Development of Data Refresh Tool Using Scaled Agile Framework in a Financial Corporation

Suresh M. Poudel

Follow this and additional works at: http://repository.stcloudstate.edu/mme_etds

Recommended Citation
Development of Data Refresh Tool Using Scaled Agile Framework in a Financial Corporation

by

Suresh Mohan Poudel

A Starred Paper
Submitted to the Graduate Faculty of St. Cloud State University in Partial Fulfillment of the Requirements for the Degree of Master of Engineering Management

December, 2015

Starred Paper Committee:
Hiral Shah, Chairperson
Ben Baliga
Balasubramanian Kasi
Abstract

The project was conducted in a financial organization providing various products of traditional banking and online banking. Being a huge enterprise, a lot of applications and customer records has to be stored and kept safely. Before lunching any new application a thorough testing of its functionality has to be done. Security breaches or vulnerability are unacceptable. Customer’s sensitive data cannot be compromised.

A thoroughly tested application is required to do all the transaction for a big organization like this. Every new application and every new updates have to be tested end to end. Many test accounts are created to test different scenarios. Creating a test account and conditioning is a major part of the time consuming process during the software development life cycle. When the code is moved to the integration region where the newly developed code is tested along with the already existing application, second round of testing is required. Again tester in integration testing has to create the test accounts and test the same functionality as of system testing. To remove this redundant task of creating same set of test account to test in both region and make a common reusable framework “Data Refresh” application was introduced.
Acknowledgements

The success and result of this project would not have been possible without the guidance and support of many people and I am extremely grateful to have got this all along the way of my project.

I would like to offer my thankfulness to my project committee Dr. Hiral Shah who helped me with her guidance to complete the project successfully. I would like to express my deep gratitude to the Department Chair Dr. Ben Baliga for his support and guidance to complete this project. I owe my profound gratitude to Prof. Gary J. Nierengarten, for her advice, guidance and encouragement on my project work without which I don’t think I would have been able to complete the project on time and successfully.

I would also like to thank my advisor Dr. Balasubramanian Kasi for his valuable suggestions and being the member of my project committee and his support.

Finally I would like to thank my friends and family for their support and help during this capstone project.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>6</td>
</tr>
<tr>
<td>List of Figures</td>
<td>7</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>I. Introduction</td>
<td>8</td>
</tr>
<tr>
<td>Introduction</td>
<td>8</td>
</tr>
<tr>
<td>Problem Statement</td>
<td>9</td>
</tr>
<tr>
<td>Nature and Significance of the Problem</td>
<td>9</td>
</tr>
<tr>
<td>Objective of the Project</td>
<td>10</td>
</tr>
<tr>
<td>Project Questions</td>
<td>10</td>
</tr>
<tr>
<td>Limitations of the Project</td>
<td>10</td>
</tr>
<tr>
<td>Definition of Terms</td>
<td>11</td>
</tr>
<tr>
<td>Summary</td>
<td>12</td>
</tr>
<tr>
<td>II. Background and Review of Literature</td>
<td>13</td>
</tr>
<tr>
<td>Introduction</td>
<td>13</td>
</tr>
<tr>
<td>Background Related to the Problem</td>
<td>13</td>
</tr>
<tr>
<td>Literature Related to the Problem</td>
<td>14</td>
</tr>
<tr>
<td>Literature Related to the Methodology</td>
<td>15</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Summary</td>
<td>22</td>
</tr>
<tr>
<td>III. Methodology</td>
<td>23</td>
</tr>
<tr>
<td>Introduction</td>
<td>23</td>
</tr>
<tr>
<td>Data Collection</td>
<td>29</td>
</tr>
<tr>
<td>Data Presentation</td>
<td>30</td>
</tr>
<tr>
<td>Budget</td>
<td>32</td>
</tr>
<tr>
<td>Timeline</td>
<td>32</td>
</tr>
<tr>
<td>Summary</td>
<td>33</td>
</tr>
<tr>
<td>IV. Data Collection and Analysis</td>
<td>34</td>
</tr>
<tr>
<td>Introduction</td>
<td>34</td>
</tr>
<tr>
<td>Data Presentation</td>
<td>34</td>
</tr>
<tr>
<td>Data Analysis</td>
<td>41</td>
</tr>
<tr>
<td>Summary</td>
<td>47</td>
</tr>
<tr>
<td>V. Results, Conclusion, and Recommendations</td>
<td>48</td>
</tr>
<tr>
<td>Introduction</td>
<td>48</td>
</tr>
<tr>
<td>Results</td>
<td>48</td>
</tr>
<tr>
<td>Conclusion</td>
<td>49</td>
</tr>
<tr>
<td>Recommendations</td>
<td>50</td>
</tr>
<tr>
<td>References</td>
<td>51</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Timeline of the Project</td>
<td>32</td>
</tr>
<tr>
<td>4.1 Request by Months</td>
<td>36</td>
</tr>
<tr>
<td>4.2 Request by Line of Business(LOB)</td>
<td>38</td>
</tr>
<tr>
<td>4.3 Request of September</td>
<td>40</td>
</tr>
<tr>
<td>4.4 Monthly Development Cost Calculation</td>
<td>43</td>
</tr>
<tr>
<td>4.5 Breakeven Analysis Table</td>
<td>46</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 REST API Architecture</td>
<td>15</td>
</tr>
<tr>
<td>2.2 SDLC Life Cycle</td>
<td>16</td>
</tr>
<tr>
<td>2.3 SAFe Portfolio Vision</td>
<td>19</td>
</tr>
<tr>
<td>2.4 SAFe Program Level Abstract</td>
<td>20</td>
</tr>
<tr>
<td>2.5 Team Level Abstract</td>
<td>21</td>
</tr>
<tr>
<td>3.1 DB2 Database View</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Mainframe Screen</td>
<td>26</td>
</tr>
<tr>
<td>3.3 Defects Section on HP ALM</td>
<td>27</td>
</tr>
<tr>
<td>3.4 Value Stream</td>
<td>27</td>
</tr>
<tr>
<td>3.5 Factor That Constrain And Define Optimum Release Train Size</td>
<td>28</td>
</tr>
<tr>
<td>3.6 Value Stream Mapping For Priority Four Defect</td>
<td>29</td>
</tr>
<tr>
<td>3.7 Pareto Chart</td>
<td>31</td>
</tr>
<tr>
<td>4.1 Line of Business</td>
<td>35</td>
</tr>
<tr>
<td>4.2 Bar-Chart of Request by Month</td>
<td>37</td>
</tr>
<tr>
<td>4.3 Line-Chart of Request by Line of Business (LOB)</td>
<td>39</td>
</tr>
<tr>
<td>4.4 Pie-Chart of Requests in Month of September</td>
<td>41</td>
</tr>
<tr>
<td>4.5 Sample Account Creation Page in Mainframe</td>
<td>42</td>
</tr>
<tr>
<td>4.6 Break Even Graph</td>
<td>47</td>
</tr>
</tbody>
</table>
Chapter I: Introduction

Introduction

Capital One Financial Corporation supports traditional and e-banking. Traditional banking includes commercial/treasury management, retail/small business, trust/wealth management and mortgage. It has a lot of web and desktop application to carry out millions of transaction per day for different sectors like credit card, bank, loan, etc. This capstone project mainly deals with the development of the tool implementing lean-agile methodology using scaled agile framework.

A lot of testing is required to bring new application like online banking, balance transfer, issue new credit card, employee information portal. Test data’s are created for each scenarios. In every new release the regression testing is required to verify if the previous functions are working as expected. During the testing a tester will do positive and negative testing and modify the test data. It is very tedious task to revert back all test data to initial condition to test when newer version of the tool comes. So to increase efficiency and accountability the Data Refresh tool acted as the repository of the freshly built data so that tester can test the same data in each new version to verify all the previous functional lists are working as expected.
Problem Statement

The company had different backend databases and mainframe applications to save the data. During each phase a tester had to test repeatedly the same set of data to verify each and every module is working as expected. The tester had to do positive, negative, integrated and regression testing. Each time the testing was done the data got modified in the backend systems. Most of the time common test data worked to test all these testing scenarios. So a lot of time was consumed while creating the test data to initial condition manually. So there was a need of a tool that would revert all test data to initial states so that they can test multiple scenarios with same set of data’s.

Nature and Significance of the Problem

Hundreds of application was supported by Capital One. Each new version had to be fully functional and tested before it was brought to the enterprise level. In many cases the tester would verify if the previous features of the application was working as expected or not. But tester would try to break the functionality in every possible way to make the application robust and handle every possible scenarios that can break the general business logic and security flaws. During the process a lot of data had to be modified as an example for an account they may change the credit limit, credit score, address, deposit etc.

Once the set of accounts were created they should be preserved in one backend so that it can be loaded to the testing environment for testing. A lot of manual effort of creating the test data decreased the efficiency and increased the cost of the application’s creation and support. Each time when there was the resource change
during the development of project incurred extra cost by losing the test data. Creation of redundant data by different feature teams for same function-test was another major drawback.

**Objective of the Project**

The main objective of this project was to create a data refresh application that would save the initial fresh data and can be loaded to respective databases by using the tool or using the external tools. Two major objectives were:

1. To create the tool to do the data refresh.
2. To create a common Automated Test Driven Design (ATDD) framework using (Representational State Transfer Application Program Interface) REST API so that any external tool can invoke the data refresh without using the tool.

**Project Questions**

The following questions will be answered at the end of the study:

1. What will be the likely cost in developing the data refresh tool?
2. What is the cost savings after developing the tool?
3. What is the customer feedback on the tool?

**Limitations of the Project**

This project only considered major databases tests. Many data’s were considered confidential so could not expose sensitive reports and findings.
Definition of Terms

The Scaled Agile Framework, or SAFe, provides a recipe for adopting Agile at enterprise scale. SAFe is based on Lean and Agile Principle and there are three levels in SAFe [2]:

1) Team

2) Program

3) Portfolio

1) At the Team Level:

Scrum with Extreme Programing engineers practices are used. Teams will deliver the fully shippable program or tool in every 2 teams which is functional and tested fully.

2) At the Program Level:

SAFe defines an Agile Release Train (ART). As iteration is to team, train is to program. 5 to 10 teams forms an ART and Each ART (or train) in an organization is the primary vehicle for value delivery at the program level. It delivers a value stream for the organization. Every 12 weeks (6 sprints) it will deliver a Potentially Shippable Increment (PSI) where they demo the product and lunch it to the end customer. Here different projects can be combined together which were developed by different teams and shipped as single product. During this PSI meeting they will prioritize projects and start working for another 3 months (PSI length). New program level roles are defined.

- System Team
- Product Manager
- System Architect
- Release Train Engineer(RTE)
- Shared Resources.
- Release Management Teams.

SAFe defines the hierarchy of Epics-Features- User Stories. Epics are defined at the Portfolio level. A backlog item is the prioritized list of features. Features are further divided in User Stories which is taken by each member of the team in every sprint.

3) At the Portfolio Level:

Strategy, funding, program management is done at portfolio level.

Summary

This chapter briefly explained the background of this project, problem statement, nature and significance of the problem, objective of this project and limitations of this project. The following chapter would give the literature review related to this project.
Chapter II: Background and Review of Literature

Introduction

This chapter will talk about the background related to the project as what kind of company the project was conducted, what is their area of expertise and what are their services. Similarly, it will discuss about the various literature reviews, articles and journals associated to the project. Furthermore, it will also provide different literatures and articles related to the methodology used in the project.

Background Related to the Problem

Capital One Financial Corporation is diversified banking company focused primarily on consumer and commercial lending and deposit origination. It does both local banking and national lending. Local banking includes consumer, small business and commercial deposits, and lending conducted within its branch network. U.S credit and debit card, auto finance comes under national lending. Global financial services sub-segment consisting of international lending activities, small business lending, installment loans, home loans, healthcare financing and many other activities.

Many enterprise applications help to do the regular bank transaction, bill pay, credit card transfer, auto loan service etc. All these application needed accounts to be tested. So the company made the common internal application called Data Refresh which had two basic purpose. First was to create the repository of the test account using tool and second was to create and update the repository using the Representational State Transfer Application Programming Interface (REST API) through common REST
I worked as a developer by using Active Server Page (ASP .net) software development framework.

**Literature Related to the Problem**

First challenge was to develop a common data refresh platform using .net framework and next to host REST services so that external application can use the common platform.

ASP.NET is used as the development platform for the application. It is an open source server side web application framework designed for web development to produce dynamic web pages. It is built on Common Language Runtime (CLR), allowing programmers to write ASP.NET code.

Representational State Transfer (REST) is a style of architecture based on a set of principles that describe how networked resources are defined and addressed. The architecture designed to facilitate the development of flexible, agile applications by reusing common components within a general model of workflow managements. By the use of the service oriented architecture language dependency is removed i.e. an application written in Java, Python, Ruby, PHP, Perl, etc. can easily use ASP.net hosted.
Microsoft Structured Query Language (Ms. SQL) was used as the database to store all the data’s and test accounts. It has huge storage capacity and constantly monitored for its health.

**Literature Related to the Methodology**

The Software Development Life Cycle is a process that ensures good software is built. Each phase in the life cycle has its own process and deliverables that feed into the next phase. There are typically 5 phases starting with the analysis and requirements gathering and ending with the implementation. Let us look in greater detail at each phase (Mares, 2013):
Requirements gathering/analysis: This phase is critical to the success of the project. Expectations need to be studied in great detail and documented. This is an iterative process with much communication taking place between stakeholders, end users and the project team. The following tasks comprise gathering requirements (Mares, 2013):

- Identify and capture stakeholder requirements using customer interviews and surveys.
- Build multiple use cases to describe each action that a user will take in the new system.
- Prototypes can be built to show the client what the end product will look like. This means taking a look at your customers, figuring out what they want, and then designing what a successful outcome would look like in the new software (Mares, 2013).
**Design:** Technical design requirements are prepared in this phase by lead development staff that can include architects and lead developers. The Business Requirements are used to define how the application will be written. Technical requirements will detail database tables to be added, new transactions to be defined, security processes and hardware and system requirements (Mares, 2013).

**Coding:** This phase is the actual coding and unit testing of the process by the development team. After each stage, the developer may demonstrate the work accomplished to the Business Analysts and enhancements may be required. It’s important in this phase for developers to be open-minded and flexible if any changes are introduced. This is normally the longest phase of the SDLC. The finished product here is input to the Testing phase (Mares, 2013).

**Testing:** Once the application is migrated to a test environment, different types of testing will be performed including integration and system testing. User acceptance testing is the last part of testing and is performed by the end users to ensure the system meets their expectations. At this point, defects may be found and more work may be required in the analysis, design or coding. Once sign-off is obtained by all relevant parties, implementation and deployment can begin (Mares, 2013).

**Implementation/deployment:** The size of the project will determine the complexity of the deployment. Training may be required for end users, operations and on-call IT staff. Roll-out of the system may be performed in stages starting with one branch then slowly adding all locations or it could be a full blown implementation (Mares, 2013).
One of two methods can be followed in a SDLC process. Waterfall is the more traditional model and has a well-structured plan and requirements to be followed. This method works well for large projects that may take many months to develop. The Agile Methodology is more flexible in the requirements, design and coding process and is very iterative. This process works best for smaller projects and expectations of continuous improvement to the application. Whether you use one over the other will also depend to a large extent on the corporation and skills of the IT department (Mares, 2013).

Scaled Agile Framework (SAFe)

The Scaled Agile Framework (SAFe), created by Dean Leffingwell, brought the great momentum in the organization level. It is currently supported by several vendors including Rally, Net Objectivex and Valtech. SAFe is very concrete. It provides specific guidance at the team, program and portfolio level- which makes it easier to understand because everyone is aware of their duties and boundaries. SAFe’s merges body of knowledge and the lessons learned from hundreds of deployments it a single framework.- a system of integrated, proven practices that has been demonstrated to bring substantial improvement in employee engagement, time-to market, solution quality and team productivity. SAFe is divided into 3 levels

1. **Portfolio Level**
The highest level of SAFe is the Portfolio Level, where programs are aligned to the enterprise business strategy along Value Stream lines. Decisions made here drive the overall economics for the portfolio. In Capital One there were multiple portfolios. Portfolio vision consists of:

*Strategic themes*: Specific, itemized business objectives that connect the portfolio(s) vision providing business context for decision making.

*Value Stream*: Long lived series of system definition, development and deployment steps used to build and deploy systems that provides a continuous flow.

*Budgets*: Fund allocated for the agile Release Trains.

*Epic*: Includes business epics and architectural epics which are technological changes that must be made the system flowing.

*Epic Owners*: People who take the responsibility of the epic as it moves through the system.

*Portfolio Backlog*: This is the highest-level backlog in the framework, and serves as a holding stage for Epic that makes it through the Kanban systems and waits for implementation.
2. **Program Level Abstract**

In this level funding for the personnel and other resources are applied to some important, long-lived *Agile Release Trains (ART)*, enterprise mission. *Program Increment (PI)* which is typically 8-12 weeks is multi-iteration time box during which significant, valuable increment of the system is deployed. *Agile Release Teams* meet with key program *stakeholders* on the PI and plan the next increment during *Release Planning* session which is typically supported by a *Release Train Engineer* who serves as chief *Scrum Master* and helps “keep the train on the tracks”.

All member meet for 1-2 days and plan together to meet program PI objectives. According to the *vision, roadmap* and *program epics features* prioritized by the *product manager* are accomplished. *Business owners* are responsible to assure that the train gets the fast market feedback it needs. *System team* and key specialists such as system architects and user experience designer and shared resources integrate, refine and validate system code.

![SAFe Program Level Abstract](image)

*Figure 2.4: SAFe Program Level Abstract*
3. **Team Level Abstract**

*Agile Teams* are the engine of software development. Each team has 7 +/- 2 team members. Team is responsible for defining/building/testing user stories from their backlog in a series of fixed-length iterations (sprints). Team will have scrum master, product owner, developers, and tester and may have shared resources. Teams are involved in planning, grooming, sprint review and all other agile ceremonies. The team backlog consist of user stories, upcoming features, and other backlog items. Most of these items are identified during release planning, when the product management presents the vision, roadmap, and program backlog identifying, maintaining, prioritizing, scheduling, elaborating, implementing, testing and accepting user stories is the primary requirements management process at work.

![Figure 2.5: Team Level Abstract](image-url)
Summary

In this chapter, we discussed the literature related to ASP.NET, REST API and Ms. SQL database. Similarly we also discussed literature on Software Quality Assurance and about literature related to Software Development Life Cycle (SDLC), different kinds of Software Development Life Cycle models, their advantages and disadvantages. Finally, we discussed literature on SAFe. The next chapter will discuss the methodology we are adopting in this project.
Chapter III: Methodology

Introduction

This chapter provides a detailed description of the methodology used in this study. This chapter will outline the data collection process including methods and techniques used in the capstone project, the budget used for the project and the project timeline.

A real time desktop application at an enterprise level was developed using agile methodology with Scrum. The SAFe focus solely on describing the best practices, roles activities and artifacts that enterprises have used to achieve the significant business, economic, and individual benefits that result from successful implementation of Lean-Agile methods at enterprise scale.

In a Scrum different people hold different roles. Product owner communicated the product vision and release goals. He continually groomed and prioritized the items in the product backlog to best achieve goals and missions. He ensured that team understands the items in the product backlog to level need and accepts or rejects work results. Scrum master in other hand ensured that the process is understood and followed. He facilitated scrum events. He enabled close cooperation across all roles and functions and removes barriers. Next, development team was cross-functional, included all skills needed to ensure the work gets “Done”. This team was accountable to complete the work, task work, and estimates work, volunteered to perform work and finally demoed work results to the end – user and stakeholders.

As the part of agile team Capital One encouraged co-locate team as often as possible, rotated members around (cross pollinate), planned to experiment with new
tools (Live Meeting, One Note and Instant Screen Sharing), use Version One (the system of record for agile work), used Virtual teams building (common portal), established core hours and norms, developed a shared team vocabulary and finally applied Scrum-of Scrum technique (S2’s). The core rule of following the SAFe was to ship the Minimum Viable Product (MVP). Minimum functionality that is useful to customers, will take less time to get into their hands and start earning money.

Basic understanding of agile is everyone can write stories which is directly tied to increase business value such as focus on the customer needs, easy to prioritize, small pieces of work, allow frequent feedback.

**INVEST Guidelines of Good Stories**

**I-** Independent: Does not depend on other stories.

**N-** Negotiable: Stories are a conversation starter

**V-** Valuable: Must deliver value directly to a user

**E-** Estimable: We have enough information to estimate the amount of work

**S-** Can be completed within a sprint

**T-** Can be tested easily and completely.

Smaller stories were always better because it was easy to understand, easy to get it done. So splitting of stories to smaller blocks helped a lot. Every stories had acceptance criteria. Acceptance criteria were the base for definition of “DONE” for the story. Stories were estimated using a relative measure of effort and size known as story points. Story points included the entire effort to complete the story. Story points were sized as the Fibonacci series like 1, 2, 3, 5, 8, 13…. After a few sprints it was clear how many story points team could typically do in a sprint; this was team’s VELOCITY.
Test accounts were created for different backend systems. Below are figures of the backend systems Db2 and IBM mainframe sessions where 90% of the data were stored.

Figure 3.1: DB2 Database View
Any defect found in the tool was reported in the HP ALM defect log window.

Each defects or bugs were resolved according to the severity. It may be P1, P2, P3 and P4. 1 as the highest priority ticket and should be resolved within 4 hours.
Value Stream

Value stream are realized by Agile Release Trains, virtual organizations formed to simplify development, eliminate unnecessary steps and to delivery of value via implementation of Lean-Agile principles and practices. Each release train is a virtual or solutions-based organization that exists only for one purpose to define develop and deliver that specific flow of value.
Value streams helped systematic analysis and improvement by the lean tool of Stream Mapping. Once value was identified, the next step was to understand where in the organization that value was created, because that was where we found the people, processes and assets that are the target of the Lean-Agile initiative.

Finding the Kidney is a metaphor for identifying the individuals, teams, and other resources need for a value stream. The Agile Release Train was the primary organizational, operational and value delivery construct in SAFe. There were a number of factors that tend to limit the size of an effective release train to about 50-100 people.

*Optimum ART size is based on:*

- **Dunbar's number** “…a suggested cognitive limit to the number of people with whom one can maintain stable social relationships”

- **Empirical evidence.** Beyond 125, logistics and inter-team dependencies are more difficult. Alignment is harder to achieve.

- **Queue size and WIP.** Larger numbers of teams create more dependencies (per team), longer delay queues, and more work in process


*Figure 3.5: Factors That Constrain And Define Optimum Release Train Size.*

Another benefit of identifying the value streams was it provided a named, identifiable, and measurable flow of value to a customer. It helped to improve velocity and quality. “Value Stream Mapping” is derived from lean manufacturing and an integral part of Six Sigma. It is the tool used to document, analyze and improve the flow of information required to produce a product or service.
Figure 3.6: Value Steam Mapping For Priority Four Defect

Total touch time = 2 hr. 42 minutes

Delivery time for priority 4 ticket = Inventory time + Actual time (Touch time)

\[ = 30 \text{ hr.} + 2 \text{ hr. 42 minutes} \]

\[ = 32 \text{ hr. 42 minutes} \]

Efficiency = Touch Time/ Delivery Time

\[ = \frac{2 \text{ hr. 42 minutes}}{32 \text{ hr. 42 minutes}} \]

\[ = \frac{162}{1962} \]

\[ = 8.25\% \]

Data Collection

As the development process went ahead, more features were delivered in each sprints to the end users. Bugs and defects were raised and handled in the successive
sprints. All the defects were collected in HP ALM application. Resolution and progress were updates about the defect where updated in same so that the end user have better visibility on its fix.

**Data Presentation**

Team had created a report view in the tool which would capture each request count from each Line of Business (LOB). Users reported bugs and defects were logged properly and presented bi-weekly. Any defects raised were categorized as major, minor and fatal according to severity level. A defect, which will cause an observable product failure or departure from requirements was categorized as major. A defect that will not cause a failure in execution of the product was categorized as minor defect. A defect that will cause the system to crash or close abruptly or effect other applications was categorized as fatal error.

Each week, the number of defects were collected and analyzed. This helped in analyzing the progress in development process, how well the teams are doing, what kind of defects were found, what should be the solution to resolve the defect issues and how should the team go ahead in the process. Pareto chart shows below the defects for the month of September.
The number of defects were reduced and resolved immediately which helped in decrease in the cost for the company affected by the interruptions. This was examined using the graphs in each week period.
Budget

The team had two Developers, one Quality Analyst (QA), one Business Analyst (BA), one Scrum Master and a Team Lead. Team was working in 2 other projects. For data refresh project developer dedicated 50% of their bandwidth and other team members around 35%.

Timeline

Table 3.1: Timeline of the Project

<table>
<thead>
<tr>
<th>Activity</th>
<th>Timeline</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literature Review Proposal</td>
<td>18 February 2015</td>
<td></td>
</tr>
<tr>
<td>Requirements Specification</td>
<td>February 25 – March 11</td>
<td>1 sprint</td>
</tr>
<tr>
<td>Analysis and Design</td>
<td>March 12 – March 25</td>
<td>1 sprint</td>
</tr>
<tr>
<td>Development</td>
<td>March 26 – June 03</td>
<td>5 sprints</td>
</tr>
<tr>
<td>Integration Testing</td>
<td>June 4 – July 08</td>
<td>2 sprints</td>
</tr>
<tr>
<td>Deployment</td>
<td>July 09 – July 22</td>
<td>1 sprints</td>
</tr>
<tr>
<td>Maintenance</td>
<td>July 23- August 25</td>
<td>2 sprints</td>
</tr>
<tr>
<td>Final Defense Presentation</td>
<td>November 2015</td>
<td></td>
</tr>
</tbody>
</table>
Summary

This chapter discussed about process of the data collection used in the project and how the data analysis was done along with the progress of the project. The budget information along with the timeline of the project was also presented. The next chapter will discuss about how the data was presented and also the analysis of the collected data.
Chapter IV: Data Collection and Analysis

Introduction

This chapter presents the actual data that was collected by the team. It presents the number of request served by the tool every month. It shows the request served by Line of Business (LOB). Finally break even analysis is done. It also shows how the data analysis was done using graphs and tables for the team involved and how the progress was studied during the whole project time.

Data Presentation

Request by Months: Although the project time line was for September, because of agile process end users started using tool as early as June. The tool was given access to the two pilot teams and our team was closely scrutinizing their defects and enhancement suggestion. By selecting the limited users early fixes were carried out.
Line of Business (LOB):

Capital One
Enterprise

- Digital (IT)
- Credit Card (Visa)
- Retail (Capital One)
- Partnership (GMC, Kohl’s)
- Home and Auto Loan
- Commercial

Figure 4.1: Line of Business
Table 4.1: Request by Months.

<table>
<thead>
<tr>
<th>Months</th>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>June</td>
<td>47</td>
</tr>
<tr>
<td>July</td>
<td>233</td>
</tr>
<tr>
<td>August</td>
<td>287</td>
</tr>
<tr>
<td>September</td>
<td>611</td>
</tr>
</tbody>
</table>

Next in July some more teams were introduced with the tool. Same process was followed in the following months reaching to greater number of request and reached 611 requests only in the month of September. This was great achievement of the project.
As discussed Capital One has many line of business like card, home loan, auto loan, retail, partnership, digital and commercial. Card refers to credit card. It has many credit cards for the customers here in US, Canada and UK. Unlike other financial services, Capital One began as “mono-line”, meaning the vast majority of its business was in credit card. It stands 4 in credit card lender in USA in the first quarter of 2015. It serves Visa and Master cards. Quick silver, Venture and Spark are the most common and popular credit card types. In retail banking it has capital one bank and 360 which is a merger of American ING Direct bank. Capital One Auto Financial Corporation is the parent company of Capital One Auto Finance Company, based in Plano, Texas. Many companies do partnership with Capital one to bring the credit card in their Brand name. HSBS- USA was merger brought the Capital One with much more companies doing partnering with it like, GMC, Best Buy, Kohl’s etc. Commercial handles and coordinate
different line of business. Digital is the middle tier of the organization. Most of the information technology related task are handled by the digital. The application development comes under Digital LOB.

Table 4.2: Request by Line of Business (LOB).

<table>
<thead>
<tr>
<th></th>
<th>Commercial</th>
<th>Home &amp; Auto Loan</th>
<th>Card</th>
<th>Bank</th>
<th>360</th>
<th>Digital</th>
<th>Partnership</th>
</tr>
</thead>
<tbody>
<tr>
<td>June</td>
<td>9</td>
<td>23</td>
<td>1</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>July</td>
<td>25</td>
<td>105</td>
<td>17</td>
<td>86</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>August</td>
<td>41</td>
<td>18</td>
<td>84</td>
<td>44</td>
<td>22</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>September</td>
<td>87</td>
<td>51</td>
<td>202</td>
<td>86</td>
<td>31</td>
<td>154</td>
<td></td>
</tr>
</tbody>
</table>

The graph below demonstrate the valuable information during the software life cycle. As we see for the first two months very less people used the tool and had very less request. Slowly other LOB’s started using the tool. By the end of the July three major LOB’s Credit Card, Retail and Digital started using it. This stage was a learning stage for the users. Many defects were identified and tool was not able to handle every business scenarios. Month of August was the nostalgic month for the development. There were lot of defects been raised and tool was not able to address users requirement. Once again critical defects were taken as high priority and again bring the users back to use the tool. During the month of September the request increased from 287 to 611 request. Which was a greatest encouragement for the team. September
release was the project dead line and we were able to successfully bring the fully matured tool.

Figure 4.3: Line-Chart of Request by Line of Business (LOB)
Request after tool deployment:

As per the timeline, September was the month of deployment. Almost all Line of Business (LOB) adapted the tool except partnership. After the massive defect fix during the month of August drastic outcome was seen. Everyone was getting used to with tool and became quick hit inside the enterprise. Credit card and digital lob request were the major.

Table 4.3: Request of September

<table>
<thead>
<tr>
<th>LOB</th>
<th>Commercial</th>
<th>Home &amp; Auto Loan</th>
<th>Card</th>
<th>Retail</th>
<th>Digital</th>
<th>Partnership</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>87</td>
<td>51</td>
<td>202</td>
<td>86</td>
<td>154</td>
<td>0</td>
</tr>
</tbody>
</table>

Getting more data refresh request indicates the popularity of the tool. Next target should be to train and include partnership Line of Business (LOB) to increase the users and make use of the data refresh tool.
Data Analysis

Breakeven analysis is carried to analyses the date. Breakeven analysis is used to determine when the business will able to cover all its expenses and begin to make a profit. It is important to identify the cost. To calculate breakeven point fixed and variable costs was identified.

The first thing before building tool was to find out what it would cost to setup a test account. It would take average of 10 minutes to 45 minutes creating an account and condition to make defined test account.
Figure 4.5: Sample Account Creation Page in Mainframe

For example user has to update each screens line by line and update. At least have to go 10 screens and have to update the values in green color like demographic information, balance, email etc. This process has to be done twice when code moves for integrated testing. To recondition the same account it may take another 15 to 20 minutes. This was huge loss for the company to do the redundant task. Thus the data refresh tool updated this process automatically and make use of same account number.
Table 4.4: Monthly Development Cost Calculation

<table>
<thead>
<tr>
<th>Job Title</th>
<th>No of individual</th>
<th>Cost per year</th>
<th>Monthly</th>
<th>Total Per Month</th>
<th>Percentage dedication</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers</td>
<td>2</td>
<td>110,000</td>
<td>9166.66667</td>
<td>18,333</td>
<td>40%</td>
<td>7333.3</td>
</tr>
<tr>
<td>Team Lead</td>
<td>1</td>
<td>110,000</td>
<td>9166.66667</td>
<td>9,167</td>
<td>30%</td>
<td>2750</td>
</tr>
<tr>
<td>BSA</td>
<td>1</td>
<td>90000</td>
<td>7500</td>
<td>7,500</td>
<td>30%</td>
<td>2250</td>
</tr>
<tr>
<td>QA</td>
<td>1</td>
<td>80000</td>
<td>6666.66667</td>
<td>6,667</td>
<td>40%</td>
<td>2666.7</td>
</tr>
<tr>
<td>Scrum Master</td>
<td>1</td>
<td>95,000</td>
<td>7916.66667</td>
<td>7,917</td>
<td>10%</td>
<td>791.67</td>
</tr>
<tr>
<td>Manager</td>
<td>1</td>
<td>110,000</td>
<td>9166.66667</td>
<td>9,167</td>
<td>10%</td>
<td>916.67</td>
</tr>
<tr>
<td>Grand Total</td>
<td></td>
<td></td>
<td></td>
<td>58750</td>
<td>1.6</td>
<td>16708</td>
</tr>
</tbody>
</table>

From the business and extensive analysis enterprise brought the per request cost = $29

Team consist of 2 Developers, 1 Team Lead, 1 Business Analysis, 1 Quality Analysis, 1 Scrum Master and 1 Manager. Since multiple people were doing multiple projects. Their dedication was not 100%. It is clearly shown in the graph under percentage dedication of each member.

Monthly Cost = \( \sum \text{No. of individuals} \times \text{Cost per year} \times \text{Percentage dedication} \)

Developer cost = \( \text{No. of developers} \times \text{Cost per year} \times \text{Percentage dedication} \)
$7333$ per month

Team lead cost = No. of team lead $\times$ Cost per year $\times$ Percentage dedication

\[
= 1 \times 110 \times 30\% \\
= 12 \\
= 2750 \text{ per month}
\]

BSA cost = No. of BSAs $\times$ Cost per year $\times$ Percentage dedication

\[
= 1 \times 90 \times 30\% \\
= 12 \\
= 2250 \text{ per month}
\]

QA cost = No. of QAs $\times$ Cost per year $\times$ Percentage dedication

\[
= 1 \times 80 \times 40\% \\
= 12 \\
= 2666 \text{ per month}
\]

Scrum Master Cost = No. of Scrum Master $\times$ Cost per year $\times$ % dedication

\[
= 1 \times 95 \times 10\% \\
= 12 \\
= 2666 \text{ per month}
\]
Manager cost = No. of Managers x Cost per year x Percentage dedication

\[
\text{12} = 1 \times 110 \times 10\%
\]

\[
\text{12} = $916 \text{ per month}
\]

Total Cost = $16708 per month

Common Hypothesis:

1. Infrastructure costs like building, servers and computers were not considered.
2. All cost were fixed for a year.
3. Each request cost are evaluated equally.
4. Salary of individual position were pulled from current market study form Glassdoor.
5. Support after completion was considered 5% of the total effort.
6. Development effort to the project was around 35% on average by the team members because multiple projects were developed same time.
Table 4.5: Breakeven Analysis Table

<table>
<thead>
<tr>
<th>Month</th>
<th>Request</th>
<th>Profit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>February</td>
<td>0</td>
<td>0</td>
<td>16708.00</td>
</tr>
<tr>
<td>March</td>
<td>0</td>
<td>0</td>
<td>16708.00</td>
</tr>
<tr>
<td>June</td>
<td>47</td>
<td>1363</td>
<td>16708.00</td>
</tr>
<tr>
<td>July</td>
<td>233</td>
<td>6757</td>
<td>16708.00</td>
</tr>
<tr>
<td>August</td>
<td>287</td>
<td>8323</td>
<td>16708.00</td>
</tr>
<tr>
<td>September</td>
<td>611</td>
<td>17719</td>
<td>16708.00</td>
</tr>
<tr>
<td>October</td>
<td>600</td>
<td>17400</td>
<td>2937.00</td>
</tr>
<tr>
<td>November</td>
<td>600</td>
<td>17400</td>
<td>2937.00</td>
</tr>
<tr>
<td>December</td>
<td>600</td>
<td>17400</td>
<td>2937.00</td>
</tr>
<tr>
<td>January</td>
<td>600</td>
<td>17400</td>
<td>2937.00</td>
</tr>
</tbody>
</table>

Development was carried in between February and September with only less than 50% dedication to this project as other projects were also going side by side. After September the cost incurred was 5% only. Average monthly maintenance cost was $2,937. Since $ 29 was the amount saved of each request. So the greater the number of request greater would be saving.
Summary

This chapter presented the actual data collected during the project in tables. Also it explained the analysis of those data with different bar charts, pie chart and also showed the compiled view of the cost and return. The next chapter will talk about the results acquired from the project, the conclusion, and the recommendations.
Chapter V: Results, Conclusion, and Recommendations

Introduction

This chapter will explain about the results obtained by doing the project, the conclusion accomplished from the results and some recommendations related to this project.

Results

The results are explained on the basis of the project questions presented in the beginning of this project.

1. What will be the likely cost in developing the data refresh tool?

The cost of the whole project comprised of the payments made to the external IT consultants, 2 developers and 1 Quality Analysts, 1 Business Analysis (BA), 1 scrum master, 1 manager, 1 team lead. On average the cost paid to the developers was approximately

$7333 per month to the Developers.
$2750 per month to Team Lead
2250 per month to Business Analyst
2666 per month to Quality Analyst
$791 per month to Scrum Master
$916 per month to Manager

Total Monthly Cost=$ 16708 per month
Total Cost of the Project= Total Monthly Cost x Number of months

= $16708 x 8
= $1, 33,664 per year

Total Maintenance Cost = Total Monthly Cost x 12 (per year)

= $2937 x 12

=$35,244 per year

2. What is the cost savings after developing the tool?

After the full development of the tool, only incurring cost for maintenance is $16,708 per month. For an average of 1000 request per month the cost without using tool would be $ 29,000. Total save will be $ 12,292 per month.

3. What is the customer feedback on the tool?

The tool have overwhelming response. It was much user friendly and saving a lot of time and effort.

Conclusion

This project was based on creating a data refresh tool and API that stored the data and metadata of the account for testing, which could be reverted back to original condition for each testing cycle. By using the tool the user can use same set of test accounts for different testing. The company payed around $1, 33,664 to develop the entire application. The company will pay around $35,244 per year for the support and maintenance of the environment. Every time the tester creates the test account with necessary status, flags, amount etc. for system testing, the same process had to be repeated for next phase of testing in User Acceptance Test Environment(UATE). Creating a test account required 15 minutes to 5 days depending upon the scenario and batch jobs. Redundant effort to create same set of test accounts was huge loss for the company.
After integrate analysis company came to the conclusion that average cost of creation of an account and conditioning sum up to $29 per account or request. Approximately 1000 such request per month would cost $29,000. Apart from this cost the tester had to have knowledge of all the backend system, which requires training and practice. Each time new resources comes he/she has to learn all the backend systems but by using this application on click of a button and with limited system knowledge a tester can create and condition the account and reuse it multiple times.

Next part of the tool is usable Application Programing Interface (API) which can be consumed by different Line of Business (LOB) applications meeting their requirements. So just by paying maintenance cost of around $2937, company was able to save $29,000 per month for around 1000 request. Number of requests may increase in the future but the maintenance cost would be almost same. Its almost 90% saving of cost without considering the initial development cost.

**Recommendations**

Following are the recommendations after completing the Data Refresh tool.

1. All the Line of Business (LOB) should use the data refresh to store their test data.
2. Company should organize different training and brown bag sessions and demos about tool and its features.
3. There should be some kind of continuous monitoring method or some resource available all the time to check and report on the status of the tool.
References


