Agile Method to Improve Delivery of Large-Scale Software Projects

Presented to

The Faculty of the Department of

General Engineering

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Engineering

By

Mary Wu

May 2011
APPROVED FOR THE DEPARTMENT OF GENERAL ENGINEERING

Dr. Mithal Albassam

Academic Advisor / Faculty Reader

Department of Industrial Systems Engineering

Signature: _______________________ Date: _________________

Michael Lee

Industrial Advisor

Sunrise Telecom Inc. / Department of Marketing Communications

Signature: _______________________ Date: _________________

Dr. Leonard Wesley

Assistant Professor

Department of Computer Engineering

Signature: _______________________ Date: _________________
# Table of Contents

1. **Abstract** ........................................................................................................................................... 8

2. **Project Overview** .............................................................................................................................. 9
   2.1 Scope .................................................................................................................................................. 9
   2.2 Description ......................................................................................................................................... 9
   2.3 Test Plan ............................................................................................................................................ 9

3. **Literature Survey** ............................................................................................................................... 10
   3.1 Comparison of Various Development Process .............................................................................. 12
   3.2 Agile Process ..................................................................................................................................... 14
   3.3 Integration of Agile Methods to Large-Scale Projects ................................................................. 15
   3.4 Agile Reporting Methods .............................................................................................................. 16
   3.5 Risk Analysis ................................................................................................................................. 17

4. **Project Design** .................................................................................................................................. 18
   4.1 Case Study A Implementation ....................................................................................................... 18
   4.2 Case Study B Implementation ....................................................................................................... 23
   4.2.1 Estimate of Potential Cost Savings to Overall Development ............................................. 26

5. **Project Implementation** .................................................................................................................... 27

6. **Performance and Analysis** ................................................................................................................ 33
   6.1 Qualitative Analysis ....................................................................................................................... 34
   6.2 Quantitative Analysis .................................................................................................................... 35
   6.2.1 Estimate of Potential Cost Savings to Software Development ............................................. 35
   6.3 S-Curve Forecasting ...................................................................................................................... 39

7. **Economic Justification** ...................................................................................................................... 40
   7.1 Executive Summary ...................................................................................................................... 40
   7.1.1 Problem Statement ................................................................................................................... 40
   7.1.2 Solution & Value Proposition ................................................................................................. 40
   7.1.3 Market Size ............................................................................................................................. 40
   7.1.4 Competitors ............................................................................................................................ 41
7.1.5 Customer ............................................................... 41
7.1.6 Cost ................................................................. 42
7.1.7 Price Point ......................................................... 44
7.1.8 SWOT Analysis .................................................... 44
7.1.9 Investment Capital Requirement ............................. 45
7.1.10 Personnel ......................................................... 47
7.1.11 Business and Revenue Model ............................... 48
7.1.12 Strategic Alliances/Partners ................................. 49
7.1.13 Profit and Loss Statement .................................... 51
7.1.14 Exit Strategy ...................................................... 52
  7.2 Return on Investment (ROI) .................................... 53
  7.3 Norden-Rayleigh Analysis ..................................... 54
8 GANTT CHART ............................................................. 57
8.1.1 Phase I .............................................................. 57
8.1.2 Phase II .............................................................. 58
9 Conclusion .................................................................. 58
10 Appendix .................................................................. 59
  10.1 Appendix A .......................................................... 59
  10.2 Appendix B – Glossary of Terms ......................... 60
11 References .............................................................. 61
LIST OF TABLES

Table 1- Comparison of Factors among the alternatives affecting Choice of development........ 35
Table 2- Estimate for a large-scale Agile Project Duration for Case Study A......................... 38
Table 3- Fixed Costs ............................................................................................................. 42
Table 4- Variable Costs ....................................................................................................... 42
Table 3- Summary of Cost vs. Sales ..................................................................................... 43
Table 8- SWOT Analysis ....................................................................................................... 44
Table 5- Summary of Cost vs. Sales ..................................................................................... 46
Table 5- Consolidated Statements of Operations for Case Study A....................................... 51
Table 7- ROI Calculation ...................................................................................................... 53

LIST OF FIGURES

Figure 1- The Planning spectrum ......................................................................................... 11
Figure 2- Project Success Rate ............................................................................................ 12
Figure 3- Flow chart of the agile process ............................................................................. 15
Figure 5- Estimating development effort with sub-iterations to estimate work hours .......... 18
Figure 6- Prototype implementation of the Agile process .................................................... 19
Figure 7- Hardware and software uses the "layer" approach (Ngpanwei, 2010) .................. 21
Figure 8- Work-in-progress work-flow document for case study A ................................... 23
Figure 16- US National Averages of Software Engineer III ................................................... 26
Figure 9- Addition of Features in the Iteration ..................................................................... 29
Figure 10- Agile Project Burn Down Chart (Seider & Vescuso, 2009) ............................... 30
Figure 11- Work Estimate using Agilefant .......................................................................... 31
Figure 12- Defining Stories in Agilefant .............................................................................. 32
Figure 13- Burn down chart for the case study ................................................................. 33
Figure 14- Agile change management ............................................................................... 36
Figure 15- The AMDD lifecycle ..................................................................................... 37
Figure 19- S-Curve for Product Lifecycle .................................................................... 39
Figure 18- Break Even Analysis ..................................................................................... 46
Figure 4- Leading the agile development ...................................................................... 48
Figure 17- Agile development to the organization ......................................................... 50
Figure 20- Norden-Rayleigh Models by Northrop Grumman ...................................... 55
1 Abstract

This master project describes the use of agile management techniques at an organization to plan, organize, and manage resources in software projects to bring a project's objective to full focus. It attempts to prove that implementing agile software development in engineering best practices ensures efficient delivery of high-quality software. It also shows that agile methods create the best result when used with operations management on large scale projects.

Researches indicated that agile methods are contrasted to the "plan-driven" method, but rather, it is related to adaptation. When agile methods are used instead of other iterative development methods or Waterfall Methodology, it proves with measurable results that the added flexibility creates greater efficiency in the short term.
2 Project Overview

2.1 Scope

Agile development process can be used on projects with different scale and complexity by having iterations of incremental code release, product verification, and bug fixes. This process has been used in many organizations to increase efficiency of development code release to product verification. I attempt to extend this concept to large-scale software projects that involves more complex FPGA and CPLD firmware code releases for both hardware and software integration. Much of the focus is on the internal side, effect of customer feedbacks and results will be for further study.

2.2 Description

The idea of using less procedures and brainstorming creative ways to implement an instruction with the shortest amount of time is the idea behind agile process. My master project is focused on using agile development to enhance the delivery and efficiency of large-scale software projects. It involves continuous planning, testing, integration, and delivering to customers for evaluations.

2.3 Test Plan

My hypothesis will be tested by simulating a number of software projects called RealGate from Sunrise Telecom Inc. This development has started in the Beijing, China location and will closely examine the benefits, development cycles, test cycles, and roadblocks encountered. Since Agile development is to use the least amount of documentation, most of the details will be in types of verbal communication and involves daily short chat sessions which are namely the "Scrum"
meetings. All the development iterations and process will be tracked by the software application called Agilefant. Next, I will examine the projects from start to finish and compare to the development time, cost, and performance of the previous software cycle that did not utilize Agile. I will then brainstorm ways to come up with ideas to "slice" FPGA and CPLD development that can turn its development into iterations and again simulate it with Agilefant. During this process, I plan to incorporate some of the best practices from a number of companies with different industry focus, such as NEC and JobScience that can possibly eliminate wasteful iterations.

3 Literature Survey

From "The Agile Manifesto", it is clear that responding to unpredictable events is more powerful than planning ahead for disasters. Most of other articles, such as "Agile and iterative development: a manager's guide" and "Agile software development methods", describe agile method as a marginalized process that is random, overlapping, and occurs in unique form with negotiated goals. This process allows developers to implement late changes as compared to strictly following with scheduled plans and stated specifications. Boehm (2002) and many sources relate Agile methods to other models as follows:
Most authors introduced the four common agile methods: Scrum, Extreme Programming (XP), and the Adaptive Software Development. Agile software processes is characterized by its ability to develop in module form, few weeks time-frame for delivery with short verifications and corrections cycle, and incremental approach. As such, this process requires careful collaboration and communication in cross-functional teams. This process can be translated to co-located development team. As this requires smaller steps, simple solutions that anticipate for continuous testing, resources need to increase and project will need to be divided into sub-projects for larger scale projects. Since less documentation is more favorable in the agile process, resources will solely focus on development and testing only. Current problem is the ability to balance the execution cycle and the framework to be used effectively across large scale projects.
3.1 Comparison of Various Development Process

![Perceived Project Success Rates](image)

**Figure 2- Project Success Rate**

The traditional method as shown in figure 2 is the waterfall model that many companies follow decade ago. (Ambler, 2010). Waterfall is a sequential design process that follows conception of design requirements, initiation, analysis, design phase, actual implementation, test/verification/production, and maintenance. At each step of the way, cost and risks of development increase as problems only uncover as the phase surfaces. Ad-Hoc is a test strategy that does not require planning or documentation; as such the iteration is low and usually does not anticipate repeated test runs. It is the least structured, but effective way for quick error location. The deficiency is that it is not a good method for regression testing because test steps needed to reproduce a bug are not documented, and users will waste time
and cost to regenerate those test steps. Traditional method has similar outcome as the Ad-Hoc method, either waste time and effort to reproduce a problem or problems cannot be uncovered until later stage. In the last decade, newer development methods like iterative and agile evolved and advocated the change in response to the weaknesses introduced by the waterfall model in the traditional method. (Wikipedia) The idea of iterative and incremental approach is to develop a small piece of code and perform development test as a joint process in repeated cycles. The advantage is to allow bringing experience from previous cycle to the next using a modular and straightforward approach. Its success rate is very similar to the agile process. The main difference that distinguishes agile from iterative approach is the inclusion of cross-functional teams to plan, test, develop, and release to customers in an iterative manner. Detailed advantages and strengths/weaknesses will be discussed in later sections of this project report. Agile methods have proved its benefits in flexibility, customers' satisfaction, and incremental releases. This technique allows integration to large-scale projects with development in different locations. Careful monitoring and coordination between teams in different locations is necessary for timely and accurate software release. Adaptability works best for smaller teams, and each project should also retain a small team structure for better coordination between sub-projects. This practice can be measured by AIM and test trials. Alongside, time, cost, and performance of each small workgroups for each sub-project easily resolve some of the issues project managers face today. Moreover, Agile scoring reflects the difference in project result that is executed with predicted/planned methods versus the adaptive approach. (Ambler, 2011) This will greatly prove viable results that can be adopted by any industry in any project scale given a small to medium-sized team. The results are displayed
with completeness, accuracy, and timeliness that stir public acclaim and recognition as an innovator. Overall, agile provided the most benefits out of all four models discussed.

3.2 Agile Process

Highsmith (2002) commented that most of all methods consist of similar phases in the life cycle. Extreme Programming (XP) has exploration, planning, iterations-to-release, production, maintenance, and death. XP proceeds by using existing developed strategies. Customers direct all of the business decisions such as timing and scope, while programmers structure the technical complexities including effort estimation. For Scrum, it is an empirical approach with application of control theory to systems development. Its management focuses on identifying defects during development. The phases include pre-game, development, and post-game. Scrum Master monitors the project team and coordinates with customers. Some important tasks for the project team include product backlog, effort estimation, and sprint. The team then produces an incremental executable product in about 30 days. Both the XP and Scrum works well in small to medium sized teams. Recent studies show that XP and Scrum can be integrated together. Scrum provides the project management framework that can be adopted by XP for execution of software development. According to "Agile Software Development", integration of XP and Scrum provide a good foundation for extending to larger scale projects.
3.3 Integration of Agile Methods to Large-Scale Projects

Many ways can support agile methods to aid in executing larger scale projects. New whiteboards can easily track "parking lot" ideas, and it's graphically effective to be used by team. They have saving feature build-in to capture ideas written on the whiteboard and transferring them to workstations. This greatly reduces and even replaces the formal planning cycle or the need for design documents. Large-scale projects can be sub-divided into smaller projects. Customers' directions are quickly re-grouped and categorized into each of the smaller projects. Scrum provides the central location for the product backlog list with priorities and effort estimation. This can be shared by multiple projects and be used by global locations. During development cycle, each project takes the advantage of pair programming as suggested by XP. Developers continuously analyze, design, plan for test, and execute testing. All codes are repeatedly stored into collective code database, such as SourceSafe or Open Source.
each piece of code is tested, and reviewers will provide feedback along with validation result for the cycle. While each location establishes a check-in mechanism for all of their projects, there will also be a centralized location for storing multiple projects. Each project will set small releases that can be integrated to other sub-projects. At the end of each month, Scrum will enter the post game phase final integration, system test, and actual incremental release to customer.

### 3.4 Agile Reporting Methods

Report documentation should also apply the Agile concept as it is aimed to focus on ensuring feature requirements are met. (Nielsen, 2008) Therefore, tedious project plans and other traditional methods can be avoided. Scrum produces the Product Backlog, Sprint Backlog, Changes Report, and Burn down Report at the end of each iteration. Product backlog contains all the brainstorming and product requirement in line item feature description with a priority associated, which can leverage tools like Wiki or Google Docs for easy sharing and modification. One way is to utilize Microsoft Project to list all the decomposed features, or other tools such as ScrumWorks, VersionOne, or IDailyScrum for mobile devices. (Freedman, 2010) The Sprint Backlog contains the specific features that would be released in each iteration. The Change Report gives flexibility to the team to modify feature requirements with tracking. For easy data and info manipulation, spreadsheet can be used to track and outline the changes, although this will not be favorable for engineering change order when hardware is involved with a revision control. The Burn down Chart records work with the number of story points as time progresses. This gives you an overview of how many features have been released and how many more yet to complete. An increasing slope indicates that the project is well underway.
and engineers understand and develop the requirement. A decreasing slope can show that there is a slow-down in feature development due to roadblocks or change in project priorities.

3.5 Risk Analysis

I will encounter roadblocks in this case study and research. There will be times when the daily meetings fail to discuss work completed along with work to be done. At times, the meetings will not happen if participants are not available. Meetings missed in this process will be critical to the project finish since tasks' progresses and issues from not addressed daily to the team can affect the hours spent for the work, both completed work and to-be-completed work. The daily meetings will be held in the US location and participants from overseas will need to be present as well. Difference in time zone can make the meeting less productive. Discussion from the US will be end of the day's work while Asia locations will be the start of the day's work. Also, database will need to be sync daily in both locations. Any database mismatch or no-sync issues can delay development. Agile development process is quite new to the culture of my current employer. It takes time for the project team to achieve collective understanding of the process and apply to its daily tasks. Some will feel the pressure of having to complete the tasks in a defined time. Others who are not used to daily meetings might feel less flexible and uncomfortable.
4 Project Design

4.1 Case Study A Implementation

- Identify the high-level scope
- Identify initial "requirements stack"
- Identify an architectural vision

- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration

- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later

- Develop working software via a test-first approach
- Details captured in the form of executable specifications

Figure 4- Estimating development effort with sub-iterations to estimate work hours
Figure 5- Prototype implementation of the Agile process

Initial requirement needs to be addressed by product managers in conjunction with the architectural requirement during the envisioning. During the iteration 1 to n, it includes modeling, storming, and test driven development. Figure 5 displays the sequence of events that are planned in each of the test and development iteration 1 to n for case study A (see Appendix A), aside from initial requirement and architectural envisioning in iteration 0. We will implement this prototype process in a pipelining fashion as shown in figure 6.

Following are the required deliverables that will be included each iteration of the test and development for the prototype implementation:

- Deliverables before test execution: Test plan (risk/assessment), test design document, test cases
- Deliverables during test execution: bug reports, results check-in to databases, bug scrub reviews

- Deliverables after test execution: test reports, test review acceptance, test coverage analysis, ECO release

- Required documents: To do List, SW release notes

- Required software: virtual machine

Following are the required deliverables that will be included in each iteration of the test development for the prototype implementation of a large-scale software projects:

- Deliverables before test execution: Test plan (risk/assessment), test design document, compliance matrix, test cases, PRD PV

- Deliverables during test execution: bug reports, results check-in to databases, bug scrub reviews

- Deliverables after test execution: test reports, test review acceptance, compliance matrix update, post mortem, test coverage analysis, ECO release, test procedures for manufacturing

- Required documents: MRD, PRD, SW/HW release notes, calibration certificates

- Required hardware: ~5-6 test units shared between SW, HW, compliance

- Required software: FPGA, CPLD, test application, kernel, RFS, device drivers, support equipment software
Hardware and software uses the "layer" approach (Ngpanwei, 2010)

Hardware and software components will be defined in "layers" and are color-coded.

(Ngpanwei, 2010) As shown in Figure 4.3, the colors are mapped to show dependency between the layers. The number on each colored block defines the "story" or feature requested. During the architecture phase, the overlapping hardware and software components will be addressed as part of iteration 0. During test and development, each iteration consists of the coded color block that can be detailed into the "story".
Flow chart as shown in Figure 7 describes the process flow between hardware, software, and test development. Traditional method requires each group of processes to complete before interacting with a different group. After the major research and architecture definition, the hardware (in yellow) and firmware (in pink) will work in parallel. Once FPGA pre-analyze is complete, it will immediately feed to schematic design for development. In traditional Waterfall process, completed work will not be reviewed by both teams until after board bring up and constraint and timing test results.

4.2 Case Study B Implementation

Below is the RTL development as suggested by case study B. This can be incorporated into RTL coding steps necessary to be integrated to software and board-level test development. Each step will be a task point that can be inserted into each iteration of RTL coding design.

RTL design phases

1. Define Functionalities

2. Define Architecture, and Partitioning between Hardware and Software

3. Estimate overall gate count, CPU performance requirement and code size estimation

4. Define micro architecture in hardware, block definition, clock definition, and reset definition
5. Code RTL Design

6. Simulation and Verification

7. RTL synthesis for FPGA

8. Place and Route

9. Gate level simulation

**RTL Verification phases**

1. Define functionalities to be tested

2. Define typical usage model

3. Define test matrix

4. Code test bench

5. Block level verification

6. System level verification

7. Regression test

**Silicon Validation phases**

1. Define functionalities to be tested
2. Define equipment needed for testing

3. Define automation needed (watchdog timer, power cycling, keyboard automation)

4. Define how to peek into the chip's internal state for debugging

5. Define system level variation (clock source, component differences, reset timing, voltage variation, and temperature variation)

6. Bring up

7. Run at lower speed test

8. Run at speed test

9. Overnight burn-in test

Software design and test phases

1. Define Functionalities and what to be tested

2. Define Architecture, micro architecture, and block definition

3. Define performance requirement, and identify critical block for performance requirement

4. Define CPU interrupt

5. Define logging capability and how to support test automation
6. Define how to upgrade software in field

7. Code

8. Verified with memory leak identifying tool

9. Test with FPGA/silicon

4.2.1 Estimate of Potential Cost Savings to Overall Development

![Graph showing median salary]

**Figure 8- US National Averages of Software Engineer III**

If we are to account for US National Average as Figure 3.3.2.1 shows, software engineer level III median salary falls at $87,221, which converts to $45/hr. This figure also represents, in our case study, the average salary of 1 US engineer and 1 overseas engineer. As such, we leverage one extra headcount for the overall development. With one extra headcount, we speed up development time by two.
5 Project Implementation

Problem to face: A lot of time it is hard to determine what has been completed and what is left to complete in the allotted amount of time. MS Project or other project management tools that has the Gantt chart describing the project in a horizontal view stating the tasks completion and to-be-completed percentage graph. It does not give an overview of the overall amount of work completed in a certain period or iteration. The completed tasks can appear at different places on the project Gantt chart and are not grouped together to visually represent completed and to-be-completed tasks.

What is the tool? A burn-down chart then provides this type of solution for project and product managers for an immediate understanding of the completed work. It is not a milestone base of project tracking system.

How? First you determine the user stories that would be developed over the course of the defined project iteration. In agile, user stories are use cases that have been broken down. Each of them is stated as a statement with a subject and the intended goal. The stories are generally defined by the product managers and will be prioritized depending on customers' urgency. Then, each story will be weighted according to the complexity as reviewed according to the scope, resources, and knowledge available by the engineering team. The stories can be prioritized as high with relatively low in complexity. In that case, you would expect the task to be "burned" fairly quickly. The chart describes the stories on the vertical axis and time frames on the horizontal axis. Time frames are described as iterations of intended product delivery. As the tasks or stories are completed, the graph will reach toward the horizontal axis toward 0.
The dotted line as shown on figure 4 represents the "velocity" of the work estimated. This line shows the likelihood of the work that can be delivered by the team. The completed work will be in hours and the amount of work hours completed will then be plotted for each story starting from story 1. As the work gets completed quicker, the slope becomes more negative.

The calculation for the slope of the line is just as a linear math model: $y=mx+b$. Each iteration should have a slope that describes the change in the amount of work hours over time elapsed. After two iterations, it will be deterministic to draw the slope to come up with the projected slope for the next few iterations. There are many factors that can change the projected slope. Factors such as available resources and funding will affect the pace of the work to be completed. Both resources and funding can change during the course of the project.

Alongside, the weighted complexity can affect the slope tremendously. If the group of stories have high cumulative weights, it would be likely for the slope to be less negative with more time spent to complete the tasks with higher complexity. The team's goal will aim to complete the tasks closest to the velocity slope.

**Benefit vs. disadvantage:** The benefit of this chart is to display how development moves along in each iteration. It promotes discussion of possible solutions to speed up or in cases slow down development. One reason for the need to slow down development in some iterations is to utilize resources in other projects or if the team decides to spend more time researching rather than constant test and development. As we would like to move away from the ideal velocity, we can also utilize the excess pace by adding more features into the task group. There are disadvantages to using the burn-down chart. The chart only shows the tasks vs. time which does not include risks or issues that can slow down or speed up the project. During an increase
in scope at a particular iteration, you can start another project line indicating that a new requirement has been added and the project progress with then shift from that point.

![Figure 9- Addition of Features in the Iteration](image)

**When?** Each day will have a 15-minute Scrum meeting to discuss about the current task completed and tasks-to-be completed. The amount of work hours completed will then be updated onto the tracking software which will be reflected in the burn-down chart. This chart can be posted on a projection screen or shared in a location for the team.
Figure 10- Agile Project Burn Down Chart (Seider & Vescuso, 2009)

According to Figure 4, the top horizontal red dotted line shows the number of features likely to release after 10 weeks, where most of the project lines will intercept with that horizontal line. The uncertainty section has only the ideal and one projected line; those features are more likely to be developed if each slope is more negative. Lastly, the unlikely features after 10 weeks are almost excluded from the projected lines. The product managers can use this to gauge the amount of stories to be completed and be able to shift priorities depending on customer change, market influence, or the general outcome of the development. One way to predict how many sprints a project will take is to draw a trend line through the bars and extend the baseline. Also, the number of sprints can also be predicted by drawing a trend line through the changes occurring at the bottom of the bars to include the addition of scope.
Figure 11- Work Estimate using Agilefant

Case study A uses Agilefant to keep track of the agile development test runs. The 4 iterations as suggested by Figure 4.2 are incorporated into this system. There are total of 86 tasks in these four iterations with a total of 692 estimated hours + 12 minutes and 15 items without hours estimated. Each iteration is a 5-week span with a total of 20-weeks for the project. Figure 6.2 will examine iteration 1 that has 270 hours 12 minutes in the 5-week cycle.
Within iteration #1, the stories are defined in a few words that relates to the group of tasks. The actual tasks are located under each story. They are stated with further detail on the description, creator, original estimate, elapsed time, state of the task, priority, and actions. We rely on the executor to document under each task with bug numbers from the internal Bugzilla bug tracking system that tie in to the task. This provides a reasonable justification of the estimated task hours and perhaps relate to risks, issues, and contingency plans that will can potentially face for the task.
6 Performance and Analysis

Figure 13- Burn down chart for the case study

As shown in figure 6.3, the blue line displays the original estimate that translates to velocity of the work tasks in weekly increments. The red dots represent the actual hours of the tasks completed or to-be-completed over time. The actual overall task hours are higher than the original estimate. At certain points, the slope is 0 because more than 1 week was needed to complete the task. This is due to unforeseen hours needed for completion and or resource allocation from one project to another project; it would be broken down into small tasks, otherwise. On the week of 4/18 to 4/25, the task points rise above the 12.5 point to indicate the additional hours needed for planning/scoping before the task hours will "burn down".
Within the same day, it can also complete the task hours and result in a vertical slope. Overall, this has shown a success due to the following observation:

1. Although the actual overall task point is above the initial point in the velocity graph, it was able to burn down the hours in the 5-week cycle.

2. Most of the task points lie below the velocity graph to demonstrate that the progress is better than the original estimate.

3. Total of 692 hours of work hours were estimated to-be-completed in the 20-week long project. This is in line with our average of 5.3 months for project completion as estimated in section 7.3.1.

6.1 Qualitative Analysis
Cost incurred the greatest in the early stages of development. During preliminary design phase, software and firmware architecture will gradually prove its way to success by delivering prove-of-concepts coding. As more feedbacks are received early on, tremendous cost will be saved from mistakes not made. (Anderson & P.E. & CMC, 2010) Moreover, both the software and firmware development teams can easily adapt to this process with minor changes to development cost. The key is to turn the traditional methods into more interactive approach with the same end-product. The team should not expect any overhead increase.
6.2 Quantitative Analysis

6.2.1 Estimate of Potential Cost Savings to Software Development

Table 1- Comparison of Factors among the alternatives affecting Choice of development

<table>
<thead>
<tr>
<th>Factors</th>
<th>Alternatives</th>
<th>Time-to-Market (3)</th>
<th>Quality (4)</th>
<th>Sources accessibility (1)</th>
<th>Employee Engagement (2)</th>
<th>Customer Satisfaction (5)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>10</td>
<td></td>
<td>135</td>
</tr>
<tr>
<td>Current</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td></td>
<td>87</td>
</tr>
<tr>
<td>Combination</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>Third-party vendors</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td></td>
<td>81</td>
</tr>
</tbody>
</table>

Factors on the scale of 1-5 (highest rating), Alternatives on the scale from 0-10 (highest rating)

From the above table, factors are those that are important to Sunrise Telecom (see Appendix A for case study A). These factors are also given a rating which corresponds to the relative importance among each other. The alternatives for development are listed on the left column, and each is rated against the factors. The factors and ratings are information gathered by interviewing the software manager from development team and the product manager from marketing team. Since Agile development received the highest score in this survey, it’s worth
taking a detailed study on how the cost can be affected. I will compare the development cost of implementing Agile across large-scale projects and the current development process.

![Figure 14- Agile change management](image)

Developers will give the work estimate for each item base on this structure. This work estimate translates into cost values that can be compared to the salary given to the employees. We will then justify the ROI.
Figure 15 - The AMDD lifecycle

Iteration 0 is the estimated time from initial requirement envisioning to initial architectural envisioning. On average, it takes 20 days to complete, review, and accepted by the project team. The rest of the iterations are estimated per project basis and can vary from project's scale and complexity. Each iteration is a development cycle based on iteration modeling (in hours) by planning the work, model storming (in minutes) to review what needs to be addressed into the requirement, and test driven development (TDD) (in hours) by using test-first approach while specification details are added. On average, it takes 8 hours to model the feature, which will be iteration 1. Then, it takes 30 minutes to review with the manager. The TDD will be roughly 24 hours (or 3 days). This is summarized in the table below:
As seen from Table 3.3.2, average hours needed is 847.5 for a large-scale project executed with Agile, which converts to 5.3 months to complete. Using the current method of development, software projects with firmware integration takes on average 9 months to complete the full development cycle. This constitutes to a saving of 3.7 months per project. We estimate that each US engineer costs $75/hr and engineers in China costs $17/hr for development costs. Since each project is comprised of 2 software engineers from the US and 1 firmware engineer from China, it renders a saving of \((2 \times 75 + 1 \times 17) \times (3.7 \text{ months} \times 160 \text{ hrs/month}) = 98,864\)
per project. With about 6 large-scale projects to launch in a year, it totals a saving of $600,000
in software development alone.

### 6.3 S-Curve Forecasting

We can also perform an S-Curve forecasting that will apply to product development as follows:

**Figure 16- S-Curve for Product Lifecycle**

The three phases will be followed in the product development from initial design to product
launch. Agile development will undergo prototype in phase II and production run in phase III. This will be time frame when key customers and major customers will be indentified and maintained. Therefore, careful execution and control of the test and development are crucial to the success of product launches and customer satisfaction.
7 Economic Justification

7.1 Executive Summary

7.1.1 Problem Statement

Agile development in this study proved to increase efficiency and quality in software development that extended to firmware development. Moreover, it efficiently reduces cost in test and development in the product lifecycle.

7.1.2 Solution & Value Proposition

Many large applications and projects are now made up of sub-components that use Agile to strengthen productivity and interaction between development teams. Customers will face smaller lead time for our products for evaluation and trial, and their constant feedbacks from these incremental releases become collaboration to our quality standards. This in turn boosts the customers’ ability to test and troubleshoot the network with assurance due to our timely releases.

7.1.3 Market Size

Potential customers are spending 30-40% lower for a comparable test solution for network service activation and troubleshooting, according to the field survey by Sunrise Telecom in April 2010. On average, 1GE test sets, such as JDSU SmartClass 10M-1G Network Testing Device RJ-45 Network, costs the customer $2500 for a new test set. Sunrise Telecom offers a similar solution at almost half the costs with twice as many features compared to JDSU. Agile development provides more rapid test and development that are able to meet more customers' requirement.
### 7.1.4 Competitors

<table>
<thead>
<tr>
<th></th>
<th>Annual Revenue</th>
<th>Geographic Location</th>
<th>Customer Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSU</td>
<td>$1,363.9 million</td>
<td>Americas, Europe, Middle East, Asia Pacific</td>
<td>the largest 100 telecommunications and cable test companies, along with 2000+ telecom customers</td>
</tr>
<tr>
<td>EXFO</td>
<td>$202.8 million</td>
<td>Americas, Europe, Middle East, Asia Pacific</td>
<td>2000 telecom customers</td>
</tr>
<tr>
<td>VeEX</td>
<td>$20 million</td>
<td>Americas, Asia Pacific</td>
<td>100 telecom customers</td>
</tr>
</tbody>
</table>

Sunrise Telecom's annual revenue and customer base lie just between EXFO and VeEX. Consistent products and features comparisons are for further research to characterize the main driving forces and market acquisitions by the competitors.

### 7.1.5 Customer

The Agile development on large scale projects will deliver to both internal and external customers. Internal customers include VP of Engineering, executive management, CTO, and CFO. The solution will roll out from engineering to cross-functional teams, and ultimately to utilize Agile development in a corporate sense. External customers include all current telecommunication customer base acquired by Sunrise Telecom. We aim to focus on customers specifically for the RealGate projects as our initial workgroup.
7.1.6 Cost

A number of fixed costs were incurred for the development of our test simulations. These costs are outlined in the table below and are strictly for testing the 3 projects using agile development during the course of the semester. As shown on Table 3, the costs are minimal compared to the overall R&D operating costs. See Table 4.

Table 3- Fixed Costs

<table>
<thead>
<tr>
<th>Fixed Costs</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC setups and servers</td>
<td>$3,000</td>
</tr>
<tr>
<td>Webcams, white boards, and other office accessories</td>
<td>$500</td>
</tr>
<tr>
<td>Licensing (one-time, shared by multiple stations)</td>
<td>$8,000</td>
</tr>
<tr>
<td>Training</td>
<td>$4,000</td>
</tr>
<tr>
<td><strong>Total Fixed Costs</strong></td>
<td><strong>$15,500</strong></td>
</tr>
</tbody>
</table>

Variable costs include labor, material, overhead, and marginal costs that vary as production volume or business activities vary.

The table below shows the expected variable cost in the first year of implementation.

Table 4- Variable Costs

<table>
<thead>
<tr>
<th>Variable Costs</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Year 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database management for robustness (OpenSource)</td>
<td>$4,000</td>
<td>$4,000</td>
<td>$6,000</td>
<td>$12,000</td>
</tr>
</tbody>
</table>
Project Managers (x2) | $160,000 | $160,000 | $200,000 | $200,000
---|---|---|---|---
Other compensations | $5,000 | $7,000 | $9,000 | $18,000
Total Variable Costs | $169,000 | $171,000 | $215,000 | $230,000

After the completion of 6 projects as a prototype run for Agile, we will continue Agile development as a "production" phase where we plan to launch this process to all projects. Database will continue to expand as more projects will structure into agile development. We plan to leverage OpenSource as the codes database management. As number of projects increase, the R&D department will likely to allocate more budget to project management.

Other compensations include project bonuses, travel costs, and other utilities. By year 3, we expect database manage, project manager salary, and other compensations to all increase as a result of agile development implementation. We expect some costs to double by year 2013 as it approaches to reach the company's mission by 2015.

Table 5- Summary of Cost vs. Sales

<table>
<thead>
<tr>
<th>Year</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed cost</td>
<td>15,500</td>
<td>15,500</td>
<td>15,500</td>
<td>15,500</td>
</tr>
<tr>
<td>Variable cost</td>
<td>169,000</td>
<td>171,000</td>
<td>215,000</td>
<td>230,000</td>
</tr>
<tr>
<td>Total cost</td>
<td>184,500</td>
<td>186,500</td>
<td>230,500</td>
<td>245,500</td>
</tr>
</tbody>
</table>

The summary of all total costs including both fixed and variable costs per year are listed on Table 3.
7.1.7 Price Point

RealGate has been selling as a product since 2006. Many software upgrades and maintenance projects followed as part of sustaining engineering. As such, we will follow the price list as suggested by the product managers. We are a leading provider for test and management solutions for RealGate servers, as such there is no comparison of competing products in the market for further analytical study. The price point is $18,995 per server. This provides customer with software application for installation on the customer’s server. This assumes those customers already has the server licenses and are seeking for software to be installed.

7.1.8 SWOT Analysis

SWOT analysis was completed to study the forces that affect the process implementation:

Table 6- SWOT Analysis

<table>
<thead>
<tr>
<th>S (strength)</th>
<th>W (weakness)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Streamlined process without additional workload of documentation releases</td>
<td>• Teams can be resistant in adopting a new process</td>
</tr>
<tr>
<td>• Cross functional teams are more in-tuned with project progress</td>
<td>• Daily meetings can interfere with normal working schedule</td>
</tr>
<tr>
<td>• Problems are visible early and consistency is reached with repeated test</td>
<td>• Increase in product delivery can create more product management that require</td>
</tr>
<tr>
<td>and development cycle</td>
<td>additional database or workload</td>
</tr>
<tr>
<td>• Hardware, software, and firmware functions become more integral</td>
<td>• One miscommunication or missed task can impact the release schedule easily because no buffer is set</td>
</tr>
</tbody>
</table>
This analysis learned was helpful in understanding what type and how much risks the project should take. More attention should be given to address the risks and plans for mitigation and contingency.

### 7.1.9 Investment Capital Requirement

The total cost includes the fixed and variable costs that will need to be justified to the number of units produced and sold. This table includes the profit relative to these costs for case study A. From Profit and Loss section, agile implementation realizes a total cost saving of 5% in overall R&D expenses. In order for the project to be justified, we learned from ISE 200 that benefit needs to be greater than cost. As for breakeven analysis, I will set a hypothetical 1% of
total sales to be realized into agile development to calculate the annual profit. In year 2010, agile implementation happened in Q4 and calculation of 1% sales reflects only Q4.

Table 7- Summary of Cost vs. Sales

<table>
<thead>
<tr>
<th>Year</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed cost</td>
<td>15,500</td>
<td>15,500</td>
<td>15,500</td>
<td>15,500</td>
</tr>
<tr>
<td>Variable cost</td>
<td>169,000</td>
<td>171,000</td>
<td>215,000</td>
<td>230,000</td>
</tr>
<tr>
<td>Total cost</td>
<td>184,500</td>
<td>186,500</td>
<td>230,500</td>
<td>245,500</td>
</tr>
<tr>
<td>Sales</td>
<td>23,948,000</td>
<td>28,737,600</td>
<td>34,485,120</td>
<td>41,382,144</td>
</tr>
<tr>
<td>1% of Sales</td>
<td>59870</td>
<td>287376</td>
<td>344851.2</td>
<td>413821.44</td>
</tr>
</tbody>
</table>

Figure 17- Break Even Analysis
The total costs incurred during first three years of implementation will be low compared to the total sales. After 2012, most of the variable cost will increase almost double as more development expenses will be spent to increase development to drive product launches by year 2013. The company hopes to reach its mission in 2015 of a 100 million in sales. As Figure 18 shows, the break-even point, where total cost and total revenue meets, occurs at the beginning of agile implementation, at $184,500. However, the exact breakeven occurs after 3-months in December 2010 when projects are realized into profit after completion. There are no changes in total cost and total revenue reported within the quarter, so the graph will only show a linear interpolation during that time period. Most of the variable costs will become steady after 2013 as no significant increase in resources is needed in agile implementation.

7.1.10 Personnel

A study of agile development was performed by VersionOne in their 5th Annual State of Agile Development Survey completed on November 7, 2010. This study analyzed the industries' definition, implementation, changes, issues, and outcome with the use agile development in their organization. The data was received from emails, website surveys, and other software development industry forum. According to the survey, the VP or Director of Development along with the Development Manager account for the most percentage as likely to be the initial leader of the agile development and deployment.
For case study A, we have the VP of Engineering as the initial champion, followed by the Hardware Manager as the project lead. The agile development project team includes a number of functional managers, such as FPGA development managers, software managers, Direct of Product Verification, product managers, and product verification engineers from global locations.

Figure 18- Leading the agile development

7.1.11 Business and Revenue Model

There are 3 business models that can be used for funding Agile projects depending on how the development and organization take risks. They are as follows:
1. **Fixed Price.** The initial price is set for the project as the project requirements (iteration 0 and some of iteration 1) are developed. COCOMO II is one example of an up-front estimating strategy which is accurate to +/- 30%. According to July 2009 State of the IT Union survey, it is recommended to set a +/- 11% average. This strategy provides the least cost flexibility.

2. **Stage Gate.** Fund the project based on an evaluation system, whether it would be a time period or a decision of the scope and proven architecture with some completed iterations. This is practiced in disciplined agile methods such as open unified process. Project team will need to provide the estimate. This strategy increases workload for the team due to consistent estimation efforts involved.

3. **Time and materials (T&M).** Project will be paid initially at a low rate up front or funded by the organization and will only continue to fund the project when customer verifies a good working software release. This triggers the team to plan according for an efficient release to the customer, although it is the most risky funding strategy as it potentially faces project closure if customer rejects the release.

**7.1.12 Strategic Alliances/Partners**

For the agile process to flow efficiently in engineering, the entire organization should also move its discipline in an agile manner.
According to figure 3.3.3.2, the Enterprise Business Modeler describes the four core roles in terms of modeling the business to the direction of agile development with the similar iterative approach.
7.1.13 Profit and Loss Statement

We can use the accounting equation as a guide and provide a rough estimate of the percentage of profit to be generated by the engineering team. It is then converted to the income statement to itemize the sales, COGS, profit, expenses, and retained earnings. Refer to the income statement of Q4 2010 for Sunrise Telecom Inc below:

Table 8- Consolidated Statements of Operations for Case Study A

CONSOLIDATED STATEMENTS OF OPERATIONS
(unaudited, in thousands, except per share data)

<table>
<thead>
<tr>
<th></th>
<th>Three Months Ended</th>
<th>Twelve Months Ended</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>December 31,</td>
<td>December 31,</td>
</tr>
<tr>
<td></td>
<td>2010</td>
<td>2009</td>
</tr>
<tr>
<td>Net sales</td>
<td>$13,200</td>
<td>$13,719</td>
</tr>
<tr>
<td>Cost of sales</td>
<td>5,594</td>
<td>7,077</td>
</tr>
<tr>
<td>Gross profit</td>
<td>7,606</td>
<td>6,642</td>
</tr>
<tr>
<td>Operating expenses:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research and development</td>
<td>2,500</td>
<td>2,487</td>
</tr>
<tr>
<td>Selling and marketing</td>
<td>3,112</td>
<td>3,043</td>
</tr>
<tr>
<td>General and administrative</td>
<td>836</td>
<td>1,391</td>
</tr>
<tr>
<td>Restructuring charges</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Total operating expenses</td>
<td>6,448</td>
<td>6,921</td>
</tr>
<tr>
<td>Operating income (loss)</td>
<td>1,158</td>
<td>(279)</td>
</tr>
<tr>
<td>Other income (expense), net</td>
<td>383</td>
<td>527</td>
</tr>
<tr>
<td>Income (loss) before income taxes</td>
<td>1,541</td>
<td>248</td>
</tr>
<tr>
<td>Income tax expense (benefit)</td>
<td>97</td>
<td>(109)</td>
</tr>
</tbody>
</table>
Net income (loss) 1,444 357 167 (9,776)

Net income (loss) per share:
Basic $ 0.03 $ 0.01 $ 0.00 $ (0.19)
Diluted $ 0.03 $ 0.01 $ 0.00 $ (0.19)

Shares used in computing net income (loss)
Basic 51,328 51,349 51,341 51,349
Diluted 51,328 51,679 51,359 51,349

R&D had an expense of 2.5 millions in the Q4 of 2010. It consumes an estimated 39% of overall operating expenses. Since we estimated that a project takes 5.3 months to complete with agile development, a project would be in 56.6% completion during Q4 and will have relative cost saving of $55960 (56.6% of $98864 as estimated from section 3.3.1) per project. This cost will not include fixed cost as suggested by Table 3. It brings to a total cost saving of $335,764 for six projects, which will bring the overall operating expense down to 34%. This 5% of saving in the quarter is a hypothetical figure assuming that all 6 agile projects have started in the beginning of Q4.

7.1.14 Exit Strategy

As the total cost would impact the revenue by end of Q2 in 2012, a new cost method should be implemented to allow ROI to stay positive. By the end of Q2 2012, the following should be considered in the new phase of agile development:

- Extension of practice to the company’s outsourced projects and contract manufacturers
- Obsolete older products and replace with newer products from faster product development
- Re-list in the stock market and become public again
- Recognition of the process effectiveness and efficiency by major customers

### 7.2 Return on Investment (ROI)

ROI is a performance indicator of how well the investment is utilized compared to the return or benefit the organization received based on that particular investment.

The return on investment formula:

\[
ROI = \frac{(\text{Gain from Investment} \cdot \text{Cost of Investment})}{\text{Cost of Investment}}
\]

#### Table 9- ROI Calculation

<table>
<thead>
<tr>
<th>Year</th>
<th>Total Cost</th>
<th>Total Revenue</th>
<th>ROI</th>
<th>ROI (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>184,500</td>
<td>59,870</td>
<td>-0.6755</td>
<td>-67.5501</td>
</tr>
<tr>
<td>2011</td>
<td>186,500</td>
<td>287,376</td>
<td>0.5409</td>
<td>54.0890</td>
</tr>
<tr>
<td>2012</td>
<td>230,500</td>
<td>344,851.20</td>
<td>0.4961</td>
<td>49.6101</td>
</tr>
<tr>
<td>2013</td>
<td>445,500</td>
<td>413,821.44</td>
<td>-0.0711</td>
<td>-7.1108</td>
</tr>
</tbody>
</table>
ROI peaks at about 54% in 2011. For ROI to maintain positive over the next few years, cost need to be kept at minimum and should not exceed $400,000 as suggested by the break-even analysis unless sales will jump significantly.

7.3 Norden-Rayleigh Analysis

Norden-Rayleigh distribution is used to calculate cumulative costs of R&D projects. It models the buildup, peak, and taper of a development program's effort over time. (Lee 1999)
Cumulative distribution function for the Rayleigh:

\[ V(t) = d(1 - e^{-at^2}) \]

- \( V(t) \) = Total effort expended
- \( a \) = Scale factor of the distribution
- \( t \) = Time
- \( d \) = Shape parameter

Probability density function for the Rayleigh:

\[ v(t) = 2adte^{-at^2} \]

Early in the program (because the ACWP is immature), the pdf parameters – \( a \) and \( d \) – can only be “found” from the schedule variables. Below are the equations for calculating \( a \) and \( d \).

\[ V(t) = d(1 - e^{-at^2}) \]

at \( t = 0 \), \( V(t) = d(1 - e^{-at^2}) \)

\[ V(t_f) = d(1 - e^{-at^2}) \]

\[ .97d = d(1 - e^{-at^2}) \]

\[ .97 = (1 - e^{-at^2}) \]

\[ e^{-at^2} = .03 \]

\[ -at^2 = \ln(.03) \]

\[ a = -\ln(.03) / t_f^2 \]

\[ d = V(t) / (1 - e^{-(-\ln(.03)/t_f^2)}) \]

where \( t_f \) is known

Because \( V(t) \) does not reach \( v_0 \) in finite time, the project’s end time is usually defined as the time at which:

\[ V(t_e) = 97\% \ of \ v_0 \]

or, \( V(t_e) = .97d \)

The authors recommend using this computation only as a rough cross check to the program plan, particularly for the curve generation.

A mismatch between this derivation of \( d \) and the program funding should be viewed as an indicator of schedule and funding misalignment.

**Figure 20- Norden-Rayleigh Models by Northrop Grumman**

Program Estimate: $184,500 \rightarrow d$

Program Start date: Q4 of 2010

Program End date: Q2 of 2012

Year 1 = 1 quarter; Year 2 = 4 quarters; Year 3 = 2 quarters
\[ t_f = 1 + 4 + 2 = 7 \rightarrow 7 \text{ quarters total in program} \]

\[ a = -\ln (0.03)/ t_f^2 = -\ln (0.03) / (7^2) \]

\[ a = 0.07156 \]

<table>
<thead>
<tr>
<th>t</th>
<th>d</th>
<th>a</th>
<th>d*(1-exp(-at^2))</th>
<th>2adt*(exp(-at^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>184,500</td>
<td>0.07156</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>1</td>
<td>184,500</td>
<td>0.07156</td>
<td>12741.49260</td>
<td>24582.07758</td>
</tr>
<tr>
<td>2</td>
<td>184,500</td>
<td>0.07156</td>
<td>45925.30927</td>
<td>39665.61947</td>
</tr>
<tr>
<td>3</td>
<td>184,500</td>
<td>0.07156</td>
<td>87606.78813</td>
<td>41602.06945</td>
</tr>
<tr>
<td>4</td>
<td>184,500</td>
<td>0.07156</td>
<td>125785.33863</td>
<td>33612.96934</td>
</tr>
<tr>
<td>5</td>
<td>184,500</td>
<td>0.07156</td>
<td>153665.02914</td>
<td>22065.50515</td>
</tr>
<tr>
<td>6</td>
<td>184,500</td>
<td>0.07156</td>
<td>170465.90869</td>
<td>12051.35489</td>
</tr>
<tr>
<td>7</td>
<td>184,500</td>
<td>0.07156</td>
<td>178964.34740</td>
<td>5545.83820</td>
</tr>
</tbody>
</table>

Norden-Rayleigh Model

\[ V(t) = d(1-exp(-at^2)) \]
\[ V(t) = 2adt(exp(-at^2)) \]
## 8 Gantt Chart

### 8.1.1 Phase I

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure</td>
<td>45 days</td>
<td>Fri 11/19/10</td>
<td>Fri 1/4/11</td>
<td>37</td>
</tr>
<tr>
<td>Gather Agile best results of projects simulated</td>
<td>10 days</td>
<td>Fri 11/19/10</td>
<td>Thu 12/2/10</td>
<td>37</td>
</tr>
<tr>
<td>Compare performance &amp; quality between current release vs. simulated Agile projects</td>
<td>10 days</td>
<td>Fri 12/3/10</td>
<td>Thu 12/16/10</td>
<td>40</td>
</tr>
<tr>
<td>Compare actual and estimated development costs</td>
<td>5 days</td>
<td>Fri 12/17/10</td>
<td>Thu 12/23/10</td>
<td>41</td>
</tr>
<tr>
<td>Progress report</td>
<td>1 day</td>
<td>Fri 12/24/10</td>
<td>Fri 12/24/10</td>
<td>42</td>
</tr>
<tr>
<td>Report and presentation write-up</td>
<td>5 days</td>
<td>Mon 12/27/10</td>
<td>Fri 12/31/10</td>
<td>43</td>
</tr>
<tr>
<td>Buffer for Measure</td>
<td>66 days</td>
<td>Mon 12/9/11</td>
<td>Fri 1/14/11</td>
<td>48</td>
</tr>
<tr>
<td>Extend small software projects</td>
<td>15 days</td>
<td>Fri 12/24/10</td>
<td>Thu 12/13/11</td>
<td>45</td>
</tr>
<tr>
<td>Compare best practices as implemented in other companies (NEC, Supermicro, Jobscience)</td>
<td>15 days</td>
<td>Fri 12/24/10</td>
<td>Thu 12/13/11</td>
<td>47</td>
</tr>
<tr>
<td>Review thoroughness of code check-in and documentation</td>
<td>5 days</td>
<td>Wed 1/26/11</td>
<td>Tue 1/12/11</td>
<td>48</td>
</tr>
<tr>
<td>Identify roadblocks and flaws, prioritize defects</td>
<td>10 days</td>
<td>Wed 2/2/11</td>
<td>Tue 2/15/11</td>
<td>49</td>
</tr>
<tr>
<td>Extension from development to manufacturing and general release</td>
<td>5 days</td>
<td>Tue 2/15/11</td>
<td>Mon 2/21/11</td>
<td>50</td>
</tr>
<tr>
<td>Create manufacturability/business model with flowchart</td>
<td>5 days</td>
<td>Tue 2/29/11</td>
<td>Mon 3/7/11</td>
<td>51</td>
</tr>
<tr>
<td>Progress report</td>
<td>1 day</td>
<td>Mon 3/7/11</td>
<td>Mon 3/7/11</td>
<td>52</td>
</tr>
<tr>
<td>Elevator Pitch Presentation slides</td>
<td>5 days</td>
<td>Tue 3/1/11</td>
<td>Mon 3/7/11</td>
<td>52</td>
</tr>
<tr>
<td>Report and presentation write-up</td>
<td>15 days</td>
<td>Mon 3/7/11</td>
<td>Fri 3/18/11</td>
<td>52</td>
</tr>
<tr>
<td>Elevator Pitch presentation</td>
<td>1 day</td>
<td>Fri 3/18/11</td>
<td>Mon 3/25/11</td>
<td>56</td>
</tr>
<tr>
<td>Buffer for Analyze</td>
<td>5 days</td>
<td>Mon 3/25/11</td>
<td>Fri 3/22/11</td>
<td>56</td>
</tr>
<tr>
<td>Improve &amp; Control</td>
<td>20 days</td>
<td>Mon 3/7/11</td>
<td>Fri 3/22/11</td>
<td>56</td>
</tr>
<tr>
<td>Continue to monitor some iterations (story/issue)</td>
<td>10 days</td>
<td>Mon 3/25/11</td>
<td>Fri 4/8/11</td>
<td>56</td>
</tr>
<tr>
<td>Eliminate or move iterations to further increase efficiency</td>
<td>3 days</td>
<td>Mon 4/8/11</td>
<td>Fri 4/12/11</td>
<td>57</td>
</tr>
<tr>
<td>Progress report</td>
<td>1 day</td>
<td>Fri 4/8/11</td>
<td>Fri 4/12/11</td>
<td>60</td>
</tr>
<tr>
<td>Report and presentation write-up</td>
<td>5 days</td>
<td>Mon 4/18/11</td>
<td>Fri 4/22/11</td>
<td>60</td>
</tr>
</tbody>
</table>
9 Conclusion

The economic justification, both quantitative and qualitative, shows a direct correlation of streamlined and efficient outcome as a result of running large-scale engineering projects in an organization. The result of this approach proved a number of detrimental effects to the viability and success of the projects. It proved a number of benefits as it replace the traditional methods.

In summary, agile development in large-scale projects has proven the following advantages:
• Greatly align to time-to-market by synchronizing development efforts

• Adaptive to customer requests by redirecting to only a portion of the development iterations

• Easily adjusted to market influences and product managers’ requests

• Attracts business stakeholders when projects are executed with Agile process

10 Appendix

10.1 Appendix A
Case Study A – Sunrise Telecom

Sunrise Telecom is a market-leading provider of test, monitoring, and workflow management solutions primarily for cable, telecom, and wireless operators. Our broad portfolio of feature-rich, easy-to-use products enable our customers to deliver premium voice, video, data and next-generation digital multimedia services faster, cheaper and with better quality than their competition. As a testament to our leadership, over 200,000 of our test units have been deployed by 2,000+ customers across 65 countries. Studies of agile development implementation on 6 projects using Agilefant. Most of the economic justification analysis will use case study A as a study. The included information in this report is proprietary of Sunrise Telecom and should not be distributed.
Case Study B – NEC

Studies of RTL implementation as a comparison to case study A with the goal of adopting a few best practices to be used as part of the agile development projects. I would like to give many thanks to Peter Teng, the senior engineering manager from NEC for sharing his expertise.

10.2 Appendix B – Glossary of Terms

**Burn-down chart**- a publicly displayed chart showing remaining work in the sprint backlog. Updated every day, it gives a simple view of the sprint progress. It also provides quick visualizations for reference.

**Increment**- features are rolled out for development only after the first features are developed and completed. In agile, we try to avoid this approach as developers and testers do not have visibility of the other features.

**Iteration**- all features are concurrently developed and released. They will be refined with additional sub-features. Problems rise early and can allocate additional tasks for bug fixes and verification based on internal and external evaluation.

**Sprint**- is a "time-boxed" period of work, with a closely defined and agreed output. In the software development world, the output is "potentially shippable" code. At the end of the sprint, the "potentially shippable" code is presented back to the client in a playback session.

**Task**- is a simple description of how to do some bit of work towards completing an item in the Work Item List

**Task points**- a relative measure that can be used for agile estimation of size. The team decides how big a point is, and based on that size, determines how many points each work item is.

**User Story**- A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it.
11 References


Antonia, G., Ph.D. (2009). Emergence of expertise in categorization of complex visual stimuli. Northwestern University


Subject Matter Expertise in Performance Analysis. Retrieved from https://www.fbo.gov/index?s=opportunity&mode=form&id=d19f56f77dd29d05a47c8dfb44a6f95f&tab=core&_cview=0


