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