GITHUB IN SOFTWARE TESTING

Simi Bajaj, Shreejai Raj, Sanket Mantri and Kewal Wadibhasme

School of Computing, Engineering and Mathematics,
Western Sydney University, Australia

ABSTRACT

Software development has an around for quite a while but the progress that has been madein the last three decade is quite remarkable. Every few years there is an emergence of new concept, new programming language or framework for software development which leads to questions around management and control of the software development process. The goal of this paper is to explore the importance of software in everyday life and the need for advanced software testing methodologies for producing reliable software products. Further, this report takes a deep dive into the challenges associated with distributed software projects such as lack of effective collaboration, awareness of the project, code conflicts and resolutions which play a vital role in successful software development and how version control systems like GitHub can prove to be a helpful tool in overcoming these challenges. GitHub is a powerful version control system and there is discussion about GitHub how tester can harness the power of GitHub in testing. This paper will shed light on how GitHub is used by software testers to gain benefits that are sometimes missing in conventional software testing methods.

KEYWORDS

Github, Software testing, collaborative software testing

1. INTRODUCTION

Software has become an integral part of our lives. Our civilisation depends on a number of systems for carrying out various tasks and 'software' has the ability to define how such systems work and behave. Systems such as the Web, network routers, transport systems, financial calculation engines, power grids and other necessary command, control and communications services, all have a software that runs to make them work. As a result, the last two decades have not only seen an immense growth in the software industry but also an upsurge in its competitiveness and its number of users. The use of software in embedded systems ranges from mundane appliances such as cell phones, microwave ovens, remote controllers, automatic garage door openers and cars to more complex and exotic applications used in air traffic control systems, spaceships and air-planes. Hundreds of processors can be found in modern households and over a thousand processors in modern cars. All these processors run on software and the consumer is optimistic while using these appliances that the software running behind these processors will never fail! (Ammann & Offutt, 2016)

This gives rise to the need of producing high quality software that meet consumers' needs. Although the software driven appliances are produced with at most attention to factors such as sound process management and careful design to produce a reliable product, the industry prefers testing as its primary tool for evaluating a software before its final delivery (Ammann & Offutt, 2016). As the development team starts the development process so does the testing team starts to write the test cases. These test cases are the most integral part of the testing

Natarajan Meghanathan et al. (Eds) : ACITY, AIAA, DPPR, CNDC, WIMNET, WEST, ICSS - 2019
pp. 221-231, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.91720

process. Over the years many techniques have been developed to make the testing process automated to save resources and speed the testing process.

The software industry is highly interested in revolutionising software testing as a means to produce successful software products. Software testing also seems to be under increased pressure with the recent growth seen in testing methodologies such as 'Agile' (Ammann & Offutt, 2016). Software projects can impose the need for collaborative efforts to achieve final product goals and the Agile methodology makes such collaboration possible by encouraging the sharing of information among departments and teams within an organization. This transparency allows team members to work together for accomplishing tasks and completing projects(Adanza, 2016).

Achieving effective collaboration, however, can be a challenge in large-scale projects where team member are dispersed across various locations across the globe(Adanza, 2016).Large and distributed projects also bring about other challenges such as communication and co-ordination breakdowns, which in turn lead to build failures and extended periods of time for resolution. Maintaining awareness about interdependencies is also a challenge associated with such projects. With the power of high speed internet and other technologies such as cloud computing, distributed computing etc. developers can work from any location at any point of time. The only problem with this approach is centralized control over the team and code review. There exist tools that provide services such as notifying about possible co-ordination needs and recommending communication needs between interdependent developers, however, these do not arrive without overheads. Even with the use of such awareness tools, conflicts seem to prevail which further call for team efforts to resolve them. Effective scheduling and division of tasks can limit, however cannot eliminate code clashes and overheads in co-ordination. Modularization of tasks is an acknowledged method to limit interdependencies, however, it is not possible to remove the interdependencies altogether (Kalliamvakou et al., 2015).

These challenges typically surface when attempting collaboration in software projects, and are especially highlighted when the team members in such projects are distributed. Collaborative Development Environments (CDE) seem to be acknowledged as a solution for the co-ordination and communication problems that arise in distributed software projects. CDEs integrate bug trackers and source code administration tools with additional features for collaboration of team members. Past few years version controlling technology is used widely to overcome this issue and it has proven to be successful and aided developers. "GitHub" is one such tool that facilitates collaboration through its interface. GitHub was launched in 2008 and has been popular since then. As of 2016 GitHub had 10 million users and 32 million monthly visitors (Craig, 2016). Other than its code hosting service, GitHub offers mechanisms for collaborative review of code, social features and a co-ordinated issue tracker. It also allows programmers to leave their remarks on issues and new submissions to the project. This helps is attaching conversations to codes which in turn helps in increasing awareness about the project (Kalliamvakou et al., 2015).

In recent years, GitHub has played a vital role in testing automation by providing the developers to develop open source software contributing well to Continuous integration and continuous development(CICD) or continuous deployment that being effectively used in software development by the industry. These techniques have catalysed the test automation process. The following sections of this paper are literature review focusing on GitHub , benefits to software testers followed by discussion on how GitHub will be useful in testing and lastly conclusion.

2. ABOUT GITHUB

GitHub is a popular web service that facilitates users to host their code online and share it with others for collaborative development. As of 2017, GitHub seems to host over 67 million repositories by 24 million developers and is used by over 117,000 business worldwide (GitHub, 2017). It provides an open platform for collaborative work with transparency of the activities that take place in a given project. This transparency is achieved through a simple user-friendly interface as well as with the help of notifications. The visibility of the project so achieved, further helps in spreading awareness about the project status among its members with minimal communication. In the last decade there has been a lot of advancement in software version control systems (Defaix, Doyle, & Wetmore, 2010), GitHub being one of the pioneers. According to (Hackernoon, 2018) Github is one of the most adapted and renowned version control system all around the world in the software industry. As a result, GitHub seems to be a good answer for the challenges faced by distributed projects where problems such as lack of awareness, conflicts in code, communication and co-ordination breakdowns are the areas of concern. As a matter of fact, this ability of GitHub is also highlighted in its motto "Collaboration without upfront co-ordination". (Kalliamvakou et al., 2015)

3. GIT AND GITHUB

GitHub works on 'git', which is a decentralized version control system (DVCS) (Kalliamvakou et al., 2015). A DVCS is a distributed system with autonomous nodes. It allows writing over local data at any time and intermittently connects with other repositories at the user's command to exchange updated information. It allows concurrent updates as different branches of development. A DVCS can simultaneously track many different branches and any conflicts that may arise between these branches can be resolved or combined by the user using a merge operation (Murphy, 2017).

DVCS are also popular for collaborative software development as it allows for creating repositories and keeps track of the version history of the file system. This version history is stored in a content-addressed graphical format that naturally performs de-duplication of unchanged files. This allows for efficient synchronization of duplicate files in a file system. De-duplication occurs naturally for all identical files within the project however, for performing de-duplication between different versions of files, Git relies on algorithms which are suited for text, but do not seem to work well with large binary files(Murphy, 2017).

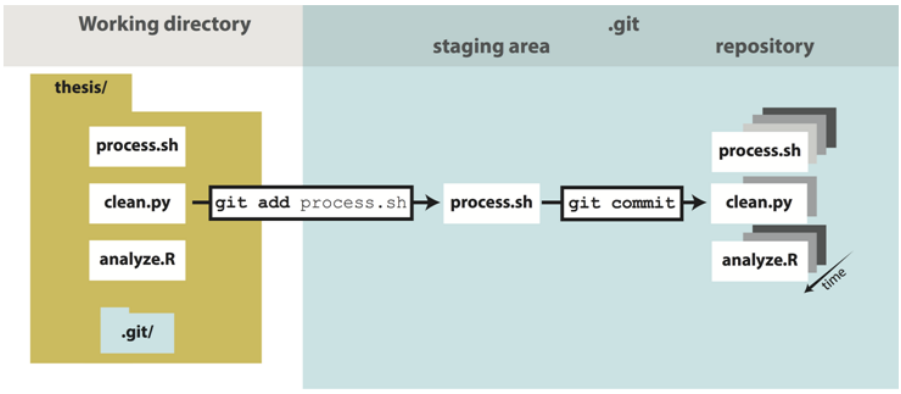

Figure 1: Git Process (Blischak, Davenport, & Wilson, 2016).

Blischak et. al. 2016 outlines the Git process as shown in Figure 1. The main code or the main repository is controlled by the core team of the software providing read only access to only contribute using branches to the developers. Developers can pull the code by pull request and further if satisfied with their code can push their work on the main branch for the core team to review and if appropriate merge with the main repository. Hence number of developers can work together at any point from any location. GitHub provides with the social coding(Lima, Rossi, & Musolesi,2014) and enables developers to broadcast new methods and learn new methods. In the research by (Dabbish, Stuart, Tsay, & Herbsleb, 2012) gives a study of how transparency affects the coding style with the help of GitHub. The core team must review each requested submitted and check for the compatibility and then merge with the main repository.

The workflow of GitHub seems to be a good fit for Open Source Software (OOS) projects, however, a rapid growth is also seen in the number of commercial projects using GitHub as a development interface. These include large corporations such as Microsoft, Walmart, Lockheed Martin and Living Social. Since such corporations in general, do not prefer allowing public access to their files, GitHub also provides for private repositories in its enterprise versions (Kalliamvakou et al., 2015).

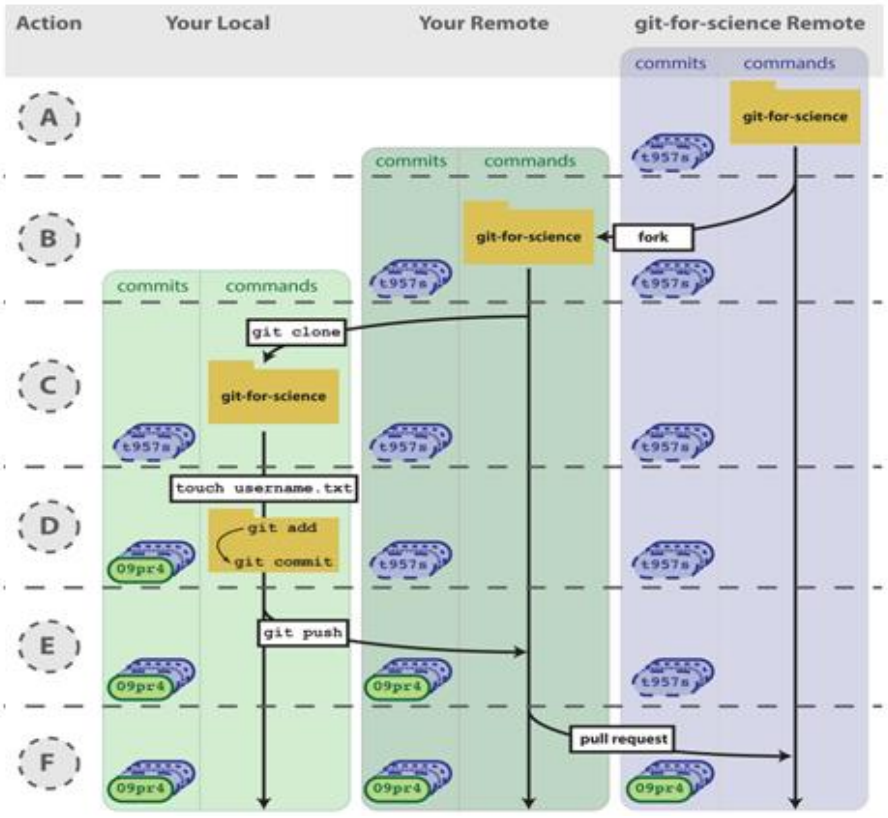

Figure 2: Working on Open Source with Git (Blischak et al.,2016)

GitHub provides various tools and charts to track the branches and help to know there responsible for bugs. Many of these repositories are open source and can be accessed either for use or to contribute for the development of the project. One of the best examples is Ruby on Rails project developed on GitHub and is still continuously improved and deployed for the users.

An online survey which involved 240 developers as well as interviews with 30 GitHub users revealed that common Open Source Software practices such as reduced communication and co-ordination needs, self-organization and independent work are being successfully adopted by many commercial software projects (Kalliamvakou et al., 2015).

4. BENEFITS FOR TESTERS

Software testing is considered to be one of the most widely practiced and studied methods to assess and improve the quality of any built software (Orso & Rothermel, 2014). Testing of any software is important to avoid any failures that may create problems for the customers using the software and potentially sue the software development company for damages.

Some notable examples of software failures include the failure of credit cards in Germany on 1st January 2010, leaving millions of Germans cashless and unable to purchase any items from retail stores without any warning beforehand, that their credit cards would suddenly stop working. Another example is of the Airbus flight A380, which had a software failure on its autopilot function that forced it to return to New York which was on route to Paris on 20th November 2009 (Homès, 2013). Such software failures can have catastrophic results, and therefore it is best to thoroughly test all software before releasing to the public.

After going through the brief overview of the concepts, let us now look at how GitHub can be used with any testing software. As GitHub is used to track updates and bugs in software code while working on a software as a team of large number of developers, the people at GitHub started adding more features to the tool that relate to project management such as issue-tracker. These additional features combined with simple version control offered initially by GitHub make it a great tool for tracking bugs found while testing a software (Gaspar, 2016).

4.1. Hosting project artefacts online

GitHub, along with its ability to let developers and testers store their code, also seems to facilitate testers in a number of other ways. According to Matt Heuser, M.D. of Excelon Development, GitHub is also useful for storing test artifacts such as test plans, how-tos and session notes in version control. Specifications of the project whether in HTML or MS Word format can also be stored at GitHub. Heauser further states that, by using GitHub, testers can gain an understanding of the developers' workflow and their jargons, which further assists in minimizing the friction between the two. It also helps testers get one step closer to the source code for better understanding the code. Using these facilities of GitHub, testers can also get better at reading the code and pointing out errors at the source code level. They may even be able to make contributions to code review. An rise is also seen in the number of non-profit as well as profit organizations that make use of GitHub to host their code (Ben Linders, 2017).

4.2. Builds portfolio by showcasing contributions

GitHub can also be used to make workflows by approving team members to perform tasks such as filing bugs and pushing sets. One more benefit of working with a GitHub account is that it also creates a public portfolio of the user which can further help members in getting hired based on their contributions. Recruiters and organizations progressively look to GitHub accounts as an approach to assess possibility for employments(Ben Linders, 2017). Many interviewers like to check the applicant's public GitHub account to assess their consistency and interest in the role (Mar, 2016). This also makes their task of finding the right candidate easier as they get to make decisions based a public work-based portfolio of the user rather

than going through his / her resume.This portfolio presents a more valuable demonstration of skills than a mere resume.(Ben Linders, 2017).

This implies that it is very likely for software professionals to utilize the GitHub work process in their professional lives. One of the benefits of using this workflow is that any changes in code and be submitted through a process called “pull requests” and both the programmers and testers can observe the differences in these codes and test those differences independently before they are merged for the final production. The ‘pull requests’ get processed by automated tests that keep running at different levels of the system giving potentially huge testing and process benefits to the team. This also helps testers to report bugs and contribute to code reviews through GitHub Issues (Ben Linders, 2017).

4.3. GITHUB Watchers

Motivated communities of programmers who collectively produce software products initiate and maintain open source software projects. Such projects must be able to efficiently sustain a pool for interested programmers to make individual contributions to the project. Moreover, there are different kinds of contributors to an open source software project. There are core contributors who focus on building the code base. There are peripheral contributors who engage in activities like reporting bugs and submitting patches. There are also other groups of users that don’t contribute code but actively participate in providing their feedback and support for the project. The success of a project can be identified in accordance with the span of the developer community around it, which goes about as a pool of potential contributors to the project. Without an in-exhaustive source of such supporters, an open source software project can become stagnant or fall flat (Sheoran et al., 2014).

GitHub, with its well-known code-hosting facility, has an expansive number of open source programming ventures. It gives social features that permit building a network of contributors around the codebase. Any user can keep a “watch” on an open GitHub repository to get notified about any changes that occur within the repository. “Watching” a GitHub repository signals the interest of the user in the activities that take place within the repository and also indicates that the user is likely to contribute to the project. This can be viewed as a passive engagement of users to a given project (Sheoran et al., 2014).

4.4. Continuous Integration and Continuous Deployment

GitHub is also capable of supporting open collaboration within organizations. Organizations see an advantage in removing barriers between projects and teams that run within them in order to promote inter-team collaborations. GitHub can be helpful in building this transparency within commercial organizations by centralizing their tools and data (Kalliamvakou et al., 2015). Collaboration brings the individual efforts of team members and co-ordinate them to work towards a common objective. It also seems to be an efficient way of managing interdependencies (Kalliamvakou et al., 2015).

Initial software development models had a lot of drawbacks such as lack of communication, time consuming, manual documentation etc. Agile methodology revolutionized the process and help enhance the process drastically. Today maximum number of companies have adopted to agile methodology and are delivering great results. New development methodologies are introduced such as DevOps that encourages the companies to use techniques that help developers and operations teams to work together. GitHub has played a vital role in DevOps method (Bissyandé et al., 2013). Apart from these methods companies and developers had to wait for the testing feedback to know the bugs and fix them. One of the

most recent methodology implemented by the companies is continuous integration (CI) and continuous development (CD).

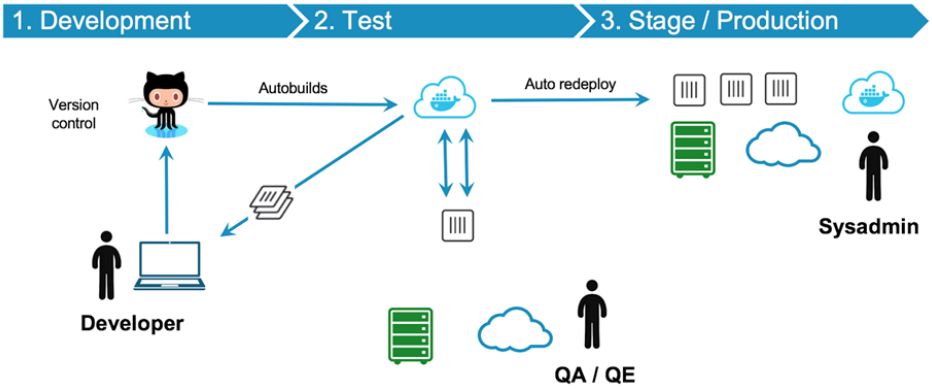

Figure 3: CI/CD Using GitHub (Heins, 2016)

As seen in the above figure developers get the feedback of the issues the moment any new merge is made. If the new code has any bugs the developer is notified, and the faulty code is not deployed on the QA server. If the new code is bug free it is auto deployed in the QA servers and the testing teams continue with further testing. This method has helped testing automation greatly and for the same GitHub has been a crucial part. Unlike previous methods for deployment CI has enabled test results to be available at early stages and help reduce bugs seeping into production (Meyer, 2014;Vasilescu, Van Schuy lenburg, Wulms, Serebrenik, & van den Brand, 2014). The primary requirement for the success of the CI/CD is the test cases to be automated and deployed on the repository server. Whenever the developers push the code for merge into the main repository the automated test cases are runand the initial test results are available to the developer as test reports. If the test cases havepassed the code is merged with the main repository and deployed on the staging server, on which the testers can carry out further testing. This ensures that the initial code is up to the requirement and error free. This also helps to reduce the time and cost for the testing which is crucial factor (Stolberg, 2009).

5. ANALYSIS OF GITHUB FOR SOFTWARE TESTING

It has been established that software testing does play an extremely important role in the software development cycle. Often companies do not consider it so important and result into a spending more on bug fixing. Using testing automation and unit testing for code can help detect bugs at proper time and fix it. One of the important documents to track and fix bugs is the issue reports. Many systems do provide with the issue reporting, but GitHub is more popular and helps to trace back the issue to the developer.

GitHub has enabled developers and the operations teams to work together to find issues and bugs with ease and on time. As with the open source systems users are able to contribute to bug fixing under the watch of the core team. GitHub has assists developers to keep track of the versions, detect the location and the origin of the bug if any. (Feliciano, Storey, & Zagalsky, 2016) discusses a case study of student perspective in using GitHub as learning tool for software engineering. This case study shows how apart from testing GitHub is aiding students to learn engineering process by using GitHub as platform. Students still feel the flexibility and the privacy are main concerns for using GitHub. Another GitHub feature that

supports reviewing work of the peers, may enable the novice developers to have better understanding of the testing process in the early stages of the software development.

The following table outlines the strengths and weaknesses of Github:

Table 1: Strengths and weaknesses of Github

Strengths	Weaknesses
The transparency of GitHub allows for effective collaboration and co-ordination of team members (Kalliamvakou et al., 2015).	The esoteric command lines can be confusing for new users (Mar, 2016).
Offers hosting project artifacts online (Ben Linders, 2017).	Version control systems were primarily designed to store source code files that do not usually measure more than tens of kilobytes in size. Engaging larger binary files such as audios, videos and images can affect its performance (Murphy, 2017).
Provides mechanism for collaborative code review (Kalliamvakou et al., 2015).	Distributed version control systems tend to keep all versions of all the files in their respective repositories. This also gives rise to excessive storage needs (Murphy, 2017).
Offers social features to build a community around a project (Sheoran et al., 2014).	
Allows developers to leave comments on commits and issues (Kalliamvakou et al., 2015).	
It helps in spreading awareness about the nature, scope and status of the project without collocation of team members (Kalliamvakou et al., 2015).	

One other important concept that allows the developers to test their software code is through making use of an assertion statement i.e. a statement that helps the developers to test an assumption they may have of a fragment of code that they may have written. The result of an assertion statement is always denoted using Boolean expression that needs to be satisfied for the execution of program statements. If any execution fails to execute, the Assertion Error is raised to flag that the test case did not run as expected. Such assertion statements could be used to check preconditions, post-conditions and other variables as defined by the developers or by the business requirements. The study by (Lo & Kochar, 2017) has highlighted the relation between assertions and defect occurrence on GitHub projects using the study by(Casalnuovo, Devanbu, Oliveira, Filkov, & Ray, 2015) who also seem to have found that the projects on GitHub that used assertions have significantly lower amount of failure rates as compared to the software projects that do not use assert statements in their projects.

GitHub has reduced the testing workload as identified in CI/CD. It has also enabled the developers to automate the merge process more efficiently and accurately. One of the good examples of GitHub providing this platform is GitHub Education which is meant for code learning and does unit testing on the go and with immediate feedback (Zagalsky, Feliciano, Storey, Zhao, & Wang, 2015). Platforms like this will enable better learning techniques for the students and for the teachers. Another breakthrough in the testing is Travis-CI that has

enabled both testing and CI/CD efficiently and was developed on GitHub (Beller, Gousios, & Zaidman, 2016).

Despite a very few research papers seem to be focused on software testing using the social software coding platforms such as GitHub, we can clearly see by the recent research that the topic seems to be gaining popularity. Further, it can be assured that the very vast codebase offered by GitHub can be used in combination with Machine Learning to build a proper software testing tool that is able to predict any issues that may arise in the software development life cycle.

On the other hand, GitHub - one of the most popular social coding platform, offers a number of features that allow the software developers to maintain history of changes to the code and raise any bug reports directly on GitHub. This makes it easier for developers to pay attention and resolve any critical issues with the code without relying on any other software or collaboration tool. This makes GitHub a great platform to perform software testing and may enable to pin-point and resolve many issues with the code which may sometimes go unnoticed.

6. CONCLUSION

GitHub seems to be a great tool for collaborative approaches in software development. New members need to accustom themselves with the command lines associated with Git to get started with using GitHub, however, the advantages of using GitHub seem to outweigh the learning curve (Mar, 2016). GitHub plays a vital role in assisting projects with a large number of distributed users by providing an open development environment and a simple user-friendly interface which showcases all the activities that take place within a given project. This promotes a better understanding of the project within the team members and a better collaboration with minimal communication needs. As a result, GitHub seems to be the answer to problems such as collaboration, lack of communication and understanding and code clashes which arise in distributed software projects. Privacy and latency are still concerning in this area which are open to future research. Open source software development will be benefited by using CI/CD to reduce bugs in the code and repair the code in time and automate. GitHub along with third party software testing tools such as Travis CI already allows us to test the code in real-time further enhancing the possibility of GitHub becoming a critical component of software testing tools.

REFERENCES

[1] Adanza, F. (2016). Collaboration: the backbone of agile testing | Zephyr. Retrieved October 17, 2018, from https://www.getzephyr.com/insights/collaboration-backbone-agile-testing

[2] Ammann, P., & Offutt, J. (2016). Introduction to software testing. Retrieved from https://books.google.com.au/books?hl=en&lr=&id=58LeDQAAQBAJ&oi=fnd&pg=PR10&dq=software+testing+process&ots=Vzf4RKRIU_&sig=DMnt_nfymEoE9vW7pFhjBJGCrtM#v=onepage&q=software testing process&f=false

[3] Beller, M., Gousios, G., & Zaidman, A. (2016). Oops, mytests broke the build: An analysis of travis cibuilds with github (2167-9843).

[4] Ben Linders. (2017). GitHub for Testers. Retrieved October 17, 2018, from https://www.infoq.com/news/2017/07/github-testers

[5] Bissyandé, T. F., Lo, D., Jiang, L., Réveillere, L., Klein,J., & Le Traon, Y. (2013). Got issues? whocares about it? a large scale investigation ofissue trackers from github. Paper presented atthe Software Reliability Engineering (ISSRE),2013 IEEE 24th International Symposium on.

[6] Blischak, J. D., Davenport, E. R., & Wilson, G. (2016). Aquick introduction to version control with Gitand GitHub. PLoS computational biology, 12(1),e1004668.

[7] Casalnuovo, C., Devanbu, P., Oliveira, A., Filkov, V., & Ray, B. (2015). Assert use in GitHubprojects. Proceedings - International Conference on Software Engineering, 1, 755–766.https://doi.org/10.1109/ICSE.2015.88
Craig, S. (2016). 10 Amazing GitHub Statistics(December 2016). Retrieved fromhttps://expandedramblings.com/index.php/github-statistics/

[8] Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012).
Social coding in GitHub: transparency andcollaboration in an open software repository.
Paper presented at the Proceedings of the ACM2012 conference on Computer Supported Cooperative Work

[9] Defaix, F., Doyle, M., & Wetmore, R. (2010). Version control system for software development. In: Google Patents.

[10] Feliciano, J., Storey, M.-A., & Zagalsky, A. (2016).Student experiences using github in softwareengineering courses: a case study. Paperpresented at the Proceedings of the 38thInternational Conference on SoftwareEngineering Companion.

[11] Gaspar, G. (2016). 7 Examples of Bug Reporting Template You Can Copy For Your Web TestingProcess. Retrieved September 24, 2017, from https://marker.io/blog/bug-reporttemplate/

[12] GitHub. (2017). GitHub - The World's leading software development platform. Retrieved September 24, 2017, from https://github.com/

[13] Hackernoon. (2018, june 2018). How Git Changed The History of Software Version Control. Retrieved from https://hackernoon.com/how-git-changed-the-history-ofsoftware-version-control-5f2c0a0850df accessed: 16/10/2018

[14] Heins, C. (2016). CI/CD WITH DOCKER CLOUD.Retrieved fromhttps://blog.docker.com/2016/04/cicdwith-docker-cloud/

[15] Homès, B. (2013). Fundamentals of Software Testing. Fundamentals of Software Testing. https://doi.org/10.1002/9781118602270

[16] Kalliamvakou, E., Damian, D., Blincoe, K., Singer, L., & German, D. M. (2015). Open Source-Style Collaborative Development Practices in Commercial Projects Using GitHub. Retrieved from https://help.github.com/articles/using-pull-requests/

[17] Lima, A., Rossi, L., & Musolesi, M. (2014). CodingTogether at Scale: GitHub as a Collaborative Social Network. Paper presented at the ICWSM.

[18] Lo, D., & Kochar, P. S. (2017). Revisiting Assert Use in GitHub Projects.

[19] Mar, W. (2016). Git and GitHub for Testers | StickyMinds. Retrieved October 17, 2018, from https://www.stickyminds.com/presentation/git-and-github-testers

[20] Meyer, M. (2014). Continuous integration and its tools.IEEE software, 31(3), 14-16.

[21] Murphy, M. (2017). Scalability of Distributed Version Control Systems. Retrieved from http://ojs.bibsys.no/index.php/NIK/article/view/434/394

[22] Orso, A., & Rothermel, G. (2014). Software testing: a research travelogue (2000–2014).Proceedings of the on Future of Software Engineering - FOSE 2014, 117–132.https://doi.org/10.1145/2593882.2593885

[23] Sheoran, J., Blincoe, K., Kalliamvakou, E., Damian, D., & Ell, J. (2014). Understanding "Watchers" on GitHub Uncovering dependencies in software ecosystems View project Understanding "Watchers" on GitHub. https://doi.org/10.1145/2597073.2597114

[24] Stolberg, S. (2009). Enabling agile testing throughcontinuous integration. Paper presented at theAgile Conference, 2009. AGILE'09.

[25] Vasilescu, B., Van Schuylenburg, S., Wulms, J.,Serebrenik, A., & van den Brand, M. G. (2014).Continuous integration in a social-coding world:Empirical evidence from GitHub. Paperpresented at the Software Maintenance andEvolution (ICSME), 2014 IEEE InternationalConference on.

[26] Zagalsky, A., Feliciano, J., Storey, M.-A., Zhao, Y., &Wang, W. (2015). The emergence of github as acollaborative platform for education. Paperpresented at the Proceedings of the 18th ACMConference on Computer Supported CooperativeWork & Social Computing.

AUTHORS

Simi Bajaj

Simi, currently Lecturer of Computing in SCEM, graduated with B.Sc. first class followed by double Mastersdegree where she was awarded a medal for her academic achievement. She won APAI scholarship to pursue her PhD in Computer Science. She has developed a REcursive, MUlti-stage Classifier for Client-side email Spam filtering using machine learning and deep learning methods. She has worked with National Australia bank in the role of Senior Analyst, nabCERT - Monitoring and Response Infrastructure and Security Services.

She is passionate about enhancing student learning and growth and collaborates with industry and institutions locally and globally. In last 5 years, Simi has joined editorial boards of journals, obtained research grants, published in journals as well as rank A and B conferences in the areas of software development and testing, information security, cyber fraud, malware detection, learning & education, and machine learning. Simi has established research partnerships with the industry and has successfully worked on projects with them.

Shreejai Raj, Sanket Mantri, Kewal Wadibhasmeare students of MICT in School of Computing, Engineering and Mathematics, Western Sydney University, Australia