Aishwarya Vatsa et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 3, Issue 2, June 2016, pp. 6-9

Software Production Issues and Mitigation Techniques: A review

Aishwarya Vatsa[1] and Shiv Kumar[2]

PhD Scholar, The ICFAI University, Jaipur, India
Assistant Professor, The ICFAI University, Jaipur, India

***Abstract*- Deployment of a software product is crucial part of Software Development Life Cycle (SDLC) and agility in SDLC infuses more challenges at every phase making deployment in agile environment difficult to handle. These difficulties introduce issues at the stage of software production. To conduct a smooth software production the associated issues need addressing and mitigation. In this paper, recent research in the described area has been studied and this study is formulated into recommendation and future scope. Following these recommendations could result in a new approach which will have the potential to mitigate software production issues.**

***Keywords*— Agile Environment, Software Deployment, Production Issues.**

I. INTRODUCTION

Agile methodology is implemented frequently in the present IT industries. The various characteristics of agile have helped it to conquer the industry. Some of the agile characteristics which make it prominent are incremental delivery, iterative development, modularity, collaborative nature and adaptive to changes. Being agile is as complex as it sounds. The characteristics of agile surely produce a product which is in accordance with the client requirements; nevertheless, there is a need to keep an eye on the quality of the product. Quality is vital in any software development paradigm and the role of quality analyst is to bring the 'quality' attribute in the product. Despite keeping an eye on the product from every viewpoint, issues are raised at the time of production. To assure quality and a pleasant deployment, Quality Analysts (QA) follows various techniques. Software Quality Assurance (SQA) is an umbrella activity which covers whole of the software development procedure and is assigned majorly to the QA personnel. It is a process that ensures that developed software meets the quality requirements of the end users.

Role of QA in assuring quality:

Other than QA, members of the team involved in software development procedure, have a very little role in assuring quality. Role of QA in agile or any other SDLC is not that different. The difference lies at how the quality assurance activities are performed under agile methodology. In Agile, QA team is expected to work with the development team and collect various issues during this time. Illustratively in a particular agile technique like scrum, they are supposed to collect every bit of information regarding the encountered issues and prepare a log with the help of any issue tracking tool (e.g. JIRA). QA team should also communicate these issues to the other team members. They are also supposed to make the necessary documents regarding their testing approach. They should first create a document regarding their testing approach towards the current work product and in accordance with this approach they need to create test cases. QA team tests the work product through various testing methods (e.g. sanity test, regression test, performance test, load test or smoke test), generates the test results and publish it for the team. Eventually when all the defects are identified and fixed by the development team, QA team perform test on the fixed work product. If the work product passes these tests then QA team certifies stories as closed i.e. work product is ready for deployment in the production environment.

Problem Areas and Complication:

One could say that QA's efforts during application development/enhancement are remarkably exhaustive. However, after putting in so much of effort, still issues are raised in production environment. These issues need to be acknowledged and categorized. Clients invest a huge amount for Quality Analysts (QA) in any particular project (in the context of IT industry). Reason for this investment is that they require an application which is bug free. In the real world scenario, software is probable to breakdown if QA leaves the application infected. QA put in a lot of efforts to make the application perfect. Nevertheless, there seem to be issues raised for the application in client environment. Many projects are bunged due to these issues. These software issues needs to discovered, studied and solved.

II. LITERATURE REVIEW

This segment presents prior work done in the introduced area. The basic idea here is to consider the relevant work and channelize them to produce a path, which when followed leads to prospected solution for issues occurring during software production.

Software Deployment Activities and Challenges:

Before considering the challenges occurring during the deployment of a software product, one should have a clear idea about the activities that are performed during this time.
Stakeholder communication, installation preparation, installation and testing the installed product [1] are the activities presumed by Mika V. Mantyla and Jari Vanhanen. During the very first activity, stakeholders are informed about the product/updates they are going to receive. In the

second activity, all the preparations are done before the actual deployment. In the third activity, which is installation, the product is deployed in the client's environment. In the last activity testing is performed as well during the installation process of the product.

Software deployment activities are step by step method which assures a clean installation of software product during deployment. These activities possess various challenges, which need to be addressed and taken care of. All the challenges are related to the activities followed during deployment. Few such challenges are to minimize the deployment effort and decreasing reliability on individual experts.

Barriers in Continuous Integration (CI) and Continuous Deployment (CD):

IT organizations are moving towards continuous integration and deployment which complements the agile environment. CD is the ability of a software procedure to deliver functionality to the customers frequently [2]. Attaining CD is difficult and needs a movement from traditional development. Helena Holmstrom Olsson, Hiva Alahyari and Jan Bosch conducted a study which revealed the barriers coming in the way of CD and strategies to overcome these. "The stairway to heaven" described by Helena, Hiva and Jan, represents the evolution of software procedure followed by IT industry.

The path consists of five stairs, traditional development, Agile R&D organization, continuous integration, continuous deployment and R&D as an experiment system. These stairs leads the software development procedure to a place where customer is involved in the development procedure and functionality is delivered within short time period with continuous feedback. However reaching the final stair faces many barriers in the path. In movement from first stair to the second, insufficient software development process and use of old tools are the barriers, which are addressed through agile working practices and introduction of feature teams. Movement from Agile R&D to CI, barriers are communication and culture shift, which is solved by a fully automated testing infrastructure that verifies the work product during its evolvement. Movement from continuous integration to continuous deployment faces the barriers that arise due to different network configuration at customer sites and lack of transparency, which is minimized through introduction of agile practices in various section of software development procedure. These movements leads the IT organizations to an ideal agile development procedure, which then faces problems during deployment, as faster deployment means more error prone product to be delivered at customer sites. Production issues occur due this reason and could be solved through categorization of these issues and solving them in isolation. Above discussed barriers could be helpful in finding the production issues.

Disciplined delivery:

When moving forward with the idea of agile, an excellent framework is required if the project is large scaled. Agility is easy to implement with small development procedure, however, when agility is to be implemented at large scale, projects becomes difficult to handle. Alan. W. Brown, Scott Ambler and Walker Royce discussed that how economic governance, measured improvement and disciplined delivery [3] construct a framework for large scale agile projects. Disciplined Agile Delivery (DAD) is a framework which ensures scalability of agile process. The framework has various characteristics such as: people first, explicit scaling support, goal driven, enterprise awareness, risk and value driven, delivery focused, IT solution focused, agile, hybrid and learning oriented. All these characteristics if embedded properly in the product development procedure will ensure its reliability. Question arises that where does deployment issues are concerned in the DAD framework. DAD framework is the second generation of agile framework; therefore it is an amalgamation of all the excellent features of different agile methodologies. Goal-driven characteristic of DAD, make it aware of the issues that is associated with each goal. This awareness of issues leads the team to look for solution beforehand. Hence at the time of deployment, the realized goals must be associated with the underlying issues. And these issues could be addressed to overcome complications arising due to them. Risk and value-driven characteristic followed by DAD lets the team to identify the common risks and deliver solutions in short span of time. The delivery-focused characteristic, manages the post-delivery activities, which is necessary to handle the issues occurring during deployment. Hence, DAD framework's approach vaguely ensures less production issues.

Composable Fault Tolerance (CFT) framework:

Amid of all the failures occurring during the development and production of software product their must lie a solution to overcome these. CFT framework [4] does the same but in a different context. Keun Soo Yim, reveals the reasons behind failures occurring during deployment of big data software. Although big data products are different from general products developed by IT organization, but the procedure and issues associated are similar moreover. The CFT framework integrates previously constructed components and also has techniques to tolerate fault. Studying the previous documents of various cases following observation was made by Keun Soo Yim:

a) Production failures are generally faced by large scale infrastructure products rather than smaller products.
b) Failures have a correlation with changes being made to the products.
c) There is less chances of propagation of failures from one correlated product to other one.
d) Faults occurring during the deployment of a product increase the chances of failures.
e) Human errors are common during the development and deployment of software products. These errors could be minimised through automation.

CFT framework is based on workflow model composed of various execution entities. The prime aim of this framework is to uncover errors and make the system fault tolerant. The

idea is increment validation, which follows failure identification, automatic fault injection, integration of fault detection/recovery with protected operation and validation of fault tolerance. After increment validation, user provided control data is verified, followed by design of automated testing environment. Having all the future failures sorted out beforehand keeps the team aware of the deployment issues.

Test Orchestration Framework:

Test orchestration is a technique for automated testing and deployment of software work products [5]. It analyse the codes, selects the tests to be conducted, schedules the tests, prepare the environment, executes the tests, analyse the results and finally deploy. Primarily, its aim is to make software reliable and bug free. All the steps involved in the process are automated. This is helpful in identification of bottlenecks early. Nikhil Rathod and Anil Surve, collaborates CI and CD to develop a pipeline which is a build-deploy-test workflow.

The visualization of deploying the builds in pre-production environment first before actual deployment is a path that when followed will minimize the production issues. The design of test orchestration framework consists of components such as build automation, test automation, reporting and deployment automation. Build automation leads to a quality software because of reusing components for all builds. In testing automation, a Test Driven Development (TDD) procedure is followed which tests the work product at every stage. And testing always makes software product reliable. Reporting is a necessary component of test orchestration framework, because previous results are helpful to develop and deploy a build with its help.

The deployment automation of test orchestration makes use of continuous integration and continuous deployment, joining them to form a pipeline for designing, developing, testing and deployment. Test orchestration characteristics which are effective for handling production issues are future prediction and testing the product throughout the development. Future predictions with the help of repository assist in minding the loopholes which could become issues during deployment. Effective testing model a one way, through which bugs in the codes could be revealed, hence a better model will probably minimize production issues.

Preventing failures through Risk Management:

Risk Management in agile environment is a topic which is focused minimally by researchers in recent years [6]. Aalaa Albadarneh, Israa Albadarneh and Abdallah Qusef discussed the recent and past research done in the area of risk management in software development procedure. RM consists of identification, assessment and prioritisation of risks which leads organization to manage events which could turn the project upside down. However, while working in an agile environment, various risks associated with the project fall back. In other words agile reduces risks associated in various areas of project development.

Agile Risk Identification is supported by Daily Stand-up Meeting (DSM), where every risk or issue associated with the project is discussed and eliminated. Agile Risk Analysis calculates that how impactful the risks will be if not mitigated. Agile Risk Prioritisation decides that what risks are most catastrophic and make the team be aware for that risk. Agile Risk-Management Planning decides what approach is to be taken for confronting the risk in hand. Agile Risk Resolution, means executing the risk management plan. Agile Risk Monitoring lets the project manager monitor the risk management plan throughout the development procedure.

Various agile models implements RM in its own specific way. However, DSDM and scrum provides better ways of managing risks than eXtreme Programming (XP). The effectiveness of RM in agile model could be handful if implemented properly. It could address deployment issues beforehand and provide the team with solutions.

III. RECOMMEDATION AND FUTURE SCOPE

In the literature that has been studied there exist gaps which necessitate to be filled in for the intention of an application which is breakdown free in software production environment. There is eminent requirement to reconsider the approach of handling issues occurring during software production. This requirement appears to channelize the development procedure which when followed will mitigate issues occurring during software production. Following are the recommendations derived from the study conducted in this paper:

a) Acknowledgement of role played by QA in software development is required as the successful deployment of product lies in the hand of QA. QA needs to keep an eye on issues that could occur at the time of deployment.
b) There is lack of research performed from the perspective of a QA. Software quality lays in the hand of QA and production issues is a matter of software quality, hence, lights needs to be thrown in this area so that a refined procedure is followed assuring a peaceful deployment.
c) Production issues have not been described from the basis of their root cause. These issues ask for a root cause analysis. Digging deep into the origin of these issues will lead to the path where these issues could be minimized.
d) A stated solution has not been put up by researchers in this particular area. More tedious study should be done to uncover a concrete solution.

Futuristically, this research when complete could be extended to form an approach which will guide the QA and other team members for successful deployment of software work product. This approach then could be generalized for software product deployment having different characteristics.

IV. CONCLUSION

In this study software deployment has been explored from the point of views of various researchers. Prime motive of this study was to look for loopholes in the whole software development procedure. Centre of study was issues occurring during software product deployment. With the evolution of integration and deployment in CI and CD, there lays more

chances of production issues. Although various frameworks, such as DAD, CFT and test orchestration, have been introduced, handling issues in its particular way, still production issues occur and it needs to be addressed. Hence, there exists requirement of an approach which mitigates software production issues.

REFERENCES

[1] Mika V. Mantyla and Jari Vanhanen; "Software Deployment Activities and Challenges – A Case Study of Four Software Product Companies", *15th European Conference on Software Maintenance and Reengineering*, IEEE, 2011, pp. 131-139.

[2] Helena Holmstrom Olsson, Hiva Alahyari and Jan Bosch; "Climbing the "Stairway to Heaven"", *38th Euromicro Conference on Software Engineering and Advanced Applications*, IEEE, 2012, pp.392-399

[3] Alan W. Brown, Scott Ambler and Walker Royce; "Agility at Scale: Economic Governance, Measured Improvement, and Disciplined Delivery", *ICSE*, IEEE, 2013, pp. 873-881.

[4] Keun Soo Yim; "Norming to Performing: Failure Analysis and Deployment Automation of Big Data Software Developed by Highly Iterative Models", *25th International Symposium on Software Reliability Engineering*, IEEE, 2014, pp. 144-155.

[5] Nikhil Rathod and Anil Surve; "Test Orchestration", *International Conference on Pervasive Computing*, IEEE, 2015.

[6] Aalaa Albadarneh, Israa Albadarneh and Abdallah Qusef; "Risk Management in Agile Software: a Comparative Study", *Jordan Conference on Applied Electrical Engineering and Computing Technologies*, IEEE, 2015.

[7] S. Jansen and S. Brinkkemper, "Definition and Validation of the Key process of Release, Delivery and Deployment for Product Software Vendors: turning the ugly duckling into a swan", *22nd IEEE International Conference on Software Maintenance, 2006. ICSM'06*, 2006, pp. 166-175.

[8] J. Highsmith and A. Cockburn, Agile software development: the business of innovation, *Computer*, vol. 34, no. 9, pp. 120-127, Sep. 2001.

[9] M. Kajko-Mattsson and P. Meyer, "Evaluating the acceptor side of EM 3: release management at SAS," *Empirical Software Engineering, 2005. 2005 International Symposium on*, 2005.

[10] E. Dolstra, E. Visser and M. de Jonge, "Imposing a Memory Management Discipline on Software Deployment," *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, 2004, pp. 583-592.

[11] R.S. Hall, D. Heimbigner and A.L. Wolf, "A cooperative approach to support software deployment using the software dock," *Proceedings of International Conference on Software Engineering*, 1999, pp. 174-183.

[12] www.infoq.com/articles/agile-fails-enterprise

[13] https://lwn.net/Articles/562333/

[14] https://codecontracts.info/2011/08/06/causes-of-failure-in-software-deployments-and-solutions/

[15] http://www.theserverside.com/tip/Finding-the-cause-helps-solve-future-application-deployment-issues

