



Our Approach in Overcoming Testing Challenges in Agile Environment

Yugank Bhatnagar, Varun Bhambri
GlobalLogic India Pvt Ltd.
B-34/1, Sector 59,
Noida 201301 Uttar Pradesh India
yugankbhatnagar@yahoo.co.in, varunbhambri@yahoo.co.in

Abstract:

In this fast changing world of software, Agile methodology is being accepted as the best and the most efficient approach towards Project Delivery. Testing under Agile offers a completely different perspective to testing and is packed with multiple challenges especially if the concept of Agile is new to the team.

Recently, we were part of a project where-in we gained more understanding of testing under Agile environment. During this, our project learning experience guided us in devising an altogether new approach, which helped us in overcoming the initial hiccups and other challenges faced during the course of the project. In this paper we would identify the benefits for our approach over the other traditional testing methods and we would also try to identify the improvement areas, if adopted, could have made our approach ideal.

Keywords:

Challenges, efficient documentation, scenario sheets, exploring the product, active communication, categorization of test cases, ready for release, adaptive testing, avoiding waste.

1. Introduction

Our approach towards implementing agile testing method was very different from the traditional testing practices. We recently successfully completed testing a product using this approach. It was a rich learning experience and helped us in gaining the knowledge of testing products in an Agile environment. The product, that we were supposed to work on, was completely a new product for us and we had no prior knowledge of it. We were responsible for the first release of this product. The following sections of this paper identify our methodology in successfully releasing it.

2. Challenges Faced

2.1. Initial Hiccups

- **There was no formal documentation provided by the client:** Requirements were in form of user stories. It was a very high-level definition of any requirement containing just enough information, which made it quite difficult to understand the scope and hence draft the test cases.



- **There was no prior knowledge of the product:** The product was an entirely new one and we were unsure of the workflows, which made it difficult to prioritize the business and technical requirements.

2.2. Roadblocks during the course of the Project

- **QA build was short-lived:** Due to the tight project deadlines, and frequent changes in the requirements, the testing time was getting continuously shortened. Initially, biweekly QA builds were delivered, but it later got reduced to daily nightly builds, which posed a great difficulty in managing our QA efforts.
- **Time to update the test case documents was not available:** Requirements were never frozen throughout the life cycle of the project. So, the documents (test cases, regression test cases) required frequent updates. With QA build cycle getting continuously shortened as the release date approached, the update task was getting quite difficult to accomplish.
- **Managing the regression task:** As per the client's instructions to perform the regression as frequently as possible to ensure that no fixed bugs get reopened, or the changes in one module doesn't affect the other modules, it became essential to perform regression testing with wide coverage. Additionally, keeping track of the bugs and updating the same in the tracking tool was equally important.

2.3. Are we ready for release?

It was challenging for the testing team to provide a definitive to answer to this question because all the QA efforts were employed in a very short time frame. With the kind of build cycles being followed and the corresponding testing, this was a question that was difficult to answer.

3. Our Approach to overcome challenges in Agile

3.1 Efficient Documentation – Capture the essence, not the details!

To minimize the risk of over-documentation, thereby wasting time on test cases which might get irrelevant with the changing requirements in the successive builds, we ensured that our documentation was highly effective and required minimal changes in case the requirements changed.

We ensured this by using the following techniques:

- **Maintained test scenario sheets:** A test scenario sheet is a matrix which includes all the possible flows which could be performed by a user in the application. In this sheet, the steps to execute these scenarios were not included. It was assumed that while execution the tester would be well aware of the steps to follow for these scenarios.

Payment Mode/Order Type	Pickup	Carry Out	On Hold	Backorder
Cash	✗	✗	✗	✗
Check	✓	✓	✓	✓
Gift Card	✓	✓	✓	✓
Credit Card	✗	✗	✓	✓
✗-Scenario Passed ✓-Scenario Failed				

Table 1: Test Scenario Sheet

- Maintained review comments sheet:** The documents we prepared based on the communication with the client, were sent for review and the review comments on these documents were maintained in a sheet known as the review comments sheet. So, instead of incorporating the changes in our documents we referred to the review comments sheet for any validation. So, it helped us to save some time in investing our efforts in documenting the changes.
- Maintained activity flow diagrams through mind mapping techniques, where-in we mapped the user's possible flows down the application.**

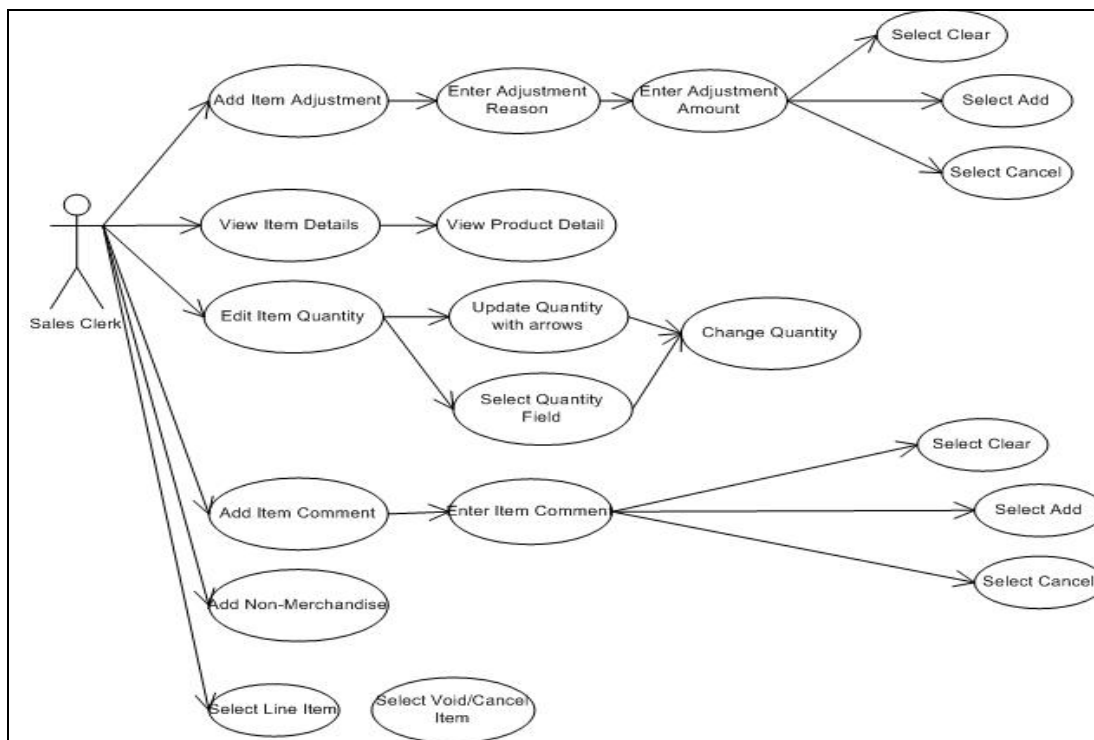


Figure 1. Activity Flow Diagram



3.2 Exploring the product – Understanding the business

As the product was new to us and we had no prior knowledge, we initially started with exploratory testing and used software from the end-user's perspective within its overall environment. This rough test helped us to uncover all the critical functionalities, defect prone areas and few important failure scenarios. Also, it helped us in prioritizing these features and areas in our future testing course.

3.3 Active Communication – Let's talk!

We adopted an approach where-in we interacted with the client and the onsite Development team, not only through the formal ways of communication like client calls and emails, but also through informal chats like IM and calling directly on their cell phones in case of any issues.

We believed that it would be a good approach to reach the client to clarify our issues on that day itself, rather than to wait for their response through any formal means of communication.

Also, we helped the onsite Development Team in reproducing the logged bugs in case they had any issues surrounding the bugs. We kept this communication as informal as possible, which established a comfort zone between the Development and the QA team.

3.4 Constant Feedbacks

We ensured to have frequent bug triages, where-in we discussed on the priority of the bugs which need to be fixed and the bugs which need to be deferred. Also, we provided feedback on the design features from the customer's perspective.

As the QA builds were short-lived, we discussed regarding the prioritization of our testing efforts and requested the client and the onsite Development team a list of the impacted areas and new modules, modified or added in the coming build which helped us in further focusing on the impacted areas accordingly.

3.5 Identification of different sets of test cases – Categorization: It always helps!

We identified and continuously prioritized executable and regression test cases with each new build. We felt the need to identify two sets of test cases:

- **Executable Test Cases:** Test cases which could be executed for the features/modules which have been built.
- **Regression Test Cases:** Test cases which needed to be run on daily basis to ensure that the newly added features do not affect the flow of the previously working modules.



3.6 Increasing the scope of the bug tracking tool

We used JIRA as the bug tracking tool in our project. Apart from logging and tracking bugs we requested the client to raise any kind of requirement changes as a change request in JIRA. This helped us in keeping the teams in sync and helped the testing team to incorporate the changes accordingly in the next test cycle.

3.7 Yes, we are ready for release – Let's go live!

When the time of the release approached, we sat with the client and judged the readiness for the build release to production. Instead of looking at the number of open bugs, we analyzed the consequence of these bugs on the overall application.

In couple of meetings with the client, we were able to identify the bugs which can be left open and does not pose a serious threat for going live. The status of these open bugs was changed to deferred and it was decided that these bugs would be resolved in the next version.

4. Benefits of our Approach

4.1. Adaptive Testing Approach – Plan as we Progress

Though the testing time was less but still we were able to execute our testing efficiently which might not have been possible had we followed any of the traditional testing approach.

4.2. Encouraged Futuristic Thinking

The brainstorming sessions between the client, the development team and the testing team helped encourage the team members to provide some valuable feedback back to the client about the product. The client highly appreciated such lateral thinking and decided to implement a few of those suggestions as enhancements in the future release.

4.3. Identification of show stoppers at early stages

Running the prioritized regression tests with each new build helped us to identify the critical bugs affecting the previously built features, when incorporated with the new modules.

4.4. Timely Delivery

With this approach, we were able to manage testing activities efficiently and hence give a green signal for the product to go live despite the tight deadlines for testing.

4.5. Avoiding waste in documentation

With our new technique to manage documents efficiently, we were able to avoid a lot of waste in terms of time and resources required for the documentation process.

5. Improvement Areas

5.1. Adding Automation for Test Management

To speed up the entire effort, we could have automated the regression tests which were being executed in each new build.

5.2. Increasing client awareness regarding our approach for future products

As this approach was planned as we progressed with the project, it was not well structured. But we were able to initiate the process at our project level to make the client aware of the process we followed throughout the product delivery cycle. This made them well-informed about our working method of testing under Agile environment.

6. Conclusion

The successful delivery and our experience with this approach made us believe that testing can be well justified even in Agile environment.

References

- [1] Ioannis G. Stamelos, Pagagiotis Sfetsos: *Agile Software Development Quality Assurance* Published by Information Science Reference
- [2] Elisabeth Hendrickson.: 2005: *Agile Testing*
<http://video.google.com/videoplay?docid=-3054974855576235846>
- [3] Naresh Jain: *Agile Testing Overview Presentation*
<http://www.slideshare.net/nashjain/agile-testing-392928>

Biography

Yugank Bhatnagar

Yugank is working with GlobalLogic India (Formerly Indus Logic). He has around 2 years of experience in software testing comprising of various steps in a project QC lifecycle which includes developing of Test Strategy, Test Environment, preparation of Test Plan, Test Cases, Test Database for Black Box testing and preparation of Test Case Report. I have been exposure in working on projects following Waterfall method and agile method for development.



Varun Bhambri

Varun is Senior Software Developer working in GlobalLogic, India. Completed B.Tech from MNNIT, Allahabad in 2006 and has been working with GlobalLogic for the last two years. Majorly responsible for designing and developing code. Involved more recently into parts of project management as well.