ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICACTRP 2017

International Conference on Advances in Computational Techniques and Research Practices

Noida Institute of Engineering & Technology, Greater Noida

Vol. 6, Special Issue 2, February 2017

Software Production Issues: Analysis and Categorisation

Aishwarya Vatsa[1], Shiv Kumar[2]

PhD Scholar, The ICFAI University, Jaipur, India[1]

Assistant Professor, The ICFAI University, Jaipur, India[2]

Abstract: Issues raised at the time of software product deployment are termed as Software Production Issues (SPI). They call for encountering and prospected solution because of the involved criticality. This paper presents a proposal of advancement towards the solution of SPI. It deals with the various aspects associated with SPI and how Quality Assurance Team (QA) deals with them in the context of Agile SDLC. The Agile environment in itself carries various tools and techniques to overcome setbacks. Nevertheless, there exist issues in the deployment phase. Aim here is to analyse SPI and place them in their respective category.

Keywords: SPI; Software Quality; Agile SDLC; Software Deployment.

I. INTRODUCTION

Software Production Issues (SPI) is the problem faced by the software development team during/after the deployment phase of a software work product. This phase is considered as a transition stage for the work product. The work product is deployed from the company's environment to the client's environment. This evolution of work product poses a handful of challenges which is dealt by the software development team. The challenges may transform to issues if left unhandled. Hence, there is a need for SPI management. This paper discusses the shortcomings identified during or after the software deployment phase which give rise to SPI and a proposal to mitigate SPI. The raised problem has been examined in the context of agile environment since it is a prominent methodology in today's world. Agile SDLC is an application for software work products which has the ability to change by the environment. This property of agile SDLC makes it suitable for products which are prone to changes and updates. However, a change in the context of research will not affect the ideology of this paper. Hence, there lays scope of generalisation. Before investing time in this idea there is a need to understand the broader area. Software Quality Assurance (SQA) which is the root of an existence of quality in software work product enforces QA to take various measures to produce qualitative software. The problem at hand is that after all the efforts put in by QA; still there lay scope of research. The prime reason behind this is SPI.

A. Research Objectives and SPI Resolution

Ideologies of the proposed research are followings:

- Identify the current status of research and software production issues.
- Identify SPI's impacts and challenges.
- Identify implications upon QA.
- Identify resolution approach.

The above-stated objectives are prospected to minimise SPI. QA play a vital role in this research. For this reason, QA's viewpoint is explored. The product development procedure involves various stages in agile SDLC. Development and QA team are responsible for different tasks. Loopholes identification in the adopted procedure is required, as it converges to issues later in the procedure. Hence, before finding the resolution approach for SPI, categorisation is required.

B. Categorisation of SPI

Software production stage in Agile SDLC is sighted to be most critical in nature. This stage if executed successfully leads to client's satisfaction. On the other hand, if any issue arises at the time of software production then it may hamper the software development team's work. Consequently, SPI identification and categorisation is required. It will ease-up the process of approach finding. Challenges and impacts associated with each SPI will further reduce the hazard.

The problem here is that how to identify the SPI and put them in their respective categories. Issues can only be identified through QA's viewpoint. Considering the chosen environment of this study, Agile, the understanding is that QA is responsible for SPI which leaks through all the testing performed over the software product. Hence, QA's perspective will lead to SPI identification and categorisation.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICACTRP 2017

International Conference on Advances in Computational Techniques and Research Practices

Noida Institute of Engineering & Technology, Greater Noida

Vol. 6, Special Issue 2, February 2017

Another aspect of SPI is its criticality which depends on the container project's characteristics. If the project is time critical, then it directly makes SPI crucial in nature. Hence, project and SPI's criticality are directly proportional to each other making SPI reduction a major 'need' of the software industry. Product Quality, which is the main focus for QA team, is not the only dimension which gets affected due to any type of SPI. It affects the project as a whole. If it becomes critical enough, then it can bring the container project to its closure. It gives rise to a requirement which imply that severity of SPI must be known before deployment. Severity further decides the priority of SPI. Reason behind categorisation is to understand the root cause of issue origin. When the origin is identified then approach proposal will resolves SPI through its origin itself.

This paper is prearranged in the following way. The next section describes related work done in the area of SPI. Section III explains the origination of SPI. Sections IV define characteristics of SPI and also place them in their respective categories. Section V reflects the impacts and challenges associated with SPI and finally Section VI presents the conclusion.

II. RELATED WORK AND TERMINOLOGY

This section discusses the relevant work done prior in this area. Available literatures introduce various techniques implemented to improve the product development procedure. Notable among the work done in the area of SPI is the study conducted by Maarit Laanti [1]. The study focuses primarily on the introduction of agile in large scale software development organisation. It establishes the fact that how constructive agile environment is and how it helps in software product development.

Yongxiang Hu [2] discusses the idea of embedding testing at every phase of software development procedure to produce a work product which is error free. Software testing is declared as the primary technique to overcome defects.

Mika V. Mantyla et al. [3] discusses the activities performed during the clean installs and updates. Further, the role of sanity test was studied in uncovering of bugs in client's environment.

Mary Poppendieck et al. [4] discuss lean software development procedure. The building integrity principle if applied in agile software development will eliminate some of the production issues. In this process of lean, developer and customer tests are used with the same versioning, synchronisation and semantics.

Helena Holmstrom Olsson et al. [5] conducted a study which reveals the barriers coming in the way of continuous deployment. Strategising a solution for these is also a part of this study.

Alan. W. Brown et al. [6] proposed Disciplined Agile Delivery (DAD) framework which is composed of various characteristics. Many of these could help with SPI.

Keun Soo Yim [7] proposed Composable Fault Tolerance (CFT) framework which implies various techniques to make the final product fault tolerant.

Many frameworks and guidelines are proposed for overcoming SPI. However, the study conducted over these frameworks unfolds the fact that a general view over the SPI is required. The work proposed in this paper is different from earlier work. Difference lie in the way SPI is studied. A focused and isolated view in this direction is required.

III. ORIGIN OF SPI

Although the responsibility of issue reduction lay over QA team, nevertheless other team members also play a vital role in their propagation in the deployment or production phase. Primarily, the cause of SPI is QA team. Not questioning the QA team's capabilities, there exist certain areas where they lack. These areas are:

A. Environmental Difference

The environment under which testing is performed is not the replica of the environment under which software is deployed. The differences may create credibility issues (e.g. RAM and disk space difference, Operating system difference, Number of cores difference, Number of processors difference, Latency change, etc.).

B. Regression Execution

Every release requires execution of the regression suite which ensures that no existing functionality of the work product was hampered. However, there might be some edge cases which were not automated due to the technicality involved. These cases may create issues during deployment.

TABLE I. REASONS FOR NON-INCLUSION OF CASES

S. No	Rationales	Description
1	Non-automatable	Cases which are not automated due to technical constraint
2	Edge Cases	Cases which has minimal chances of occurrence
3	Unidentified Cases	Cases which remain unknown by the QA team
4	Time Taking Cases	Cases which takes too long for execution and verification

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICACTRP 2017

International Conference on Advances in Computational Techniques and Research Practices

Noida Institute of Engineering & Technology, Greater Noida

Vol. 6, Special Issue 2, February 2017

5	Dynamic Test Data	Cases which have dynamic output
6	Complex Flow	Cases whose flow of execution is difficult to handle

Table I. describes all the reasons which force the QA team in non-inclusion of test cases in the regression suite.

C. Functional Cases

Identification of functional test cases which will execute against any release is a difficult job. If few of the functionalities are not covered by the functional testing, then it will remain unverified till deployment which could produce an issue at the time of work product release.

D. Minimal Time

In Agile environment, a strict schedule is followed for work product release, comprising of the time spent over testing. It increases the chances of failure. QA team needs to adhere to their testing plan under any circumstances which will ensure that the work product holds the quality attribute in the production environment as well.

E. Third Party Module Failure

Application, during the time of its execution, is supported by various third party modules. If any of these modules fail to execute, then it could affect application at runtime.

F. Network Glitch

In the QA environment, few scenarios remain untested. A network failure is one of them, which is an undefined exception affecting the work product in the production environment.

G. Lack of Domain Knowledge

If QA is deficient in the thorough knowledge of the domain on which he/she is working, then it could affect their testing capability.

H. Miscommunication

Agile manifesto suggests maximum communication and collaboration between the team members. Miscommunication could lead to misunderstandings between the team members and affect the work product in an undefined manner. All the above-explained points give rise to SPI. The discussion is valid for the generalised group of software development procedure. QA play a vital role during product deployment which is the reason that they need to be more focused. All the possible areas from where SPI could origin must be kept in control.

IV. CATEGORISATION OF ISSUES

Numerous frameworks, methodologies and techniques have been introduced and employed. As the number grows, there are very few of these which are indulged entirely in the reduction of SPI. A framework or an approach, which is an amalgamation of already existing theory, is entailed for SPI mitigation. It could be done by first identifying the issues and placing them in their respective category. The categorisation process will further lead to the challenges and impacts associated with SPI.

SPI have two crucial characteristics. First one is its severity and the second one is its priority. Severity describes that if a particular SPI occur at the time of deployment, then how it will impact the container project. On the other hand, priority defines that how important that issue is during the deployment procedure. Following are the cases that arise due to SPI severity and priority:

A. High severity and high priority

These issues are highly catastrophic and associate themselves with the high priority level. They need immediate action and fixture. If they were not immediately attended, then they will impact the product as well as the customer.

B. High severity and low priority

An issue under this case impacts any specific functionality of the software product sternly. However, the functionality of the product which gets affected is of less importance (for e.g. functionality which is rarely used by the customers). Hence, these issues have less priority over others.

C. Low severity and high priority

These are the issues which doesn't concern the product much, as they do not affect any particular functionality. However, they are of high priority because they need urgent fixture. For instance, a syntactic or semantic error over the homepage of a website is of high priority, but it doesn't affect the product's functionality.

D. Low severity and low priority

These are the issues which don't affect the functionality and is low on the priority level. For instance, an alignment issue on the least visited page of a website.

Severity characteristic of SPI decides that how the product will get affected and how it will influence the clients and their respective customers. Following are its types:

1) Critical Severity: These are the issues which occur in the live environment and breakdown the product.

2) Major Severity: These issues have high impact factor over the product but it would not breakdown the product when it is live.

3) Medium Severity: These issues are deferred for some time as its probability of occurrence in the live environment is very little. However, if it occurs, then it would surely breakdown the product.

4) Minor Severity: Issues which doesn't impact the product's functionality and could be ignored if the priority of the issue is also low.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICACTRP 2017

International Conference on Advances in Computational Techniques and Research Practices

Noida Institute of Engineering & Technology, Greater Noida

Vol. 6, Special Issue 2, February 2017

Priority characteristic of SPI is deciding factor for the timeline of issue fix. Following priority levels has been recognised:

1) High-Level Priority: Issues mentioned in high priority zone needs immediate fixture before entering into another stage

2) Medium-Level Priority: Issues mentioned in medium priority zone does not need urgent fixture. They are delayed for two to three sprints (in any agile environment).

3) Low-Level Priority: Issues mentioned in low priority zone does not need fixing as it has minimum chances of occurrence.

Fig. 1 consists of all the possible categories of SPI. All these categories are further bifurcated in its consequent subcategories. Following are the types of SPI:

A. Functional SPI

An Issue arising due to the functionalities of any software product are termed as functional SPI. In this category, issue occurring due to the difference between actual and expected outputs of test cases were included.

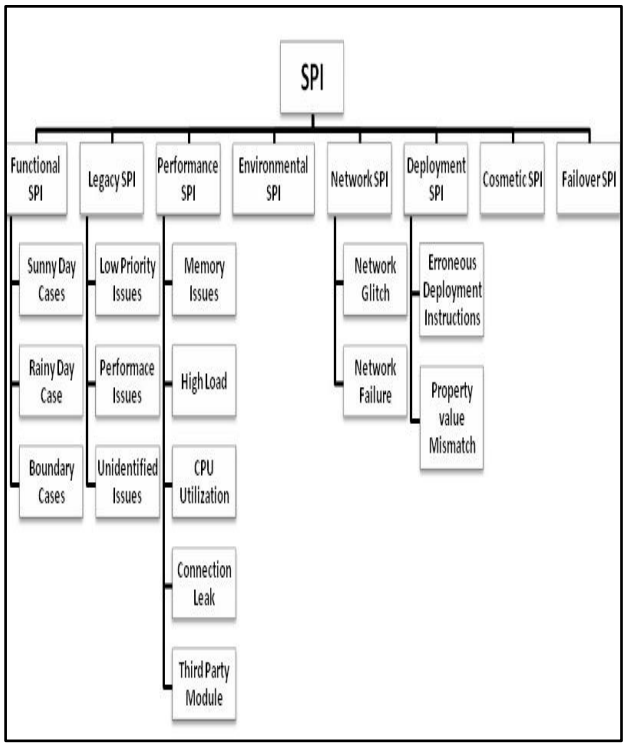

Fig. 1. SPI Categories

Below are the functional SPI types:

1) Sunny Day Cases: These are the cases which impact the application functionalities directly. These cases are easy to replicate and test. The QA team is expected to follow the procedure of replication and removal of the SPI. For instance, a booking cancellation test case for ticket booking application is easy to replicate and test.

2) Rainy Day Cases: These are the cases which are not easy to replicate and brings the QA team in a doubtful situation. They are not much likely to occur, for instance, the database server goes down while booking a ticket on a ticket booking website.

3) Boundary Cases: These are the cases where some variables of any work product have a boundary value resolution problem. For instance, if a user on a ticket booking website can book the ticket for the next 60 days, then the case of the sixtieth day lays on the boundary, and need specialised handling.

B. Legacy SPI

Legacy SPI is the carry over issues in any software product. In agile SDLC, issues get carried over from one sprint to another. Their existence in the software work product does not affect the deployment procedure. Following are the types of legacy SPI:

1) Low Priority Issues: These are the issues which do not necessarily need fixing as it does not harm the software product.

2) Performance Issues: All the performance issues related to the software work product does not require removal. These issues do not harm the software product. For instance, higher load in the production environment than expected load is a performance issue, but the load factor will not affect the software work product functionality.

3) Unidentified Issues: These are the issues which came into the picture after incremental deployment. They do not impact the software work product.

C. Performance SPI

These are the issues which affect the performance of the software work product in the production environment. There are various factors which give rise to this SPI. Following are their explanation:

1) Memory Issues: In the production environment, the load on the software product increases continuously. Primary memory (RAM) of the server plays a significant role in such scenario as it directly impacts the performance of software product.

2) High Load: High load value may harm the response time of customers. If the load on the software is not supported, then it could transform the state of the software and make it unresponsive.

3) CPU Utilisation: Under normal circumstances, high CPU utilisation is favoured. However, If the utilisation threshold of CPU is reached then, it may breakdown. This scenario needs to be avoided.

4) Connection Leak: The software product connects to various external modules in live environment. The numbers of connections have a minimum and maximum doorsill. If the connection violates these levels, then performance could be affected. If more than maximum

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICACTRP 2017

International Conference on Advances in Computational Techniques and Research Practices

Noida Institute of Engineering & Technology, Greater Noida

Vol. 6, Special Issue 2, February 2017

connections allowed are established then, the software product will exhaust in the production environment.

5) Third Party Modules: External Modules connected to the software may have degraded performance numbers which will further propagate the issue in the software work product.

D. Environmental SPI

Any software product in agile SDLC is developed, tested and then deployed. In the deployment phase, the product is supposed to deploy in development, QA and production environment. All these environments have different variables. When the software is deployed, these variables should have correct values. Slight variation in these values could create an issue at the time of deployment.

E. Network SPI

Software product in production environment gets requests and responses over the network. They also connect to third-party module over the network. Network SPI are the one which affects the software product over the network connection. Following are the types of network SPI:

1) Network Glitch: At times network can face some malfunctioning. For instance, a number of requests paused for some period and outburst of the paused requests at the same time resulting in request-spike.

2) Network Failure: Huge loss is expected if the network crashes which could impact customer and software product as well.

F. Deployment SPI

Software deployment is a complicated procedure. Various issues can occur at the time of deployment in the production environment. The deployment SPI is the issue which occurs due to some incorrect steps taken during the deployment process. Following are the types of deployment SPI:

1) Erroneous Deployment Instruction: Deployment procedure is conducted by Dev ops team, who follow the deployment instruction, step by step. If there is any error in these steps or some instructions are missing, then deployment can be unsuccessful.

2) Properties Value Mismatch: If there is any mismatch in the application properties value, then it can cause an error at the time of start-up.

G. Cosmetic SPI

Cosmetic SPI is the issue which affect the layout or interface of the software product. If any software product has a dedicated GUI, then there arise chances that it could have minor cosmetic issues, which are raised by the customers.

H. Failover SPI

These are the issues which occur due to the dependency between primary and secondary external modules. The software product is supported by multiple instances of external modules. For instance, the primary database is supported by the secondary database, which is meant for failover cases. If the primary database goes down then, software product should switch to secondary database in minimal time duration to reduce the impact of the failure. The failover cases needs handling at the product's code level.

Above described are the variously identified categories of SPI. These issues could become severe if not handled at the earliest. The severity could affect the software product at critical times.

V. IMPACTS AND CHALLENGES OF SPI

As discussed in the previous section, the severity of SPI decides that how it is going to affect the functionality of any software product. However, there are various dimensions of SPI's impact in any IT organisation. Following are the experienced dimensions:

1) Product Quality: Quality is the most important property of any product which is hampered by SPI, severely.

2) Client's Satisfaction: If issues propagate in any product then it could further lead to failure or unacceptable behaviour.

3) Client and their customer's relation: IT company's clients may be further associated with their own customers. If the product is encumbered by SPI, then it would also impact clients' reputation.

4) Company's Goodwill: If the company loses any of their clients due to SPI then it will impact their goodwill in the market.

5) Rework: Huge amount of rework is required if any product is brought back to desk due to any issue in the production environment.

All these impacts pose assorted challenges associated with the software product and SPI. Primary challenge is to overcome all the types of SPI and control them at the time of their origin.

VI. CONCLUSION

Software Production Issues is discussed to its core in the above sections, which is the first step towards an approach introduction for SPI mitigation. The introduced categories are defined purposefully and impact associated with SPI has been identified. It is the proposal of SPI categories which will further give way to motivated research in this particular direction. It was important to understand the vitality of SPI and how it affects various IT projects. All the categories, their priority and severity characteristics, will place them in an appropriate amount of light. It will help in understanding that how solution will be presented for this problem. To avoid SPI in the software product, a revised approach is required. The well-defined categories

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICACTRP 2017

International Conference on Advances in Computational Techniques and Research Practices

Noida Institute of Engineering & Technology, Greater Noida

Vol. 6, Special Issue 2, February 2017

and their respective derivation is the first step towards this approach.

Software Production is a multifaceted area. It involves various transitions, configuration changes, customers' involvement and product integration. SPI exposure will address the problems associated with all these areas. In this paper only a single aspect of SPI is covered. Ideologies presented at the beginning of this paper are not covered to its entirety. Tedious work will unfold the implications on QA and present an approach which will help in resolvng Software Producton Issues.

REFERENCES

[1] M. Laanti, Implementing Program Model with Agile Principles in a Large Software Development Organisation, Annual IEEE International Computer Software and Applications Conference, 2008, pp. 1383-1391.

[2] Y. Hu, The Application and Research of Software Testing on Agile software development, International Conference on E-Business and E-Government (IEEE), 2010, pp. 5540-5542

[3] M. V. Mäntylä and J. 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.

[4] M. Poppendieck and M. A. Cusumano, "Lean Software Development: A Tutorial", IEEE Software, 2012, pp. 26-32.

[5] H. H. Olsson, H. Alahyari and J. Bosch, "Climbing the "Stairway to Heaven"", 38th Euromicro Conference on Software Engineering and Advanced Applications (IEEE), 2012, pp. 392-399.

[6] A. W. Brown, S. Ambler and W. Royce, "Agility at Scale: Economic Governance, Measured Improvement, and Disciplined Delivery", ICSE (IEEE), 2013, pp. 873-881.

[7] K. S. 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.

[8] A. Vatsa and S. Kumar, "Software Production Issues and Mitigation Techniques: A review", IJRRA, Vol. 3, Issue 2, June 2016, pp. 6-9.

[9] N. D. Fogelström, T. Gorschek, M. Svahnberg, and P. Olsson, "The Impact of Agile Principles on Market-Driven Software Product Development", Journal of Software Maintenance and Evolution: Research and Practice, 2010, Vol: 22, pp. 53-80.

[10] I. Ruiz-Rube, J. M. Dodero and R. Colomo-Palacios, "A framework for software process deployment and evaluation", Information and Software Technology (Elsevier), 2015, pp. 205-221.

[11] X. Zhao, X. Xuan, A. Wangy , D. Liuz and L. Zhengz, "Software Quality Control via Exit Criteria Methodology: An Industrial Experience Report", 21st Asia-Pacific Software Engineering Conference (IEEE), 2014, pp. 23-26.

[12] T. Kanij, R. Merkel and J. Grundy, "An Empirical Study to Review and Revise Job Responsibilities of Software Testers", IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2014, pp. 89-92.

[13] E. Collins, A. Dias-Neto and V. F. de Lucena Jr, "Strategies for Agile Software Testing Automation: An Industrial Experience", 36th International Conference on Computer Software and Applications Workshops (IEEE), 2012, pp. 440-445.

[14] D. Marijan, Multi-Perspective Regression Test Prioritization for Time-constrained Environments, International Conference on Software Quality, Reliability and Security (IEEE), 2015, pp. 157-162.

[15] G. Schermann, J. Cito, P. Leitner, and H. C. Gall, "Towards Quality Gates in Continuous Delivery and Deployment", ICPC (IEEE), 2016, pp. 1-4.

[16] A. Mattila, T. Lehtonen, H. Terho, T. Mikkonen and K. Systa, "Mashing Up Software Issue Management, Development, and Usage Data", IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering, 2015, pp 26-29.

[17] X. Su, H. Liu, Z. Wu, D. Zuo and X. Yang, "SA Based Software Deployment Reliability Estimation: Problem Space, Challenges and Strategies", International Conference on Educational and Information Technology (IEEE), 2010, pp. V2130-V2135.

[18] N. Rathod and A. Surve, "Test Orchestration", International Conference on Pervasive Computing, IEEE, 2015.

[19] 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.

[20] 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.

[21] G. G. Claps, R. B. Svensson and A. Aurum, "On the journey to continuous deployment: Technical and social challenges along the way", Information and Software Technology 57 (Elsevier), 2015, pp. 21-31.