Regression Testing from the Developers' Perspective

Hanwen Wang
260778557
hanwen.wang@mail.mcgill.ca

Yudong Zhou
260721223
yudong.zhou@mail.mcgill.ca

ABSTRACT
The focus of this paper is to report the regression testing approaches described by Stephane Parenti and Chao Wang during our interviews with them. Testing techniques in different companies and teams are discussed by comparing and contrasting. Furthermore, the advantages and disadvantages of each approach are inspected, with potential improvements suggested.

1. INTRODUCTION
During the software development process, many changes need to be made on the current software program. It involves creating new logic to correct an error or implementing a change and incorporating the new logic into the current program [1]. In order not to increase bugs or undesirable side-effects caused by those changes, regression testing is implemented. It is also done to make sure that a new code change does not affect the existing functionality of the product. The aim of this paper is to compare and contrast the approaches provided by two interviewees, identify strengths and weaknesses and suggest some improvements for each of approaches.

1.1 Stephane Parenti
Stephane Parenti graduated from Pierre and Marie Curie University in 2010, with a bachelor's degree in Computer Science. He has more than 10 years of working experience in companies like Renault, CS CANADA Inc. and Air Canada. He is now working as a Functional Safety Engineer at CS CANADA Inc. as the software team lead [2].

1.2 Chao Wang
Chao Wang graduated from McGill University in 2012 with a master's degree in Engineering (Aerodynamics in CFD). He also completed a master program in Computer Science in Concordia University after that. Chao Wang has experience working as a quality assurance analyst, a Java automation developer, a C++ developer and a cloud service Java developer. He is now working at MindGeek as a member in the data team [3].

1.3 More Information
This paper will consider Stephane Parenti and Chao Wang's description of regression testing according to their working experience in CS and MindGeek respectively. During the interviews, both interviewees not only answered general questions about the regression testing process, but also gave several stories about regression testing that happened in their work, including a success story, a failure story and a typical story.

2. COMPARING APPROACHES

2.1 Software Development Process
Regardless of the different types of companies and services provided, both software development processes start with a requirement requested by the client or the product owner. After that, the team identifies what new features should be added, which part of the design will be modified and what tests should be implemented. Then they use their own tools to do regression testing depending on the company, the team and the culture. After all, they report the result to the client or the product owner.

2.2 Quality Assurance
Quality assurance is very important in both software development processes. In order to assure the quality of new features or functions of a software, many regression tests are done before the update is finally released. The team has to do regression testing on modified modules and old modules until all tests are passed. If some test does not pass, they need to find the reason why the test fails and try to correct it. If it still fails, then they cannot add this new change to the new version of the product.

2.3 Safety Net
For both companies and both teams, the aim of regression testing is the same. As stated by Stephane Parenti, regression testing is a safety net, which aims to catch human mistakes [2]. Although developers can use automation tools during the development and testing processes, the impacts of the modifications are sometimes identified by themselves. Since errors can be made by humans while implementing new features or doing changes to current features, regression

testing can help developers to catch the bugs, so that they can make changes to the erroneous part of the code.

2.4 Reusability

Since regression testing not only tests whether newly implemented features are correct or not, but also tests whether the old features are still working as expected or not. Even if there is a change in the program, tests for the old features are reusable as long as their top-level requirements remain unchanged. Both interviewees mentioned this part that brings much convenience to their teams [2][3].

2.5 Test Tracking

As expressed by both interviewees, the test tracking process is tracked following a pipeline pattern [2][3]. At first, the requirements are analyzed and the new feature is identified. Then, all the tests are implemented and executed. If there is any test failing, the team needs to find where the problem is and fix it, otherwise, the new feature is not allowed to be deployed on the production environment.

3. CONTRASTING APPROACHES

3.1 Testing Tools

As mentioned by Chao Wang, the testing tools used in his team are mainly public testing frameworks such as unit testing and Selenium (a web testing framework) to test web APIs. Due to the characteristics of the services provided by Chao's company, the tests are mostly written to mimic the behavior of end users [3]. However, as described by Stephane Parenti, his team has internal tools which are based on CSV files. They also use some common tools such as Jenkins and Mantis Bug Tracker which are also different from the ones used in Chao's team [2]. The difference in testing tools can be caused by the different types of services provided by two companies. In Chao's company, the service is mainly for end users such as customers, so they have to pay attention to the experience of end users, thus putting emphasis on the simulation of end users' steps. While in Stephane's company, the service is mainly for industries such as Aeronautics, hence, there is less emphasis on user experience than in Chao's company.

3.2 Task distribution

In Chao's team, test developers are responsible for the test implementation and execution [3]. While in Stephane's team, the tests are written by one group of people and executed by another group of people [2]. The separation of work in Stephane's team enables an extra step to review the tests written, which can help to improve the test quality. However, it can also lead to increasing communication costs, as there may be discrepancy in test standards.

4. STRENGTHS

4.1 Stephane's Approach

4.1.1 Automation Tools

As mentioned by Stephane Parenti, in his working place CS and in his team, two automation tools are used in the process of regression testing [2]. Jenkins is a continuous integration tool that is used to build and test software projects continuously [4]. As stated by Stephane, they first define a list of tests that they want to run, then they split the work among different computers to run pipelines by feeding the program with a CSV file including all those tests. Apart from this main tool, a bug tracker tool named Mantis is also used to run tests and track bugs. With these automation tools, he and his colleagues can save much time by comparing previous data and current data of two simulations automatically. This allows them to perform regression testing on the entire software or some of the modules before they report the impact of the change to the client.

4.2 Chao's Approach

4.2.1 User Analysis

In the typical story described by Chao, he states that there are some automation developers writing tests that can behave like normal users [3]. Due to the characteristics of the services provided by his company, this user analysis phase is critical to the success of the products. In addition, it also gives the team a chance to think like a user of their product, so that the developers are given insights from a new perspective.

4.2.2 Automation Tools

In the success story described by Chao, he mentions that the automation tools used for test generation has greatly reduced the team's work. For example, when the team is implementing a banned word system for a search engine, developers only need to have a few test words to make sure the functions are working, and they do not need to test every existing banned word. The complete banned word list will be tested in the regression test by the automation developers using some automation tools. As a developer who writes the main logic, Chao does not have to directly add the test in his code, and he only needs to make modifications to the code when there is any issue arising from the tests.

5. WEAKNESSES

5.1 Stephane's Approach

5.1.1 Non-Automated Tools

As mentioned by Stephane Parenti, many companies and even his working place still use Excel to track different tests [2]. When he first joined the company, he and his colleagues would lose much time running tests that had not been updated, because they had to identify tests that they needed to run and review them manually. Because of their limited time, they would contain the test to only a specific section which was related to the modified requirements. What is more, the accuracy of manual tests cannot be ensured. Developers may include many human-made mistakes in the tests because of their carelessness.

5.1.2 Carry Over

As stated by Stephane Parenti, regardless of whether the regression tests are automated or not, if developers do not develop their software correctly, then it is possible that the problem carries over [2]. The general process of regression testing is as follows. First, developers receive a modification from the client, then they identify which requirements and which part of the design have to be modified. Afterwards, they use tools to identify which tests have to be run, and those tests become the definition of the test campaign that will have to be executed. For each defined test, a developer will run it and other developers will review it, then the status of that test, whether passed or failed, will be updated. However, the problem of carry over can take place during this process. When a developer validates a test in the next campaign, even if there is a very slight difference that he/she does not notice, if other developers also do not catch this difference, then this incorrect test will become the new baseline. Developers will carry on this bug again and again until someone discovers it later in the future. According to Stephane Parenti's own experience, the same problem he encountered did not have any impact on the project, because it was a slight difference and was not a malfunction of the system. Nevertheless, he still thinks that when developers run a test, they have to match the definition of the test which corresponds to the requirement of the client. When this carry over problem is finally found later, the team will have to spend some extra time to do much rework and investigation related to that problem, which is very inefficient and may delay the whole team's current process.

5.2 Chao's Approach

5.2.1 Test Review

As mentioned by Chao, when new features are going to be implemented, developers, quality assurance engineers and product owners will discuss together about the new features [3]. The tests are written and executed solely by the quality assurance engineers, which does not provide sufficient guarantee for the test quality. Just like any software development activity, development of test scripts is tedious and error prone [5]. If there is a bug in a test case which is not identified, then the implemented feature will be problematic. Therefore, test reviewing measures are needed to ensure the test quality.

5.2.2 Insufficient Requirement Analysis

During the interview, a failure story was described by Chao. His team was developing a new feature which could show some commercial videos about cars to the users. However, there was a missing geo-location constraint, and after the feature was released, the team found that some promoted cars were not sold in a specific region [3]. For example, users in Canada could see commercial videos about cars that they could not buy in Canada, which was not expected. The regression test did not catch this problem. Then the team added this geo-location constraint into the requirements and fixed the issue. From this story, it can be seen that, in the process of software development, any issue in the current phase will potentially be propagated to the next steps, especially those in the requirement analysis phase.

6. IMPROVEMENTS

6.1 Stephane's Approach

6.1.1 Test Automation

"Testing is a destructive task in which the goal is to find relevant defects as early as possible. It requires automation to reduce cost and ensure high regression, thus delivering determined quality [6]." According to Stephane Parenti's statement, although they already used 2 automation tools to do regression testing, some tests were still done manually. In order to achieve full automation in each step of the process, another automation tool should be introduced to replace Excel or add other automation extensions supported by Excel. With the help of test automation, developers can detect problems and bugs in a very short time, which increases the efficiency [7]. Also, if the team does not need to do any manual testing, the company can save budget on testing its products. What is more, a test will become tedious when repeated manually for many times, which may cause developers to lose patience and make mistakes like skipping some steps and making typo errors. This can

impact the quality of tests. However, with the help of automation tests, people can simply add the new tests they write to the automated test suite, so that it is easier to operate the tests and increase the coverage of tests compared to manual testing. Another benefit is its reusability, tests can be reused in other use cases and other projects, and usually it does not need to be updated. As mentioned by Stephane Parenti, the only time that the test will potentially need to be updated is when the test platform is changed [2]. In other words, only if a requirement at the top level is changed, then all tests at this level and the downstream tests associated with it should be modified. Apart from the above-mentioned benefits, automated tests can run 24 hours a day without a break, which allows developers to efficiently use the time at night and automatically run the tests at specific intervals. Lastly, automated tests can result in better accuracy. It prevents some human-made errors caused by carelessness during the manual testing process.

6.2 Chao's Approach

6.2.1 Test Review

One way to improve the test quality is to include test reviews in the testing process, which is implemented in Stephane's team by having different groups of people for writing and executing tests [2]. Peer code review, a manual inspection of source code by developers other than the author, is recognized as a valuable tool for reducing software defects and improving the quality of the code [8]. If no code review is done on the test code, then the team will have little chance to find the defects in the tests, which can probably lead to more serious bugs in the tested software. However, it requires extra communication work and can be affected by inconsistent standards, hence, a uniform test standard should be introduced first before doing the test review, where a test planning step may help.

6.2.2 Test Plan

A critical problem in many software development projects is missing some important requirements until late in the development life cycle or even the production stage. The failure story told by Chao is a typical example of this problem. Building a thorough test plan very early in the product development cycle has the potential for early discovery of missing requirements with attendant reduction in project costs and schedule improvement [9]. In a test plan, the test scope is defined, where the requirement boundaries are explored, thus revealing the potential omission of the tests and preventing problems caused by unclear requirement boundaries. Moreover, from another perspective, Test planning at the management level would include estimating the time required for all test activities, scheduling and resourcing of test activities, monitoring the progress of testing, and taking any actions required to keep the entire test effort on track [10]. Therefore, a test plan is a sensible way to improve the test quality for the team.

7. ACKNOWLEDGEMENT

We really appreciate that both interviewees could take off their time and attend our interviews to let us learn more about regression testing by sharing their experience in working. From the interviews, we could learn and understand how regression testing is implemented and why it is needed in the software development process. Furthermore, we were able to find similarities and differences between their approaches and conclude the aspects that could be improved, which can be really helpful in our career in the future.

8. REFERENCES

[1] H. K. N. Leung and L. White, "Insights into regression testing (software testing)," Proceedings. Conference on Software Maintenance - 1989, Miami, FL, USA, 1989, pp. 60-69, doi: 10.1109/ICSM.1989.65194.
[2] Stephane Parenti. (2020, Oct. 10). interview_transcript_Hanwen_Wang.pdf.
[3] Chao Wang. (2020, Oct. 11). interview_transcript_Yudong_Zhou.pdf.
[4] Saurabh. "What is Jenkins | Jenkins For Continuous Integration | Edureka." edureka!. https://www.edureka.co/blog/what-is-jenkins/ (accessed Nov. 25, 2020)
[5] V. Garousi Yusifoğlu, Y. Amannejad, and A. Betin Can, "Software test-code engineering: A systematic mapping," Information and Software Technology, vol. 58, pp. 123–147, Feb. 2015, doi: 10.1016/j.infsof.2014.06.009.
[6] M. Polo, P. Reales, M. Piattini and C. Ebert, "Test Automation," in IEEE Software, vol. 30, no. 1, pp. 84-89, Jan.-Feb. 2013, doi: 10.1109/MS.2013.15.
[7] "5 Reasons Why You Should Automate Regression Testing." Leapwork. https://www.leapwork.com/blog/5-reasons-why-you-should-automate-regression-testing (accessed Nov. 26, 2020)
[8] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," presented at the 2013 35th International Conference on Software Engineering (ICSE), May 2013, doi: 10.1109/icse.2013.6606617.
[9] J. Nindel-Edwards and G. Steinke (2007) "The Development of a Thorough Test Plan in the Analysis Phase leading to more Successful Software

Development Projects," Journal of International Technology and Information Management: Vol. 16 : Iss. 1 , Article 5.
Available at: https://scholarworks.lib.csusb.edu/jitim/vol16/iss1/5
[10] M. Fewster and D. Graham. Software Test Automation: Effective use of test execution tools. ACM Press, 1999.