Information and Cyber Security: Automating the Quality Assurance Process

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Jack Warner

Fall, 2022

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

Rosanne Vrugtman, Department of Computer Science

Briana Morrison, Department of Computer Science

Information and Cyber Security: Automating the Quality Assurance Process

CS4991 Capstone Report, 2022
Jack Warner
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA

Abstract

A Fortune 50 bank familiarizes its interns with the code base of their product in the field of Information and Cyber Security by giving them the time consuming, monotonous work of Quality Assurance. By effectively understanding how the product works and the expected results, the process can be automated using a Python script and two Shell scripts. The product in question is utilized in Shell and produces an output based on the user inputs. All possible combinations of possible inputs must be tested, but by utilizing the script, the time required for QA can be decreased dramatically. The more comprehensive testing of input combinations results in better test coverage, leading to more bug-finding. A possible future implementation would be altering the script to be more user-friendly.

1. Introduction

Back in 2015, the new top-of-the-line tablet called Innotab, designed by VTech, was all the rage among parents. These kid-friendly smart devices have the ability to take pictures, store information, connect to the internet, and send messages to parents' phones.

A hacker decided to tinker with the tablet, as there are many active online communities that dedicate themselves to hacking Internet of Things devices. As he familiarized himself with the device, he noticed one of the VTech websites was vulnerable to SQL injection. With that knowledge, he received root access in almost no time. Since the hacker now has obtained the highest level of authorization privilege, he is able to find databases containing millions of pictures, messages, and personal data of children.

With the poor implementation of quality assurance, newly-created devices and applications will have significantly more vulnerabilities once launched. This begs the question regarding quality assurance, an integral part of the software development process: What are the ethical implications of automating the QA process? Is it even practical?

2. Background

Software can be simply defined as a set of instructions able to accomplish a certain task. Software quality assurance, as defined by the Institute of Electrical and Electronics Engineers is: "A set of activities that define and assess the adequacy of software processes to provide evidence that establishes confidence that the software processes are appropriate for and produce software products of suitable quality for their intended purposes" (Laporte, 2018).

Many aspects go into software quality

assurance. Aspects such as security vulnerabilities, test coverage, expected results, and many more, depend on the application that is being tested. SQA focuses on the maintenance and evolution of the software. Software maintenance is one of the most important if not the most costly part of any software's life cycle.

With software, maintenance come updates and improvements to the software. Keeping the software updated reduces possible vulnerabilities, and creates a better product immediately, not to mention the performance improvements that increase customer satisfaction in the long run.

Another important element of quality assurance is the protection of vulnerabilities and possible breaches. If an application experiences a breach due to an unforeseen vulnerability, the company that produced the software will lose the customer's trust, as well as have to allocate resources to make sure the problem is fixed and never happens again. It is a lot more costly to fix a problem than to prevent one.

3. Related Works

Today software is becoming more and more prevalent and the turnaround time for product development is shrinking. Dustin, et. al (1999) found that more than 90% of developers have missed ship dates, and missing deadlines is a routine occurrence for 67% of developers.

On top of expenses, businesses must find a way to keep costs down as well as streamline the development process in order to meet deadlines. This is where the possibility of automating software testing comes into play. Reusable scripts may be run as many times as needed, offering significant payback.

The main challenge regarding testing software is that customers as well as companies want more software functionality to be delivered faster as well as cheaper; however, the quality of the software must be on par with modern capabilities. A higher functionality software means each part of the software becomes extraordinarily more complex, requiring developers to run more test cases for each delivery. Faster deliveries mean less time to test and develop. Because one often cannot control and automate the development process, the time has to be cut from the testing portion of the production. As Dustin, et. al., (2009) point out, the more functionality provided the more tests are required and the longer it takes to deploy.

4. Process Design

After initial onboarding, the team I worked with decided the best way to introduce interns to the code base is to have us fully understand the functionality of the application (QA). Working with our mentor we were able to understand the application fully from a black-box testing perspective. We worked in an agile workflow environment, meaning every day we would have standup meetings discussing what we had accomplished the day prior and what we expected to accomplish later that day or that week.

Agile workflow provides the team with a "story" point system, in which all tasks are assigned a certain value depending on how long they would take to accomplish. The workflow is almost always development, code review, quality assurance (test generating), and quality assurance (test execution). In general, both parts of quality assurance would generate three story points, meaning it would take between 2-3 days to

accomplish the task. If at any stage after development a bug is found, the ticket would have to be renewed and would have to go back to the development phase, starting the process in its entirety all over again. For all tests executed and generated in the quality assurance process, if it goes back to development, all tests must be rerun; no old tests can be carried over.

The application we tested had to deal with registering and deregistering hosts, which presents many flags, such as hostname, IP, verbosity, template, OS, etc. All the possible combinations have to be tested. Some flags are required, while others are optional. The way we designed the automation is to have a python script capable of reading text document input, containing tests to be run. The script parses the document line by line and generates a shell script able to call the application to register or deregister.

Along with the script, there are two shell scripts. One shell script registers dummy hosts in order to test the deregisters function; the other shell script deregisters all the hosts that have been registered so it is a clean-up file after testing the register function. There are still many edge cases to be tested manually. For example, the IP flag could take either an IPv4 address or an IPv6 address, so those also need to be taken into consideration.

5. Results

Utilizing the automation script described above, the team saw a significant impact on total bugs caught as well as a decrease in total time spent for each QA ticket. Instead of finishing one QA ticket every 2-3 days, the efficiency has increased to 2-3 tickets every day. The speed at which we were finishing the tickets caught the attention of our managers, who thought it was impossible to finish these tickets at such speed. Once we explained the script we had developed, our manager was extremely happy with the result. The script, which now belongs to the company, is now being utilized in the day-to-day QA process, as well as onboarding new team members.

6. Conclusion

The quality assurance process is an essential part of the software development life cycle. QA ensures a product or service meets the proper standards, expectations, and requirements of the users. It ensures a higher level of product quality by preventing product defects before the product is released to the users.

However, the process also has potentially detrimental outcomes. For instance, it is tedious and time-consuming, as one unexpected result can restart the entire software development life cycle for that aspect. It is not an unreasonable choice to automate the process but many factors need to be taken into consideration as automation is theoretically a product itself.

The benefit of automating this process is a faster software development life cycle, in addition to better-vetted code. However, the automation ultimately comes down to the QA engineer's fundamental knowledge and experience with the application and it is essential during the design phase of the automation that all aspects of the software are considered.

7. Future Work

In terms of the future development of the script, my personal impact has come to an end. The script we created is in no way perfect or polished. The script can be better implemented to include an automatic

screenshot process to increase the documentation process of QA. The script along with its two shell script counterparts can be merged into a single script. The manual deregistering and registering dummy node process can be automated, as well. Of course, as the application continues to grow, the automation needs to be able to pivot or it will become completely obsolete. Automating the QA process should be further discussed in any software development life cycle. The benefits of automation outweigh the cons; however, there needs to be a proper regulation where the automation needs to be kept in check so it does not hinder the overall software development life cycle.

References

Laporte, C. Y., April, A., Wiley Online Library UBCM Engineering, Wiley Online Library UBCM Math & Statistics, Wiley Online Library UBCM German Language, O'Reilly Online Learning: Academic/Public Library Edition, & Wiley Online Library UBCM All Obooks (2018). Software Quality Assurance (1 ed.). Hoboken, NJ: Wiley-IEEE Computer Society, Inc.

Dustin, E., Rashka, J., Paul, J., & O'Reilly Online Learning: Academic/Public Library Edition (1999). Automated Software Testing: Introduction, Management, and Performance. Boston: Addison Wesley Professional.

Jena, A. K., Das, H., Mohapatra, D. P., & Springer Intelligent Technologies and Robotics eBooks English/International (2020). Automated Software Testing Foundations, Applications and Challenges. S.l.: Springer Singapore.

Dustin, E., Garrett, T., Gauf, B., & O'Reilly Online Learning: Academic/Public Library Edition (2009). Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality. Upper Saddle River, NJ: Addison-Wesley.