How cloudization of development processes can improve software processes

Ivo Willemsen
Email: ivo.willemsen@outlook.com

***Abstract*— This essay raises ideas relating to how cloud-based development solutions can improve software processes in general and quality, schedule and cost properties of project management in particular**

Keywords: Quality characteristics measurements, ISO 25010 quality model, cloud-based development, software processes

I. Introduction

Fuggetta and Di Nitto [1] give an overview of development of software processes in the last 15 years. This essay focuses on how the cloudization of software processes can improve quality, cost and time aspects of project execution. In the *Ideas* section, the first two research questions focus on the usage of cloud-based development solutions, in particular cloud-based source control and continuous integration and delivery platforms and how these tools can help in measuring *quality* related characteristics of open-source components. The last research question is an attempt, at a very high macro level, to raise ideas about how untapping the potential of software developers in developing countries can help reduce *cost and schedule* related constraints of project management. Finally, the *Future work* section gives a follow up on possible interesting ideas for future investigation with regards to the research questions that are posed in this essay.

II. Ideas

(RQ1): Do the success of the agile approach and cloudization of tools that implement software processes in the SLC, warrant a revision of the ISO 25010 quality model?

In my opinion, having read the quality model in detail, it seems that the Agile Manifesto (2001) did not have a lot of impact on the *evolution* of the quality model ISO 9126 to the latest model, ISO 25010, which is, to say the least, surprising. The quality model ISO 25010 [2] was issued in 2011 and supersedes the ISO 9126 quality model [3] and it provides a hierarchical decomposition of quality characteristics which, for a given software system, can be mapped to stakeholders' goals and objectives for the system. One would expect that a time span of 10 years would be sufficient to shed a light on the impact of the Agile approach on the (at that moment) current quality model ISO 9126, but not a lot of agile aspects can be found in the revised model ISO 25010.

Ford et al. [4] discuss the effect that software evolution can have on the architecture of a system and it defines *evolvability* as the *ability to support guided, incremental change across multiple dimensions*. Quoting Ford:

> *To build architectures that truly evolve, architects must support genuine change, not jury-rigged solutions. Going back to our biological metaphor, evolutionary is about the process of having a system that is fit for purpose and can survive the ever-changing environment in which it operates. Systems may have individual adaptations, but as architects, we should care about the overall evolvable system.*

When applying an agile approach to software development, business requirement changes, changes in the environment and changes in market conditions are enough ingredients to change the architecture recipe of a system and might result in a re-evaluation and change of the overall architecture of a system. Changes to the architecture should not impact the continuous integration and delivery processes and therefore, judging the *evolvability* of a system gives a good picture of the degree in which a software system is immune to those architectural changes. It would be beneficial for projects that follow the agile approach, to include another main quality characteristic *Agility* with its first sub-characteristic *Evolvability* and define how this sub-characteristic can be measured.

One of the goals of the Agile approach is to iteratively deliver MVPs[1], reducing deployment cycle times to a minimum while at the same time not obstructing business operations. A system's architecture contributes to promoting or restricting the *deployability* of a system. If sub-optimal design decisions are taken as the system is architected and later on evolves through the time, critical activities such as continuous integration and delivery and automated testing can become a problem. It is therefore important to introduce deployability as a quality characteristic.

The idea of deployability as a quality attribute is not a novelty and Khohm et al. [5] researched the relationship between reduced cycle times and perceived quality. Actually, the first mention (not in an agile setting) of deployability as a quality attribute is made by Bass et al. [6] when discussing modifiability quality attributes of a system:

> *In some projects, deployability is an important quality attribute that measures how easy it is to get a new version of the system into the hands of its users. This might mean a trip to your auto dealer or transmitting updates over the Internet. It also includes the time it takes to install the update once it arrives. In projects that measure deployability separately, should the cost of a modification stop when the new version is ready to ship? Justify your answer.*

In the footsteps of what was said with regards to the introduction of a new sub-characteristic related with the definition of the quality attribute *evolvability* and it the light of *agilizing* the ISO 25010 quality model, adding the sub-characteristic *deployability* to the main characteristic *Agility*, would be a welcome revision.

(RQ2): How do cloudization of source code repositories and continuous integration tools, impact the characteristics of the ISO 25010 quality model in order to make sound judgments pertaining to the selection of useful open-source components?

The usage of open source components at the enterprise level is becoming a widespread phenomenon and cloud-based solutions like Github[2] and BitBucket[3] not only provide functionalities to manage software sources, but also allow projects to design CI/CD [4] pipelines which help projects to successfully implement agile aspects of software development by setting up automatic tests and deployments to improve related software processes. Speaking from personal experience, it is a daunting task to make a good selection of open source components to be used in projects because proper quality characteristics measurements of those open source components are missing. As an alternative method of selection, a lot of time is wasted in investigating support forums like stackoverflow[5] and subreddits[6] in order to extract useful information that might help projects make sound selection decisions. In the absence of proper selection information, projects might be tempted into making precocious decisions and select open-source components that might not fit their set of non-functional requirements, only to find out later that the

[1] Minimal Viable Products
[2] `https://github.com/`
[3] `https://bitbucket.org/`
[4] Continuous Integration/Continuous Deployment
[5] `https://stackoverflow.com/`
[6] `https://reddit.com`

decision must be revoked because of a mismatch and losing time and resources and jeopardizing the success of the project.

In order to circumvent these situations, it would be beneficial to enterprises to have these quality measurements of open-source components at their disposal, which would allow them to decide on components that best match the project's quality requirements. The ISO 25010 quality model can be consulted to extract these quality characteristics.

Page 10 of the quality model [2] lists the main quality characteristic categories. Of those main categories, *Functional suitability*, *Performance efficiency* and *Usability* are not affected by the application of open-source code repositories and CI/CD tools, as they are primarily related with the business requirements and usage of the system, which are quality characteristics that cannot be directly deduced from the source code itself or from statistics that are extracted from CI/CD tools. The other main categories do at least inhibit one quality characteristic that is affected by the usage of cloud-based source control and CI/CD platforms:

- Interoperability (Compatibility) - The extent to which a certain component is compatible with other components can be deduced by checking the dependencies of components. Dependency management is performed by tools like NPM[7] (Node) and Maven [8] (Java) and statistics with regards to validity and deprecation of dependent components could be collected during the build-phase performed by CI/CD tools
- Maturity (Reliability) - The maturity of a component cannot be directly deduced, because it is based on the usage of the system. Notwithstanding, the age of the component, the activeness (number of commits) and results of unit and integration tests can be used in order to calculate a statistic that features the maturity of a component
- Fault tolerance (Reliability) - Likewise the previous quality characteristic, fault tolerance is directly deduced from the usage of the system, but a reliable statistic can be deduced by checking the results of unit and integration tests which can be collected by CI/CD tools, and keeping a history of the results in order to check tendency changes
- Confidentiality and Integrity (Security) - An open-source component that is used by an enterprise should use data only according to its intended usage. The degree to which a component ensures that data are accessible only to those authorized to have access is a quality characteristic that is difficult to collect statistics for. Research by Denning [7] and Poll [8] shows that dynamic and static code analysis tools can be developed that would possess the functionality to discover whether confidentiality would be jeopardized by using open-source components. These tools or libraries would have to be integrated with cloud-based source control platforms and CI/CD tools
- Modularity, Reusability, Analyzability and Modifiability (Maintainability) - These are all quality characteristics that are derivations of design properties like cohesion and coupling, and adhering to SOLID design principles is the basis of a good design [9]. Lanza et al. [10] have published an article regarding the generation of visualizations based on reverse-engineered code source code. These visualizations give insight into the quality characteristics which are under discussion here. Likewise, instead of generating visualizations, statistics could be generated that concretize these quality characteristics in a numerical manner
- Testability (Maintainability) - A reliable statistic can be deduced by checking the results of unit and integration tests which can be collected by CI/CD tools
- Installability (Portability) - Package managers like NPM and Maven include steps that take care of packaging and installation of components and these steps can be configured in the build-phase of a component. Cloud-based source control platforms and CI/CD tools keep track of the statistics pertaining to the outcome of the installation and packaging in the build-phase and these statistics thus can be used to qualify the *installability* of a component
- Deployability (Agility) - As suggested in the previous research question (RQ1), adding *deployability* to the ISO 25010 quality model would give agile projects an extra quality characteristic that could aid them in making better decisions with regards to the selection of open-source components. Calculating the deployability of a component is straightforward as it just requires keeping track of the success rate of deployments of components by CI/CD tools

An interesting research was performed by Trockman et al. [11] which elaborates on the usage of quality related repository badges, which are offered by platforms like Github and Bitbucket. One of the research questions they pose is what the effect is of those badges on people who are *shopping* for open-source packages. These badges can either be generated by the platform itself (assessment signals) or can be added by the repository maintainers (conventional signals) in order to convey certain quality characteristics. The outcome of the research was that a large amount of users of open-source packages find those badges useful and admit they are influenced especially by the presence of badges that express quality characteristics related with continuous integration (deployability, testability), static analysis (maintainability) and dependency management (Interoperability). Trockman et al. provide a different categorization of quality characteristics than the ISO 25010 model, and the usage of badges implies in general a more binary representation of the state of the quality characteristic (build passed or build failed). Future work could focus on investigating how data could be collected in order to generate measurements that possess a more continuous character (like a percentage for example).

(RQ3): How can cloud-based integrated development environments help in tapping into the pool of nascent software developers of developing countries and as a result improve cost and schedule related aspects of project management?

From a project management point of perspective, the previous two research questions were related with the quality aspect of project execution. Project outcome can be maximized not only by focusing on maximizing quality, but also by attempting to *minimize time and cost related parameters*.

The cloudization aspects of the previous two research questions were related with discussing quality derivable characteristics of the application of cloud-based source control and CI/CD platforms. Currently, I am writing this essay by using Overleaf, an online environment for writing academic papers. I can focus on writing and there is no need for me to download applications and configure my system in order to run the LaTeX application, thus substantially reducing the time it takes to get started. I don't need to worry about making backups or losing time when my laptop breaks down and in which case I would have to repeat the installation steps again on another laptop. The resource requirements for my laptop, just for the concept of writing this essay, are very limited, as it only requires me to have a browser and a minimal bandwidth internet connection. As a result I *minimized on the cost and schedule requirements* of writing this essay.

A parallel can be drawn by considering the development process and the availability of online IDEs[9]. A plethora of online IDE providers are available, the best known being AWS Cloud9 by Amazon [10]. Cloud-based IDEs allow developers to code, run and debug application by just using a browser, a minimal bandwidth internet connection and a very low-cost laptop. The last remotely managed project I have worked on as a developer suffered from *configuration hell*: All developers were obliged to configure their laptops themselves with

[7] https://www.npmjs.com/
[8] https://maven.apache.org/
[9] Integrated Development Environments
[10] https://aws.amazon.com/cloud9/

no prescribed setup constraints and as a result, the project suffered in the initial phase and it also required developers to have costly laptops with high-end specifications.

The shortage of IT professionals in the world is not the best kept secret, and software developers have been at the top of the hardest to fill jobs in the developed world. At the same time, interest in computer science bachelor and master degree education is lacking behind in the USA [12]. In the last decade, despite the rise in demand for highly-paid software developers, awarded bachelor and major degrees have stayed flat. The last two decades, Indian consultancy service businesses have filled the void created by the ever increasing demand for IT professionals and have caused a true migration invasion of people working in the USA on H1-B visas that eclipses the current migration stream consisting of people from Central American countries looking for better economic situations in the USA. Adding to this the current stance and trend of developed countries against migration in general, it is interesting to research other opportunities of overcoming the mismatch between the increasing demand and diminishing supply of software developers in developed countries and this is where matching Silicon Valley directly with software developers in developing countries could be the solution. But in order to reap the harvest, the seeds must be sown first, which means that access to education must be guaranteed.

Universal access to education was researched by Goel et al. [13] and the following paragraph caught my attention:

> *Developing countries usually suffer from remote location, lack of infrastructure like inadequate telecommunications facilities, lack of trained staff, the cost and the availability of computers and electricity. These are all major problems that bring on poor quality in education and lack of access to it. There is a need for innovative methods of delivery which can help bridge this gap and thus provide universal access to education.*

Silicon Valley and other IT hubs around the world should take a holistic approach by taking responsibility regarding combined education and employment opportunities and consider bundling forces with cloud-based MOOC platforms like Coursera[11] and Udemy[12]. These platforms offer complete educations and technical courses, where Coursera focuses on offering academic (unaccredited) education and careers (affiliated with the most influential universities around the world) and Udemy focuses more on offering technical courses. The costs of those education tracks and technical courses are very low (up to 40-hour courses on Udemy sell for as low as 10 dollar and Coursera charges 40 dollar for complete academic tracks). Inadequate government spending in developing countries, quoting Goel et al. [13] "Inadequate government spending in the education sector is primarily blamed for poor education as an aspect in hampered national development" in combination with a multiple-decades history of corruption, gives little hope for involving local governments in these efforts and matching IT solution provider directly with software developers in developing countries might be an option.

Cloud-based IDEs require a low initial investment with regards to hardware, no software licenses if open-source components are used and a low bandwidth internet connection. Adding to this the usage of cloud-based MOOC platforms for education purposes, coupled with a proactive approach by the IT services sector in unlocking an untapped market pool of software developers in developing countries, could help decrease schedule and cost aspects of project execution.

III. Future work

Given the fact that the intent of this essay is to raise ideas and not to do a full-blown research, there is room for future work. The following ideas could be investigated in a more academic and elaborate fashion:

- *(RQ1, RQ2) - What kind of APIs and interfaces need to be developed in order to automatically collect quality measurements statistics ?*
- *(RQ1, RQ2) - Adding identity information to collected quality measurement statistics*. Research questions RQ1 and RQ2 are related with collecting measurements of quality characteristics of open-source components by cloud-based source control and CI/CD platforms. The basis of these statistics are not the open-source components itself, but the contributors who commit work to the source control platforms. By collecting statistics with regards to the quality of open-source components, implicitly, also the skills, capabilities and competences of those contributors can be deduced. Possible follow up research questions that I can imagine:
  - *How can quality related measurement statistics be used to generate resumes of contributors to open-source projects ?* Putting incorrect content in resumes is sometimes referred to as *Resume Dressing* and is an activity that is a pain in the rear for many employers and recruiters. On the other hand, it could be considered a tool for contributors to showcase their capabilities and to leave a global record of skills and competences
  - *Which controls should be put in place to manage the amount of personal information that contributors would like to share ?* GDPR and other privacy regulations are put in place to protect privacy sensitive information. Which controls should contributors be given to decide upon the amount of personal information that would be shared with employers and recruiters
  - *How can contributors to open-source components be uniquely identified ?* Contributors to open-source projects don't go through a KYC process [13]. Is there a need for a KYC process when matching identity information with quality measurement statistics of open-source components ?

[11] https://www.coursera.org/

[12] https://www.udemy.com/

[13] https://en.wikipedia.org/wiki/Know_your_customer

References

[1] A. Fuggetta and E. Di Nitto, "Software process," in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 1–12.

[2] ISO/IEC, *Systems and software engineering Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models*. BSI Standards Publication, 2011. [Online]. Available: https://pdfs.semanticscholar.org/57a5/b99eceff9da205e244337c9f4678b5b23d25.pdf

[3] ISO, *ISO/IEC 9126-1:2001 - Software engineering – Product quality – Part 1: Quality model*. ISO, 2001. [Online]. Available: https://www.iso.org/standard/22749.html

[4] N. Ford, *Building Evolutionary Architectures: Support Constant Change*. O'Reilly Media, 2017.

[5] F. Khomh, T. Dhaliwal, Y. Zou, and B. Adams, "Do faster releases improve software quality? an empirical case study of mozilla firefox," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, June 2012, pp. 179–188.

[6] P. C. L. Bass and R. Kazman, *Software Architecture in Practice, 3rd ed. Boston*. MA: Addison-Wesley, 2012.

[7] D. E. Denning and P. J. Denning, "Certification of programs for secure information flow," *Commun. ACM*, vol. 20, no. 7, pp. 504–513, Jul. 1977. [Online]. Available: http://doi.acm.org/10.1145/359636.359712

[8] E. Poll, *Lecture Notes on Language-based Security*. Radboud University Nijmegen, 2017. [Online]. Available: http://www.cs.ru.nl/~erikpoll/papers/language_based_security.pdf

[9] "Solid." [Online]. Available: https://en.wikipedia.org/wiki/SOLID

[10] M. Lanza, "Codecrawler - polymetric views in action," in *Proceedings of the 19th IEEE International Conference on Automated Software Engineering*, ser. ASE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 394–395. [Online]. Available: https://www.si.usi.ch/assets/publications/conf/kbse/ase2004/Lanza04.pdf

[11] A. Trockman, "Adding sparkle to social coding: An empirical study of repository badges in the npm ecosystem," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 524–526. [Online]. Available: https://www.cs.cmu.edu/~ckaestne/pdf/icse18badges.pdf

[12] "Principles behind the agile manifesto." [Online]. Available: https://nces.ed.gov/programs/digest/d16/tables/dt16_322.10.asp?current=yes

[13] A. W. S. T. R. Goel, C. Donald, "Universal access to education: A study of innovative strategies," 2012. [Online]. Available: https://www.yumpu.com/en/document/view/19474110/universal-access-to-education-a-study-of-innovative-strategies-erim