT, I will attempt to determine the current testing methods commonly used by healthcare software companies. As the precise working structure of companies will be difficult to observe, SCOT will become a very useful tool in this analysis. Rather than looking at

a company's internal process, I will look at commonly invested stakeholders of software companies to determine the efficacy of modern testing systems. One large stakeholder in software quality is government, so I will look at the six major policy software compliance standards: the General Data Protection Regulation (GDPR), ISO 27001, HIPAA, the Payment Card Industry Data Security Standard (PCI DISS), SOC 2, FedRAMP, and the Federal Information Security Management Act (FISMA) (Ali 2024). Using SCOT, as well as my previous internship experience, I will analyze the strictness and effectiveness of each compliance standard, analyzing how these policies and regulations influence testing systems. Through viewing public health incident reports, as well as different compliance standards, the impact of testing regulations on software quality will be determined, leading to the proposal of new solutions for more effective code regulation.

Conclusion

Electronic health records, and other healthcare technologies, have a large responsibility to ensure the well-being of their users. This analysis should determine how specific EHRs are performing, as well as how general testing procedures and regulations are impacting the healthcare world. The results of this paper should be a helpful guideline for how to most effectively test software, pulling from the strengths of different EHRs and constructively criticizing the weaknesses of EHRs to create a well-informed testing framework. The expected results of the paper are that software has a significant impact on healthcare, and that EHRs are not testing at the level that they should be. We have seen from previous studies mentioned above that test driven development is not popular amongst developers, and that hospitals' incident

reports reference defects in EHRs, so this paper should also reflect that, hopefully providing even more insight into how this problem might be fixed.

Resources

Ali, N. (2024, June 3). *Navigating the maze: What you need to know about software compliance standards*. Beagle Security. Retrieved from https://beaglesecurity.com/blog/article/software-compliance-standards.html

Baresi, L., & Pezzè, M. (2006). An introduction to software testing. *Electronic Notes in Theoretical Computer Science*, 148(1), 89-111. https://doi.org/10.1016/j.entcs.2005.12.014

Bijker, W. E., Pinch, T. J., & Hughes, T. P. (Eds.). (1987). *The social construction of technological systems: New directions in the sociology and history of technology*. MIT Press.

Bowman S. (2013). Impact of electronic health record systems on information integrity: quality and safety implications. *Perspectives in health information management*, *10*(Fall), 1c.

Cohen, J. (2024, July 23). CrowdStrike's faulty Windows update causes BSOD issues. The Verge. https://www.theverge.com/2024/7/23/24204196/crowdstrike-windows-bsodfaulty-update-microsoft-responses

Davis, S. (2024, February 12). Explaining the largest IT outage in history and what's next. TechTarget. https://www.techtarget.com/WhatIs/feature/Explaining-the-largest-IToutage-in-history-and-whats-next

Diffblue. (2020, May 21). *2020 DevOps and testing report*. Diffblue. https://www.diffblue.com/resources/2020-devops-and-testing-report/

General Data Protection Regulation (GDPR). (2018, May 25). *GDPR info*. Retrieved December 6, 2024, from https://gdpr-info.eu/

George, B., & Williams, L. (2003). An initial investigation of test driven development in industry. *Proceedings of the 2003 ACM Symposium on Applied Computing* (SAC '03), 1135–1139. https://doi.org/10.1145/952532.952753

Howe, J. L., Lammers, E. J., & Baker, S. (2018). Electronic health record usability issues and potential contribution to patient harm. *Journal of the American Medical Association, 319*(12), 1276–1278. https://doi.org/10.1001/jama.2018.1171

Joseph, Sb & Ness, Pedro. (2023). The Relationship Between Computers and Society, Impacts, Challenges, and Opportunities.

Khan, N. (2024). *What are the top 5 EHR systems?* Folio3 Software Inc. https://digitalhealth.folio3.com/blog/what-are-the-top-5-ehr-systems/

Kumar, D., & Mishra, K. K. (2016). The impacts of test automation on software's cost, quality, and time to market. *Procedia Computer Science, 79*, 8–15. https://doi.org/10.1016/j.procs.2016.03.003

Makinen, Simo & Münch, Jürgen. (2014). Effects of Test-Driven Development: A Comparative Analysis of Empirical Studies. Lecture Notes in Business Information Processing. 166. 10.1007/978-3-319-03602-1_10.

National Institute of Standards and Technology. (2022). *CVSS severity distribution over time*. National Vulnerability Database. https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cvss-severity-distribution-over-time#CVSSSeverityOverTime

Parsa, S., Zakeri-Nasrabadi, M., & Turhan, B. (2025). Testability-driven development: An improvement to the TDD efficiency. *Computer Standards & Interfaces, 91*, 103877. https://doi.org/10.1016/j.csi.2024.103877

PCC. (n.d.). PCC pediatric EHR solutions. Retrieved September 29, 2024, from https://www.pcc.com

Pew Charitable Trusts. (2018, August 28). *Ways to improve electronic health record safety*. https://www.pewtrusts.org/en/research-and-analysis/reports/2018/08/28/ways-to-improve-electronic-health-record-safety

U.S. Department of Health and Human Services. (n.d.). *Health Information Privacy (HIPAA)*. Retrieved December 4, 2024, from https://www.hhs.gov/hipaa/index.html

Zippia. (2024, April 5). *Software tester demographics and statistics in the U.S.* Zippia. https://www.zippia.com/software-tester-jobs/demographics/