Volume 5, Issue 2
April 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Acceptance Test Driven Development in Web Based Banking Applications

[1]Nosheen Khan,

Department of Computer Science,

Internal Islamic University, Islamabad, Pakistan
E-mail: [1]nosheen.msse357@iiu.edu.pk

[2]Ahthasham Sajid
Department of Computing
SZABIST, Islamabad, Pakistan
E-mail: [2]gullje2008@hotmail.com

[3]Natasha Khan and [3]Shayan Ejaz
Department of Computer Science
COMSATS Institute of Information Technology, Islamabad, Pakistan
E-mail: [3]kevaa.92@gmail.com
[3]shayan.ejaz91@gmail.com

ABSTRACT

Banking sectors have transitioned from old style core banking procedures to the new era by implementing IT infrastructure as it sheer brute force to reach the corporate and random clients at their desks. E-banking services have moved ahead of being just a mere ATM card to mobile wallets and internet banking accounts to be accessed from the most remote areas of the world without any limitation of time. The banking industry has switched to web based applications in order to achieve reliability and efficiency. Products ranging from wallet cards to premium term financing has been enabled at just a few keyboard clicks for the customer. Banking sector has developed into one of the most profitable and wide client sector for the software industry by providing high scope industry based projects to be developed as well as hiring their services for the support. For the development of such systems, software houses adopt different development processes. In this paper we evaluate the efficiency of using Acceptance Testing driven development process in web based applications in the banking industry by defining a failed banking industry project and how it was re-modeled and re-developed using the acceptance driven development techniques ultimately proving to be a financially profitable solution.

Keywords*: E-banking; extreme programming; agile; test driven development.*

1. INTRODUCTION

During the last several years, multi-national companies and customer oriented structures such as banking sector has transitioned much of its operations to the web based solutions. Web-based solutions allow banks and credit unions to reduce operation costs and increase efficiency by reducing manpower, cutting the cost of IT overhead and are easily tailored to fulfill the requirements. These web based solutions, generally websites, include variety of options to be provided to the customers such as Online banking, funds transfer, Automatic Teller Machines, Mini Statements, Mobile Banking and Channel banking. Banks have decided to target the corporate clients as well as the ordinary users who only have 2 to 3 transactions in a month enabling it to address wide and multi cultured array of customers. Electronic Banking solutions have been provided to the customers enabling the banks to reach the customers' desks. These all e-banking solutions can be merged to form a single bank website that allows all these facilities to the customers just by a few clicks. To develop these websites different types of practices are used by the development teams that include Agile Methodologies, Scrum, and Incremental Models and sometimes even Extreme Programming.

While developing a website for a bank that will be hosting such mission critical systems that involves customer's money as well as 24 hour customer interaction such as e-banking services, it is very important for the development teams to adopt proper methodologies to ensure user acceptance in order to ensure the efficiency of the system. Development teams that select agile methodologies easily accommodate the changes that user suggests. Successful Web-based system development and deployment is a process, not just an event as currently perceived and practiced by many developers and academics [8]. A bank's website was developed by a software house by just collecting random user stories from the bank's staff and no proper requirement gathering or elicitation was done in order to start the development. After 4 months of development

and almost negligible testing done, the website was finally launched for the customers to access the e-banking solutions hosted by the bank. The project was a complete disaster with lots of bugs in the system and Users found it very difficult to use. The IT department received hundreds of error log forms which ultimately ended up in the system being called off.

This paper describes the advantage of using Test driven development in industrial environment, in the process of re-developing the website for the bank to host its e-banking services in a much efficient way. Test Driven development is the practice of writing tests for a software module and then using the tests as a guide for writing reliable implementation for the software module. This methodology is known as test first development [2]. For the re-development, Acceptance testing driven development was used so that the new system completely fulfills the requirements of not only the bank staff but the end user as well. Acceptance test driven development is characterized by acceptance tests written by customers usually with the help of the development team to drive the software development process, i.e. only when the test cases are specified, programmers start writing the functional code to satisfy those test cases [3]. For development of a system that should ensure the highest degree of Usability and Learnability, Acceptance Test driven development ensures that all the test cases provided are catered. Evaluation data and reviews from the end users are formulated to prove the effectiveness of Acceptance testing driven development in the given environment.

2. RELATED WORK

(ATDD) is a software development technique to develop software in increments. In this technique, test cases are specified before the functional code [3]. Two rules defined by Kent beck for Test Driven Development (TDD) are as follows: The first principle which is also the foundation for the TDD methodology is that a developer first needs to write a test and then the implementation code. Refactoring, the second principle of TDD is to improve the code without introducing/making any change in the existing functionalities [9].

Programmers can start writing the functional code after the identification of the test cases to gratify these tests. In test-driven software development processes testing starts from the preliminary development stages which further drives to the full development process. [3]
TDD seemed as one of the most successful development practices in literature to write clean and flexible code on time [3]. ATDD is found very useful (tool/technique/methodology) to outline the project's scope and validate the requirements in a very unpretentious manner during the specification phase. It provides better understanding of the requirements to developer. [4]

Our goal is to show a specific actual development process where these practices were successfully applied [3].If our challenges are time constraints, lack of domain knowledge and Lack of specifications than we can go for ATDD [3].

A test-driven development technique helps software and business related people to create unambiguous and clear-cut requirements hence assisting the organization to deliver precisely what the customer wants. TDD allows the users to maintain the code easily, and to evaluate and implement the changes in fast and effective way. The code is clear and simple, the tests have all the information required for the requirement, and hence, there is no need to review the documents in it [3].

TDD provides comparatively better reliability then traditional development approaches [9]. ATDD requires a wide support from the customer to clearly capture and validate the requirements. ATDD increases interactions between customers and developers, advances production, reduce defects and permits to automatically test software at the business level. [3] TDD is effective for any project type, size and environment [9]. ATDD needs lesser time for initial detailed design than those of traditional methodologies because the initial design is less formal. The detailed design ascends and grows with time or during the work cycle [3].
Availability and using of the automated testing tool is mandatory for ATDD [9].

For the team to use ATDD, they must know specific tools that enable ATDD [5]. There are several test automation tools in the market like JFCUnit, Selenium, FitNesse, Autolt, Proven [1]. Switching to the test-first approach or TDD doesn't guaranties one to find solution quickly or more speedy. [R1]TDD increases the quality of software but decreases the productivity [3].

In comparison of TDD projects with non-TDD projects, the total number of unseen bugs are lower in TDD projects, because the errors were found mostly during the execution of the unit tests. TDD approach increases customer's satisfaction because it condense the errors (which is) left to be found in future testing, hence the number of errors found by the customer are very low during the rollout. The code generated from TDD is of high quality because of constant integration and verification of the code as well as the internal quality in terms of the number of lines of code (LOC). The code size is clearly smaller [7].

In reality, it's not easy to attribute whether the success of a project is due to ATDD adoption or to other factors like the presence of the customer representative, the capability of the new development team, etc. In many cases ATDD or UTDD are not usually applied in commercial projects because the prevalent idea that they are costly and uncertain [3].

The paper is organized as follows. Section 2 gives an overview of the Acceptance Testing Driven Development. Section 3 presents the project background and explains the factors that lead to the adoption of Acceptance Testing Driven Development. Section 4 describes how the ACTD techniques

Volume 5, Issue 2
April 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

were implemented in the system. Section 5 presents the results and draws the conclusion.

ATDD stands for acceptance test driven development. In ATDD developers first write the automated test cases for any new functionality and then writes the functional code to satisfy the test cases. [3] Acceptance TDD helps the team to deliver exactly what the customer wants when they want it.

It is a technique where the customer is involved before coding has begun. ATDD is a collaborative practice where users, testers, and developers define automated acceptance criteria. It helps to ensure that all project members understand precisely what needs to be done and implemented. Failing tests provide quick feedback that the requirements are not being met.

ATDD has many advantages over other development methodologies like it provides better communication between the business and development teams, comforts developer to get improved understanding of the requirements. Defects are caught in the early stages of software development, ATDD provides testable code, which is easy to maintain and is of high quality. ATDD approach can decrease defect density for approximately 40 %.

From the productivity point of view, the use of ATDD has positive and negative effects it takes 16% more time for development. [9] Implementing automated ATDD is a whole lot of work, some of which is quite technical and requires the effort of everyone from the development team. ATDD is problematic to use and demands a high level of discipline from the developers. The most difficult part is to understand the need of writing unit test ahead of time in ATDD and the importance of the fast incremental cycles between writing failing unit tests and writing functional code to pass these tests. [3]

3. DEVELOPMENT THROUGH EXTREME PROGRAMMING

The bank's executives took an initiative to launch the e-banking services as it was a condition implemented by the Governing body. To launch such a mission critical system that will involve finance of the bank as well customer's money, the bank contacted several software houses so that they may submit their proposals and quotations in order to develop the website. As it included a tender system, the bank opted for the bid that involved the lowest cost with the highest number of features additionally added to the bank's requirements, without any information regarding what technology and development cycle will be used.

The contract was allocated on the basis of features and amount of time that will be taken to complete the project. The requirement engineer sent by the software house was communicated incomplete requirements in the form of user stories because there was no technical individual having expertise to define the bank's and customers' requirements in a proper manner. The software house opted extreme programming to develop the website as it involved user stories. The requirements were informally noted down on 2-3 sheets of paper, more over they were simply written user stories communicated by the core banking staff.

The Software house was not provided the proper user requirements that were required to develop the website, they still started the development process giving no priority to the fact that the system they were developing will have end users that would include illiterates and liegeman that might have never used such systems or would not be having the technical knowledge possessed by the core banking staff that had communicated the user stories. The requirements were not base lined. This was because every other day the bank staff sent e-mails so as to change specific requirements and transform them into some new ones. No end user was involved on the time of conveying the requirements ultimately ending up in the system being developed taking the usage abilities of the banking staff into consideration.

The system was tested by the developers, the main core banking functionalities seemed fine and the optimum level for the functional requirements was achieved. The Graphical user interface was very specific and seemed difficult to be used. It included terms and options that were generally known to only the bank staff. 3 officers form the bank staff tested the system, having expertise from three separate areas of the banking field. The system was tested having main focus on the accounts, finance and modules integration. No bank's representative gave priority to the Graphical user interface taking into consideration that whether the end users will be able to use it efficiently or not.

The system passed the test cases developed by the accounts and finance division of the bank. The Quality assurance department cleared it to be launched and deployed at the earliest. The system was made live and the bank's customers were informed about the newly e-banking services launched by the bank. To subscribe the new facility, the customers were charged as well. For the Quality assurance, a system was introduced, it included that if a customer experienced an error or bug in the system there will be a log form available on the website. Customer will fill the log form and forward that to the bank either on the website or it can be submitted in a printed form as well. This was done to ensure the quality of the system because customers were charged 0.25$ per month plus it included separate charges per transaction. To use the facilities the customers were assigned user ids and passwords.

The system was launched with all the facilities for the customers. The newly launched system highly attracted the customers and more than 55% of the customers signed in for the system in the first month (Value taken form the Annual report of the bank). The system started to show its negligence and only in the second month the IT department of the bank started receiving Error Log forms. The customers were not able to use the system properly and there was no support available either from the bank's IT department or the bank

Volume 5, Issue 2
April 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

staff. In a course of 3 months hundreds of error forms were reported which included the issues mainly on Usability and functionality bugs.

In such a deteriorating condition, the Bank hired 3 resources and setup an e-banking department to thoroughly inspect the issues. In the inspection report it was discussed that the main reasons for the system to fail was the use of extreme programming technique for developing such a mission critical system. There was no proper requirement elicitation and the requirements that were gathered did not include any participation from the end users. The bank management ultimately decided to call off the website and instructed the e-Banking department to launch a new re-modeled website fulfilling all the technical requirements. E-Banking department decided to include the end user views for the development life cycle ultimately choosing Acceptance Testing Driven Development process.

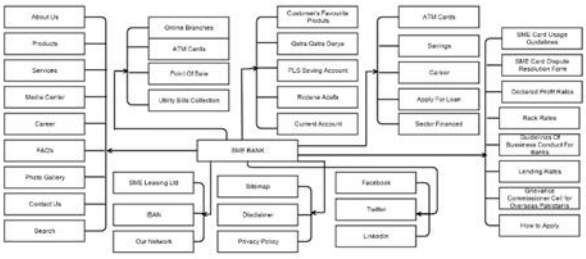
4. DEVELOPMENT THROUGH ATDD APPROACH

The E-banking department hired a new and reputable software house for the re-development of the existing system but with mutual understanding and considering the performance constraints of the existing system it was decided that the system will be developed from the scratch using the Acceptance testing driven development to involve the end users, in this case the bank customers. 7 users from different working environment and educational backgrounds were requested by the bank to allocate 2 hours on a weekend. With that the e-banking department acquired 3 experienced core banking personnel from the head office having a minimum experience of at least 7 to 8 years to ensure thorough understanding of the core banking. The time duration for requirement elicitation was decided to be 3 weeks. Software are evaluated by measuring the quality of attributes such as reliability, usability, and maintainability, yet academics often fail to acknowledge that the basic economics behind software production has a strong impact on the development process [8].

In the first phase, the functional requirements were elicited with detailed discussions with the banking personnel and the requirements were noted down on a proper file template. Further on this requirement document was forward to the bank departments' heads to ensure the functionalities to be accurate. There were minor changes that were easily incorporated. The functional requirements involved transactions of every module, the credit and debit transactions, basic banking account details, formats of Mini statement and balance statements, procedures of account linking and de-linking, term financing and further on linking the ATM cards and credit cards with the online accounts provided to the users by the bank.

The second phase involved the requirements for the Graphical user interface; therefore the end users were invited at the bank's head office. These users were frequent users of banking services. It was made sure that some of them possessed ATM cards while the others possessed Credit cards. One of these users was educated only till the 8th standard and represented the community that might use such a high tech system for the first time. There were some paper sketch prototypes prepared by the requirement analysts using and improving the pre-built called off system. The users suggested many changes which they found that would enable them to use the functionalities much efficiently. The new added requirements by the customers in the form of user stories were transformed into proper user requirements by the requirement analysts and the next week again, i.e. in the next iteration the formulated requirements were discussed with the end users. All the requirements were base lined and added to the SRS document. The final SRS document was then circulated to all the department heads and to the Software House.

In the third phase the requirements team using the SRS document prepared a Usage model for the system. Diagram 4.1 shows the usage model prepared for the system. The usage model was then further used by the Quality assurance team to develop the test cases for all the iteration of the usage model to ensure that each and every path is addressed and tested. The development life cycle was the acceptance testing driven development; therefore all the test cases were prepared and automated prior to the start of the development process. The test cases that were prepared were then discussed with the e-banking department for the authenticity of the test cases.
The fourth phase was the design and development phase. The coding teams were allocated the test cases that were relevant to their assigned tasks. The coding was done in a way that it fulfills all the test cases concerned to its domain. The domains were mainly core banking and graphical user interface. The core banking domain was developed by an entire separate team than the one developing the Graphical user interface of the banking website. Once a certain domain module was implemented and the test cases were executed and cleared, the Domain representatives, i.e. the end users or the banking personnel were informed and invited to perform the final testing of the module and to suggest any further changes. Fewer changes were incorporated because the test cases were expertly generated from the usage model and the testing team comprised of some highly skilled and experienced testers. The tool used for generating and executing the test cases was FITNESSE. FitNesse is a test automation tool based upon the Fit test framework [1].



Volume 5, Issue 2
April 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Fig. 4.1 Usage Model Created By the Test Team to Formulate the Test Cases.

After the completion of the fourth phase, the website was then deployed for a test run and the 7 users that were invited for the requirements phase were allowed to use the website for a week. An additional week was included for the banking staff to ensure the domain functionalities operable at an optimum level. After a process of 3 months the system was re-launched with the customers being notified to start reusing the system and to compensate the time wastage the users that had previously availed the e-banking facility were allowed 3 months free usage of the system.

5. CONCLUSION

The system was re-launched keeping in view the risks involved and the reputation loss that the bank suffered previously. The e-banking was solely assigned to look after the functionalities. The e-banking team comprised of 3 resources that were very experienced coders and knew a lot about the core banking functionalities as well as the system domains. To ensure the quality of the system, Standards of procedures (SOPS) were also available in the website portal for the user s to enable them solutions to the problems faced by the end users. The wide domain of the website was summarized into 5 bug domains which further included all the functionalities that were available for the users. These domains included Login bugs, e-banking bugs, Core banking Bugs, GUI bugs and the security risk bugs.

After the launch of the website, the first month's response of the website was a very good one where almost negligible bugs were reported by the customers. The e-banking department decided to extend the analysis phase from one month to six months so that a total comparative analysis could be done with the previously built system. After six months of analysis the bugs reported by the customers were compared with the bugs that were reported for the previously built system. Comparatively, very less number of bugs were reported for the new system and the bugs that were reported, the changes to incorporate the bugs were addressed very easily. E-banking department provided a full time support to the customers providing them assistance in the learnability and usability issues. Figure 5.1 portraits the comparative study and the empirical results that were collected.

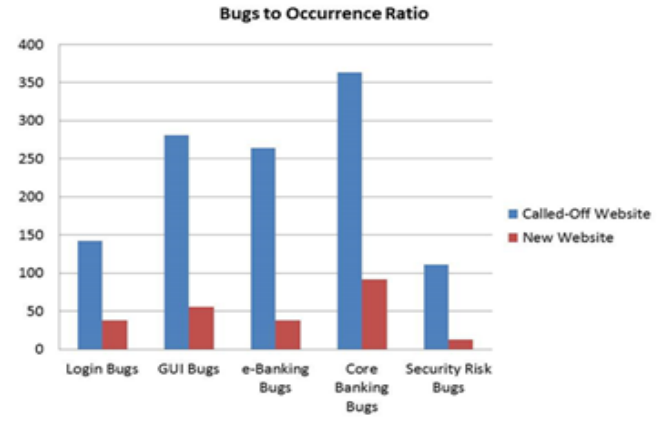


FIG. 5.1 shows the number of bugs that were reported for both the systems in the first 6 months of their first implementation.

The comparative analysis report for the first 6 months for both the systems revealed that the re-developed system had less number of flaws in it and the user satisfaction level increased significantly enabling the bank to attract more customers. The use of Acceptance testing driven development enabled the end users requirement to be addressed at each and every level of the development process ultimately resulting in development of such a mission critical system. Acceptance testing not only involved the Test cases developed by the Bank personnel or the end users, but the test cases that were generated by the senior quality assurance resources ensuring the effectiveness, reliability and learnability of the system.

The study gave an overview of the usage of extreme programming and acceptance testing driven development in the banking industry. Extreme programming is not recommended to be used in the banking sector where end user and core banking staff is involved. Whereas at the banking sector and the industrial level, Acceptance testing driven development is recommended for the projects that shall have a user array comprising of all age levels, educational backgrounds and financial conditions as the end user requirements and the core functionalities are equally addressed.

REFERENCES

[1] Marian JURECZKO, and Michal MLYNARSKI. "Automated Acceptance Testing Tools for Web Applications Using Test-Driven Development." PRZEGLĄD ELEKTROTECHNICZNY (Electrical Review), ISSN 0033-2097, R. 86 NR 9/2010 (2010)I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[2] Test-Driven Development. (2008). Software Development Rhythms, 265-289.

[3] Latorre, R. (2013). A successful application of a Test-Driven Development strategy in the industrial environment. Empirical Software Engineering Empir Software Eng, 753-773.

[4] Kam, Ben, and Thomas R. Dean. "Lessons learned from a survey of Web Applications Testing." Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on. IEEE, 2009.

[5] M. Alalfi, J. R. Cordy, and T. R. Dean, A Survey of Analysis Models and Methods in Website Verification and Testing, Proc 7th International conference on Web

Engineering (ICWE2007). Como, Italy July 2007, pp.306-311.

[6] Sara Sprenkle, Emily Gibson, Sreedevi Sampath, Lori Pollock, Automated Relay and Failure Detection for Web Applications, Proceedings of the 20th IEEE/ACM International Conference on Automated software engineering ASE '05. November 2005

[7] Lyons, R. (2007). A Survey of Analysis Models and Methods in Website Verification and Testing. Reston, Va.: [American Society of Engineers].

[8] Kazimierz Worwa, and Jerzy Stanik. Journal of Internet Banking and Commerce. N.p.: Journal of Internet Banking and Commerce, December 2010, Vol. 15, No.3, n.d. Web.

[9] Aleksandar Bulajic, and Samuel Sambasivam. "Overview of the Test Driven Development Research Projects and Experiments." Proceedings of Informing Science & IT Education Conference (InSITE) 2012 (2012)