International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-2S, December 2018

A White Paper Attempt on Defining Key Process and Benefits of Effective Defect Management System in AGILE Based Projects - Effective Defect Management System in AGILE Methodology

Rajarajeswari Natarajan, Vaithyasubramanian S

Abstract— In Software Development & Testing life cycle, Agile approach had transformed the project delivery models dramatically and today there are innovations around key processes within the Agile umbrella. Overall, any Agile-based project methodologies promotes continuous iteration & integration of the development bringing testing services in parallel. While the overall scope keeps evolving, each "sprint" manages a subset of the scope been developed and tested and the subsequent sprints keeps building the system integrated. Agile has the benefit of bringing the end system early to market with the complete visualization of evolving or build-in-process system to the users. Unlike the traditional waterfall approach where users validate the final product directly to identify complete list of defects, there are challenges in managing and tracking defects on a growing system from base. This paper attempts on Defining Key Process and Benefits of Effective Defect Management System in AGILE Based Projects. This paper presents Workflow Chart of a Simple Agile Project with Single Phase Implementation Target, Potential Applicability for Agile model and benefits of effective defect tracking system in Agile Projects.

I. INTRODUCTION

'Never look at Testing as a separate function but blend it always with the entire flavor of Analysis, Design & Development' is what Agile speaks out! Still the popular Waterfall or V models exists and spells the success stories in Assurance services, Agile model has been slowly conquering most of the space on a challenging path [1]. While a Waterfall model defines a sequential design process that flows down to Testing at the end of the SDLC (Software Development Life cycle) to validate the entire volume of development conceived through the requirements analyzed, V-model brought in the concept of performing various Validation phases of testing against each Verification phase of the requirements, design & development cycle as Unit Testing, Integration Testing, System Testing, User Acceptance Testing, Operational Acceptance Testing. When such models looked at Testing as a distinct function of "Final Approver" on the SDLC, Agile model integrated QA along each phase of the Development to quickly identify the pitfalls and help in bridging quality gaps immediately, bringing better shape for the system while it grows [2,4, 11,14] .

An Agile project keeps building the system amidst the quality assurance on existing package hence demands for an effective, well- planned Defect management and tracking methodology [5, 7]. Working an effective Defect management system amidst the dynamic design & Development upgrades/changes, integrated component changes, Upstream/ Downstream systems consistency/ availability, Scope creeps during solutioning on raised defects, Integrated Component's Version/ Configuration maintenance, Redesigns &Deviations due to technical constraints, Assurance Team knowledge base, unstable code package during the test cycle have been real challenges on the Agile projects to execute organized defect tracking [8, 9, 10]. This Whitepaper serves to propose few process oriented alignments which can help in successful project management and better Defect tracking.

II. BACKGROUND

The basic rule of any test process would be to perform a controlled Sanity test whenever an organized code is delivered to the testing team. On an agile project, the development delivers the first meaningful piece of code based on the LLD model which is picked by the testing team with the first level of sanity, followed by comprehensive and Regression testing. Unlike a structured QA model, the challenges here begins when the next or subsequent deployments for Testing falls before the completion of first (Being under test) code level. This may again include revisions or changes to already tested function of the first code due to application fix or technical demands or business changes on running developments. The test process should get extremely robust & Adaptive for such agile model to detect potential failure points or bug-prone areas, hit the critical junction of business importance, absorb incoming defect fixed codes and logically restructure a second level Sanity to validate the complete level of code including the modifications. When Stakeholders gets the first glimpse of their required system, it should have emerged pushing back the pitfalls very efficiently in a steady growth rate of Agile [3, 6, 12, 13, 15].

Manuscript published on 30 December 2018.
***** Correspondence Author (s)
Rajarajeswari Natarajan, Test Manager, Maveric Systems, Chennai, Tamil Nadu, India. (E-Mail: rajisri.80.try@gmail.com)
Vaithyasubramanian S, Assistant Professor, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, India.(E-Mail: discretevs@gmail.com)

Table 1 Assurance Projects fitness to Agile Modeling

Project Type/ Size	Potential Applicability for Agile model
An End-to-End Requirements-to-Implementation model for new application	Extremely high
Phased approach on new functional module on existing application	High
Implementation of extension modules with existing application for new region rollout	High
Enhancements and Business user specifications rollout on existing application	Low
Configuration changes only	Low
Migration of existing application to a new platform (Hardware/ Software)	Low
Migration of upstream/downstream systems only to ensure compatibility	High
Requirements study and System Integration phase approach	High

III. BENEFITS OF EFFECTIVE DEFECT TRACKING SYSTEM IN AGILE PROJECTS

Starting Defect management early during growth cycle:

QA models like Waterfall or V, delivers almost the complete product or logically organised set of product features for testing phase. Hence a planned systematic QA phase with functional teams gets executed for the major release version and subsequent defect fixing versions. Defects are tagged to corresponding functionalities which failed, and we get the feel of testing the end product – the heavier and total version. The complete package is developed and pushed to QA to detect the pitfalls, gaps which would then be bridged to completeness. Unlike here, Agile has high number of random deployment cycles which brings in fixes along with code advancement – We test the everyday growing product. An agile sprint delivers multiple functionalities of the product but may be partially, demanding strategic test planning, key workflow tests, Retests under various test data/ environment diversity, managing tests across different versions and many times capturing defects of instability or behaviour on similar tests compared across various versions. We start monitoring the requirements and existing gaps very early, but in consistency with pace of system growth as well accommodating the changing requirements of business ever.

Integrating Defect fixes with next product stage:

The beauty of defect tracking on agile model project is that failure detection is on compounded code at any point of time, hence the system gradually progress towards the perfection. Each release (New code and defect fixes prioritised from earlier versions) gets built on previous functionality tested. With a clear and sound configuration management system, defect tracking of Agile can bring in a controlled and well tested application. For instance, if a critical bug is reported in Version 1.3, there might be a possibility of corrective behaviour (auto-fix) with related code changes in Version 1.4 or a necessity of fix to be worked upon and delivered with V 1.4.Defect gets retested in V1.4 and either closed or opens up for new bugs. This way, agile team responds to unpredictability by incremental & Iterative cadences. We start monitoring the quality with first logical code segment of the system and move-at-par with developing application package.

Information radiator:

Just because Agile always measures the quality of current integrated Product bundle, we tend to see a more realistic project status on health of the system, defect count on the whole product given, risks to be addressed etc. Unlike in traditional, where 'x' functions are tested and tracked against all 'y' functionalities of the system available at specific time (move gradually over testing phase), Agile would be monitoring 'x' functions against 'x' functions hence the numbers on Status and defect reports are latest dashboards display. Moreover, this helps in prioritising on key defect fixes as untested segments on latest code may be closer to 0.

System Stability through consistent tracking of defects:

A defect reported may not be completely fixed at first instance always, leading to failure on certain test conditions only or trigger related defects after fix. There are real time scenarios where we reopen a closed defect on future versions. Say we raise a defect on functionality ABC in Version 1.8, we retest and close the defect in version 2.1 but subsequently on testing further versions for 'ABC' with additional feature / different test data/ user profile, the defect which is at the back of our mind gets tested many times adding value. Chances of Defects (high risk prone areas) monitoring on future advanced codes makes system more stable& robust.

Self-managing and well connecting team :

To deliver the working product enhancements within short iterations adapting to changing business needs of users, pushing behind the technical challenges, solving critical issues for test progress and facing the quicker time-to-market, the project success follows a well-coordinated team. Daily Status checks, Empherical feedback, Defect analysis and ETA planning, System health monitoring, User acknowledgement- These demands connectivity between Development, QA, Business and BA in bringing a transparency across the project teams, establishing accountability, aligning to product relevancy, sharing common goals & Self manage to deliver on time.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Adapting to changing business demands to bring product relevancy:

Accessing direction of the project during development & Defect fixing optimises dev. cost and time to market. When requirement changes by business needs, existing gaps widens further that gets relevant attention and analysis to be logically closed & yet tracked. Without an efficient Defect management system, it would be tough to reflect those business changes accommodated between Product Conception and Completion which would otherwise demand investment, Effort and Time to enhance/ alter the end product to be competitive in live-market.

IV. APPROACH FOR EFFECTIVE EXECUTION &DEFECT MANAGEMENT: PROCESS ORIENTATIONS & RESULTS

Target the actual Client business:

Eyeing for critical defects on every new deployment, which means critical for the business but not just functionality would shape up application consistently for client satisfaction. A defect causing key logic failure for the system functionality may look critical/ major for QA & Dev while performing a comprehensive testing but from business need it may be a minimal volume segment or less preferred end-user feature which could be prioritized medium. Understanding real business targets and aligning the effort accordingly brings in efficiency faster. Any deviation from expected of the system should always be recorded, but prioritized with an alignment to client business objective.

Blend potential failure check points into Master Test pack:

The test pack should be constantly managed and enhanced throughout an agile styled project, since system keeps evolving in offerings during its SIT. Generating Test cases out of defects seen (Reverse methodology) or sometimes pointers for observations of possible defects (say specific environment or Source data) would help through subsequent regression cycles and BAU.

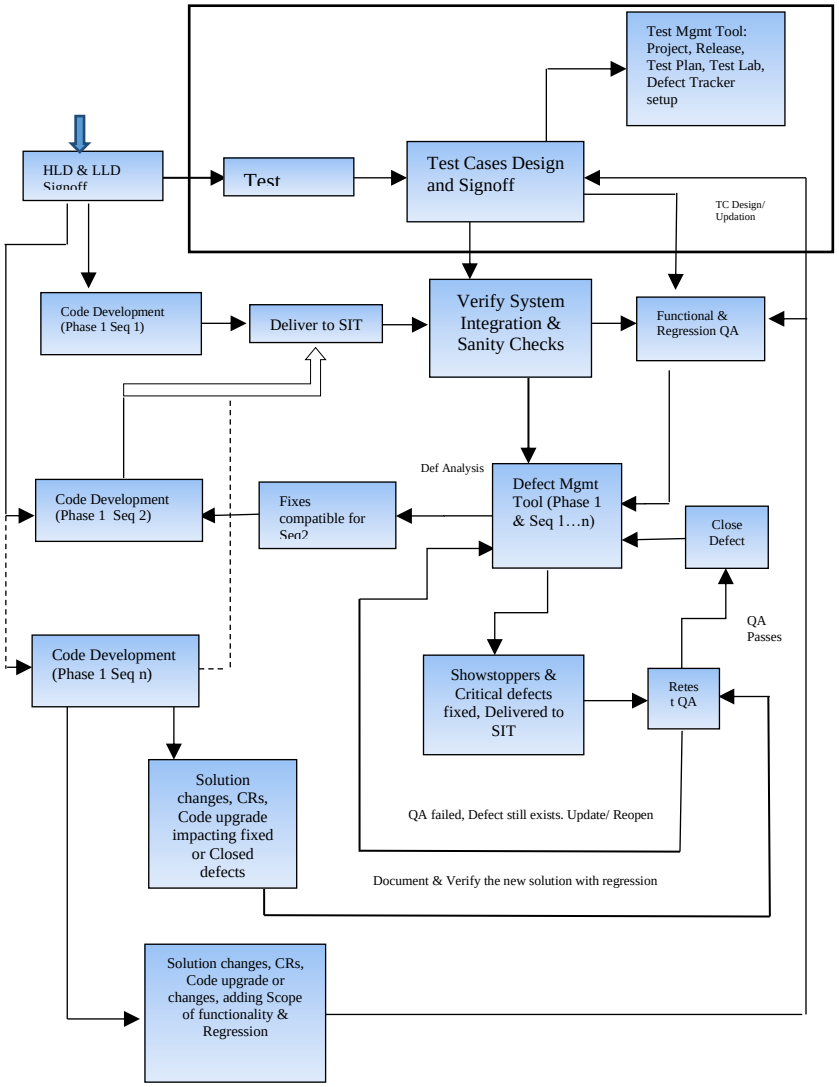

Figure 1 Workflow Chart of a Simple Agile Project with Single Phase Implementation Target

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Construct and Expand the Workflow until system is complete:

Due to code maintenance & code merge throughout the phase of development in Agile, even QA tested/ passed components have certain level of uncertainty till system gains shape &stability. Designing a business functionality workflow for the system from QA start and expanding the workflow during each stage of application building/ expansion would simplify the detection of defects across dependent components or functionalities and as well help in detecting failures on a specific version while it worked in previous versions.

Track any defect to all possible causes and impacts:

Instances occur, where a straight simple defect might have been fixed, retested and closed. Observation of another failure occurring later & its fix, could have impact to this closed defect which a functional tester cannot trace easily as Base Code dependency is the language of Developers & Technical team. Relating the defects logically with help of Dev & Business supports, wherein a subsequent impact or block due to any critical reported issues are tracked closure to the master defect for a logical test & closure.

Record & Track defects by Versioning:

Version management in Development & QA is a key contributor to faster and effective implementation in Agile based projects. Defect tools do support attributes like 'Detected version', 'Tested version', 'Fixed in Version' etc which has to be consistently understood and followed by project teams. Raising defect with Detected version where it's tested and failed at first and tracking the defect with fixed version to retest until closure, shows the actual existence and ageing of defect until the last version. A good practice is to ensure commenting on integrated component version & Interface version in defect so that retest is appropriate.

Enforce Reopening& not recreation when defect reoccurs:

Whether a previous fixed and closed defect reoccurs or a new defect impacts any segment/ test data for a previously closed defect, reopening the original defect would optimize the usage of defect management system. This tracks the quality as well throws insight on all possible failure areas due to the defect as well reference to the original solution provided earlier, while fixing the new problem.

Tag each defect to its module and monitor the defect pattern of modules:

The list of functionality/ modules delivered in each version can be added to corresponding attribute in the Defect management tool (Eg 'Component') and QA should be encouraged to associate each defect being raised to its appropriate functionality. Studying the defect pattern on each module would help stakeholders plan the fixes on next scrum to optimize the effort and quicker progress towards planned schedule. A deeper study on functionality defect pattern over the deployed cycles including severity, business impact of the defect, reopen rate etc would measure the real enhancement to its code quality at project level.

Monitor quality across all delivered components parallel in a version:

Agile poses challenge of validating any component thoroughly in a stable environment due to a) Frequent planned deployment cycles with feature upgrades b) Unplanned deployments with fixes between planned deployments for critical fixes. Planning to verify quality of all deployed functional components of a version in parallel through an effective sanity health pack and recording all critical defects across these features would help defect management at a earlier stage to prioritize system development with major bugs fading away &minimize the defect fix cycles. Strategically, not freezing functionality/ module testing to same team member in each cycle would enhance the quality due to Test approach variations as well multiply the team competency with required cross-functional knowledge.

Provide pointers to Test data failed in the defect raised:

During defect creation, the description should target the exact point of failure/ deviation and should summarize the exact issue. The defect details should further brief the workflow stepwise to recreate the same issue highlighting the step failed/ blocked and what has been the system behavior at that stage. There should be samples of Test data used and all Reference IDs of the failure product. This should help track the actual problem, sometimes which is not easily replicable with all test data- Say, a specific behavior of error might occur only when a settlement cycle is T+1 but not with T+2, T+3 etc. In case the actual occurrence of failure condition is determined by QA or Dev at later stage, Defect comments should be updated with this analyzed data. Reference to defect data has a grip in agile projects since a developed& tested code can undergo changes throughout development and cannot be controlled by one level of test.

Retest a defect with all possible interface data sourcing as per business needs:

At times, source data may not be generated just in the system being tested but flows in from few upstream. And processed data might outflow to several downstream for further functions or reporting. A defect should be validated with all possible options of data source/ output and as needed monitored for complete lifecycle of the event tested. A good practice would be to initiate from pre requisite and source data creation on the fixed version to retest the defect, instead of using existing record of the past to check only failure point of the defect.

Review and check any impact on Defect due to a new CR or Solution & Scope change:

A system redefines its shape and content during the entire agile development and that ensures certain deviations or extensions to tested components. From a single value property update or a column inclusion or View change, this can go up to business product workflow categorization or delivery of a key functionality itself. Solution updates, Scope changes, CRs should be carefully analyzed for impact to existing functionalities and complete Defect list (irrespective of current defect status) has to be reviewed to check possible failures.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

A good practice would be to reverify these defects after the change implementations, atleast once before QA closure.

Brief out a Defect Status and relevant impacts to stakeholders' day before next planned deployment:

Defects captured with all required details saves huge effort and cost while decision making. Defect management team comprising QA and Development can summarize a defect status with key attention to Pending Critical/ showstopper defects, Defects not closed as per SLA guidelines, Defects causing blocks to Test progress, Defects awaiting business answers, Defects with certain restrictions to fix, Defects which can only be fixed with yet to be developed components etc. All possible impacts of system quality/ Defect status on further code upgrades has to be analyzed with Defect review committee and stakeholders to decide on progressing on planned activities. For instance, an open defect in current version x.0 if not fixed/ closed before next QA deployment x.1 might cause a huge rework to QA subsequently when fixed at later stage by repeating much of x.1 tests done. A logical decision from such impact analysis of defect state brings in better efficiency.

Reflect the defect trend in every WSR to reveal system quality:

An Agile project, depending on the type and complexity of the system demands code deployments at specific intervals. Considering example of E2E New application build, Code deliveries are in short frequencies say twice a week at the beginning to once a week at mid-project stage. Defects might show a sharp rise at the start, then a gradual rise followed by stability in functional modules. Defect trend, one of the key monitoring aspects to measure system quality across phases should be captured at a weekly/ monthly frequency and reported to Project owner group. Defect Review committee can analyze the defect pattern across components and recommend decisions.

Record all changes on a defect solution/ fix in the tool by date and version:

A defect raised in version x.0 might go through various levels of fixes and solutioning until it's fixed completely. Sometimes dependent code changes during development might impact solutioning and demand refix at later stages. Development as well QA should ensure ownership to record/ update the defect with each fix and business solution version-wise with date. Defect history & fix quality should be traceable in the defect management tool.

V. RECOMMENDED DEFECT MANAGEMENT TOOL: QC ALM

Reasons: ALM Defect manager supports vast list of Attributes for defect capture which can be customized based on the project complexity/ type. Also, as one-time activity, specific users can align the selected attributes in specific style for defect data handling &consistent Reporting. Defect dashboards are easy to generate. Execution lab Failed and blocked test cases can be easily linked to defects for tracking.

REFERENCES

1. Arturs Rasnacis, Solvita Berzisa "Method for Adaptation and Implementation of Agile Project Management Methodology" Procedia Computer Science, 104 (2017) 43 – 50.
2. https://en.wikipedia.org/wiki/Agile_software_development
3. Shariq Aziz Butt "Study of agile methodology with the cloud" Pacific Science Review B: Humanities and Social Sciences, 2 (2016) 22 - 28.
4. Georgios Papadopoulos "Moving from traditional to agile software development methodologies also on large, distributed projects" Procedia - Social and Behavioral Sciences, 175 (2015) 455 – 463.
5. Sergio Galvan, Manuel Mora, Rory V. O'Connor, Francisco Acosta, Francisco Alvarez "A Compliance Analysis of Agile Methodologies with the ISO/IEC 29110 Project Management Process" Procedia Computer Science, 64 (2015) 188 – 195.
6. M.Balaji, V.Velmurugan, C.Subashree " TADS: An assessment methodology for agile supply chains" Journal of Applied Research and Technology, 13 (2015) 504–509.
7. Mario Špundak "Mixed agile/traditional project management methodology – reality or illusion?" Procedia - Social and Behavioral Sciences, 119 (2014) 939 – 948.
8. Piotr Nowotarski, Jerzy Paslawski "Barriers in running construction SME – case study on introduction of agile methodology to electrical subcontractor" Procedia Engineering, 122 (2015) 47 – 56.
9. Cezary Orłowski, Artur Ziółkowski, Grzegorz Paciorkiewicz "Quantitative Assessment of the IT Agile Transformation" Procedia Engineering, 182 (2017) 524 – 531.
10. Varinder Kumar Mittal, Rahul Sindhwani, Vivek Kalsariya, Faizan Salroo, Kuldip Singh Sangwan, Punj Lata Singh " Adoption of Integrated Lean-Green-Agile Strategies for Modern Manufacturing Systems" Procedia CIRP 61 (2017) 463 – 468.
11. Yngve Lindsjørn, Dag I.K. Sjøberg, Torgeir Dingsøyr, Gunnar R. Bergersen, Tore Dybå "Teamwork quality and project success in software development: A survey of agile development teams" The Journal of Systems and Software, 122 (2016) 274–286.
12. Kim Dikert, Maria Paasivaara, Casper Lassenius " Challenges and success factors for large-scale agile transformations: A systematic literature review" The Journal of Systems and Software, 119 (2016) 87–108.
13. Afshin Jalali Sohi, Marcel Hertogh, Marian Bosch-Rekveldt, Rianne Blom "Does lean & agile project management help coping with project complexity?" Procedia - Social and Behavioral Sciences, 226 (2016) 252 – 259.
14. Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, Nils Brede Moe " A decade of agile methodologies: Towards explaining agile software development" The Journal of Systems and Software, 85 (2012) 1213–1221.
15. Ashish Agrawal, Mohd. Aurangzeb Atiq, L.S.Maurya "A Current Study on the Limitations of Agile Methods in Industry Using Secure Google Forms" Procedia Computer Science, 78 (2016) 291 – 297.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)