A MANAGED APPROACH OF INTERACTION BETWEEN AGILE SCRUM AND SOFTWARE CONFIGURATION MANAGEMENT SYSTEM

Abu Wahid Md. Masud Parvez

Abstract - In current age the agile software development is one of the most popular software development methodology but due the mismanagement and lack of efficient handling of agile scrum and software configuration management system our software industry is facing a high rate of failed product, keeping this as my motivation, I have designed a efficient checklist which will help the industry to organized the interaction between agile scrum process and software configuration management system in a efficient and managed way and definitely that will increase the successful project in the software industry.

Index-term : Agile Scrums, Software development, Software configuration management system, Checklist, Successful project.

1. INTRODUCTION

Agile software development actually a collection of software development methodologies. This software development methodology stands on iterative and incremental development, where requirements and solutions evolve through self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. Software configuration management system (SCM) is combination of many important factors such as Software configuration management (SCM) comprises of factors such as compliance, workflow, security, process management, code review, build management and team work. It practices and clearly defines for development teams how software will be developed and eventually released.

2. OUR MOTIVATION

For most of the developers in the software industry, the agile methodology is nothing new. Most folks know that agile was a direct response to the dominant project management paradigm, waterfall, and borrows many principles from lean manufacturing. [2. 5, 8] There is often a painful conflict happen between SCM practices because of the difficulty of understanding how much structure will be needed. For an instance, a company's branching model can be big and much more details.[3, 5] But when they go for implementation then that may not match with the delivery cycle. Normally a company give effort for frequent product releases, and normally use large complex branching structures and that take too long to merge code. Two-thirds of all software projects fail, because of improper usage of software configuration management (SCM). So there is a big blame always on SCM. After project management IT users cite configuration management as the process that most needs improvement, to make the whole development system more efficient. In this paper I have designed a structured checklist that will make the team effective and make the development process more efficient. This checklist introduces a structured process inside the team to build better product. So the time I was producing the approach then the following three vital points always I kept in my consideration:

2.1 Velocity Method:

An Agile and Scrum-based method for distributed software R&D

2.2 Velocity Platform:

An integrated suite of various best-of-breed open source and open source-derived tools for collaboration, tracking, end-to-end traceability and product lifecycle management

2.3 Velocity Objects: reusable software product frameworks that can be used with standardized

[1]Software Quality Architect, Tech Propulsion Labs, San Francisco, USA
niloy.cit1@gmail.com,
masud@techpropulsionlabs.com,

architecture and technology stacks for distributed software development.

If we ensure all these points in our development program then we can make our whole system more at least 30% effective.

3. OUR DEVELOPED CHECKLIST

My Checklist on SCM will help engineers focus on the importance of the development process and not the rigid structure and day-to-day activities associated with an SCM system. In my checklist there is mainly six check points those need to be followed and maintain during the execution of project. The Checkpoints are as follows:

- Change set and issue tracking
- Introduce effective Continuous integration
- Impose efficient distributed development
- Frequent merge and integration

4. CHANGE SET AND ISSUE TRACKING

Development team always working with backlogs, bugs, and requirements these are always under some process but the problem is often team missed to link between these items and the location of the actual code changes. Issue tracking systems (ITS) is the primary step of the whole process as an issue is in development, there are frequent changes introduce against an existing code base in order to complete it. There should be a link as it is critical to tracking an issue back to the planning tool. This procedure makes the process easy for other team members to help complete the issue or even track its status as well. According to traditional SCM, developers manually indicate the linkage between the code and the story it is associated with as the code is checked in. But the main issue is, when the development is finished, it's hard to determine for the team which team is totally completed or party completed or need to plan for the coming sprint. Then team find though comment fields to find the status about stories but unfortunately that is not only inefficient, but fraught with errors. To get rid of this problem, we introduce tighter integration, such as syncing issues from your issue tracking system to your SCM system using tools and scripts. It is also known as "change set."

4.1. Implementation Feedback

Change set links any change in files and directories with a reason for the change and data to give full visibility into history. It becomes very easy for version control, as just pull code and build. It's easy to look back in history and see what changes were made and what tasks were associated with them but developer need to be commit tide integration. Change set can create a task-driven process out of linkage, and manage change effectively. As we move issues through the different stages of their lifecycle, it can be easy to follow that code's status during the process. A single issue may go through many different stages, like-

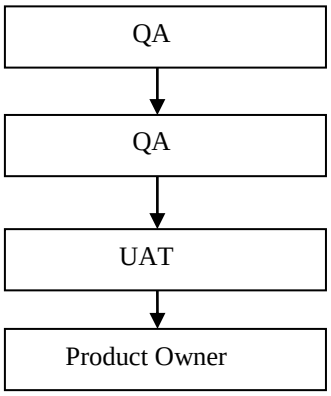

Fig. 1 in interaction inside the team

Having our code match those statuses at the same time that the issues migrate through the process is the key to pushing a successful build release out the door.

5. INTRODUCE EFFECTIVENESS OF CONTINUOUS INTEGRATION

Without continuous integration (CI) the total process is very inefficient as code is integrated several just every night or once a week. A more dangerous situation arises when the storage computer *does* allow the document to be opened by multiple developer (Such an example: Bob and Susan are two dev) at the same time. Here is what could happen:

- Bob opens the document on his computer and works on his section.
- Susan opens the document on her computer and works on her section.
- Bob completes his changes and saves the document on the storage computer.
- Susan completes her changes and saves the document on the storage computer.

So we can realize, what happen here? All the changes made by Bob are being overwritten by Susan.

CI is the process of creating builds very often and doing some basic sanity tests on the build to know if it's good or not. It helps the teams to focus on problems immediately and fix the problems instead of carry the defect in the build long time. A company should plan about it from the time of designing and implementing a continuous integration system to further reap its benefits. To gain the best benefit of CI, I would propose the following ways:

5.1. Add version of all source code to SCM

All the code components and the tests

5.2. Allowing developer local builds

It make the developer team's implementation flexible, in special case they can follow it as well.

5.3. Automated build in every change or stages:

Build have to produce on every aspect of change in the code project

5.4. Staging and private environment

Should provide the staging and private environment so that dev team can implement in private and finally push that to staging environment.

If we follow these ways then any errors arise, then developers can realize that, correcting them immediately. This process is certainly easier when utilizing an SCM system with atomic commit capabilities

6. IMPOSE EFFICIENT DISTRIBUTED ENVIRONMENT

Developing in a distributed system always is a challenge. Like your development team and your testing team can be set in two different locations. This distribution of teams not only strains the development process but always there are security auditing and integration problems throughout the process. Dealing with a distributed SCM is always challenging, even with the recent addition of DVCS (Distributed Version Control) systems. It can become more complex if a remote team asks for its own repository or server and becomes decentralized. But if we impose the following perquisite then we will able to take the real advantages of distribution development.

6.1. Shared Repositories

By imposing the most updated internet enabled tools which ensure the feasibility for distributed software developers to get up-to-date information with shared repositories. The mutually shared repositories need to provide a consolidated, real-time view of all assets, their current states, and their development histories of all the users, product lifecycle management, change management, project management help to manage processes and offer integrated collaborative characteristics like messaging, threaded discussions, etc.

6.2. Competent Network

The system needs acclimatize to provisional network outages in order to reduce the effect of the network as a restriction on performance and availability. For this the connection between users and the repository ought to utilize nominal bandwidth. To ensure that users do not feel the need to circumvent SCM procedures due to the amount of time required to check in or check out files, bandwidth efficiency is essential. For a user to work productively during an outage, network outages should be minimally disruptive and be simple for users to resolve changes made offline when connectivity is reinstated.

6.3. Backing for multi-platform:

Large organizations have numerous development groups working in cross-platform environment. Additional challenges to deployment also arise from the need to integrate code, as there are lot of update are adding in to code at the same time. There is also a surge in the data between groups that follow different development methodologies and use different tool sets. It is therefore necessary for a SCM tool to support all major hardware platforms and operating systems.

7. FREQUENT MERGE AND INTEGRATE

Frequent or early Merge and often is the practice of bringing together branches or changes of code should be committed. As the probability of a large merge is much higher. Teams have to pick the best SCM approach for

committing merging and integration. On here situation here can draw the following points:

7.1. Push the merge and schedule

Developers should be able to create private workspaces and branches to allow them to create builds, releases and tests of code before they push those changes to other team members.

7.2. Correct brunch and merge patterns

Branching patterns should make it easy for developers and team members to move and integrate code between different branches. It should be easy and straight-forward to know where a change belongs at any given time

8. CONCLUSION

Beside this checklist can be used to scale an enterprise SCM rollout, and keep development teams focusing on to the important part which is according to the SCM system. I welcome others to work with the more effectiveness interaction between defect tracking system and continuous integration, more efficient automated brunching system's architecture.

REFERENCES

[1] Coetzee, S.; Cox, S.; Herring, J, "Configuration management of a system of interdependent standards",2011, Standardization and Innovation in Information Technology (SIIT), page(s): 1 – 12

[2] S. M. Metev and V. P. Veiko, "formal Engineering method for software quality assurance", 2012, FCS.

[3] Cvitak, Lara Drvodelic; Car, Zeljka "Impact of agile development implementation on Configuration and change management in telecom domain", 2010, MIPRO, 2010 Proceedings of The 33rd International Convention, Page(s): 377 - 381

[4] IBM, "Software Configuration strategies management", vol. 2, pp. 45–291,

[5] Sutherland, Jeff, "Introduction to Agile Software Development: Lean, Distributed, and Scalable Minitrack"2012, System Science (HICSS), Page(s): 5441 – 5441

[6] Steve Berczuk, Brad Appleton., "Software configuration management patterns : effective teamwork, practical integration "

[7] Read, A.; Briggs, R.O.," The Many Lives of an Agile Story: Design Processes, Design Products, and Understandings in a Large-Scale Agile Development Project",2012, System Science (HICSS)

[8] Bass, J.M, "Influences on Agile Practice Tailoring in Enterprise Software Development", 2012, AGILE India, Page(s): 1 – 9.

[9] Anne Mette Jonassen Hass *"*Configuration management and practise*"*, 2011, web article.

[10] Traditional software configuration process model , online open source

[Online].Available:http://ibiblio.org/gferg/ldp/SCM-OpenSource/scm-traditional.html

Authors

Abu Wahid Md. Masud Parvez received his bachelor degree in Computer Science and Information
Technology from Islamic University of technology (IUT), OIC. He was bored at
23rd april 1987. Masud parvez is currently working as Software Quality Architect at
Tech propulsion labs (USA), currently he is appointed in Asia brunch of the company. Previously he
was working as Research Engineer in Electronics Research and development center, Walton.