To what degree can automation help developers deliver software projects more reliably?

Continuous delivery (CD) aims at reducing time for building, testing, and deploying software, ensuring that software can be released reliably. Automated building pipelines are used to run these processes automatically. Generally it can ensure the reliability in two major aspects: reduce manual time and allows developers spend time on more complicated tasks, and avoid human-made among those processes.

Software build automation is a necessary pre-condition for continuous integration and continuous delivery. It can avoid developers' inconsistency with IDEs, including different configuration settings, environment variables, and compilation options. Furthermore, manual builds can have severe human-made errors. Sometimes developers may forget to commit a bug fix, or forget to pull down the latest code to get other people's changes before compiling. Automation can eliminate such problems, and ensures software reliability[1]. Automated building tools varies, including Github[1], and Team City[2]

Automated testing simplifies some repetitive but necessary testing tasks. When conducting unit, functional, integration and performance tests, automation allows parallel testing, which saves developers'time. Additionally, different parts of builds do not need to pass the full performance-testing suite, but only necessary required tests, and after passing tests they are just moved off to a safer place [2]. Automated testing also improves reliability by eliminating potential risk of regression error that might take place in manual testing, which means is more likely to encourage more frequent releases or merges. Various automated testing tools come into industry with increasing demands for automated testing, such as the most trending Selenium[3], TestingWhiz[4], TestComplete[5], etc.

Apart from two aspects mentioned at the beginning, automated deployment also allows testers deploy a build to anywhere, and trigger a deployment of that build themselves[2]. This means if this build needs to be deployed to a new test environment, or if a new client needs to install it, automation can increase reliability by decreasing the overhead of deployment, for the reason that it is repeatable and configurable. Octopus[6] is one of the typical tools used for automated deployment.

A practical example for automation increasing reliability can be found in feedback of experiencing continuous delivery and Kanban by LeanKit. They uses Kanban to mirror automation process, allowing developers to instantly see where work flows and where work gets stuck, in which way they can stop in time if automation a bad process during development. Additionally, it reduces time for building new functionalities and delivering to customers. Once delivery is faster, feedback from customers loops faster, which allows LearnKit make quicker adjustment based on customer feedback[3]. Therefore it improves efficiency of whole develop process and reliability of the builds as well.

References

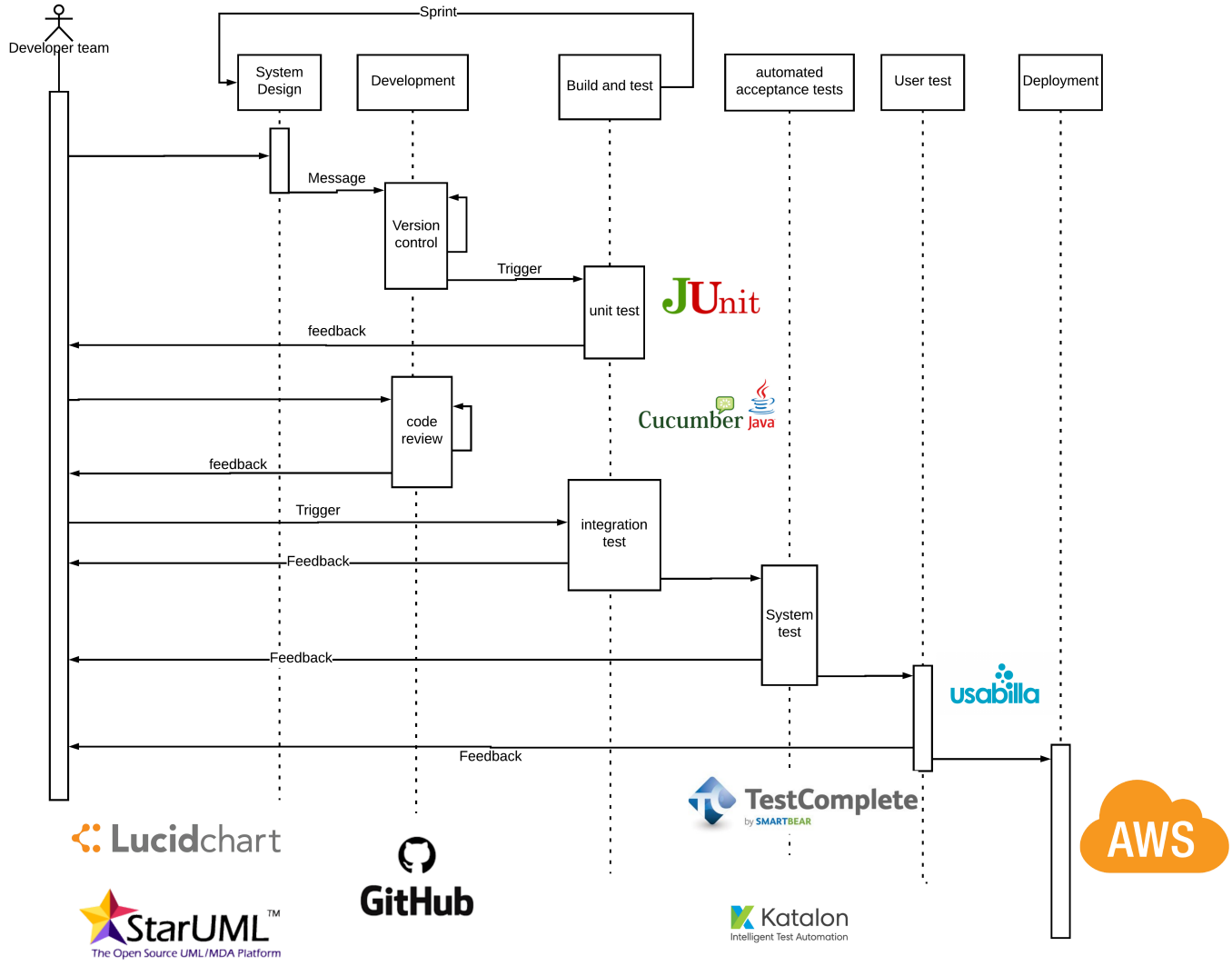
[1] Enos J. **Automated Builds: The Key to Consistency** https://www.infoq.com/articles/Automated-Builds *Date Accessed: 2 February 2019*

[2] Humble J., Read C., North D. **The Deployment Production Line** http://continuousdelivery.com/wp-content/uploads/2011/04/deployment_production_line.pdf *Date Accessed: 31 January 2019*

[3] George C. **How Kanban Enables Continuous Delivery: LeanKit's Perspective** https://leankit.com/blog/2014/06/kanban-and-continuous-delivery/ *Date Accessed: 2 February 2019*

[1]https://github.com
[2]https://www.jetbrains.com/teamcity/
[3]https://www.seleniumhq.org/about/
[4]https://www.testing-whiz.com
[5]https://smartbear.com/product/testcomplete/overview/
[6]https://octopus.com

Part II:



The figure above shows a set of processes and tools to build a deployment pipeline based on Java programming language. Before building a deployment pipeline, the developer team designs the whole system frame, using tools like Lucidchart, startUML. After every developer finishes the system design and pushes his/her code to a remote repository, the deployment pipeline starts. Firstly, in the development stage, developers need to do the version control and code review (usually these two tasks are in parallel.). GitHub is highly used in this stage. It is remarkable that the agile development cycle is used in the development cycle. There are few sprints and each spring involved in function implementation, code review, version control, testing and sending feedback back to developers. If developers get good feedback in this stage, then unit test, integration test, automated acceptance test and user test will be triggered in sequence. JUnit is used to finish the unit test and Cucumber can be used in the integration test phase. During the automated acceptance stage, the system needs to pass system test using tools like TestComplete and Katalon. After that, Usabilla is used in the user testing stage. All stages are necessary to make corresponding feedback so that team members can learn problems as soon as possible and are able to fix them as quickly as possible. Finally, if all the tests pass as well, the whole system can be promoted to production, such as hosting on AWS. The pipeline helps developers make effort to the software more efficiently. Making sure the continuous delivery in the meantime it avoids the drawback of lack of feedback like waterfall method.