Task-Based Estimation and Planning for Application Development Projects and Resources: Models, Methods and Applications

Chidambaram Subbiah
M.B.A, University of Missouri, Saint Louis, 2002
B.E., Computer Science Engineering, Bhopal University, Bhopal 1995

A Dissertation submitted to The Graduate School at the
University of Missouri – St. Louis in
partial fulfillment of the requirements for the degree
Doctor of Philosophy in Business Administration
with an emphasis in Logistics & Supply Chain Management

December, 2019

Dissertation Committee

Haitao Li, Ph.D., Chairperson
Andrea Hupman, Ph.D.
Donald Sweeney, Ph.D.
Joseph Simonis, Ph.D.
Kailash Joshi, Ph.D.

Contents

List of Tables 4

List of Figures 4

Abstract 7

Chapter 1: Introduction 8

1.1 Background and Research Motivation 8

1.2 The Framework of the Dissertation 12

Chapter 2: Literature Review 17

2.1 Bottom-up estimating and measuring the size of software applications 18

2.2 Agile Release Planning Methodology 26

2.2.1 Definition of terms for an Agile project execution 26

2.2.2 Agile release planning methodology literature review 29

2.3 Predictive modeling in the context of estimating effort for open systems web development projects 37

2.4 Combining different methodologies and assessing the quality of the estimate 44

Chapter 3: Research Purpose and Contributions 50

3.1 Advance the state of the art of estimating software development costs using design patterns. 51

3.2 Quantify the economic impact of resources and other factors on overall labor cost. 52

3.3 Implementing multiple estimating models by combining executive judgment with quantitative modeling 53

3.4 Applying prescriptive analytics (optimization) to build a new decision-support framework for Agile project planning 54

Chapter 4: Methodologies 55

4.1 Estimating Tool Development 57

4.1.1 Function points related predictive models 57

4.1.2: Two Stage Least Squares (2SLS) model and Decision Tree Analysis 60

4.1.3 Function points related suite of estimates 64

4.1.4 Development of the task based estimating model 64

4.1.5 Estimating tool requirements: 67
4.2 Assessing the quality of the estimate from the estimating tool 69
4.3 Optimization model for allocating resources to tasks in an Agile setting 70
Chapter 5: Data 78
5.1: Data for predictive models 78
5.2: Use case for Agile Software Development epic and release planning 93
5.2.1: Introduction to optimization use case 93
5.2.3: Use Case 95
Chapter 6: Model Testing Results and Experiments 105
6.1 Estimating tool development – related models and development of the tool 105
6.1.1 Function points related predictive models: 105
6.1.2 Two Stage Least Squares (2SLS) model results and Decision Tree analysis 111
6.1.3 Function points related suite of estimates: 123
6.1.4 Implementation of the Estimating Tool: 126
6.1.5 Creation of the Estimating Tool 135
6.1.6 Summary of the new estimating Process: 139
6.2 Assessing the Quality of the Estimate from the Estimating Tool 143
6.3 Optimization Model Results and Experiments 154
6.3.1 Current process of executing projects 160
6.3.2 Base Case 161
6.3.3 Scenario 1: Bringing more efficiencies by cross training resources on multiple skills 165
6.3.4 Scenario 2: Combining all development tasks separately. 167
6.3.5 Medium size case with 10 projects and 114 stories 170
6.3.6 Large size case with 20 projects and 228 stories 171
6.3.7 How to use the model at an organization 173
6.3.8 Computational Experiments 176
Chapter 7: Conclusion and Future Work 190
7.1 Summary 190
7.2 Limitations 195

7.3 Future Work 196
References 197

List of Tables

Table 1: Bottom-up estimating and measuring the size of software applications research 18
Table 2: Agile release planning methodology research 30
Table 2.1: Agile release planning optimization models research 34
Table 3: Predictive modeling in the context of estimating effort for open systems web development projects 38
Table 4: Combining different methodologies and assessing the quality of the estimate.. 44
Table 5: Function point repository 80
Table 6: High level data analysis of project data 89
Table 7: Project data by phases 89
Table 8: Data analysis by project phases 90
Table 9: Data analysis by resource type 90
Table 10: Data analysis based on usage of temporary resources 91
Table 11: Data analysis based on quality metrics 92
Table 12: Project data for optimization use case 101
Table 13: Skills data for use case 103
Table 14: Resource types for use case 104
Table 15: Summary statistics for Function points related metrics 108
Table 16: Tableau Dashboard of Function Point Suite of Estimates and differences of the estimates from the actual labor cost 125
Table 17: Sample survey 132
Table 18: Snippet of aggregated survey responses 134
Table 19: Processing of survey results in SAS 135
Table 20: Summary of task level data 154
Table 21: Sample efficiency score prototype 156
Table 22: Computational experiment results from the batch run 177
Table 23: Sensitivity Analysis dataset 181

List of Figures

Figure 1: Project Steps – Data Collection to Estimating Model to Resource Plannning.. 67
Figure 2: Function points dataset 79
Figure 3: Actual labor cost dataset sample 81

Figure 4: Aggregated task level data sample 84
Figure 5: Flow to get the aggregated dataset for modeling 87
Figure 6: Scatter plot of function point count with total labour hours............................. 87
Figure 7: Project execution with optimal resource allocation 97
Figure 8: Effect on project duration due to resource constraints 97
Figure 9: High-level diagram for use case 100
Figure 10: Tentative project plan 105
Figure 11: Function point repository visulization... 106
Figure 12: Scatter plot of adjusted function point and total labor hours 106
Figure 13: Bar graph of cost per FP for total cost across all projects sorted in ascending order 107
Figure 14: Bar graph of cost per FP for design and development across all projects sorted in ascending order 108
Figure 15: Box Whisker Plot of the difference between Actual Labor Cost and Predicted Cost. 111
Figure 16: Histogram of total project labor hours 112
Figure 17: Model description and summary for Two Stage Least Squares model with hours less than 7500 hours. 114
Figure 18: Scatter plot of residuals with predicted values for 2SLS (less than 7550 hours) 115
Figure 19: Test on residuals for 2SLS model with hours less than 7500 hours............. 116
Figure 20: Model description and summary for Two Stage Least Squares model with hours less than 10000 hours. 117
Figure 21: Scatter plot of residuals with predicted values for 2SLS (less than 10000 hours) 118
Figure 22: Test on residuals for 2SLS model with hours less than 10000 hours........... 119
Figure 23: Model description and summary for Two Stage Least Squares model with hours less than 10000 hours less one outlier.. 120
Figure 24: Decision Tree Analysis, Regression Tree (CART) 121
Figure 25: Decision Tree Analysis, Regression Tree (CHAID) 122
Figure 26: Snapshots of implemented COCOMO II tool 123
Figure 27: Masked high level design patterns at ABC Inc. 128
Figure 28 - Estimate Comparative tab in Estimating Tool 136
Figure 29: Design and development tasks entered by the SME 136
Figure 30: Pie charts in the overhead and summaries tab of Estimating Tool 136
Figure 31: Cost breakdown summary in Estimating Tool 137
Figure 32: Project data by phases for the past 12 months 138
Figure 33: Cost projection table in Estimating Tool... 141
Figure 34: Overhead and Summaries tab in Estimating Tool... 142

Figure 35: Suite of seven estimates 142
Figure 36: New software development estimating lifecycle 143
Figure 37: Tableau reporting for processing data from finished projects 144
Figure 38: Tableau Analysis for analyzing task level data 145
Figure 39: Comparison of actual hours and estimate from Estimating Tool 146
Figure 40: Project comparison by type (Create vs. Modify) 147
Figure 41: Project comparison by complexity 148
Figure 42: Project comparison by category (Open Systems vs. Back-end Systems) 148
Figure 43: Histograms assessing the quality of the estimate for five projects 150
Figure 44: Statistics on processing times at the task level 153
Figure 45: High level representation of use case data 162
Figure 46: Resource utilization for the base case execution 164
Figure 47: Sprint load across the five sprints 164
Figure 47a: Allocation of stories to the five sprints in base case 165
Figure 48: Sprint load across five sprints for the first scenario 166
Figure 49: Resource load utilization for the first scenario 167
Figure 50: Resource load for scenario two 168
Figure 51: Sprint load for scenario two 169
Figure 52: Resource load for medium size case (Ten Projects) 170
Figure 53: Sprint load for medium size case 171
Figure 54: Resource load for large size case (20 Projects) 172
Figure 55: Sprint Load for large size case (20 projects) 173
Figure 56: Proposed new flow incorporating Optimization Model 174
Figure 57: Result of computational experiments – Objective value analysis 179
Figure 58: Result of computational experiments – Number of stories allocated 180
Figure 59: Sensitivity analysis of objective value to varying efficiency scores and sizes 182
Figure 60: Sensitivity analysis of number of stories allocated to varying efficiency scores and sizes 183
Figure 61: Sensitivity analysis of objective value to varying workload availability and sizes 184
Figure 62: Sensitivity analysis of number of stories allocated to varying workload availability and sizes 185
Figure 63: Sensitivity analysis of objective value to varying payrates and sizes 186
Figure 64: Sensitivity analysis of number of stories allocated to varying payrates and sizes 187
Figure 65: Computational Experiments results summary 188
Figure 66: Duration for all runs in the batch process 189
Figure 67: Assignment of resources to a project 192

Abstract

This dissertation takes a new approach to software development effort estimation from the perspective of design patterns at an organization. A new estimating tool is developed to provide bottom up estimates based on the design patterns of the organization. The research also offers guidelines for extracting the unique design patterns specific to an organization that are used to obtain baseline task level estimates in the Estimating Tool. In addition, the tool provides a suite of seven estimates using predictive analytics to estimate the labor hours required for a project using historical data, a bottom-up estimate that is rooted in the design patterns of the organization, a recommended estimate, plus four other estimates that are based on the function point count. The four estimates include a predictive model to estimate the project cost based on the function point count and three estimates are variants of the early design Construction Cost Model II (COCOMO II). Direct benefits of the tool include reduction of process variability thereby resulting in consistency of estimates across teams in an organization.

In Agile IT Project Management, there is an important need to better plan project timelines and to make better scheduling and resource allocation decisions to facilitate on-time and on-budget project deliveries. This research thus also addresses the prescriptive aspect of the application, by making better resource allocation decisions via a new mixed-integer linear programing (MILP) optimization model. The model provides data-driven decision-support for companies looking to make a transition from the waterfall to the agile paradigm with a structured approach to focus on skills development at the organization. A

real-world application of the use of the new Estimating Tool and the proposed models at a large firm is showcased as part of the research. A comprehensive sensitivity analysis was conducted as part of a set of computational experiments to obtain managerial insights.

Chapter 1: Introduction

1.1 Background and Research Motivation

Almost all types of organizations, either large or small, perform Application Development to meet the growing needs of business. Application Development often involves estimating the size of the effort upfront so that the organization can make appropriate budgeting and scheduling decisions. According to Gartner (2014), Application Development accounts for 34% of Information Technology budgets. One of the biggest challenges companies face is their inability to size the associated effort properly. Chemuturi (2009) defines *Software Development Estimation* as “the estimation of software size, software development effort, software development cost, and software development schedule for a specified software project in a specified environment, using defined methods, tools and techniques." The effort is measured in person-days or person-hours, and the schedule is optimized based on the capacity of resources available. Every organization conducts software estimation differently, and there is very little consistency in the industry. While there are off-the-shelf tools that can be used for software development estimation, these tools do not account for the unique application environment and business rules under which companies operate, and cannot be customized for individual companies. Shepperd (2007) states that accurately predicting software development effort is a crucial concern of many organizations. Halkjelsvik and Jørgensen (2012) present evidence that the average cost and effort overrun for software projects is about thirty percent.

Application Development is often managed as a project, and the term *project estimation* can be used interchangeably with software development estimation. It is often the case where there are so many risk contingencies built into the estimate that one must peel off the layers to know the true cost of a project. I have personally been involved in Application Development for more than 22 years in various roles, including being a developer, subject matter expert (SME), and project leader. I have seen a wide range of methodologies and tools being used in this space to estimate projects. Most of these methods often rely on expert estimation or the subjective judgment of SMEs. Albrecht and Gaffney (1983) define function point as "a unit of measurement to express the amount of business functionality an Information system or application provides to a user." Many organizations use function points as a sizing measure. The estimating process can often go through multiple layers, including risk contingency through the approval hierarchy of the organization before obtaining the final estimate. The biggest disadvantage of this process is the lack of consistency in the process between different application teams that use the same technology stack to develop similar applications. There is significant variability in the estimates obtained by different teams for the same type of work.

This research was motivated by the initiative at a service organization to develop a tool for project estimation. It is a large international firm with a Program Management Office (PMO) that manages the execution of information system projects. The information system projects are centrally managed, and most of the projects are web-based projects that are executed to bring efficiencies in the workflow for their customers. The functionality delivered by these projects are unique to the organization, but the process and methodology can be applied to other organizations that use a similar technology stack. It is a mature organization, already using function points to size projects. The organization was looking to make a transition from the traditional Waterfall way of executing projects to the Agile environment. There were a few pilot initiatives in the Agile space. Jones (2007) makes the case for why an organization needs to use function points:

1. Projects with function point analysis early in the process have lower “requirements creep” than uncounted projects.
2. Projects with function point analysis have about 15% lower cost overruns than uncounted projects.
3. Projects with function point analysis have about 25% less schedule slip than uncounted projects.
4. Preliminary analysis indicates that function point analysis saves $15 to $50 per function point. ROI = 15 to 1 roughly.

One of the most significant advantages is that the project team needs to go through a structured process to get the function points for a project counted. This often can get the team thinking about the first level design, and it lays the foundation for a quality design phase.

The organization had set up the foundation to use the COCOMO II model (COnstruction COst MOdel II), which is an approach to estimate the cost, effort, and schedule for a new software development project. It takes the number of function points as its input, and it is a constructive software estimation model based mainly on regression techniques. The model is defined in terms of scaling factors and effort multipliers, which are used for estimating overall effort and cost (Cocomo, 2000). The company relied primarily on expert estimation, with SMEs in each area performing the design and estimates. It then used executive judgment in addition to the estimate from the SME to estimate the cost and effort associated with the project, which was then used to make scheduling decisions. As mentioned earlier, the two major disadvantages were the lack of consistency in estimates and variability that existed in estimates for work that was very similar.

There is abundant research in software development estimation since the early 1980s. Sehra, Brar, Kaur, and Sehra (2017) find that there are several articles in this space beginning in the early 1990s, and it continues to be actively researched. Myrtveit, Stensrud, and Shepperd (2005) mention that software cost estimation has been a complex and difficult task evidenced by the amount of research in this space. There have been numerous studies over the years comparing the accuracy of estimates regarding overall cost, duration and effort using different estimating models, but many of these approaches fail to consider the specific environment and business rules that are unique to the organization which is often the driver of costs in a software development effort. The business rules (or operations) of the organization often dictate what design patterns are used by the organization. There has been very little research around developing a task based estimating model while considering the unique design patterns used by an organization. A *design pattern* is a generic solution to a commonly occurring issue in software design and development.

Design patterns are useful in that they can speed up the development process in that a solution is often implemented the same way, and design patterns frequently can be reused. Many companies find their niche in rapid application development by standardizing on reusable design patterns. Organizations often rely on expert estimates, and those estimates are often from subject matter experts who have worked in those systems for a long time. Big organizations standardize on technology design patterns, and application development teams use the same technology stack and design patterns for most application development. This lends itself to bottom up estimating that is rooted at the task level. The key to estimating then becomes uncovering the underlying unique design pattern of the organization and utilize it for better project effort estimation.

Another thrust of my dissertation is to propose augmenting the use of the estimating tool by building a resource allocation and project scheduling framework that will help make better scheduling project tasks in an Agile environment. My goal is to build an optimization model that will account for the varied skill sets and domain expertise in the resource pool. The research on the estimating models will set the foundation for the resource allocation model.

1.2 The Framework of the Dissertation

My research aims to fill the gap in the existing research literature and practices in the real world. One thrust of my proposal is to develop a process to find the underlying design pattern of the organization, then decompose the design pattern into scenarios and tasks based on interviews with a small group of subject matter experts or technical architects in the company. Baselines for the tasks and scenarios can be constructed from surveys of

SME estimates. These baselines can be the basis for creating a bottom-up estimating model. One main contribution of this research is the development of an estimating tool that helps a team complete a bottom up estimate right after completing a thorough design for the project. The tool will incorporate the capability to factor in the local environment of the organization and the tasks/scenarios that were decomposed from the design patterns at the company. It will also have other features to enhance the usability of the tool, that cater to the unique environment of the company. Some of these features will include the capability to breakdown the estimate into smaller chunks by functionality, requirements, etc., and to override the baseline at the task level. The tool will then show the estimate from the subject matter expert with the override and compare it with the recommended cost from the tool without the overrides.

This research has been conducted in the context of a service company that does 20-25 application development projects every year. I was granted access to data from all application development projects going back seven years. The data from the past seven years was used in the development of a suite of predictive models to estimate the total labor hours needed to complete a project. Software development in the early 1990s used to have two major costs, system costs and labor costs. Over time, the cost of computing in terms of system costs has drastically decreased and is no longer a significant contributor to costs. Labor costs today are one of the main cost component in Software Application Development.

There has been little attempt made by organizations to approach software estimating by predicting the labor cost associated with the project and identifying the underlying drivers of labor cost. This research fills this gap by creating an estimating model that estimates the total labor hours for a project, identifies the most important explanatory factors that contribute to the overall labor cost, and quantifies the economic impact of resources. Due to the endogenous nature of some explanatory variables in the models, we build two-stage regression predictive models to predict the number of labor hours that will be required for a project. We then translate the labor hours to a cost and add other costs associated with the project, including vendor costs, hardware costs, among others to come up with an overall estimate for the project. The fundamental assumption underlying these models is based on the analysis of the existing cost data, which indicates that labor costs are the single most important contributor to application development costs at the company.

The models provide a suite of cost estimates to the decision makers, who integrate expert executive judgment with data science. The organization was already using function points to measure the size of development efforts, and I was also granted access to this data. I derived a predictive model to predict the cost of a project based on the historical data using function points and actual labor costs of a project. I also used the COCOMO II early design model to come up with three estimates based on varying a couple of scaling factors. The bottom-up estimate based on the design pattern completed by the subject matter expert and recommended estimate from the tool complete the suite of estimates. Overall, I obtained a suite of seven estimates for decision makers to assess and use their executive judgment to challenge the experts on the estimate. The suite of estimates are explained below.

SME/TL Estimate: This is the estimate of the subject matter expert/Technical Lead, including the override time that they have chosen for all tasks in the newly created estimating tool.

Recommended Cost: This the estimate for all the tasks considering only the time that the tool recommends leaving out the override time chosen for the individual tasks.

Predictive Cost: This is the predicted cost for the project based on the historical data of projects from 2011 to 2017. This estimate considers factors such as # of associates, # of contractors, # of experienced contractors, duration of the project, # of development tasks, time spent in testing, time spent in planning, which area is doing the project, what type of project, etc. The most significant contributors to labor cost at the organization are considered to predict the cost. These contributors and associated weightages will be determined from the newly created two stage regression model.

Function point baseline: This is the predicted cost at the organization for the project based on the historical data in the function point's database. It is calculated based on calculated function points. We take the function point number and adjust it for a requirement creep of two percent per month and use the adjusted number to predict the cost of the project.

COCOMO II Best Case, Middle of the Road, and Costliest: These three numbers also consider the adjusted function point count. It gives one a range of estimates based on adjusting scale factors and cost drivers in the model. We only change two of the cost drivers in the model. One is able to obtain a range based on the function point count.

- The best case assumes one to have a high performing team and schedule pressure is relaxed.
- The middle of the road case assumes one to have a nominal team and schedule pressure is normal.
- The costliest case assumes one to have a nominal team and schedule pressure is high.

All three scenarios assume the following scale factors and cost drivers.

The scale factors were chosen in discussion with technology leaders at the firm. The process maturity is nominal, experience of similar projects is high, flexibility required in the system is nominal, team cohesiveness is high, project risk and architectural complexity is low.

The cost drivers again were chosen in discussion with the technology leaders at the firm. System reliability, complexity, and size indicator are nominal, reusability is nominal, platform difficulty is nominal, application language and tool experience are high, and using case tools for development is high.

The estimating models set the stage for the development of a resource allocation framework that will help make better scheduling decisions. This framework relies on an optimization model that considers skill and domain expertise while assigning resources to tasks in an Agile Project Methodology setting. Agile practitioners typically do not approach the planning and scheduling of resources from an allocation framework that considers the skill and domain expertise of individuals. My model will address that need for small projects in an Agile setting and will lay the foundation for project managers to make better scheduling decisions based on the best resource available to work on a task. Many organizations are considering or making a transition from a traditional Waterfall to an Agile methodology for execution projects. This research lays out a framework for making this transition focusing on skills development to best utilize the workload of resources available, and it employs a prescriptive optimization model at the core of this framework.

Chapter 2: Literature Review

There is abundant research on software development estimation since the early 1980s. Sehra et al. (2017) find that there are several articles in this area beginning in the early 1990s, and it continues to be actively studied. There are more than 80 papers that have been published over a period of ten years from 2007 through 2016. Albrecht and Gaffney (1983) make the case that one of the most important problems faced by software developers and users is the prediction of the size of a programming system and its development effort. The literature review in this chapter is organized into four main sections.

1. The first section reviews the papers that focus on bottom-up estimating and on various approaches to measure the size of software applications.
2. The second section reviews the papers that focus on executing projects using the Agile Project Methodology, especially those dealing with release planning. This

section also reviews papers on optimizing resource allocation and scheduling stories in the Agile context.

3. The third section reviews the papers that focus on estimating models in the context of estimating effort for open systems such as web development projects.
4. The fourth section reviews the papers that make the case that a suite of estimates is better than having one estimate.

2.1 Bottom-up estimating and measuring the size of software applications

Bottom-up estimating is a way in which a big project is broken down into smaller subcomponents, and each subcomponent is estimated in terms of time needed for the subcomponent or task. The aggregation of the individual estimates of the subcomponents then becomes the estimate for the overall software project. Bottom-up estimating facilitates the generation of a work breakdown structure, which is then used for project scheduling and resource allocation. Table 1 summarizes the literature regarding bottom-up estimating and measuring the size of software applications.

Table 1: Bottom-up estimating and measuring the size of software applications research

Author	**Methodology**	**Modeling Technique**
Albrecht and Gaffney (1983)	Estimate the amount of "function" the software is to perform.	Paper Introduction Function Points
Low and Jeffery (1990)	Empirical Research	Compare Estimates from different methodologies
Symons (1991)	Mark II Function points	Paper Introducing Mark II function points

Author	**Methodology**	**Modeling Technique**
Verner and Tate (1992)	Case Study	Regression
Miyazaki, Terakado, Ozaki, and Nozaki (1994)	Case Study	Robust Regression
Hakuta, Tone, and Ohminami (1997)	Empirical Research and Survey for baselines	Regression
Pfleeger, Jeffery, Curtis, and Kitchenham (1997)	Literature Review	
Raz and Elnathan (1999)	Generic Model for costing of Projects	Activity based Costing.
Dolado (2000)	Case Study	Regression, Machine Learning, Neural Networks, Genetic Programming
Hill, Thomas, and Allen (2000)	Case Study	Regression
MacDonell (2003)	Comparison of Regression with Fuzzy Logic Modeling	Fuzzy Logic Modeling
Regolin, De Souza, Pozo, and Vergilio (2003)	Size Estimation	Machine Learning
Moløkken and Jørgensen (2003)	Literature Review	
(Regolin, De Souza, Pozo, & Vergilio, 2003)	Case Study using ISBSG data	Genetic Programming, Neural Networks
Jørgensen (2004a)	Literature Review	
Jørgensen (2004b)	Case Study	Generic Statistical Models
Costagliola, Ferrucci, Tortora, and Vitiello (2005)	Introduces the Class Point approach	Regression
Pendharkar (2004)	Case Study	Regression, Decision trees

Author	**Methodology**	**Modeling Technique**
Jørgensen (2007)	Literature Review	
Kanmani, Kathiravan, Senthil Kumar, and Shanmugam (2007)	Case study using 40 student projects	Machine Learning
Forsyth and Burt (2008)	Study of two experiments in a case study.	Generic Statistical Models
Wijayasiriwardhane and Lai (2008)	Introduce Component Point approach.	
Cunha, Cruz, Costa, Rodrigues, and Vieira (2012)	Case Study	Comparison of Estimation Approaches
Zhou, Yang, Xu, Leung, and Zhou (2014)	Objective class points	Regression
Jørgensen (2014)	Case Study	

Hill et al. (2000) analyze the accuracy of expert estimates for tasks. They study the link between task time and the number of subtasks involved in the task and find a significant relationship between the two attributes. The authors make the case that the way we breakdown tasks in a work breakdown structure to identify subtasks is also one of the most useful things that experts can do to estimate the times of the tasks better. Forsyth and Burt (2008) conduct three experiments in a non-software context to compare time allocation for a single task with the total time allocation given to all the subtasks that make up the single task. The authors advance the concept of a segmentation effect where they find higher estimates when tasks are decomposed. One other interesting conclusion from the paper is that the decomposed estimates are more accurate when predicting larger subtasks as opposed to smaller subtasks. Cunha et al. (2012) present a case study implementing a bottom-up estimating approach in a software company. The authors find that the estimates greatly improved using the bottom-up estimating approach.

Albrecht and Gaffney (1983) present a method to measure size based on the concept of function points. They define a function point as "a unit of measurement to express the amount of business functionality an Information system or application provides to a user." Low and Jeffery (1990) show how function points can be used to compute a functional value of an application in terms of external inputs, external outputs, external inquiries, internal logical files, and external interface files. Function points have been in use since the early 1990s and are used by companies to size a software development effort. Verner and Tate (1992) take up the case of a bottom up estimation based on size because size was critical to explaining overall effort. They identify the factors affecting size and obtain equations to predict size based on the explanatory factors. The authors present a method to come up with an overall system size based on the individual size of the subcomponents. The results were accurate to the extent to which knowledge about the system was known.

Expert judgment plays an important role in the development of these models. Miyazaki et al. (1994) build on the model by Verner and Tate to propose improvements in measuring size by using the least squares of inverted balanced relative errors instead of the ordinary least squares method. Hakuta et al. (1997) introduce an estimation model that is independent of program type and is more generic in nature. They introduce the concept of a processing unit. The processing unit is defined as a module that completes a specific function, and the size of the processing unit is estimated based on reference modules. It has adjustments for language level, complexity, and environmental factors. The authors make the case for refinement of size estimates based on the availability of more information.

There is plenty of research that is being done in the field of software estimating, and new models are being developed. The existing research has a gap from a practitioner's perspective in that the models being developed in academic research do not meet the everyday needs of a practitioner. The models are not intuitive enough to be employed in practice. Pfleeger et al. (1997) talk about the gap between measurement research and practice. They talk about software measurement as a broad term, including everything in the software development process that deals with metrics including monitoring the schedule and how managers look for measurable milestones to indicate the project health in terms of effort and schedule deliverables. They refer to the academic researchers as the measurement community and advocate the idea that the measurement community should not remain separate from the mainstream software engineering field. The authors make the case that academic researchers should try to understand the practitioners' needs so that models that are more relevant for the real world can be produced.

Raz and Elnathan (1999) outline an Activity-Based Costing (ABC) approach for software projects. ABC is a two-stage method of allocating overhead costs to various cost drivers at different levels of activity. The authors propose using the ABC method to track overhead costs for software projects. The focus in ABC is on consumption of resources as represented by the activities in the projects, while the traditional work breakdown structure

(WBS) in Waterfall project management groups together work packages according to the resources responsible for execution of that activity.

Symons (1991) defines Mark II function points as a method that identifies and categorizes functional user requirements of the software into three types, inputs, exits, and objects. The functional size of the system is counted based on the count of the individual requirements. Dolado (2000) applies Genetic Programing to estimate the size of the software project. The author proposes various approaches to estimate the size of the project in terms of lines of code and demonstrates a relationship between lines of code and number of components (NOC). The model is a generic model, and the data for project effort was fitted using Mark II function points. MacDonell (2003) uses fuzzy logic modeling to predict effort size and evaluates if it could be used as an alternative to least squares regression. Regolin et al. (2003) uses machine learning (Genetic programming and Neural Networks) to predict the number of lines of code and number of components (NOC). The authors advocate for the use of machine learning techniques in size estimation.

Moløkken and Jørgensen (2003) complete a systematic literature review on software estimation. Two important findings point to the fact that 30-40% overruns are common for most software projects, and the mostly used estimation approach is the expert-based estimation method. Jørgensen (2004a) also completes a systematic review of papers on software development effort expert estimation. The author concludes that expert estimation is the preferred method when estimating for software development projects and that there is no hard evidence to prove the superiority of model estimates over that of the estimates from experts. Jørgensen (2004b) compares expert estimation from a top down perspective and bottom up perspective. The author concludes that expert estimation might work better in a bottom up context when the estimators have less access to recall of similar projects. Jørgensen (2007) completes a systematic review of papers that used expert judgment, formal models, and a combination of both approaches to estimate software projects. The author finds that the models do not perform better than the experts when estimating the overall effort and makes the case for a combination approach in which both expert judgment and models are used. The author suggests that this might be the preferred approach.

Costagliola et al. (2005) present a function point like approach called class point to estimate the size of a project. They consider factors such as number of external methods, number of services requested, and a complexity measure for each class. The class point approach provides a system-level size measure by considering specific aspects of a single class. The authors propose two measures, namely CP1 and CP2. CP1 is used early in development to get a measure of size, and CP2 is used later in the development process when information about the attributes associated with a class is known. Pendharkar (2004) introduces the concept of estimating object-oriented component size using regression. He uses regression tree models to predict object-oriented component size based on factors such as the number of Graphical User Interface (GUI) elements, number of methods, number of subclasses, etc. Kanmani et al. (2007) build on the concept of class points to estimate size of the effort and use neural networks as the estimation method. The authors make the case that the results were comparable to the results obtained from regression models. Wijayasiriwardhane and Lai (2008) introduce the concept of component points, which are like function points to estimate size of a project. They apply it in the context of component-based systems. Zhou et al. (2014) introduce the concept of objective class points to estimate the size of an effort.

Jørgensen (2014) summarizes the state of software estimating based on decades of research in this space. The author points out that there is no “best” effort estimation model or method despite years of research. There are many studies that compare the accuracy of estimation models and methods. The author states that "A major reason for this lack of result stability seems to be that several core relationships, such as the one between development effort and project size, differs from context to context." The author’s findings imply that companies should focus on the local context and try to build their own estimation models. Jørgensen gives a set of recommendations, and one of the main recommendations was to develop and use a simple estimation model that is based on the local context in combination with expert estimation. He also recommends avoiding using early estimates based on highly incomplete information.

A majority of papers reviewed thus far deal with the development of models and measuring the accuracy of models. We found various methodologies and models that predict size as it relates to software estimating, starting with the function points methodology to various other models dealing with the estimation of size. We see variants of the function points methodology, including Mark II function points, class points, component points, objective class points, etc. We also see various methodologies including regression, machine learning, and other predictive methodologies used in the context of predicting size, and there is a lot of literature on comparing the accuracy of the models.

There exists a branch of literature that deals with comparing estimates from experts with those from models. Findings suggest to combine expert-based estimating with model-based estimating. Research also shows a case being made for bottom up estimating as yielding better estimates. There is a need to develop and use simple estimation models that meet the needs of the local context and can be used in conjunction with expert estimation.

A design pattern is a generic solution to a commonly occurring issue in software design and development. Design patterns are useful in that they can speed up the development process in that a solution is often implemented the same way, and design patterns yield themselves to be reused. Design patterns often become a language in which experts in the software development field communicate. Many big organizations approach application development from the perspective of reusable standardized design patterns. There exists a gap in the literature where software estimating is not being viewed from the perspective of design patterns. Every mature organization is going to have its own niche in terms of reusable design patterns, and if we can find a way to identify the core design pattern and the tasks that form the basis for the design pattern, we will have the foundation for a new estimating model. Every design pattern can then be decomposed to a set of tasks or components and scenarios. We can then come up with a stepwise process to come up with a simple estimating model that meets the needs of a local context and one that combines expert estimation in the context of standardized design patterns.

2.2 Agile Release Planning Methodology

2.2.1 Definition of terms for an Agile project execution

CPrime (2013) defines Scrum as a subset of Agile. It is one of the most widely used process framework for Agile development. The Scrum team is self-organizing in that the team comes up with a solution to completing a task, and there is no hierarchy within the team. The projects move through a sequence of iterations or sprints in the Scrum model.

At the outset, an idea for a project is considered and is taken through **design sprints**. The product owner or the business sponsor champions the project and requirements for the application are put together as a collection of stories into software like JIRA. JIRA is one of the more popular software programs used by organizations to organize sprints.

The stories are prioritized in order of importance. Each story is assigned story points when they are put into the system. Usually, more story points indicate more effort is involved in implementing the story. Each team might have its own method of assigning story points to a story. We take a collection of stories that form the crux of an application or deliverable and plan a release or epic. The collection of stories is sometimes referred to as a backlog of items. Long-term planning takes places at the release level. Short term planning takes place at the Sprint or Iteration level.

An **iteration or sprint** is a time window during which development takes place. It usually varies between one to four weeks, and the duration is fixed for a given project. The team decides to execute 2-4-week sprints, and they usually stick to that for the duration of the project. The entire release will consist of a sequence of sprints. When the team plans for a sprint, they pick up the highest prioritized stories from the backlog. The backlog can be viewed as the list of pending stories that need to be completed to deliver the functionality. When stories are prioritized, they can be classified as must have, should have, and nice to have. The must-have stories with the highest priority are usually taken up first for execution.

Stories can be interdependent and related. For example, one story might be dependent on another story, and it often is useful to take up all the dependent stories together.

The **velocity of the team** is the number of story points completed in a sprint. We can obtain a high-level estimate of the remaining work in a project based on the velocity and number of story points remaining to be completed.

There is a **working agreement** for the team. The entire team negotiates and agrees to the working agreement upfront before the project starts. Each time a team member leaves the team, or a new team member enters the team, the working agreement is renegotiated.

At the end of each sprint, we might have some rework based on feedback that then gets prioritized into the planning of the next sprint. At the sprint level, we could look at each story as a task. Each story is divided into sub-tasks and estimated in hours at the iteration planning level.

Pair programming is a methodology where two resources share a single workstation. One person is usually the driver, and the other person is considered the navigator. This methodology is often used when there is a need to bring in a new person on the team or when there is a need to develop a subject matter expert, etc.

Planning poker is one way of assigning story points to stories or hours to tasks. The business sponsor usually explains the intent of the story, and each person on the team picks a number. Usually, the people with the lowest and highest estimate are requested to give justification, and the scrum master then helps the team reach a consensus towards an acceptable number.

2.2.2 Agile release planning methodology literature review

Logue and McDaid (2008) find that release planning is a crucial activity in the software development process. The estimate to develop a particular functionality and the likely return are subject to many uncertainties. In the Agile methodology, a sprint is period of time in which a defined unit of work or specific functionality has to be completed and be made ready for review. The unit of work is usually comprised of a set of user stories. A set of sprints often comprise a Release or an Epic. Decisions need to be made on which stories to include in a sprint and within the planning horizon of a release. This will often require a delicate balance between competing benefits and risks. G. Ruhe and Saliu (2005) identify the characteristics of a good release plan, which includes increasing the overall business value, satisfying the needs of the stakeholders, meeting the resource constraints, and accounting for dependencies between features. Table 2 summarizes the literature on Agile release planning and the related methodologies.

Table 2: Agile release planning methodology research

Author	Agile Concept Addressed	Methodology
G. Ruhe and Saliu (2005)	Release Planning	Multi Objective Optimization problem.
(Cusumano, 2007)	Extreme Programming and Iterative development	Personal Experience, Interviews
(Logue & McDaid, 2008)	Release Planning	Case Study
(Moløkken-Østvold, Haugen, & Benestad, 2008)	Estimating using planning poker	Case Study
Szoke (2009)	Iteration Planning.	Resource-constrained project scheduling optimization problem (RCPSP).
Szoke (2010)	Feature Planning	Optimization Model
(Ktata & Lévesque, 2010)	Estimating user stories	Case Study, Interviews
(Abdel-Hamid & Abdel-Kader, 2011)	Velocity	Case Study
(Da Silva, Martin, Maurer, & Silveira, 2011)	User centered design and Agile methods	Literature Review
(Dong, Yang, Wang, Zhai, & Ruhe, 2011)	Extreme Programming and Iteration planning	Case Study. Knapsack based optimization solution.
(Van Valkenhoef, Tervonen, De Brock, & Postmus, 2011)	Release Planning and Extreme Programming	Knapsack based optimization solution.
Szke (2011)	Release Planning.	Multiple knapsack-based optimization model
(Mahnič & Hovelja, 2012)	Planning poker and user stories	Case Study
(Boschetti, Golfarelli, Rizzi, & Turricchia, 2014)	Sprint Planning	Optimization Model

Author	**Agile Concept Addressed**	**Methodology**
(Grapenthin, Poggel, Book, & Gruhn, 2015)	Task breakdown	Case Study
(Dragicevic, Celar, & Turic, 2017)	Effort Estimation	
(Bilgaiyan, Sagnika, Mishra, & Das, 2017)	Cost Estimation	Literature Review
(Hannay, Benestad, & Strand, 2017)	Earned Business value	Knapsack based Optimization Model.
(Hoda, Salleh, Grundy, & Tee, 2017)	Agile Software Development	Literature Review
(Usman, Mendes, Weidt, & Britto, 2014)	Effort Estimation	Literature Review
(Usman, Mendes, & Börstler, 2015)	Agile Software Development	Survey
Dragicevic, Celar, and Turic (2017)	Estimate Tasks in an Agile Setting.	Bayesian Model
(Usman, Börstler, & Petersen, 2017)	Agile Software Development	Survey

Cusumano (2007) compares extreme programming with iterative development on multiple key concepts. One of the key concepts in the paper is the building of the product in small increments of functionality at regular intervals. The author introduces the concept of releases to deliver functionality in regular and small intervals. The key in this model is to get regular feedback from customers, and it results in an evolving product. Logue and McDaid (2008) complete a case study on a company using data from two projects and provide a method for decision makers to plan out a release in terms of which stories to include in the release. They lay out the release planning process based on requesting users to give minimum, most likely and maximum estimates for each task.

Planning poker is a way of assigning story points to stories or hours to tasks. The business sponsor usually explains the intent of the story, and each person on the team picks a number. The scrum master is a facilitator for an Agile development team. Usually, the people with the lowest and highest estimate are requested to give justification, and the scrum master then helps the team reach a consensus towards an acceptable number. Moløkken-Østvold, Haugen, and Benestad (2008) present a case study on a company to compare the estimate accuracy on a project. They compare the estimates using the planning poker methodology with that of estimates provided by experts or subject matter experts. They find very little difference between the two methods, but there were some tangible side benefits like possible improved code quality, etc. that was perceived from planning poker. The authors conclude that the use of planning poker moderates the effect of optimism and provides more accurate estimates as compared with expert estimation. Ktata and Lévesque (2010) complete a case study using data collected from structured interviews. The authors introduce the concept of technical debt, which is defined as any side of the current system that is considered sub-optimal from the technical perspective. The main contributor to technical debt is the desire of developers to cut corners to meet a deadline, and this results in more technical debt. The authors also research the causes for the errors in estimating user stories. Abdel-Hamid and Abdel-Kader (2011) complete a case study implementing Agile methodologies at five companies. Their study indicates implementing Agile practices results in improved velocity and better project morale. Grapenthin, Poggel, Book, and Gruhn (2015) complete a case study of two large scrum-based projects. They find increasing communication as part of the process before and during the sprint help in better identification of tasks in a timely manner before the planning of a sprint.

G. Ruhe and Saliu (2005) show how different objectives related to implementation and requirements can be optimized using a multi-objective optimization model in the context of release planning. Szoke (2009) approaches the issue of iteration planning as a resource-constrained project scheduling optimization problem (RCPSP). The author identifies the developer as a unique resource and makes the case that the complexity in scheduling arises from various implicit and explicit dependencies around tasks like interdependence between tasks, priority for a task, etc. Szoke (2010) approaches the issue of feature planning and assigning features to teams as an optimization problem. Dong, Yang, Wang, Zhai, and Ruhe (2011) complete a case study at a Chinese software company and approach the problem of assigning user stories in an extreme programming environment as a knapsack problem. They provide an optimization model to help with iteration planning. Van Valkenhoef, Tervonen, De Brock, and Postmus (2011) approach the issue of release planning in an extreme programming environment in terms of a nested knapsack problem. Szke (2011) makes the case that Agile approaches tend to lean towards delivering software incrementally, where there is a sequence of small releases as opposed to delivering the whole system at once. He proposes "a conceptual model for Agile scheduling and a multiple knapsack-based optimization model with a branch-and-bound optimization algorithm for Agile release scheduling." Boschetti, Golfarelli, Rizzi, and Turricchia (2014) complete a case study using data from two projects in Italy and present an optimization model for sprint planning in an Agile environment to assign stories to a sprint. They assign a business value to each story in an approach rooted in the concept of increasing the overall business value. The authors extend the model to use Lagrangian heuristic methods to reduce the solving time for the model. Hannay, Benestad, and Strand (2017) propose a knapsack problem where they maximize business value within a fixed cost. The authors make the case to maximize business value within fixed cost and introduce a new concept called business points, which are assigned to stories and tasks like story points. The authors introduce a new concept called earned business value and propose using the new metric in lieu of functionality delivered by a story.

Table 2.1 compares this research with the other papers in this space that have used optimization models to address similar problems.

Table 2.1: Agile release planning optimization models research

	This Research	G. Ruhe et al.(2005)	Szoke (2009)	Szoke (2010)	Dong et al. (2011)	Valkenhoef et al. (2011)	Szke (2011)	Boschetti et al. (2014)	Hannay et al. (2017)	
Value of a story/Risk associated with a story	✓					✓		✓	✓	
Value of a feature/Theme		✓				✓				
Payrate of Resources	✓									
Efficiency Score at Skill Level	✓									
Iteration/Sprint Planning	✓		✓		✓	✓	✓	✓	✓	
Release/Epic (multiple sprints) Planning	✓	✓	✓				✓	✓	✓	
Team Level Contraint				✓						
Precedence Contraints	✓	✓	✓		✓	✓	✓			
Coupling Contraint (Stories to be done together)	✓	✓			✓			✓		
Incompatible Stories Contraint	✓							✓		
Workload Availability Contraint	✓					✓	✓	✓		
Urgency of a feature		✓								
Resource Estimation	✓									
Skills Matrix	✓									
End Date for Individual Features	✓									
Scheduling of stories	✓		✓				✓	✓	✓	
Scheduling of stories by Skills	✓									
Assignment of Features to Team/Pair of resources				✓	✓					

Numerous systematic reviews of Agile methodologies exist. Da Silva, Martin, Maurer, and Silveira (2011) complete a systematic review of literature dealing with papers that combine Agile methods with user centered design. The authors review 58 papers in this space and propose an integrated approach to incorporate design into the Agile process.

Bilgaiyan, Sagnika, Mishra, and Das (2017) complete a systematic literature review of all papers involving cost estimation in the Agile methodology space. They identify 26 relevant papers and find neural networks and expert judgment to be the most used techniques for estimating projects. Hoda, Salleh, Grundy, and Tee (2017) look at all systematic literature reviews in the Agile space and identify 28 relevant papers. These papers deal with various issues within the Agile Methodology like usability, human and social factors, etc. They conclude that there is no comprehensive study dealing with Agile practices around release planning. Usman, Mendes, Weidt, and Britto (2014) finish a systematic literature review of effort estimation techniques within Agile software development. They identify 20 relevant papers in this space and conclude that the most researched methods in the estimating context within the Agile space revolve around expert judgment, planning poker, and use case points. They also realize that this is an area that has still not been researched deeply. Usman, Mendes, and Börstler (2015) present and analyze a survey of 60 Agile practitioner’s in 16 countries. The authors conclude the most popular estimation techniques are planning poker, analogy, and expert judgment. They also find that story points were the most frequently used size metric. The respondents to the survey felt the dominant trend was towards under estimation in Agile projects. Usman, Börstler, and Petersen (2017) extend the previous study to include Globally Distributed Agile practices and compare results in terms of Effort Estimation techniques. They identify that the main differences were related to the secondary effort estimation techniques. Analogy was the preferred method for distributed Agile practitioners and use case point was the preferred metric for distributed Agile practitioners while function points were the preferred metric for co-located Agile practitioners.

Dragicevic et al. (2017) develop a Bayesian model to help estimate tasks in an Agile setting. The authors propose a model that can be used in the early project phase to predict task effort. They make the case that the model is independent of Agile methods used. The authors validate the model against a database of 160 tasks from real Agile projects, and they found it to be accurate for estimating tasks.

A majority of papers deal with the qualitative aspects of Agile methodology. Most of these papers are case studies and use the survey methodology to assess and understand the qualitative aspects. There are a few papers that deal with extreme programming. Most of the papers mostly deal with issues like various aspects of Agile, like usability, human and social factors, etc. There is no comprehensive study as such that deals with Agile practices around release planning. I found seven papers that deal with planning resources in a release planning context. Most of these papers approach the planning from an optimization context as a packing problem. They approach it as a knapsack problem and address the problem of assigning resources to tasks or stories based on given constraints. There is a gap in the literature where the skills of resources and domain expertise of resources is not considered when assigning resources to stories in a sprint and in a wider context of a release or an epic. There is also a gap in the literature where the results of developing a task-based bottom-up estimating model that is rooted in design patterns used in the company creates the foundation and sets the stage for developing an optimization model that helps in release planning. Agile project management is a technique of delivering software projects and has not been researched deeply from the perspective of optimizing the use of planning resources and schedules while retaining some tenets of traditional Waterfall-based project management such as centralized control by a Project Management

Office (PMO). There is a definite gap in the literature for a model that addresses the need of planning resources in a hybrid model where resources are managed as a pool of resources in the PMO while accounting for economies of scale that would result from accounting for skills and domain expertise of an individual.

2.3 Predictive modeling in the context of estimating effort for open systems web development projects

MacDonell and Shepperd (2003a) argue that the software community faces a significant challenge when it comes to effective resource prediction, and they make the case for accurately and consistently predicting resource requirements for effectively managing software projects. They make the case in 2003, and this continues to be the truth to date. Dejaeger, Verbeke, Martens, and Baesens (2012) find personnel costs are a significant contributor to expenses in the budget of software development companies. Software development companies often do a poor job estimating the number of resources needed to complete a software project. Labor costs today are one of the biggest contributors to cost in Software Application Development. There is limited research in this area that approaches software development cost giving more weightage to the labor costs in comparison to the overall cost. Table 3 summarizes the literature in the context of estimating effort for open systems web development projects.

Table 3: Predictive modeling in the context of estimating effort for open systems web development projects

Author	**Methodology**	**Modeling Technique**
MacDonell and Shepperd (2003a)	Expert Judgment, Case based reasoning and Linear Regression	Case Based Reasoning, Linear Regression
Dejaeger et al. (2012)	Comparative Study	Various estimating methodologies
Reifer (2000)	Based on Web Objects. Similar to COCOMO II	WebMO model for estimating web projects
Mendes and Counsell (2000)	Case Study	Estimation by Analogy
Mendes, Watson, Triggs, Mosley, and Counsell (2002)	Experimental study	Multiple Regression and Case based Reasoning
Mendes, Mosley, and Counsell (2003)	Survey and Case Study using an industry database	Case based reasoning
Baresi, Morasca, and Paolini (2003)	Empirical Study	
M. Ruhe, Jeffery, and Wieczorek (2003a)	Case Study	Combination of Cost Estimation, Benchmarking and Risk Assessment models
M. Ruhe, Jeffery, and Wieczorek (2003b)	Case Study	Validation of the WEBMO methodology
Mendes, Mosley, and Counsell (2005)	Surveys and Case study	Regression
Costagliola et al. (2006)	Experimental study	COSMIC Full Function Point approach
F. Ferrucci, Gravino, and Di Martino (2008)	Case Study	Regression
Filomena Ferrucci, Gravino, Oliveto, Sarro, and Mendes (2010)	Case Study using Industry database	Tabu search and Support Vector regression
Mendes, Abutalib, and Counsell (2012)	Case Study	Expert Elicitation

Author	**Methodology**	**Modeling Technique**
Seo and Bae (2013)	Case Study using industry datasets	Outlier Elimination Models, Regression and Estimation by Analogy
Čeke and Milašinović (2015)	Case Study	Combination of Size and Conceptual models
Turhan and Mendes (2014)	Comparative Study using cross company and single company datasets	Stepwise Regression and Nearest Neighbor filtering.

Reifer (2000) presents a model for web effort estimation that is rooted in specific metrics that applies to web-based development. He introduces the concept of Web Objects (WO) and WebMO as the new estimation model. WO was an extension to the function points approach, and WebMO took its inspiration from the COCOMO II model. The WebMO model uses only nine cost drivers.

Mendes and Counsell (2000) perform a study to estimate efforts for web application development by using analogy. The authors use a tool (ANGEL) to obtain the most optimal combination of variables to predict efforts. They complete an analysis using data from 70 web projects and conclude that analogy-based estimation is a viable method for estimating web projects. Mendes et al. (2002) perform a study on 37 web hypermedia applications to compare the prediction accuracy of case-based reasoning (CBR) techniques and then perform comparative analysis between the better performing CBR technique and other techniques including Stepwise Regression, Multiple Linear Regression and Regression Tree (CART) models. They find the prediction accuracy of Multiple linear and Stepwise regressions was better when compared with the prediction accuracy of the CBR and CART models. Mendes et al. (2003) conduct a study to obtain early size measure for estimation of web application and investigate the prediction accuracy of company specific data with multi-organizational databases. The size measures were expressed using attributes of length, functionality, and complexity. They use 26 projects to research the accuracy of prediction models on company based and multi-company-based datasets. They conclude the prediction accuracy of company specific dataset outperformed multi-company datasets.

Baresi et al. (2003) perform a study to investigate the impact of design efforts on aggregate web development efforts. This study identifies various dependent and independent attributes that impacts design efforts. They use Ordinary Least Squares (OLS) method to measure the impact and conclude that the design phase plays an important role in total effort estimation.

M. Ruhe et al. (2003a) perform effort estimation of web application by using COBRA method (Cost Estimation, Benchmarking, and Risk Assessment), and they find analogy-based estimation performs better in sixty percent of cases. They introduce a new method Web-COBRA to enhance the accuracy of the COBRA model for web projects. They validate their model using 12 web projects from a company in Australia and conclude that their new model outperformed the Ordinary Least Squares (OLS) method. M. Ruhe et al. (2003b) in their subsequent study compare traditional function points and Web Objects using OLS. They complete a case study using 12 web projects, and the results obtained in this study reveal that size expressed in WO were almost 55% more in comparison to FPs, and this difference increases as the complexity of the projects increases. The authors conclude that the results reveal effort estimation by Web Objects (WO) with OLS regression tree produced significantly better estimates and WO outperforms estimates based on expert opinion as well.

Mendes et al. (2005) conduct a study to research web size measures and cost drivers used in early effort estimation of web application development. They validate their results using the Tukutuku database, a database containing information from multiple companies about web application projects. They analyze 67 web application projects and conclude the total number of web pages and number of features and functionality were the two most influential factors for effort prediction.

Di Martino, Ferrucci, Gravino, and Mendes (2007) explain the genesis of the second generation of function points that were defined by the Common Software Measurement International Consortium (COSMIC), and it was known as the COSMIC-FFP (COSMIC Full Function Point). COSMIC FFP was considered to be the second generation functional sizing method. Costagliola et al. (2006) research about the effectiveness of COSMIC function points to estimate for web application development. They use data from 44 web applications developed by students and conclude that the counting of data movements in an application is an important contributor for estimating the effort associated with the development of web applications.

F. Ferrucci et al. (2008) conduct a study to compare the performance of COSMIC FP and Web Objects to predict accuracy in web development efforts. They use data from 15 web applications and conclude that both COSMIC FP and web objects were good methodologies to predict effort size. Filomena Ferrucci et al. (2010) research the effectiveness of Tabu Search in combination with Support Vector Regression for estimating web applications. They use 195 projects from the Tukutuku database to measure the accuracy of the estimated efforts. They compare their results with another study by Mendes that used the same database and conclude that Tabu Search in combination with Support Vector Regression produces better estimates.

Mendes et al. (2012) developed an expert-based Bayesian Network model to estimate web application efforts. They did a case study using a single company dataset and used 22 web projects to build the model. They conclude that the results obtained show that expert-based Bayesian Network models can be used to estimate effort for web development projects.

Seo and Bae (2013) study the effect of outliers in software effort estimation. They research the effect of outliers on the overall estimation accuracy using publicly available repositories like ISBSG. They use estimation by analogy and ordinary least squares for the research and conclude that removing outliers has a positive effect on the accuracy of the estimate.

Čeke and Milašinović (2015) develop a hybrid model by combining COSMIC-FP and Unified Modeling Language (UML). They use data from 19 web-based projects and validate their results using simple linear regression. They infer that their results show the model they developed was appropriate to estimate efforts in the early stages of web development.

Turhan and Mendes (2014) use data from 125 web projects to compare the accuracy of estimates from industry datasets and single-company models. They find stepwise regression performs better in the field of software effort estimation when applied to cross-company or industry datasets. They also suggest and make the case for companies to build their own estimating model using their own data when feasible.

Most of papers reviewed in the context of estimating effort for open systems web development projects again deal with the development of models and measuring the accuracy of models. We see various methodologies and models to predict size that builds on the function points methodology like Web Objects Methodology and COSMIC function points. We see various models that use methodologies like regression, machine learning, Bayesian networks, and there has been one predominant database, the Tukutuku database used by a lot of researchers in this space. There are a lot of papers that again compare estimates from experts with those from models and comparing models from an accuracy perspective. We see a branch of literature that deals with researching web size measures and cost drivers that can be used for early effort estimation of web development applications.

Software development in the early 1990s used to have two major costs, system costs, and labor costs. Over time, the cost of computing in terms of system costs has drastically come down and is no longer a significant contributor to costs. Labor costs today are one of the biggest contributors to cost in Software Application Development. There is a gap in the literature in that there very few attempts have been made to approach software estimating by predicting the labor cost associated with the project and identifying the underlying drivers of labor cost. For example, some of the drivers of cost include the use of temporary labor through hiring contractors to help fill the temporary need of completing a software project, a common practice at many companies. This research creates an estimating model that estimates the total labor hours for a project, identifies the most important explanatory factors that contribute to the overall labor cost, and quantifies the economic impact of resources and other factors on overall labor cost.

2.4 Combining different methodologies and assessing the quality of the estimate

MacDonell and Shepperd (2003a) make the case for combining estimation techniques. They argue there is never one method that is superior to other methods. MacDonell and Shepperd (2003b) in a later study make the case for the use of more than one predictive modeling technique. Combining estimates from different models and methodologies usually sets the stage for combining expert judgment with data science, and this can lead to more robust discussion and overall better estimates. Table 4 summarizes the literature in the context of combining different methodologies and assessing the quality of the estimate.

Table 4: Combining different methodologies and assessing the quality of the estimate

Author	**Methodology**	**Modeling Technique**
MacDonell and Shepperd (2003a)	Expert Judgment, Case based reasoning and Linear Regression	Case Based Reasoning, Linear Regression
MacDonell and Shepperd (2003b)	Case study based on historical project data.	Linear Regression
Mair and Shepperd (2005)	Literature Review	Focused on Regression and Analogy based models.

Author	**Methodology**	**Modeling Technique**
Jørgensen (2007)	Literature Review	Expert Judgment, Formal methods and a combination of these approaches
Bibi and Stamelos (2006)	Literature Review	Machine Learning Techniques
Bibi, Stamelos, and Angelis (2008)	Case Study	Machine Learning Techniques
Hsu, Rodas, Huang, and Peng (2010)	Empirical Research using existing datasets	Combination Forecast using multiple methodologies, including regression, machine learning, etc.
Mittas and Angelis (2010)	Empirical Research	Combination Methodology using regression and estimation by Analogy
Dejaeger et al. (2012)	Comparative Study	Various estimating methodologies
Kocaguneli, Menzies, and Keung (2012)	Case Study	Ensemble Estimates from multiple methodologies
Wen, Li, Lin, Hu, and Huang (2012)	Systematic Literature Review	Machine Learning Based Estimating models
Wu, Li, and Liang (2013)	Combination Methods	Case Based Reasoning
Idri, Amazal, and Abran (2015)	Systematic Literature Review	Analogy based estimating
Idri, Hosni, and Abran (2016a)	Empirical Study	Ensemble of Classical and Fuzzy analogy models.
Idri, Hosni, and Abran (2016b)	Systematic Literature Review	Ensemble Effort Estimation

MacDonell and Shepperd (2003a) compare three estimating techniques using data from the medical records information system. The three techniques chosen were expert judgment, least squares linear regression, and case-based reasoning. The authors make the case for using a combination of techniques based on their conclusion that there was not one superior method. MacDonell and Shepperd (2003b), in a later study, analyze effort distribution in major Waterfall phases across 16 projects. They conclude that expert estimates can be improved by using estimating models that are based on historical data. They also conclude that the use of more than one predictive modeling technique usually led to more accurate estimates. Mair and Shepperd (2005) identify 20 empirical studies through a systematic literature review to compare the relative accuracy levels yielded by regression and analogy methods for effort estimation. However the results were inconclusive in terms of which technique should be preferred. Jørgensen (2007) completes a systematic literature review of papers that compare estimating techniques using expert judgment, formal models and a combination of these two approaches. He concludes that a combined model usually produces a better estimate than an individual model.

Bibi and Stamelos (2006) investigate five machine learning methods, including association rule, bayesian belief network, regression and classification trees, neural networks, and clustering approaches. The authors propose using a decision tree to select the best estimation technique, and they make the argument that the performance of the five techniques can change depending on the dataset and weights assigned to the model features. Bibi et al. (2008) come up with a model combining a couple of machine learning techniques. They combine Association Rules (AR) and Classification and Regression Trees (CART) to create a new conceptual estimation framework. Hsu et al. (2010) makes a case to use linearly weighted combination methods. They make the case to combine methods and argue that this will improve the accuracy of software effort estimation.

Mittas and Angelis (2010) combine Regression and Estimation by Analogy to create a semi-parametric model for Software Cost Estimation. They find an improvement when the combination models were used as compared to individually using regression or estimation by analogy. Dejaeger et al. (2012) investigate 13 different data mining techniques, representing different kinds of models on nine data sets. These techniques include various regression techniques and machine learning techniques. They make the case that their results indicate that data mining techniques make a valuable contribution to software estimation techniques and should be a complement to expert judgment.

A standard machine learning approach is to try multiple methods on the available data and recommend the approach that performs the best. Ensemble learning, on the other hand, improves machine learning results by combining several models. Ensemble methods at a high level combine several machine learning techniques into one predictive model. Kocaguneli et al. (2012) make the case for ensemble methods to combine the estimates from multiple estimators. They make the case for combining various effort estimation methods and argue that this approach performs better in the scenario where there is no single best estimation method. Wen et al. (2012) complete a systematic literature review and identify eight types of machine learning techniques. The techniques identified include Case-Based Reasoning (CBR), Artificial Neural Networks (ANN), Decision Trees (DT), Bayesian Networks (BN), Support Vector Regression (SVR), Genetic Algorithms (GA), Genetic Programming (GP), and Association Rules (AR). They make the case that CBR, ANN, and DT are used most frequently and that both CBR and ANN are more accurate than DT. They also make the case that a machine learning model is more accurate than a non-machine learning model in general. The CBR method is the process of identifying similar projects from the pool of historical projects that most closely match the current project and then setting the stage for deriving cost estimates based on the similar projects. Wu et al. (2013) make the case for a hybrid model combining CBR with particle swarm optimization (PSO) method to estimate the software effort and conclude that the hybrid model outperforms the independent methods.

Idri et al. (2015) conduct a systematic literature review to examine the use of analogy-based software effort estimation (ASEE) techniques. They review 65 papers and conclude that the usage of ASEE techniques led to more acceptable estimates that outperform other prediction models. They also make the case that estimation accuracy is improved when analogy is used in combination with another technique like fuzzy logic, genetic algorithms, the model tree, and collaborative filtering to generate estimates. Idri et al. (2016a) analyze the possibility of improving the estimation accuracy through the usage of fuzzy and classical analogy ensemble techniques. They conclude that "Classical Analogy ensembles outperform solo Classical Analogy techniques, Fuzzy Analogy ensembles outperform solo Fuzzy Analogy techniques and that Fuzzy Analogy ensembles generally outperform the Classical Analogy ensembles." Idri et al. (2016b) complete a systematic literature review of studies on ensemble effort estimation techniques and conclude that in a majority of cases, the ensemble techniques are more accurate than any single model.

Many papers reviewed thus far in the context of combining estimating efforts from different methodologies approach it from the perspective of comparing estimates from an accuracy standpoint. Another stream of literature deals with creating an ensemble of estimates. They conclude that we can always come up with the best combination of effort estimation methods even when there is no best single estimation method and that this approach performs better. We saw earlier that there is stream of literature that deals with bottom up estimating.

There is yet another stream of literature that is rooted in the estimation of efforts based on size. This research is rooted in the use of function points to estimate the size of the project. Many the papers reviewed thus far seem to deal with development of an estimation method using existing datasets and employed cost models based on function points. There is a gap in the literature where there is an opportunity to bring together executive judgment with quantitative modeling. We can try to bring together an ensemble of estimates that brings together a set of models that is rooted in bottom-up estimating and combine it with an ensemble of estimates that is rooted in the estimation of size based on function points. This will give an opportunity for executives to assess the quality of the estimates setting the stage to combine executive judgment with quantitative modeling.

There is a case to be made to develop and use simple estimation models tailored to local or individual context of an organization in combination with expert estimation. There is a gap in the literature where no one has looked at estimating from the outside in perspective of design patterns. Design Patterns have been in been in vogue in the technical space of developing solutions for a long time, but it is not being extended to the concept of estimating software solutions. There is a gap in the literature where estimating is looked at more from the context of understanding the underlying labor cost associated with the project. Labor cost are usually the biggest contributor to costs these days, and there is an opportunity here to understand the effects of various types of resources, including contract labor and employees on the overall cost of a project using real-world data. There are a lot of papers that make the case for an ensemble of estimates to be used, but we found that very little research has been done where both top-down and bottom-up approaches in addition to predictive modeling are brought together in an ensemble of estimates giving an opportunity for executives to combine executive judgment with quantitative modeling. Many papers explain various facets of the Agile methodology, and a few papers dealt with implementing the Agile process using optimization methods. There is an opportunity to develop a resource allocation model in an Agile context that will consider the individual skills of resources. The above-mentioned gaps have formed the basis for my Research Purpose and Contribution as outlined in the next chapter.

Chapter 3: Research Purpose and Contributions

The central theme of this dissertation is to advance the state of the art of estimating software development by approaching it from a new perspective. This research applies both predictive and prescriptive analytics to (1) develop a new estimating tool that produces a suite of estimates, and (2) develop a new optimization modeling framework for better planning and resource utilization in the Agile project environment. The main estimate will be based on the local environment of the company and on reusing the design patterns. This estimate will rely on a subject matter expert to break down the design into manageable slices of functionality or scenarios. The tool aims to provide estimates for these scenarios while at the same time giving the subject matter expert the capability to override an estimate by providing a reason, when deemed appropriate. The optimization modeling framework will set the foundation for making better scheduling and resource assignment decisions, while considering the skill and domain expertise of resources. We now highlight these components of my research contributions in greater detail.

3.1 Advance the state of the art of estimating software development costs using design patterns.

One of the main purposes of this research is to develop a cost estimating model that is based upon reusing design patterns in a bottom-up estimating context that brings consistency across software development in a large service organization. My contribution will be a detailed exploratory case study of a large service company that involves bringing disparate internal and external data sources together to clean, analyze and aggregate data to develop new software cost estimating models for traditional (Waterfall), new (Agile), and hybrid techniques of managing projects and then investigate implementing them as part of the project lifecycle.

One outcome of this research is an estimating tool that explicitly accounts for local design patterns. This research outlines the development of the estimating model in generic terms so that the process can be repeated by other organizations to develop their own version of the estimating tool. This will be a tool that will set the stage to continuously improve estimates at the task level with the following functionalities:

1. Analyze tasks from historical projects at the task level to come up with a set of design patterns that apply in the local context.
2. Validate the design patterns with enterprise architects using thorough interviews.

3. Develop scenarios at the task level that are based on design patterns.
4. Validate the scenarios with the enterprise architects and a selected subset of subject matter experts.
5. Design a survey based on the above-defined scenarios. The purpose of the survey is to get estimates from subject matter experts for tasks that make up the scenarios.
6. Use the survey to get input from subject matter experts who currently estimate software projects.
7. Use the input from the survey in conjunction with actual past estimates to produce baseline estimates at the task level.
8. Develop estimating models that are based on the scenarios developed above that use the baseline estimates for the tasks.
9. Bring together the estimating models into a new estimating tool.

3.2 Quantify the economic impact of resources and other factors on overall labor cost.

The second purpose of this research is to derive a predictive model to estimate the labor hours for a project that will be based on the underlying explanatory factors for the project. Typically, a project can draw on two different types of resources: employees and temporary labor in the form of contractors. Further, the relevant experience of resources can differentiate costs and productivity. This research considers the economic impact of resources and other factors on overall labor cost by developing a two-stage regression model where it predicts the labor cost. The research accounts for possible endogeneity in the model and identifies the instrumental variables that help to predict the values of the endogenous variables in the model.

3.3 Implementing multiple estimating models by combining executive judgment with quantitative modeling

The third purpose of this research is to produce two ensembles of estimating models. The first ensemble will bring together two estimates that are rooted in the new estimating model based on design patterns and one estimate that is based on the two-stage regression predictive model. It is based on subject matter experts completing a design and giving estimates at the task level based on scenarios. This ensemble is based on the bottom-up estimating context.

The first ensemble will consist of three estimates.

1. SME/TL Estimate

2. Recommended Cost

3. Predictive Cost

The second ensemble will rely on the company continuing to use function points to size projects. This ensemble uses more of the top-down approach to size projects. Function points are usually completed early in the design phase. It will consist of the following four estimates.

1. Function Point Baseline Estimate
2. COCOMO II Best Case Estimate
3. COCOMO II Middle of the road Estimate

4. COCOMO II Costlier Estimate

The estimates in the two ensembles will provide an opportunity for the company to combine executive judgment with data science to reach consensus on the final estimate for a project. More importantly, it will set the stage for executives to have discussions with project managers and technical leaders, helping to derive a more balanced estimate.

3.4 Applying prescriptive analytics (optimization) to build a new decision-support framework for Agile project planning

The final purpose of this research is to augment the use of the estimating model to build a resource allocation framework based on an optimization model that takes into consideration the varied skills and domain expertise of resources. The goal is to factor in the concepts of differentiated skill sets and domain expertise to improve the overall efficiency of a team over time and include possible economies of scale in the management of the resource pool. In the Agile release planning models, most of the existing literature deals with how best to increase the business value by allocating user stories to sprints and releases. There is little existing work that incorporates the concepts of considering skill levels, domain expertise, and possible economies of scale. Our initial research sets us up well for this final objective as we know the individual scenario level estimates and task level estimates. We can build on this knowledge by using skill levels for resources for individual scenarios or stories and come up with a model for resource planning in an Agile context where there are many small projects in the pipeline.

Chapter 4: Methodologies

Business Analytics is comprised of three distinct types of methodologies: descriptive, predictive, and prescriptive (Mortenson, Doherty, and Robinson (2015). They define descriptive analytics as statistical methods designed to explore “what happened?”, predictive analytics as methods designed to predict “what will happen next” and prescriptive analytics as Operational Research/Management Science (OR/MS) methods designed to answer, “what should the business do next.” My research will approach the issue of software estimating from all three perspectives.

According to Evans and Lindner (2012), descriptive analytics are the most commonly used and most well-understood type of analytics. The techniques used in this phase help us better understand, visualize the data, and set the stage to extract useful information for understanding the underlying story behind the data. I will use descriptive analytics to summarize the historical project data to identify trends and patterns with better visualization of the data for executives. This research maps the individual tasks in 20 of the most recent projects at ABC Inc. into design patterns, and this forms the basis for the design of the survey. The survey will bes used to elicit expert opinions that is then used to build a bottom-up estimating tool based on design patterns.

Evans and Lindner (2012) explain that predictive analytics examines historical data to detect correlations or relationships in the data, then extrapolates these relationships forward in time to predict what will happen in the future. The authors also mention that we may find relationships in data that are not clear with traditional analyses. We will use predictive analytics models to estimate project costs. The predictive analytics models will lay the foundation for the descriptive analytics work next.

Prescriptive analytics employs optimization to identify the best alternatives to minimize or maximize some objective (Evans and Lindner (2012). They explain the benefit of combining optimization with the mathematical and statistical techniques of predictive analytics to help make better decisions. This research develops a prescriptive analytics model built upon the detailed inputs provided by descriptive and predictive approaches. This results in better allocation of resources in an Agile project setting and helps create more efficient schedules and better manage resources.

This research will develop a mixed integer linear programming (MILP) model to optimally assign resources in an Agile software project development environment. The model considers the multi-skill requirement of each project and matches them with the best available resource with the corresponding skills. The MILP model can be solved by the exact branch and bound (B&B) and branch-and-cut (B&C) approach (Nemhauser and Wolsey (1988), which is readily available in many off-the-shelf solvers, such as IBM ILOG Cplex, Gurobi, Xpress, among others The IBM ILOG Cplex Studio is employed in this research.

4.1 Estimating Tool Development

Models based on function point data are developed for estimation purposes. We also combine executive judgment with the quantitative models.

4.1.1 Function points related predictive models

Simple Linear Regression

The first function points-based model attempts to identify a simple relationship between the function point count of a project and the actual cost of the project. The data consists of a historical database of projects with function point counts and the final labor cost associated with the project.

In this model, the dependent variable is the estimated labor hours for a project, and the explanatory variable is the adjusted function point count of the project. A simple linear regression model is given as follows:

$$y = \beta_0 + \beta_1 x + \epsilon,$$

where β_0 and β_1 are parameters to be estimated in the model, and y is the dependent variable, and x is the explanatory variable. ϵ is the error terms associated with the model. Parlati (2011) mentions that the cost of a single function point can be estimated from past projects. We have a historical database of 80 projects available with actual design and development costs, plus the actual total labor costs associated with the projects.

Two simple models are implemented in this research to understand the cost of a single function point.

$E(Design\ and\ Development\ Cost) = \beta_0 + \ \beta_1 * Function\ Points\ Count$

$E(Total\ Labor\ Cost) = \beta_0 + \ \beta_1 * Function\ Points\ Count$

Multiple Regression

In a multiple regression model, the dependent variable is explained by multiple explanatory variables. The multiple regression model can be written as follows:

$y = \ \beta_0 + \ \beta_1 x_1 + \beta_2 x_2 + \ \beta_3 x_3 + \cdots + \beta_n x_n + \ \epsilon,$

where β_0, β_1, β_3 $\ldots .\beta_n$are parameters in the model, and ϵ is the error terms associated with the model.

One of the major activities in the planning phase of the project in the organization is the definition of business requirements for the project. The business users define these requirements in elaborate detail, which form the basis for the design phase of the project. There are typically over a hundred requirements associated with each project. One multiple regression model that is implemented to understand the impact of requirements and function point counts on the overall labor cost associated with the project can be written as:

$E(Estimated\ Cost) = \beta_0 + \ \beta_1 * Function\ Points\ Count + \ \beta_2 * \#\ of\ Requirements$

The function point count consists of five main components. One may view it as the partitioning of the overall count. Longstreet (2002) defines the five major components of function points as follows:

Internal Logical File (ILF) refers to user identifiable group of data than can be grouped together. This group of data resides entirely within the application boundary. External

Interface File (EIF) is a user identifiable group of logically related data that resides entirely outside the application boundary and is maintained in an ILF by another application. This data is used by the application for reference purposes only. The ILF and EIF are referred to as data functions within function points.

It is valuable to understand the impact of ILF and EIF separately on the overall labor cost via the following multiple regression model:

$E(Estimated\ Cost) = \beta_0 + \beta_1 * \#\ of\ ILF + \beta_2 * \#\ of\ EIF$

The other three components associated with function points are referred to as transaction function and are as defined by Longstreet (2002) as given below. External Input (EI) refers to a process in which data crosses the boundary from outside to inside. External Output (EO) refers to a process in which data crosses the boundary from inside to outside. The data typically creates reports or output files. External Inquiry (EQ) refers to a process in which data is retrieved from one or more internal logical files and external interface files. It involves both input and output components.

It will be valuable to understand the impact of the transaction functions on the overall labor cost. This research employs the following three such models to study the impact of the transaction functions on the overall cost.

$E(Estimated\ Cost) = \beta_0 + \beta_1 * \#\ of\ EQ + \beta_2 * \#\ of\ EI + \beta_3 * \#\ of\ EO$

$E(Estimated\ Cost) = \beta_0 + \beta_1 * \#\ of\ EQ + \beta_2 * \#\ of\ EI + \beta_3 * \#\ of\ EO + \beta_4 * \#\ of\ EIF + \beta_5 * \#\ of\ ILF$

$E(Estimated\ Cost) = \beta_0 + \beta_1 * \#\ of\ EQ + \beta_2 * \#\ of\ EI + \beta_3 * \#\ of\ EO + \beta_4 * \#\ of\ EIF + \beta_5 * \#\ of\ ILF + \beta_6 * \#\ of\ requirements$

4.1.2: Two Stage Least Squares (2SLS) model and Decision Tree Analysis

The two stage least squares model relates to the research contribution explained in Section 3.2 with the purpose of quantifying the economic impact of resources and other factors on the overall labor cost. The two stage least squares (2SLS) technique is an extension of the ordinary least squares (OLS) method. It is used when the dependent variable's error terms are correlated with the independent variables. We have to take into account the possible existence of endogenous variables in our model. Nagler (1999) defines endogenous variables as variables that are functions of other variables present in the system.

In a multiple regression model, the dependent variable is explained by multiple explanatory variables. The goal of this research is to predict the total labor hours that will be needed for a project. The dependent variable, in this case, will be total labor hours. The following could be additional explanatory variables that can be considered in the regression models.

1. Project Type
2. Number of tasks
3. Number of tasks by each phase, design, development, testing, implementation, and planning
4. Number of resources
5. Number of contracting resources considering experience of resources
6. Number of associates
7. Percentage time spent on design
8. Percentage time spent on development
9. Percentage time spent on planning
10. Percentage time spent on verification/testing by IS
11. Percentage time spent on meetings
12. Percentage time spent by employees/contractors on the project
13. Number of defects by category (sev1, sev2, sev3, and sev4)

14. Number of requirements
15. Number of test plans
16. Area doing the project
17. Function point information
18. Duration of the project
19. Project type

One example of the model can be written as:

$E(Labor\ hours) = \beta_0 + \beta_1 * \#\ of\ employees + \beta_2 * \#\ of\ contractors + \beta_3 * \#\ of\ development\ tasks + \beta_4 * Time\ spent\ in\ testing + \beta_5 * Area\ doing\ the\ project$

In our model above, it is quite possible that the variables ***# of employees*** and ***# of contractors*** are functions of other variables and hence could be endogenous variables.

We could have a sample model as outlined below that estimates the number of contracting resources

$E(\#\ of\ contractors) = \beta_0 + \beta_1 * \#\ of\ experienced\ contractors + \beta_2 * \#\ of\ Total\ tasks + \beta_3 * \#\ of\ requirements$

Nagler (1999) observes that when a variable is endogenous, it will be correlated with the disturbance term resulting in the violation of the general model assumptions and make the Ordinary Least Squares (OLS) estimate biased.

This endogeneity calls for the need of an instrumental variable, which is another variable used in regression analysis to deal with endogeneity in the model.

The multiple regression model is given as follows

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n + \epsilon,$

where β_0, β_1, $\beta_3 \ldots . \beta_n$are parameters in the model, ϵ is the error terms associated with the model.

Suppose $x_1$is an endogenous variable (as a function of x_2 and x_3) and define IV_1 as an instrumental variable such that:

$$\widehat{x_1} = \gamma_0 + \gamma_1 IV_1 + \gamma_2 x_2 + \gamma_3 x_3 + \nu$$

We can then plug in the fitted values of $\widehat{x_1}$ into the original linear regression equation

$$y = \beta_0 + \beta_1 \widehat{x_1} + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n + \nu,$$

where ν is a composite error term that is uncorrelated with $\widehat{x_1}, x_2 and\ x_3$.

My work constructs the appropriate instrumental variables to explain the endogenous variables. We then use two stage least squares regression method as outlined by Kmenta (2011) to estimate the labor hours needed for a project. We also run the diagnostic tests as outlined by Kmenta (2011) to validate the model.

Decision Trees

The Decision Trees based model relates to the research contribution explained in Section 3.2 with the purpose of quantifying the economic impact of resources and other factors on overall labor cost.

Bernard (2015) shows how decision tree models may be used for classification of occurrences into prespecified groups, for prediction of values of a dependent variable based on values of independent variables, and for data exploration in model building. The author also shows why decision trees have several advantages over other models, which include the capability of handling nonlinear relationships between variables, insights into input /

output relationships via data partitioning, estimated risk factor contained in each path of the tree, and the intuitive output. Various decision tree algorithms, including Classification and Regression Tree (CART) and Chi-square Automatic Interaction Dedication (CHAID), build and prune decision trees in differing ways.

Gray and MacDonell (1997) show how decision trees can be used to predict testing time for software development. The authors implement the decision trees that built upon variables such as program length, development time, number of screens, mean testing time, etc. This is an example of the Classification and Regression Tree (CART), where the algorithm creates binary trees by splitting records at each node. Bernard (2015) mentions that CHAID creates wider, non-binary trees often with many terminal nodes connected to a single branch, and automatically prunes the decision tree to avoid over fitting of the model.

This research applies decision trees, using the main contributing explanatory variables to predict labor hours needed for a project. The explanatory variables that will be used to fit the tree include variables such as the number of employees used in a project, the number of contractors used in a project, the number of experienced contractors used in a project, the time spent in each phase of the project including in planning, design, development, testing, implementation and post-implementation, the number of tasks in each phase of the project, meeting time, etc. This research will not use the results from the decision tree analysis to estimate labor hours but rather use the results to better understand the underlying data and the most important explanatory variables. Results from the decision tree approach are then used in conjunction with the results from the multiple regression model to understand and validate the most important explanatory variables to explain the total labor hours in the underlying data.

4.1.3 Function points related suite of estimates

The creation of the suite of estimates relates to the research contribution explained in Section 3.4 with the purpose of providing an opportunity for the company to combine executive judgment with data science to reach consensus on the final estimate for a project. The function point repository was split by function point ranges. The ranges that were used are 0-100, 100-200, 200-300, 300-400, 400-500, and above 500 function points.

This research created a new Excel based tool to calculate the suite of estimates based on the function point count for each individual project. These estimates were calculated for all projects in the repository, and the difference of each estimate from the actual cost of the project was also calculated. We then calculated the average for each of these differences across all function point ranges, as defined above. Based on the average difference in each range, a determination was made as to which method offered the better estimate in each range based on the average difference for each estimate.

4.1.4 Development of the task based estimating model

The development of the task based estimating model relates to the research contribution explained in Section 3.1 with the purpose of advancing the state of the art of estimating software development costs using design patterns and incorporating continuous improvement over time.

Expert Feedback and Surveys

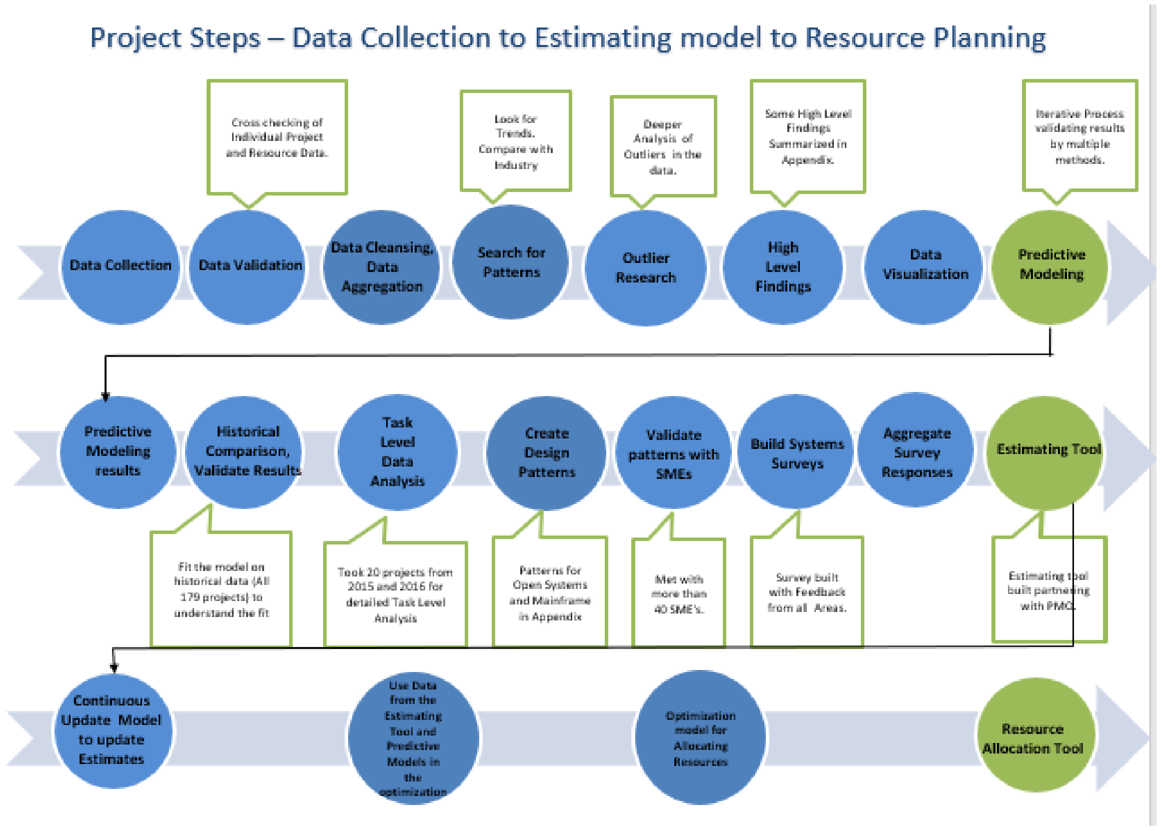
Jørgensen (2007) finds that expert estimation is one of the primary strategies when estimating software development effort and concludes that there is little evidence to support the superiority of model-based estimates over the estimates of experts. He also develops a combined model with estimates from both models and experts. Jørgensen, Boehm, and Rifkin (2009) find that expert judgment-based effort estimation methods lead to more accurate effort estimates than using sophisticated formal models. They also mention that bringing more structure to the estimation process such as introducing experience-based estimation checklists and a structured group process will further improve the estimation quality. My goal in this research is to leverage feedback from experts in the organization and bring consistency to the estimation process. Moløkken and Jørgensen (2003) complete a systematic review. They observe that expert estimation is the most frequently used method and that there was no evidence to show that the use of estimation methods based on models led to better estimates. They also find that project overruns are frequent, but at the same time major overruns were a rarity. Jenkins, Naumann, and Wetherbe (1984), Lederer and Prasad (1995), Moløkken-Østvold et al. (2004) use surveys to address the topic of estimation accuracy. Lederer and Prasad (1992), Moløkken-Østvold et al. (2004) address the issue of which estimation approach is better. Jørgensen and Shepperd (2007) conduct a systematic literature review of software estimation studies and find that nine percent of the studies reviewed used surveys as the primary research methodology. This research employs a combination of expert feedback and surveys to come up with baselines for the estimation tool.

This research study builds and analyzes responses from a survey as outlined below.

1. Understand the data by taking up to 20 of the latest projects that were executed by the organization.
    a. Look at the individual tasks used in each phase of the project while paying more attention to the design and development tasks.
2. Look for design patterns in the tasks and visualize the design patterns.
    a. We anticipate identifying multiple design patterns for the organization based on the type and nature of work being done in the projects.
3. Validate the design patterns with experts (enterprise architects) in the organization.
4. Come up with scenarios to estimate based on the design patterns.
5. Validate the scenarios with experts (enterprise architects) in the organization.
6. Convert the scenarios to individual tasks based on feedback from the historical data.
7. Design a survey to get estimates at the task level based on the scenarios in the design patterns accounting for complexity at the task level.
8. Validate the surveys with the experts (enterprise architects) in the organization.
9. Send out the survey to all the technical leads or subject matter experts in the organization.
10. Collect the responses from the survey.
11. These responses will be validated again with the experts and will form the baseline for bottom-up estimates in the estimating tool.

All the project steps are outlined in a visual representation in Figure 1. The diagram below shows all the steps that were undertaken as part of the research in working with the organization.

Figure 1: Project Steps – Data Collection to Estimating Model to Resource Plannning


Project Steps – Data Collection to Estimating model to Resource Planning
Data Collection
Cross checking of Individual Project and Resource Data.
Data Validation
Data Cleansing, Data Aggregation
Look for Trends. Compare with Industry
Search for Patterns
Deeper Analysis of Outliers in the data.
Outlier Research
Some High Level Findings Summarized in Appendix.
High Level Findings
Data Visualization
Iterative Process validating results by multiple methods.
Predictive Modeling
Predictive Modeling results
Historical Comparison, Validate Results
Fit the model on historical data (All 179 projects) to understand the fit
Task Level Data Analysis
Took 20 projects from 2015 and 2016 for detailed Task Level Analysis
Create Design Patterns
Patterns for Open Systems and Mainframe in Appendix
Validate patterns with SMEs
Met with more than 40 SME's.
Build Systems Surveys
Survey built with Feedback from all Areas.
Aggregate Survey Responses
Estimating Tool
Estimating tool built partnering with PMO.
Continuous Update Model to update Estimates
Use Data from the Estimating Tool and Predictive Models in the optimization
Optimization model for Allocating Resources
Resource Allocation Tool

4.1.5 Estimating tool requirements:

The requirements for the estimating tool were:

1. The design patterns had to be reflected in the tool.
2. The tool had to be user friendly and intuitive to use.
3. The tool had to provide the capability to create a bottom-up estimate that was rooted in the design patterns of the organization.

4. The tool had to be capable of creating a subject matter expert estimate and a recommended estimate. The subject matter expert estimate also had to form the basis for the predictive model estimate.
5. The above mentioned two estimates were part of a suite of estimates.
6. The tool had to accommodate the capability to enter function points. The function point count would then form the basis for four estimates.
7. All seven estimates would then be part of an estimates comparative tab in the tool. This would give the capability to combine executive judgment with data science.
8. The tool had to have the capability to partition the overall cost by high level features and requirements.
9. The tool had to have the capability to upload the final estimate to the time booking system for tracking purposes.
10. The tool had to have the capability of producing a running total each time a task was added.
11. The tool had to have the capability to keep track of time by each phase of the project and compare it to historical averages for the organization and each individual area.
12. The tool had to have the built-in capability for the subject matter expert to override the recommended cost.
13. The tool had to have the built-in capability to account for project management time and tech lead contingency time and meeting time. This time had to be allocated across phases based on the input of the project manager.

The above-mentioned requirements were implemented as part of the new estimating tool.

4.2 Assessing the quality of the estimate from the estimating tool

There was no existing framework to process the actual data from the execution of the project and compare it with the initial estimation data at the task level. This research created a framework to embed a hidden field containing all attributes of the individual task, including the original estimate with each task in the bottom-up estimate. That hidden field can be pulled out after the execution of the project to compare the actual time with the original estimate. The framework for processing this data was built in Tableau, and it is set to accommodate tasks from all projects as they complete.

The foundation of the estimating tool was based on survey responses from the subject matter experts. There were 14 responses from the open systems SMEs and 14 responses from the server-side SMEs. Each of the SMEs had given their estimates for the tasks that were based on the design pattern of the organization. The estimates for each task were given on a complexity scale of high, medium, and low.

An array of estimates for each of the projects was created using the survey responses from each SME as the basis. A Java program was written to go through each task in the estimate and match it up with the survey responses from the SME's. The Java program produced 14 open systems estimates and 14 server-side estimates for each project based on the assumption that each individual task was being estimated by a SME who completed a survey response. The Java program then matched up every open system's estimate with each of the server-side estimates to generate 196 estimates. Some projects did not have a server-side component, and so each project would have between 14 and 196 possible estimates.

4.3 Optimization model for allocating resources to tasks in an Agile setting

The development of the optimization model furthers the research contribution explained in section 3.4 by applying prescriptive analytics (optimization) to build a new decision-support framework for Agile project planning. The Agile project planning and resource allocation problem (APP-RAP) aims to optimally assign personnel with the appropriate skills to stories in every sprint to maximize the total discounted return of assigned stories. The mixed-integer linear programming (MILP) model can be formulated as follows...

Sets and Parameters

R: Set of potential projects to be completed during the planning horizon

S^r: Set of stories belonging to project r ∈ R

S: Set of all stories across all projects

T: Set of all sprints in the planning horizon

K^s: Set of skills required by story s ∈ S

K: Set of all skills

I: Set of individuals available to be assigned to work on stories

M^r: Maximum number of sprints allowed from the start date to the complete date for project r ∈ R

Π_{sr}:value (in \$) of story s ∈ S in project r ∈ R

ROI^r: ROI (in \$) of a project

$$\Pi_{sr} = ROI^r * \frac{Nominal\ time\ of\ story\ s}{Total\ time\ for\ all\ stories\ in\ project\ r}$$

The nominal time refers to the estimated time for story s in this context.

The ROI is given at the project level. We calculate the value of a story in a project by calculating the contribution of a single story in terms of its estimate as a percentage contribution of the total estimate for all stories in a project

For example, if we have three stories s_1 , s_2 and s_3 in project 1.

p_1 is the nominal processing time for story s_1 and is estimated to take 40 hours.

p_2 is the nominal processing time for story s_2 and is estimated to take 80 hours.

p_3 is the nominal processing time for story s_3 and is estimated to take 80 hours.

Assume the ROI of the Project 1 is \$47,469, and the total estimated time for all stories in project 1 is 540 hours

The value of Story 1: \$47469 * (40/540) = \$3516

The value of Story 2: \$47469 * (80/540) = \$7032

The value of Story 3: \$47469 * (80/540) = \$7032

W_{it}: Workload of individual $i \in I$ in sprint $t \in T$

P_s:The nominal processing time of story $s \in S$

P_{sk} : The nominal processing time for skill $k \in K$ in story $s \in S$
The nominal time above refers to the estimated time for a story and the estimated time for a skill in the story. For example, we have a story that has an estimate of 40 hours. This story is comprised of four skills S1, S2, S3, and S12. The individual estimate to complete those skills is S1(4 hours), S2(20 hours), S3(2 hours) and S12(14 hours).

θ_{ik}: Efficiency score of individual $i \in I$ to work on skill $k \in K$

The efficiency score of an individual for a skill is computed based on the following factors:

1. Total experience of an individual in a skill.
2. The date when the skill was last employed by an individual.
3. The self-rating for the skill by the individual.
4. Rating of the individual on the skill by the SME or by all the other team members

This research applies a weighted score based on the above factors. A higher efficiency score means the individual is better suited for the skill, whereas a lower score close to zero means the individual is completely incapable for the corresponding skill. The lower bound for an efficiency score is 0.01, and the upper bound for an efficiency score is 1.

$\frac{P_{sk}}{\theta_{ik}}$: Time taken by individual i to perform skill k in story s in hours

C_i: Pay Rate per hour of individual i ∈ I

D: Discounting Factor. We use a discount factor of 0.01 in the execution of the model for all runs.

S^{AND}: Set of pairs of stories that must be assigned to the same sprint. It accounts for all stories that have the constraint which must be implemented in the same sprint.

For example, if we have a requirement that stories 1, 2 and 3 must be completed in the same sprint, we would define it as shown below

SAND = [{1, 2}, {2, 3}];

S^{OR}: Set of pairs of incompatible stories. It includes all the stories that cannot be implemented in the same sprint.

For example, the definition below shows a case where stories 1 and 10 have to be executed in different sprints. Similarly, stories 11 and 23 have to be executed in different sprints.

SOR = [{1, 10},{11, 23}];

F: Set of precedent relationships of pairs of stories . For example, when story s need to precede story s$'$, the set $\{s, s'\}$ must be an element of S^{AND} or F.

Assumptions:

Each story has been decomposed into sub-tasks that each require a single skill. Our first assumption is that each sub-task requiring a single skill will be completed in a single sprint. The estimate associated with these sub-tasks is often at or below 40 hours, which is typically the case in real life operations. The second assumption is that all skills/sub-tasks associated with an assigned story are completed in the same sprint. It could often be the case that these sub-tasks can be worked on in parallel, thus this assumption is in-line with what happens in the real world where a story is often decomposed in such a way that it can be completed in one sprint. The time for a single story is 80 hours or less, and this is an assumption that will help avoid carryover stories from sprint to sprint and is in-line with real life operations. It implies that one story including all its sub-tasks can be completed in a single sprint. The final assumption we make is all projects can start at the same time. We have a constraint to enforce the deadline by which a project must be completed. This constraint ensures that all the stories with its associated sub-tasks/skills are available to be allocated to resources.

Decision Variables:

$X_{itsk} = 1$ if individual i is assigned to story s to perform skill k in sprint t

$= 0$ otherwise

$Y_{st} = 1$ if story s is assigned to sprint t

$= 0$ otherwise

Y_{st} is an auxiliary or derived decision variable that denotes whether or not story s is performed in sprint t.

Objective Function:

The total net return is calculated as the difference between the total discounted return and the total staffing cost. The cost was not discounted in the objective function with the assumption that the staffing cost can be treated as sunk cost for the internal employees. The objective function can be easily revised to discount cost if needed.

Maximize the total net return: Total discounted return - Total cost

Maximize

$$\sum_{t \in T} \sum_{r \in R} \sum_{s \in S} \frac{(\Pi_{sr} * Y_{st})}{(1+D)^t} - \sum_{i \in I} \sum_{t \in T} \sum_{s \in S} \sum_{k \in K} C_i * (X_{itsk} * \frac{P_{sk}}{\theta_{ik}}) \qquad (1)$$

Constraints:

The constraint (2) below ensures that each story is assigned to at most one sprint. It is consistent with the possibility that all the stories may not be completed during the planning horizon.

$$\sum_{t \in T} Y_{st} \leq 1 \qquad \forall\, s \in S \qquad (2)$$

The constraint (3) below ensures that each skill is performed by exactly one individual for each story assigned to a sprint. Note that no individual is assigned to a story if the story is not executed in a particular sprint.

$$\sum_{i \in I} X_{itsk} = Y_{st} \qquad \forall\, s \in S, k \in K^s, t \in T \qquad (3)$$

We need to meet the requirements that the maximum workload of each individual cannot be exceeded. The maximum workload of individual *i* in sprint *t* is denoted by W_{it} . The constraint (4) below ensures that the total time spent by individual i to work on the story is less than or equal to the maximum workload of individual i in sprint t

$$\sum_{s\in S}\sum_{k\in K^s}\left(X_{itsk} * \frac{P_{sk}}{\theta_{ik}}\right) \leq W_{it} \qquad \forall\ t \in T\,, i \in I \qquad (4)$$

<u>**Side Constraints**</u>

While Constraints (2) through (4) are the main constraints of the optimization problem, some additional requirements of can be modeled by the incorporating the following side constraints. For example, there might be a requirement that a set of stories must be all be completed in the same sprint. Constraint (5) below ensures that all the stories in the set S^{AND} are assigned in the same sprint. Specifically, for each set γ of stories in S^{AND}, the number of assigned stories must equal the cardinality of γ, i.e., all the stories in γ are assigned.

$$\sum_{s\in\gamma}\sum_{t\in T} Y_{st} = |\gamma| \qquad \forall\ \gamma \in S^{AND} \qquad (5)$$

Constraint (6) below ensures that all the stories in the collection S^{AND} are assigned to the same sprint.

$$\sum_{t\in T} Y_{st} * t = \sum_{t\in T} Y_{s't} * t \qquad \forall\ s, s' \in \gamma, s > s' \qquad (6)$$

Recall that S^{OR} contains the stories that are incompatible with each other, thus only one story from S^{OR} can be assigned in a sprint. The constraint (7) guarantees that at most one story from the set S^{OR} will be assigned to a sprint.

$$\sum_{s \in \gamma} Y_{st} \leq 1 \qquad \forall\, \gamma \in S^{OR}, t \in T \qquad (7)$$

We often have time-dependency constraints, i.e., precedence relationships, where story s needs to be completed before story s'. These set of stories are contained in the set F. The constraint (8) below ensures story s is assigned before story s'.

$$\sum_{t \in T} Y_{st} * t \leq \sum_{t \in T} Y_{s't} * t \qquad \forall\, (s, s') \in F \qquad (8)$$

We have a need to consider the constraint for the deadline on makespan of the project. We introduced a new input parameter M^r, i.e., the maximum number of sprints allowed per project from the start date to complete the project for each project $r \in R$. For example, if a project's end date is two months out, M^r would have a value of three assuming each sprint is three weeks long. Similarly, if the project's end date is one month out, M^r would have a value of two. The constraint (9) below ensures all stories s^r belonging to a particular project r will be completed before the end of M^r which is the maximum number of sprints allowed to complete the projects. This ensures the end date constraint for each project is met.

$$\sum_{t \in T} Y_{st} * t \leq M^r \qquad \forall\, s \in s^r \qquad (9)$$

Chapter 5: Data

The first phase of the research creates a tool for project leaders to get a high-level estimate of effort for a project after the function points are calculated in the early design phase of the project. Data for the analysis were collected from the existing function point repository of projects completed since 2014.

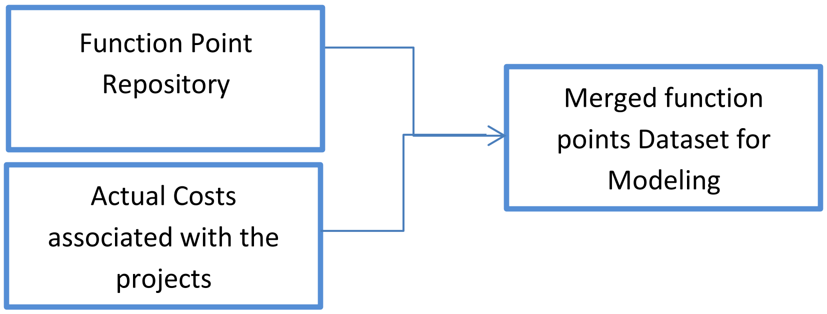
"ISBSG (International Software Benchmarking Standards Group) is a not-for-profit organization. The ISBSG was founded in 1997 by a group of national software metrics associations. They aim to promote the use of IT industry data to improve software processes and products" ("About ISBSG,"). ISBSG has a repository of more than 8000 projects. I gained access to this repository as a doctoral student. A high-level analysis of projects in the relevant comparable industry sector which used the same technology stack as the service company were analyzed, and a project delivery rate of 18 hours per function point was used as the ISBSG benchmark for our analysis.

We had access to multiple data sources at the service company with over seven years of data from the Program Management Office (PMO) and data from the operational excellence area of the company for one part of the research. Real world data is being used in the research, and the models that have been developed have been made part of a custom-tailored estimating tool that is currently being used by the company.

5.1: Data for predictive models

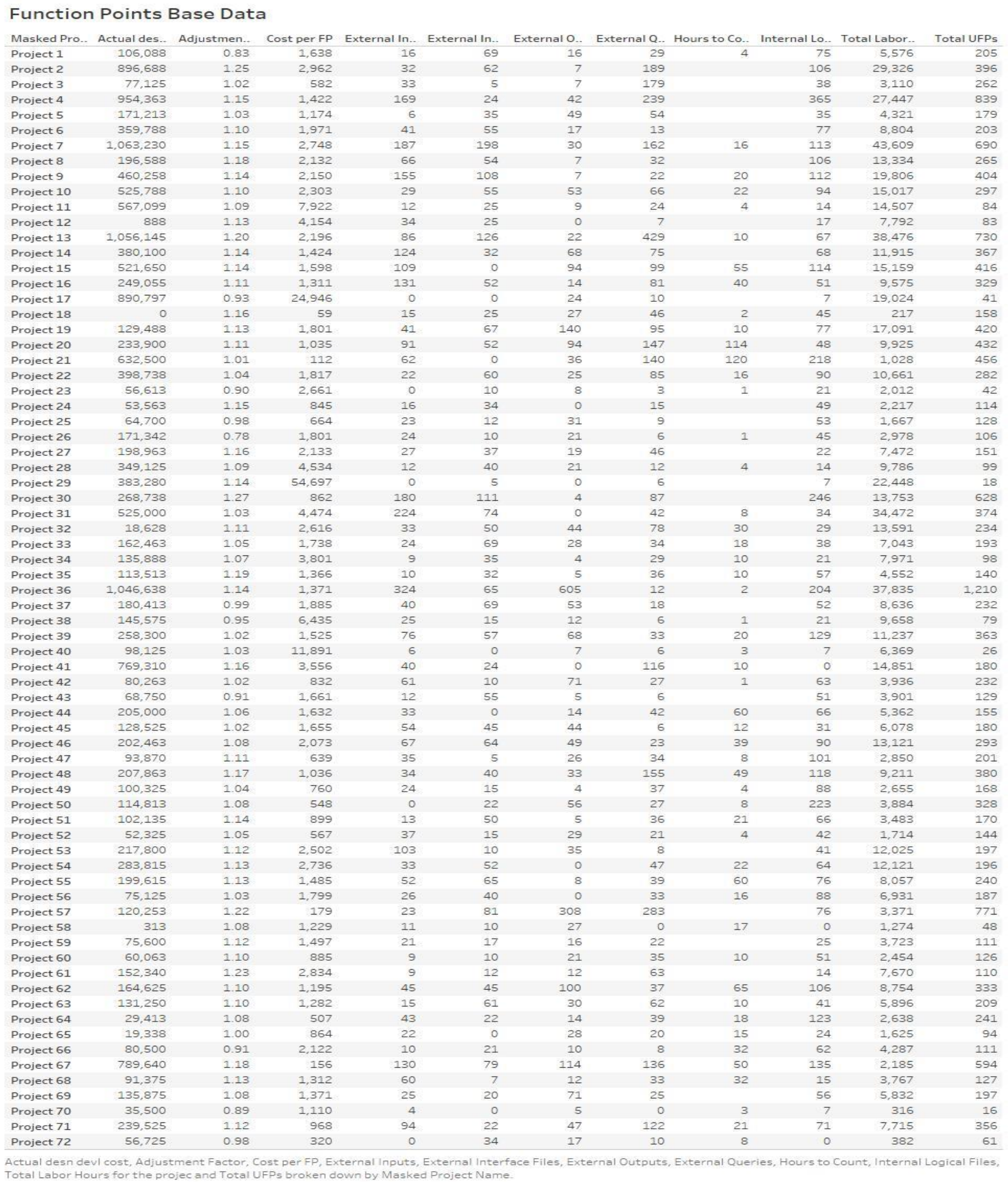
We had access to multiple datasets. The first dataset we had access to was the function point data repository. The organization collects and preserves data on projects that have been through the function points counting process. We were given access to the entire function point repository. It consisted of over 70 projects over the past four years, and estimated costs which were inputs for calculating the counts. We now have access to the actual labor costs associated with the project in a separate dataset. Two datasets are merged for the modeling purpose.

Figure 2: Function points dataset


Function Point Repository
Actual Costs associated with the projects
Merged function points Dataset for Modeling

The key to merging the two datasets was the expense code/project ID associated with the project. We merged the two datasets using the query builder utility in SAS Enterprise Guide. See Table 5 below a few snapshots of the available data. The data has been masked to hide the proprietary information such as expense code, project name, director name, and team name.

Table 5: Function point repository



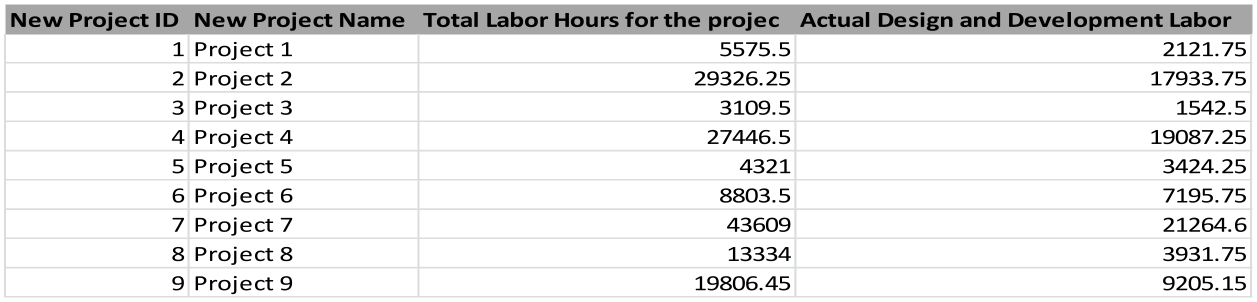
Function Points Base Data

Masked Pro..	Actual des..	Adjustmen..	Cost per FP	External In..	External In..	External O..	External Q..	Hours to Co..	Internal Lo..	Total Labor..	Total UFPs
Project 1	106,088	0.83	1,638	16	69	16	29	4	75	5,576	205
Project 2	896,688	1.25	2,962	32	62	7	189		106	29,326	396
Project 3	77,125	1.02	582	33	5	7	179		38	3,110	262
Project 4	954,363	1.15	1,422	169	24	42	239		365	27,447	839
Project 5	171,213	1.03	1,174	6	35	49	54		35	4,321	179
Project 6	359,788	1.10	1,971	41	55	17	13		77	8,804	203
Project 7	1,063,230	1.15	2,748	187	198	30	162	16	113	43,609	690
Project 8	196,588	1.18	2,132	66	54	7	32		106	13,334	265
Project 9	460,258	1.14	2,150	155	108	7	22	20	112	19,806	404
Project 10	525,788	1.10	2,303	29	55	53	66	22	94	15,017	297
Project 11	567,099	1.09	7,922	12	25	9	24	4	14	14,507	84
Project 12	888	1.13	4,154	34	25	0	7		17	7,792	83
Project 13	1,056,145	1.20	2,196	86	126	22	429	10	67	38,476	730
Project 14	380,100	1.14	1,424	124	32	68	75		68	11,915	367
Project 15	521,650	1.14	1,598	109	0	94	99	55	114	15,159	416
Project 16	249,055	1.11	1,311	131	52	14	81	40	51	9,575	329
Project 17	890,797	0.93	24,946	0	0	24	10		7	19,024	41
Project 18	0	1.16	59	15	25	27	46	2	45	217	158
Project 19	129,488	1.13	1,801	41	67	140	95	10	77	17,091	420
Project 20	233,900	1.11	1,035	91	52	94	147	114	48	9,925	432
Project 21	632,500	1.01	112	62	0	36	140	120	218	1,028	456
Project 22	398,738	1.04	1,817	22	60	25	85	16	90	10,661	282
Project 23	56,613	0.90	2,661	0	10	8	3	1	21	2,012	42
Project 24	53,563	1.15	845	16	34	0	15		49	2,217	114
Project 25	64,700	0.98	664	23	12	31	9		53	1,667	128
Project 26	171,342	0.78	1,801	24	10	21	6	1	45	2,978	106
Project 27	198,963	1.16	2,133	27	37	19	46		22	7,472	151
Project 28	349,125	1.09	4,534	12	40	21	12	4	14	9,786	99
Project 29	383,280	1.14	54,697	0	5	0	6		7	22,448	18
Project 30	268,738	1.27	862	180	111	4	87		246	13,753	628
Project 31	525,000	1.03	4,474	224	74	0	42	8	34	34,472	374
Project 32	18,628	1.11	2,616	33	50	44	78	30	29	13,591	234
Project 33	162,463	1.05	1,738	24	69	28	34	18	38	7,043	193
Project 34	135,888	1.07	3,801	9	35	4	29	10	21	7,971	98
Project 35	113,513	1.19	1,366	10	32	5	36	10	57	4,552	140
Project 36	1,046,638	1.14	1,371	324	65	605	12	2	204	37,835	1,210
Project 37	180,413	0.99	1,885	40	69	53	18		52	8,636	232
Project 38	145,575	0.95	6,435	25	15	12	6	1	21	9,658	79
Project 39	258,300	1.02	1,525	76	57	68	33	20	129	11,237	363
Project 40	98,125	1.03	11,891	6	0	7	6	3	7	6,369	26
Project 41	769,310	1.16	3,556	40	24	0	116	10	0	14,851	180
Project 42	80,263	1.02	832	61	10	71	27	1	63	3,936	232
Project 43	68,750	0.91	1,661	12	55	5	6		51	3,901	129
Project 44	205,000	1.06	1,632	33	0	14	42	60	66	5,362	155
Project 45	128,525	1.02	1,655	54	45	44	6	12	31	6,078	180
Project 46	202,463	1.08	2,073	67	64	49	23	39	90	13,121	293
Project 47	93,870	1.11	639	35	5	26	34	8	101	2,850	201
Project 48	207,863	1.17	1,036	34	40	33	155	49	118	9,211	380
Project 49	100,325	1.04	760	24	15	4	37	4	88	2,655	168
Project 50	114,813	1.08	548	0	22	56	27	8	223	3,884	328
Project 51	102,135	1.14	899	13	50	5	36	21	66	3,483	170
Project 52	52,325	1.05	567	37	15	29	21	4	42	1,714	144
Project 53	217,800	1.12	2,502	103	10	35	8		41	12,025	197
Project 54	283,815	1.13	2,736	33	52	0	47	22	64	12,121	196
Project 55	199,615	1.13	1,485	52	65	8	39	60	76	8,057	240
Project 56	75,125	1.03	1,799	26	40	0	33	16	88	6,931	187
Project 57	120,253	1.22	179	23	81	308	283		76	3,371	771
Project 58	313	1.08	1,229	11	10	27	0	17	0	1,274	48
Project 59	75,600	1.12	1,497	21	17	16	22		25	3,723	111
Project 60	60,063	1.10	885	9	10	21	35	10	51	2,454	126
Project 61	152,340	1.23	2,834	9	12	12	63		14	7,670	110
Project 62	164,625	1.10	1,195	45	45	100	37	65	106	8,754	333
Project 63	131,250	1.10	1,282	15	61	30	62	10	41	5,896	209
Project 64	29,413	1.08	507	43	22	14	39	18	123	2,638	241
Project 65	19,338	1.00	864	22	0	28	20	15	24	1,625	94
Project 66	80,500	0.91	2,122	10	21	10	8	32	62	4,287	111
Project 67	789,640	1.18	156	130	79	114	136	50	135	2,185	594
Project 68	91,375	1.13	1,312	60	7	12	33	32	15	3,767	127
Project 69	135,875	1.08	1,371	25	20	71	25		56	5,832	197
Project 70	35,500	0.89	1,110	4	0	5	0	3	7	316	16
Project 71	239,525	1.12	968	94	22	47	122	21	71	7,715	356
Project 72	56,725	0.98	320	0	34	17	10	8	0	382	61

Actual desn devl cost, Adjustment Factor, Cost per FP, External Inputs, External Interface Files, External Outputs, External Queries, Hours to Count, Internal Logical Files, Total Labor Hours for the projec and Total UFPs broken down by Masked Project Name.

Actual Labor cost dataset: Please see Figure 3 below for a snapshot of the data that was provided. We had information on the total labor hours across all phases of the project and total labor hours for the design and development phases of the project.

Figure 3: Actual labor cost dataset sample



New Project ID	New Project Name	Total Labor Hours for the projec	Actual Design and Development Labor
1	Project 1	5575.5	2121.75
2	Project 2	29326.25	17933.75
3	Project 3	3109.5	1542.5
4	Project 4	27446.5	19087.25
5	Project 5	4321	3424.25
6	Project 6	8803.5	7195.75
7	Project 7	43609	21264.6
8	Project 8	13334	3931.75
9	Project 9	19806.45	9205.15

Description of Individual Variables in the merged dataset:

This dataset contains the following data fields.

a. Project Name: This field has been masked.

b. Expense Code / Project ID: Unique ID associated with the project. This field is been masked.

c. Team Name: Denotes the application development team responsible for the project. This field is been masked.

d. Area Name: Domain area within the Application Development division executing the project. This field is been masked as well. Each area consists of 4-9 individual teams.

e. Total Labor Hours for the Project: The total labor hours across all phases of the project, including planning, design, development, testing, implementation, and post-implementation.

f. Total Design and Development hours for the project: The total design and development time in hours spent on the project.

g. Total Labor Cost: The total labor cost in dollars associated with all phases of the project.

h. Total Design and Development Cost: Total labor cost in dollars associated with the design and development phase of the project.

i. Hours to Count: The total number of hours spent on doing the actual function point count by the application development teams.

j. Total UFPs: Total unadjusted function points.

k. Function Point Count: The function point count is made up of five components, and they are also given in the dataset

    i. Internal Logical Files (ILF)

    ii. External Interface Files (EIF)

    iii. External Inputs (EI)

    iv. External Outputs (EO)

    v. External Queries (EQ)

l. Value adjustment Factor: The adjustment factor is calculated by the team for the project. The value adjustment factor is based on 14 system characteristics that rate the functionality in the application being counted, and these 14 characteristics are given values by the application development team.

m. Adjusted Function Point Count: The unadjusted function points multiplied by the value adjustment factor.

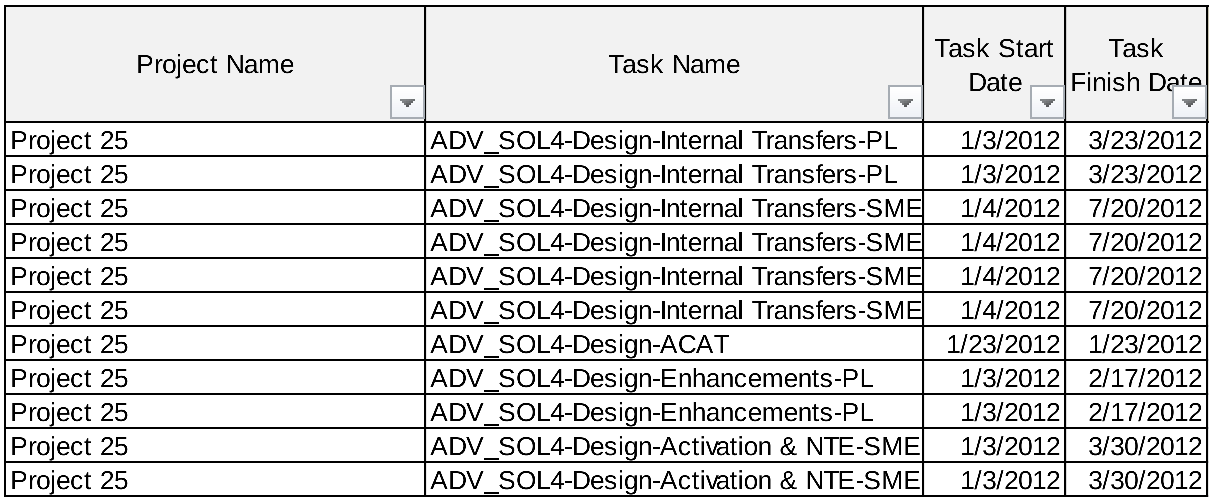
n. Cost per FP: Total labor cost divided by the adjusted function point.

2. Task level data for all projects that had executed in the past seven years. Task level data for each project was aggregated, resulting in nearly 60,000 rows containing information about each task for 400 projects. The task level data contained the following variables in the dataset. The project name and resource names have been masked in the dataset to protect proprietary information.

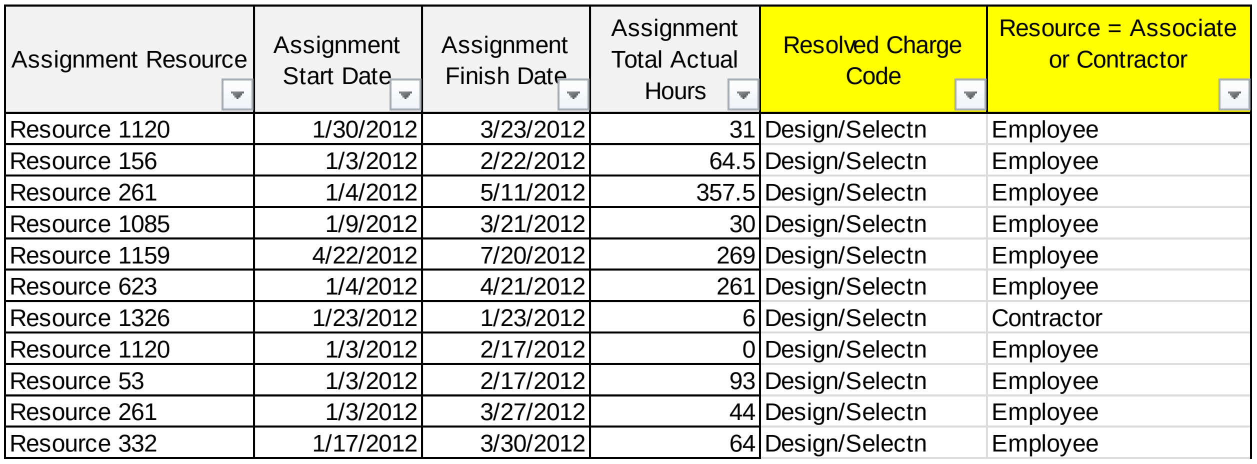
    a. Project Name: This field is been masked.
    b. Task Name: Every project is made up of hundreds of tasks split across all phases of the project. This field is the name given to the task in the time tracking system.
    c. Task Start Date: The date the task was put into the time tracking system.
    d. Task End Date: The date the task was closed in the time tracking system.
    e. Assignment Resource Name: This field identifies the resource who worked on the project. This field is being masked.
    f. Assignment Start Date: The date when the task was assigned.
    g. Assignment End Date: The date when the assigned task was completed.
    h. Assignment Total Actual Hours: The total time in hours taken by the resource to complete the task:
    i. Charge Code associated with the task: Identifies the phase associated with the project. The options were planning, design, development, testing, implementation, and post-implementation.
    j. Resource Type: Identifies if the resource was an employee or contractor.

At the high level, we aggregated the data to compute the following variables at the project level from the provided dataset. We used SAS Enterprise guide to aggregate individual variables from the above-given data and merged them to aggregate information at the project level. A few snapshots of the given data are given below in Figure 4.

Figure 4: Aggregated task level data sample



Project Name	Task Name	Task Start Date	Task Finish Date
Project 25	ADV_SOL4-Design-Internal Transfers-PL	1/3/2012	3/23/2012
Project 25	ADV_SOL4-Design-Internal Transfers-PL	1/3/2012	3/23/2012
Project 25	ADV_SOL4-Design-Internal Transfers-SME	1/4/2012	7/20/2012
Project 25	ADV_SOL4-Design-Internal Transfers-SME	1/4/2012	7/20/2012
Project 25	ADV_SOL4-Design-Internal Transfers-SME	1/4/2012	7/20/2012
Project 25	ADV_SOL4-Design-Internal Transfers-SME	1/4/2012	7/20/2012
Project 25	ADV_SOL4-Design-ACAT	1/23/2012	1/23/2012
Project 25	ADV_SOL4-Design-Enhancements-PL	1/3/2012	2/17/2012
Project 25	ADV_SOL4-Design-Enhancements-PL	1/3/2012	2/17/2012
Project 25	ADV_SOL4-Design-Activation & NTE-SME	1/3/2012	3/30/2012
Project 25	ADV_SOL4-Design-Activation & NTE-SME	1/3/2012	3/30/2012



Assignment Resource	Assignment Start Date	Assignment Finish Date	Assignment Total Actual Hours	Resolved Charge Code	Resource = Associate or Contractor
Resource 1120	1/30/2012	3/23/2012	31	Design/Selectn	Employee
Resource 156	1/3/2012	2/22/2012	64.5	Design/Selectn	Employee
Resource 261	1/4/2012	5/11/2012	357.5	Design/Selectn	Employee
Resource 1085	1/9/2012	3/21/2012	30	Design/Selectn	Employee
Resource 1159	4/22/2012	7/20/2012	269	Design/Selectn	Employee
Resource 623	1/4/2012	4/21/2012	261	Design/Selectn	Employee
Resource 1326	1/23/2012	1/23/2012	6	Design/Selectn	Contractor
Resource 1120	1/3/2012	2/17/2012	0	Design/Selectn	Employee
Resource 53	1/3/2012	2/17/2012	93	Design/Selectn	Employee
Resource 261	1/3/2012	3/27/2012	44	Design/Selectn	Employee
Resource 332	1/17/2012	3/30/2012	64	Design/Selectn	Employee

The aggregated dataset contains the following fields.

- Project ID

- Project Name
- Number of tasks associated with the project
    - Number of tasks by each phase of the project namely design, development, testing, implementation and planning
- Number of resources used in the project
    - Number of contracting resources considering the experience of contractors
    - Number of associates
- Total labor hours spent on the project across all phases of the project.
- Total time in hours spent on design
- Total time in hours spent on development
- Total time in hours spent on planning
- Total time in hours spent on verification/testing by IS
- Total time in hours spent on implementation
- Total time in hours spent on post implementation
- Percentage time spent on design
- Percentage time spent on development
- Percentage time spent on planning
- Percentage time spent on verification/testing by IS
- Percentage time spent on implementation
- Percentage time spent on post-implementation
- Percentage time spent on meetings
- Percentage work completed by employees on the project
- Percentage work completed by associates on the project

- Number of core resources on a project: A core resource was defined as someone who stayed for the full length of the project or contributed
- Number of supporting resources on a project

We also had access to another dataset with summary project-level information. This dataset contained the following variables.

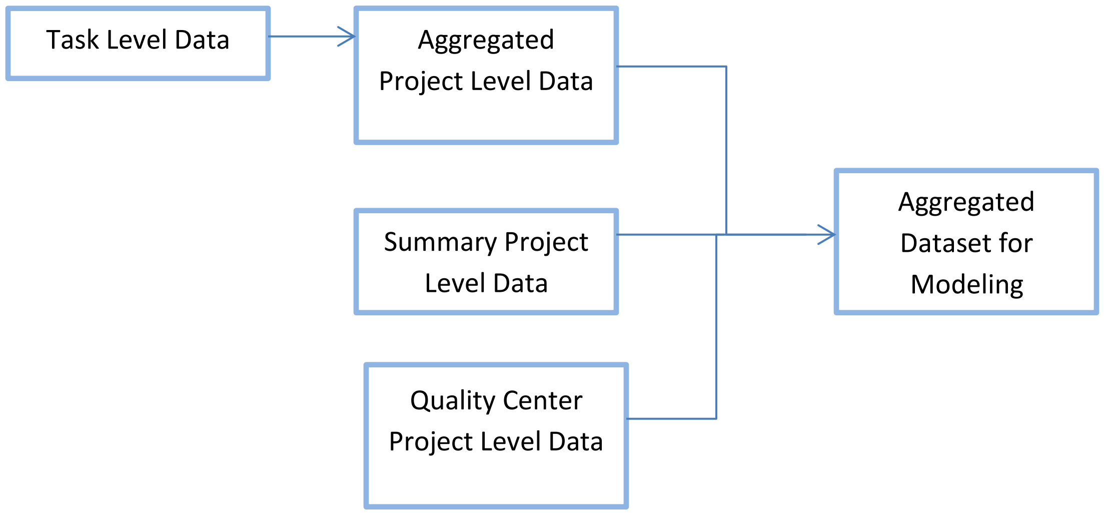
- Project ID
- Project Name
- Project Type: The project types could be Application Development, Vendor related or Infrastructure
- Regulatory Project: Indicates whether the project was executed to meet a regulatory requirement
- Area doing the project: The area within Information Systems that executed the project.
- Project Manager: The project manager who managed the project.
- Project Start Date
- Project End Date
- Duration of the project in days

We needed to build another dataset with aggregate quality information at the project level. We had this data for only 103 projects. This dataset contained the following information.

- Project ID
- Project Name

- Number of requirements in the project
- Number of test plans in the project
- Number of defects by defect severity: Defects could be classified as Sev1, Sev2, Sev3, and Sev4, and we had data for each category.

Figure 5: Flow to get the aggregated dataset for modeling

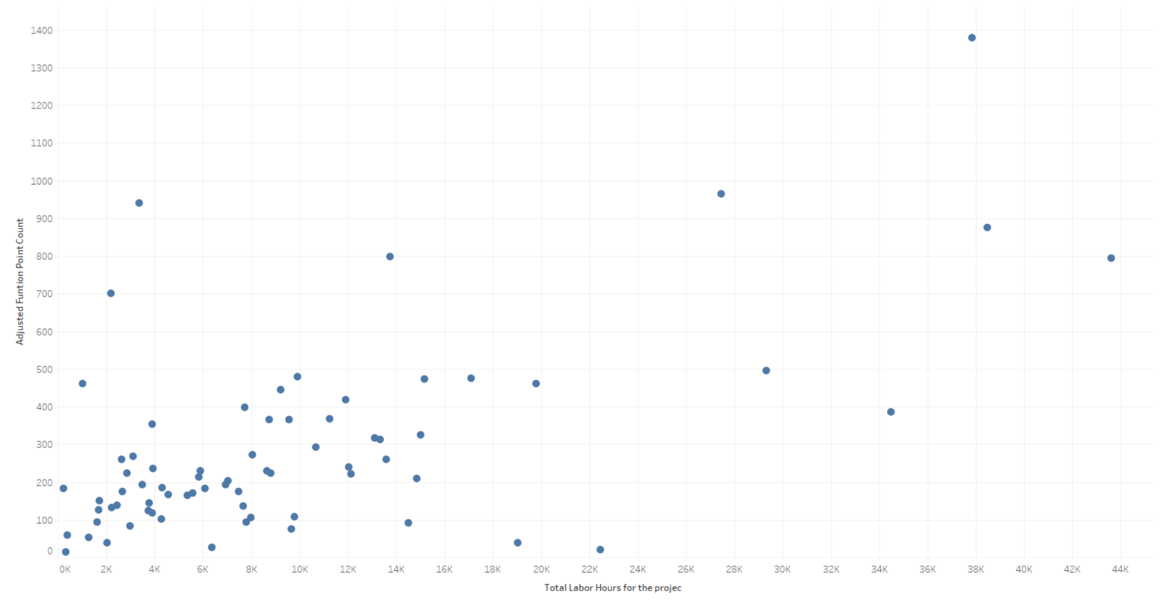

Task Level Data
Aggregated Project Level Data
Summary Project Level Data
Quality Center Project Level Data
Aggregated Dataset for Modeling

High-Level Data Analysis

We started with the analysis of the function points aggregated dataset. The scatter plot of the function point count and the total labor hours for the project is as shown below in Figure 6.

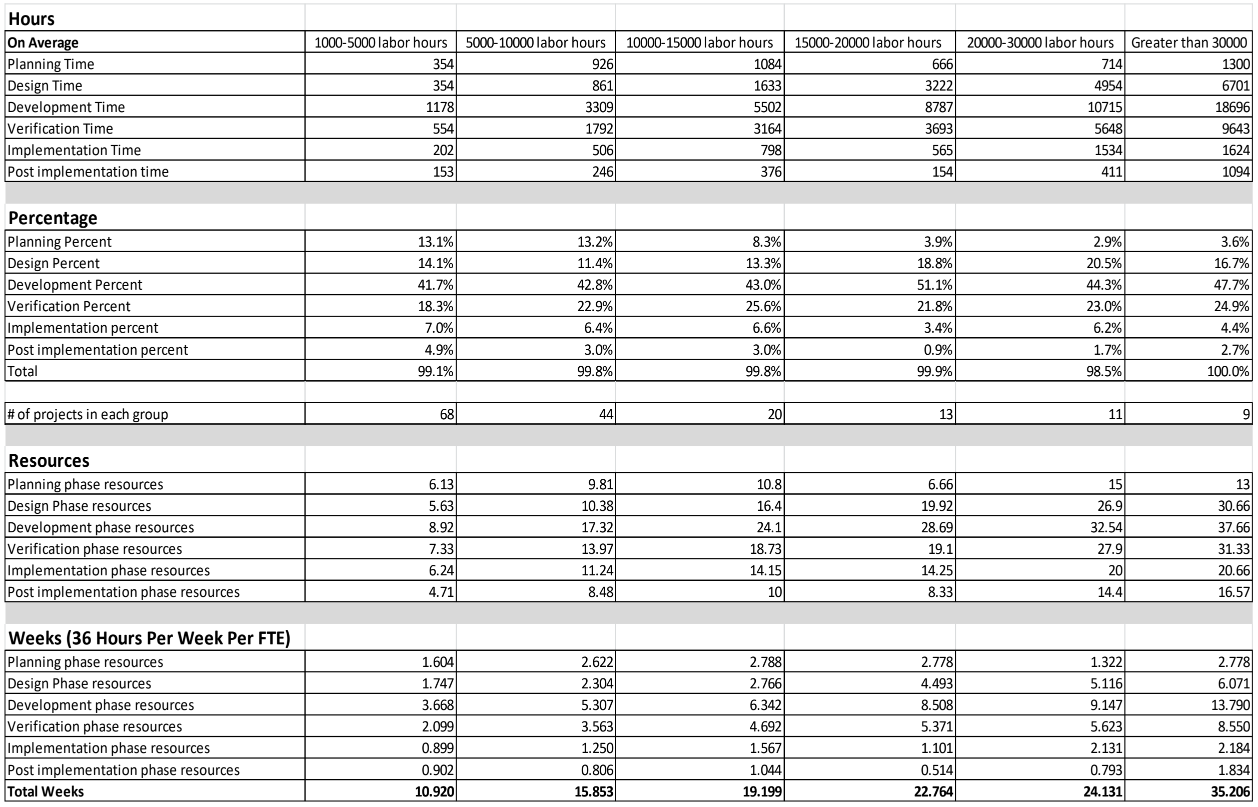
Figure 6: Scatter plot of function point count with total labour hours

Scatter Plot of Adjusted Function Point and Total Labor Hours



At the outset, I divided the aggregated dataset by project size, as shown in Table 6, to better understand the data. This view helped us understand how the time was being spent on average across each phase of the project for different project sizes. This also helped us understand how we were using resources in different categories of projects based on different project sizes. The insight from this analysis was a better understanding of how resources were utilized across the various phases of the project. The resource utilization starts off slowly during the planning and design phases, and it ramps up during development and testing. The utilization drops off gradually during implementation, and only a few resources are kept for post implementation. We analyzed the data further from a descriptive analytics point of view, and those results are shown in Chapter 6.

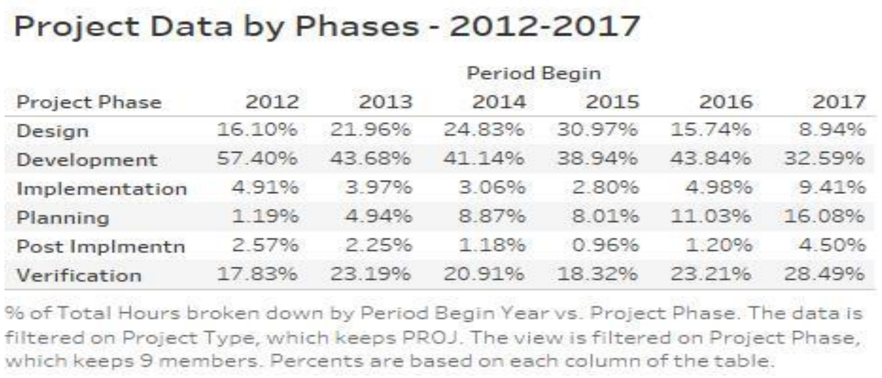
Table 6: High level data analysis of project data



Hours						
On Average	1000-5000 labor hours	5000-10000 labor hours	10000-15000 labor hours	15000-20000 labor hours	20000-30000 labor hours	Greater than 30000
Planning Time	354	926	1084	666	714	1300
Design Time	354	861	1633	3222	4954	6701
Development Time	1178	3309	5502	8787	10715	18696
Verification Time	554	1792	3164	3693	5648	9643
Implementation Time	202	506	798	565	1534	1624
Post implementation time	153	246	376	154	411	1094
Percentage						
Planning Percent	13.1%	13.2%	8.3%	3.9%	2.9%	3.6%
Design Percent	14.1%	11.4%	13.3%	18.8%	20.5%	16.7%
Development Percent	41.7%	42.8%	43.0%	51.1%	44.3%	47.7%
Verification Percent	18.3%	22.9%	25.6%	21.8%	23.0%	24.9%
Implementation percent	7.0%	6.4%	6.6%	3.4%	6.2%	4.4%
Post implementation percent	4.9%	3.0%	3.0%	0.9%	1.7%	2.7%
Total	99.1%	99.8%	99.8%	99.9%	98.5%	100.0%
# of projects in each group	68	44	20	13	11	9
Resources						
Planning phase resources	6.13	9.81	10.8	6.66	15	13
Design Phase resources	5.63	10.38	16.4	19.92	26.9	30.66
Development phase resources	8.92	17.32	24.1	28.69	32.54	37.66
Verification phase resources	7.33	13.97	18.73	19.1	27.9	31.33
Implementation phase resources	6.24	11.24	14.15	14.25	20	20.66
Post implementation phase resources	4.71	8.48	10	8.33	14.4	16.57
Weeks (36 Hours Per Week Per FTE)						
Planning phase resources	1.604	2.622	2.788	2.778	1.322	2.778
Design Phase resources	1.747	2.304	2.766	4.493	5.116	6.071
Development phase resources	3.668	5.307	6.342	8.508	9.147	13.790
Verification phase resources	2.099	3.563	4.692	5.371	5.623	8.550
Implementation phase resources	0.899	1.250	1.567	1.101	2.131	2.184
Post implementation phase resources	0.902	0.806	1.044	0.514	0.793	1.834
Total Weeks	**10.920**	**15.853**	**19.199**	**22.764**	**24.131**	**35.206**

The overall project data from 2012-2017 by phases is summarized below in Table 7. It shows the percentage of time spent each year across the six project phases.

Table 7: Project data by phases



Project Data by Phases - 2012-2017

	Period Begin					
Project Phase	2012	2013	2014	2015	2016	2017
Design	16.10%	21.96%	24.83%	30.97%	15.74%	8.94%
Development	57.40%	43.68%	41.14%	38.94%	43.84%	32.59%
Implementation	4.91%	3.97%	3.06%	2.80%	4.98%	9.41%
Planning	1.19%	4.94%	8.87%	8.01%	11.03%	16.08%
Post Implmentn	2.57%	2.25%	1.18%	0.96%	1.20%	4.50%
Verification	17.83%	23.19%	20.91%	18.32%	23.21%	28.49%

% of Total Hours broken down by Period Begin Year vs. Project Phase. The data is filtered on Project Type, which keeps PROJ. The view is filtered on Project Phase, which keeps 9 members. Percents are based on each column of the table.

We looked at the application development projects executed by each area within the organization, and these are the high-level statistics. These are average numbers for each phase by application development area.

Table 8: Data analysis by project phases

	# of projects	Design % (Variance)	Devl% (Variance)	Test% (Variance)	Plan% (Variance)	Impl% (Varianc
Application Development Area 1	57	11(1.8)	49(4.7)	17(1.5)	12(4.3)	9(1.3)
Application Development Area 2	11	15(3.2)	41(2.8)	18(2.6)	17(4.1)	8(0.5)
Application Development Area 3	19	19(5.1)	45(3.1)	16(1.9)	5.5(0.4)	13(1.7)
Application Development Area 4	37	13(2.9)	39(4.1)	23(2.1)	14(6.5)	9(1.3)
Application Development Area 5	33	16(2.0)	45(3.0)	22(1.8)	4(0.5)	12(1.8)
Application Development Area 6	17	12(1.9)	43(4.5)	26(3.5)	12(4.1)	5(0.3)
Other projects	5	27(3.4)	27(3.8)	16(2.5)	17(8.7)	12(3.8)

We looked at the average duration of a project and how each application development area used the contingent workforce of contractors in their projects. All the numbers indicated below are based on averages.

Table 9: Data analysis by resource type

	# of projects	Avg Duration (days)	# of contractors	# of employees	Contractors percent contribution
Application Development Area 1	57	466 days	7.6	12.7	42%
Application Development Area 2	11	454 days	1.5	13.8	10%
Application Development Area 3	19	488 days	4.6	10.3	35%

Application Development Area 4	37	542 days	5.6	12.7	36%
Application Development Area 5	33	504 days	10	19	43%
Application Development Area 6	17	506 days	6.6	15	35%
Other projects	5	345 days	2.8	12.2	12.4%

We looked at how the contingent workforce was used in 2016, 2015, and in all years prior to that. We found that contractors contributed on average around 34-36% of the overall labor. It was further found that contractors were used more during the development and testing phases. Their contribution peaked during those two phases and declined during the implementation phases. This finding was validated in our conversation with the management at the organization as it was employees who worked on the project during the planning and design phases of the project. They carried the workload during those phases, and the contingent workforce was brought in to help during the development phase of the project.

Table 10: Data analysis based on usage of temporary resources

	# of contractors (on average)	# of employees (on average)	Contractors percent contributio (on average)
2016	5.3	10.3	36%
2015	8.3	16.8	36%
Prior to 2015	6.4	14.2	34%

We had data about requirements, test plans, and defects for 103 projects, and this is the high-level analysis of the data. The numbers provided are based on averages for each area.

Table 11: Data analysis based on quality metrics

	# of projects	Requirements	Test plans	Defects	Defect ratio	Sev1	Sev2	Sev3	Sev 4
Application Development Area 1	35	817	597	326	0.54	53	123	92	40
Application Development Area 2	6	413	213	114	0.53	9	20	14	10
Application Development Area 3	10	609	299	256	0.85	25	129	69	27
Application Development Area 4	21	1480	1641	362	0.22	90	172	74	20
Application Development Area 5	18	1202	1022	413	0.40	65	202	89	36
Application Development Area 6	10	508	1102	213	0.19	31	82	48	18
Other projects	3	311	516	410	0.79	93	161	99	50

5.2: Use case for Agile Software Development epic and release planning

5.2.1: Introduction to optimization use case

Sliger and Broderick (2008) explain that a key feature of the Waterfall approach is the capturing and documentation of all requirements before the design and development of the software. The requirements are typically not subject to change once development starts. The risk in this approach is that testing takes place after development, and because of incomplete requirements, design flaws are uncovered only in the testing phase. The industry saw the need to address some of the inherent risks of the Waterfall methodology. They wanted to address how requirements were defined in advance of project execution and how the finished product is tested post development. The solution was an Agile development process that was rooted in an iterative approach. Variants of the Agile approach like Scrum, Extreme Programming, Pair Programming now exists. Beck et al. (2001) defined a new process for delivering software known as Agile programming, and it was called the Agile Manifesto. The manifesto includes four statements and 12 principles that describe the overall philosophy. Five key principles from among the 12 principles in the manifesto are

1. "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for a shorter timescale.
2. Working software is the primary measure of progress.
3. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

4. Continuous attention to technical excellence and good design enhances agility.
5. At regular intervals, the team reflects on how to become more effective then tunes and adjusts its behavior accordingly."

Agile programming is like Waterfall development in that some of the steps in the workflow are the same. We still collect requirements, design, develop code, test code, and deploy. The main difference is that Agile executes these steps in an iterative approach that fosters continuous improvement of the product. Software requirements are still initially captured but at a less detailed level and documented in a product backlog. These requirements are often captured as user stories, which are requirements from the user perspective.

Hayata and Han (2011) explain that some organizations employ a hybrid model where the best of both models is incorporated. These organizations look to blend Scrum, an Agile method, into traditional plan-driven project development and management. Kuhrmann et al. (2017) make the case that a hybrid software development approach is a combination of Agile and traditional approaches that an organization adopts and customizes to its own context needs. They also argue that hybrid approaches are widely used in practice and found that hybrid approaches have become more prevalent and are used by all types of companies. West, Gilpin, Grant, and Anderson (2011) coined the term “Water-Scrum-Fall” and hypothesized that hybrid development methods would become the standard." Our use case is rooted in a hybrid setting, and it will be based on a hypothetical case where requirements are spelled out, and overall project scopes is defined upfront. The projects will be relatively small with less than 1500 labor hours. The execution of the project could adopt Agile principles such as delivering working software frequently, promoting sustainable development, reflecting on how to become more effective, etc. with some leeway for changing requirements from one iteration to the next.

This research contributes to the literature by presenting a model of a scenario where the estimates that are developed at the task and scenario level are used in an Agile setting to optimize the use of limited resources. Our scenario accounts for economies of scale that can result from managing the skills and domain expertise of resources that are part of multiple projects. This model is important because it can be used in an environment of multiple small size projects to manage a pool of resources given that we always have a backlog of tasks to complete. This scenario will be a case study providing a proof of concept to optimize executing multiple small projects at the same time. The small projects are rooted in the design patterns of the company and will cater to the local context.

5.2.3: Use Case

In addition to executing large enterprise wide projects, ABC Inc. works on a lot of small projects that typically take between 1000-2000 hours. These projects are typically enhancements to existing systems, and there is a perceived decent return of investment in the short term (one year) and long term (3-5 years). These projects are typically seen as efforts that can be completed in a short span of time, but these projects also go through the planning, design, development, testing and implementation phases in quick succession. Today, these projects are completed using the traditional software development model. The resources that work on these projects are typically skilled in 1-2 skills, and the teams are confined to a single domain area of expertise. A project typically will require a good subset of the following set of resources

1. Business Subject Matter Expert (SME)
2. Information Systems SME
3. Business Project Manager
4. Information System Project Manager
5. Design Thinking coordinator
6. User Experience (UX) specialist
7. Front End Developers
8. Back End Developers
9. Quality Assurance Tester
10. Database Administrator

Some of these resources might be needed only for a limited time, and they typically might not be dedicated on the effort. Currently the issue is these projects take substantially more time to complete execution from initiation to finish due to the lack of resources with the right skill set to work on a task at the time when the task is ready to be executed. The resources are often prioritized for higher priority work. The tasks are typically scheduled in a sequence, and the following Gantt chart (Figure 7) shows a hypothetical scenario for project execution.

Figure 7: Project execution with optimal resource allocation

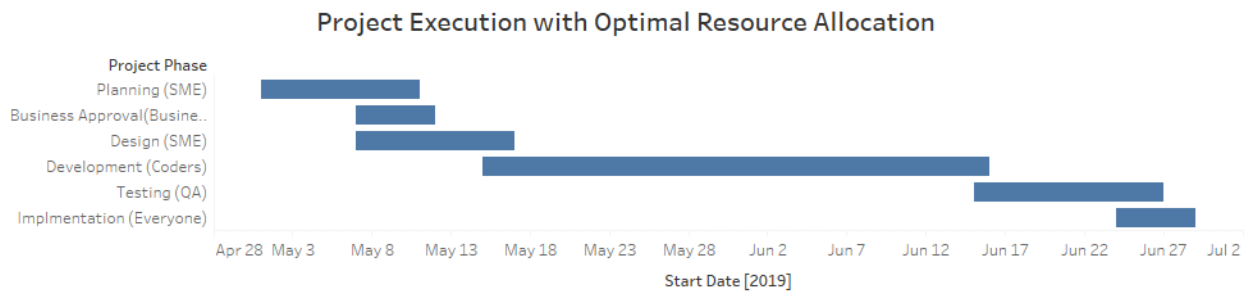
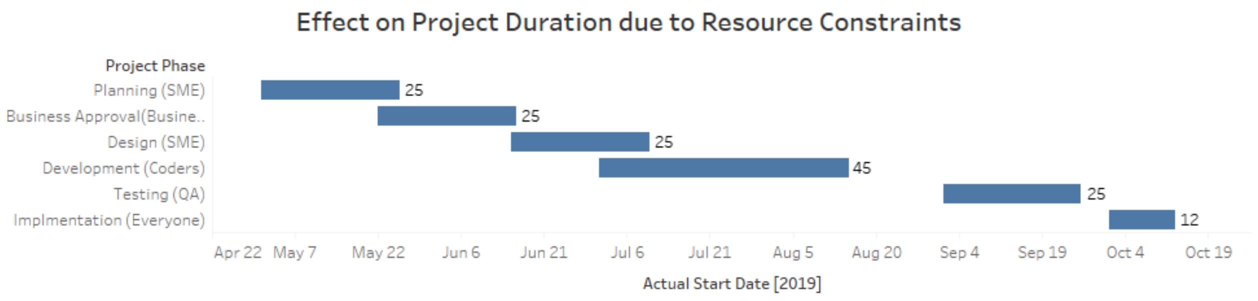


Figure 8: Effect on project duration due to resource constraints



You can observe in the scenario above shown in Figure 8 that a project execution could be delayed by close to three months when it is not prioritized for the effort, and all the scarce resources are allocated across multiple projects. This typically happens in the case of subject matter experts both in the Information Systems space and in the Business areas.

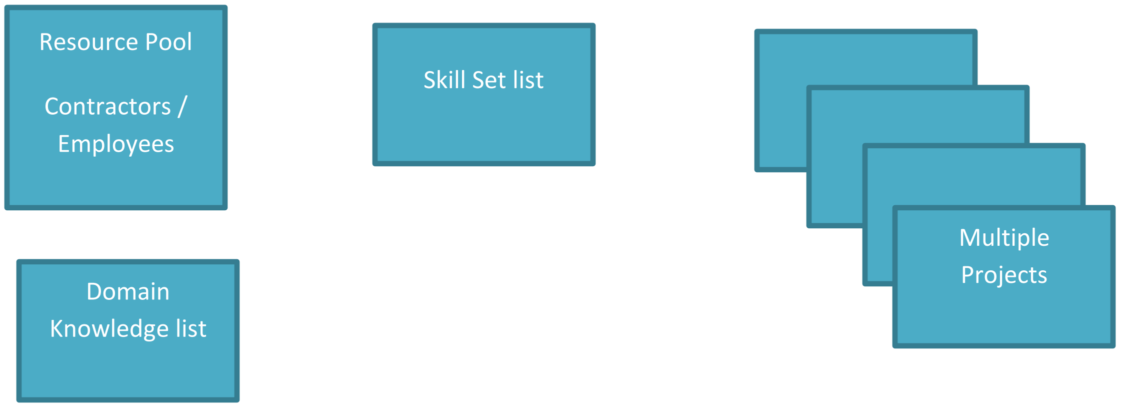
ABC Inc. is also starting to execute some of its projects using the Agile methodology, and these projects would be ideal candidates to be executed using the new approach. ABC Inc. is also starting to cross train employees and is encouraging employees to have development plans that encourage employees to become proficient in multiple skills. The company also has the concept of centers of excellence and is starting to evaluate cross functional teams within the application development area of Information Systems. It will be useful to have a model to evaluate how to plan out a schedule for multiple projects that are very similar in terms of skills needed for the individual stories in the project. There is an opportunity to evaluate the effect of having resources with multiple skills, the effect of cross training employees, the effect of centers of excellence and finally the effect of cross functional teams.

The problem in this traditional method of project execution is that resources, for the most part, are not allocated based on matching of skills needed for a single task. A single task can be decomposed into multiple sub-tasks that each require a skill. A typical example could be the development of a webpage. This task is going to require SME skills, design thinking skills, front-end UX skills, back-end server-side skills, database administrator skills, testing skills and finally implementation skills. Schedules are built, and resources are assigned to tasks based on who becomes available first. It is often the case that a single task might require multiple skills and the entire task as it pertains to a single phase of the project is assigned to the same resource. There is perceived scarcity of certain resources with certain skill sets, and there is no set measure of proficiency in a certain skill for a resource. The gap in the current approach is there is no way to evaluate the effect of cross training resources on multiple skills and to measure the efficiencies it could bring to the overall process. Currently there is no way to evaluate how the schedule would look when you can try to match resources based on each of the skills needed for a task as opposed to allocating them on a first available basis. There is also no way today to incorporate a measure like an efficiency score for a skill, and finally there is no way to evaluate how the schedule would change when the workload availability of resources changes.

To evaluate the model performance, this research addresses a hypothetical scenario where the same set of projects which were executed using the traditional methodology is instead executed in an Agile setting. Each of the tasks is perceived to be a story in an Agile setting, and we have taken the effort to identify all the skills needed to complete each story. I also have a listing of resources with the current set of skills that they have. This data has been transformed to include an efficiency score on the skills for each of the resources. This data is mocked up to show the effect of the efficiency score in such a model. I had access to very high-level work load availability data, and this data is also been mocked up to show the effect of workload availability in the model. I want to consider the effect of matching resources with the right set of skills needed for a story. We want to bring in the concept of efficiency score for a skill for each resource and further want to understand the effect of having resources who are proficient in multiple skills and in measuring the proficiency of resources in a particular skill.

The following high-level diagram explains the core principle that there is a resource pool that can be comprised or contractors and employees who each have a certain skill set. Each project is comprised of tasks that can decomposed into sub-tasks that each require one or more skills for them to be completed.

Figure 9: High-level diagram for use case



The use case is set in a scenario where you have a sequence of small projects in the pipeline that need to be completed. Each of these projects will be less than 1500 hours, preferably and at the maximum can go up to 2000 hours. Each project has a set of stories associated with it across each phase of the traditional development model. These stories have an associated estimated time to complete. Each story also has a set of skills that are required for it to be completed. Each resource also has a set of skills that they are familiar with. Our goal will be to match up the stories with the right resources and pack the maximum possible stories into a sprint based on the right allocation of resources while at the same time ensuring the timely completion of projects.

<u>Project and Resource Information</u>

1. There are multiple projects that need to be started and completed in the next few months. We have data for five projects. Each project consists of multiple phases, as indicated below.

    a. There are stories within each phase.

b. We have an estimate for each story.

c. Each story needs a set of skills, and they are indicated. The estimate for the overall story is further decomposed into the estimate needed for each skill component associated with the story.

d. We are given the ROI information for each project.

Table 12: Project data for optimization use case

Project 1		1st Year - $47,469	3rd Year - $144,456
Sprint	**Stories**	**Estimate**	**Skills Required**
Sprint: BC Development	Story 1:	40	S1(4), S2(20), S3(2), S12(14)
Sprint: Designing Requirements	Story 2:	80	S1(20), S2(50), S19(10)
	Story 3:	40	S2(20), S11 (20)
Sprint: Development & Coding	Story 4:	80	S3(60), S8(20)
	Story 5:	80	S6(80)
	Story 6:	80	S6(80)
	Story 7:	40	S6(40)
Sprint: Testing & Validation	Story 8:	40	S2(10), S9(10), S18 (20)
	Story 9:	40	S17(30), S10 (10)
Sprint: Benefit Analysis	Story 10:	20	S2(10), S12 (10)

Project 2		**1st Year - $150,003**	**3rd Year - $444,965**
Sprint	**Stories**	**Estimate**	**Skills Required**
Sprint: BC Development	Story 11:	70	S1(5), S2(25), S3(5), S12(35)
	Story 12:	70	S12(40), S3(30)
Sprint: Designing Requirements	Story 13:	60	S1(30), S2(30)
	Story 14:	60	S17 (60)
	Story 15:	80	S13(40), S14(40)
	Story 16:	60	S13(60)
	Story 17:	80	S16(80)
	Story 18:	80	S16(80)
	Story 19:	60	S16(60)
Sprint: Development & Coding	Story 20:	*80*	S16(80)
	Story 21:	80	S2(10), S9(70)
	Story 22:	60	S17(30), S10 (10), S18(20)
Sprint: Benefit Analysis	Story 23:	40	S2(20), S12 (20)

Project 3		**1st Year - $117,228**	**3rd Year - $357,275**
Sprint	**Stories**	**Estimate**	**Skills Required**
Sprint: BC Development	Story 24:	20	S1(4), S2(8), S3(2), S12(6)
	Story 25:	60	S2(40), S20(20)
Sprint: Designing Requirements	Story 26:	30	S1(20), S2(10), S19(10)
	Story 27:	80	S2(80)
	Story 28:	80	S11 (80)
	Story 29:	80	S3(80)
	Story 30:	80	S6(80)
	Story 31:	80	S6(40), S7(40)
	Story 32:	80	S7(60), S8(20)
Sprint: Development & Coding	Story 33:	80	S8(80)
Sprint: Testing & Validation	Story 34:	80	S2(20), S9(5), S18 (55)
	Story 35:	40	S17(30), S10 (11)
Sprint: Benefit Analysis	Story 36:	40	S2(25), S12 (15)

Project 4		**1st Year - $5,438**	**3rd Year - $58,181**
Sprint	**Stories**	**Estimate**	**Skills Required**
Sprint: BC Development	Story 36:	100	S1(5), S2(30), S3(5), S12(60)
Sprint: Designing Requirements	Story 37:	80	S1(20), S2(50), S19(10)
	Story 38:	80	S3(80)
	Story 39:	80	S3(70), S5(10)
	Story 40:	80	S13(80)
	Story 41:	80	S13(80)
	Story 42:	80	S13(80)
Sprint: Development & Coding	Story 43:	80	S13(80)
	Story 44:	60	S2(10), S9(10), S18 (40)
Sprint: Testing & Validation	Story 45:	80	S17(80)
	Story 46:	15	S10 (15)
Sprint: Benefit Analysis	Story 47:	40	S2(20), S12 (20)

Project 5		1st Year - $4,238	3rd Year - $39,066
Sprint	**Stories**	**Estimate**	**Skills Required**
Sprint: BC Development	Story 48:	20	S1(2), S2(11), S3(2), S12(5)
Sprint: Designing Requirements	Story 49:	30	S1(10), S2(15), S19(5)
Sprint: Development & Coding	Story 50:	80	S3(60), S5(20)
	Story 51:	80	S7(80)
	Story 52:	80	S7(80)
	Story 53:	80	S7(80)
Sprint: Testing & Validation	Story 54:	40	S2(10), S9(10), S18 (20)
	Story 55:	67	S17(37), S10 (30)
Sprint: Benefit Analysis	Story 56:	10	S2(20), S12 (20)

2. You have a set of skills. We are going to need resources with certain skills for a project based on the needs of a project. We are going to keep the skills generic to cater to the design patterns of the company. The table below gives us information on the skills and indicates which resources possess those skills. We are also given the hourly pay rate for each type of resource.

Table 13: Skills data for use case

Skill	Skill ID	Resource	Pay (hourly rate)
Skill 1	S1	Business SME	xx
Skill 2	S2	Business SME	xx
Skill 3	S3	IS SME	xx
Skill 4	S4	IS SME	xx
Skill 5	S5	Developer	xx
Skill 6	S6	Developer	xx
Skill 7	S7	Developer	xx
Skill 8	S8	Developer	xx
Skill 9	S9	Developer	xx
Skill 10	S10	IS SME	xx
Skill 11	S11	Legal Analyst	xx
Skill 12	S12	Project Manager	xx
Skill 13	S13	Developer	xx
Skill 14	S14	Developer	xx
Skill 15	S15	Project Manager	xx
Skill 16	S16	Developer	xx
Skill 17	S17	IS SME	xx
Skill 18	S18	Business SME	xx
Skill 19	S19	Project Manager	xx
Skill 20	S20	Design Thinking	xx

3. You have a resource pool comprising of associates and contractors. A typical project will need one Architect or Subject Matter Expert (SME) for the estimating, one SME to run the project/answer questions, and a bunch of other resources based on skill sets. The set of resources as indicated in the table above are shown below in Table 14.

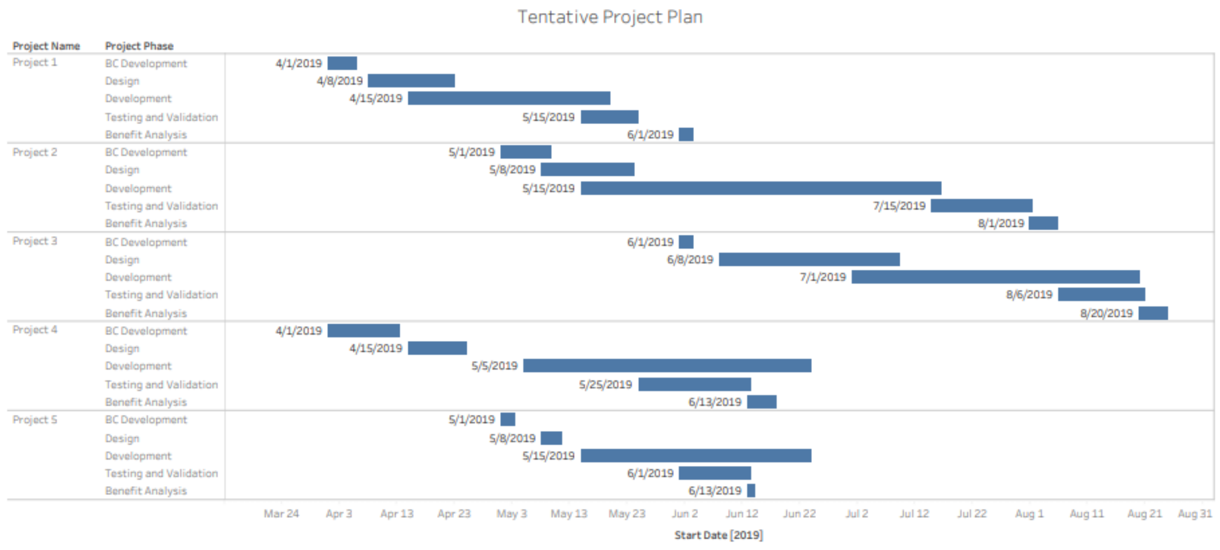
Table 14: Resource types for use case

Resource Name
Business SME
Technical Lead
Developer
Project Manager
Design Thinking Specialist
Legal Analyst

Our goal is to pack as many stories as possible into the next sprint while ensuring that all projects with end dates in that sprint are completed, while at the same time ensuring the most efficient allocation of resources to each of the stories.

The tentative project plan for the five projects is as given below in Figure 10

Figure 10: Tentative project plan



Chapter 6: Model Testing Results and Experiments

6.1 Estimating tool development – related models and development of the tool

6.1.1 Function points related predictive models:

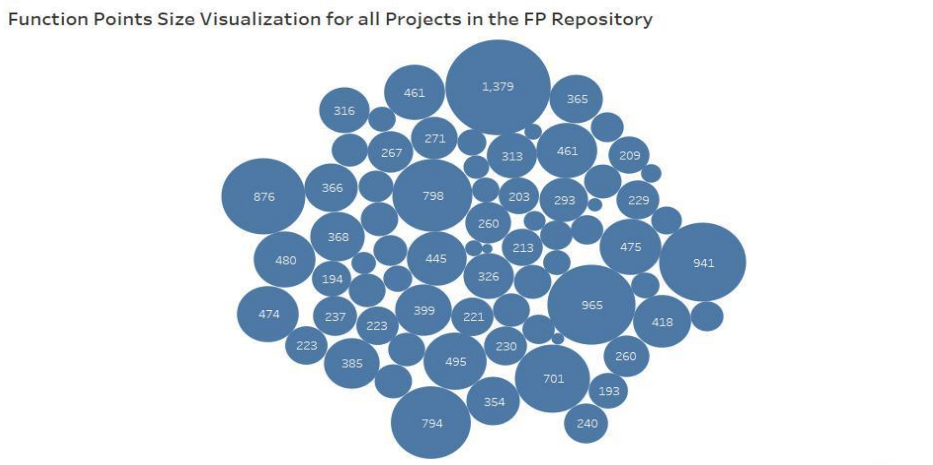
This research uses simple linear regression and multiple regression to evaluate the effect of the Adjusted Function Points Count on the total design and development cost and on the total labor cost, resulting in the following models.

$$E(Design\ and\ Development\ Cost) = \beta_0 + \beta_1 * Adjusted\ Function\ Points\ Count$$

$$E(Total\ Labor\ Cost) = \beta_0 + \beta_1 * Adjusted\ Function\ Points\ Count$$

We analyzed the data and obtained the summary statistics. The data for all projects in the repository can be summarized by the following visualization shown below in Figure 11.

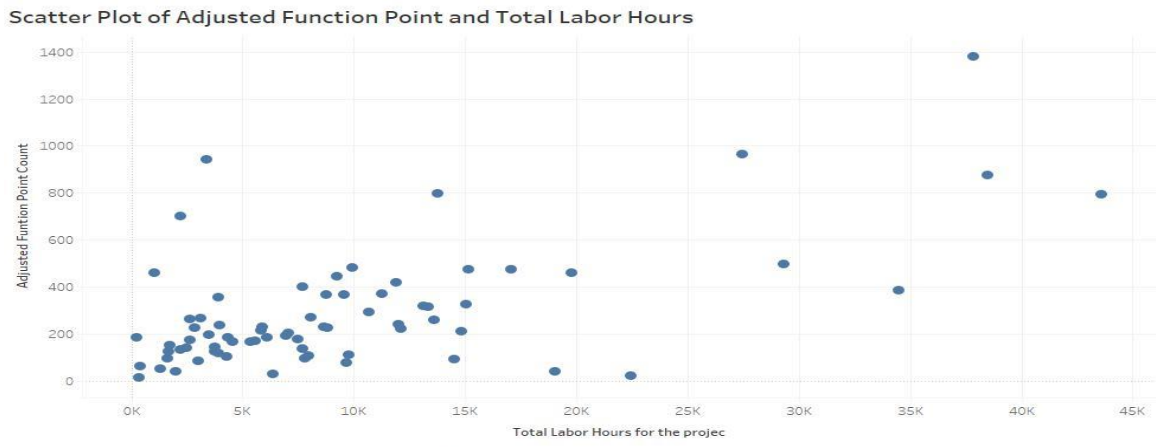
Figure 11: Function point repository visulization



Project Name and sum of Adjusted Funtion Point Count. Size shows sum of Adjusted Funtion Point Count. The marks are labeled by Project Name and sum of Adjusted Funtion Point Count.

The scatter plot of the Adjusted Function Point count with the total labor hours is shown below in Figure 12. The scatter plot shows that a bulk of the projects are below 500 function points, and it also shows that there are a few outliers in the data.

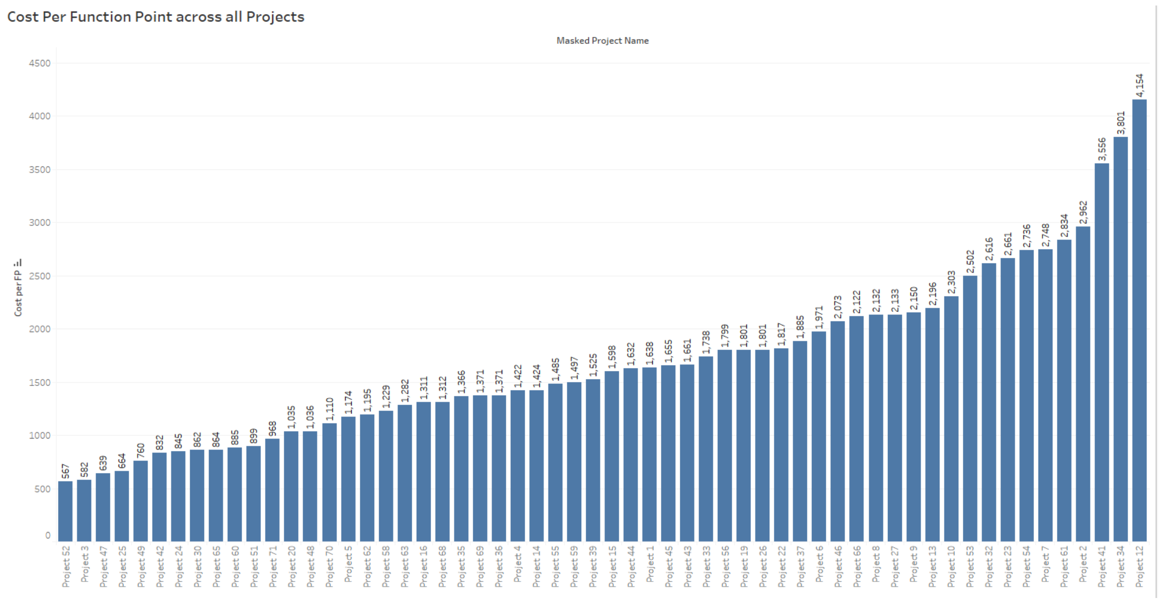
Figure 12: Scatter plot of adjusted function point and total labor hours



Sum of Total Labor Hours for the projec vs. sum of Adjusted Funtion Point Count. Details are shown for Project Name.

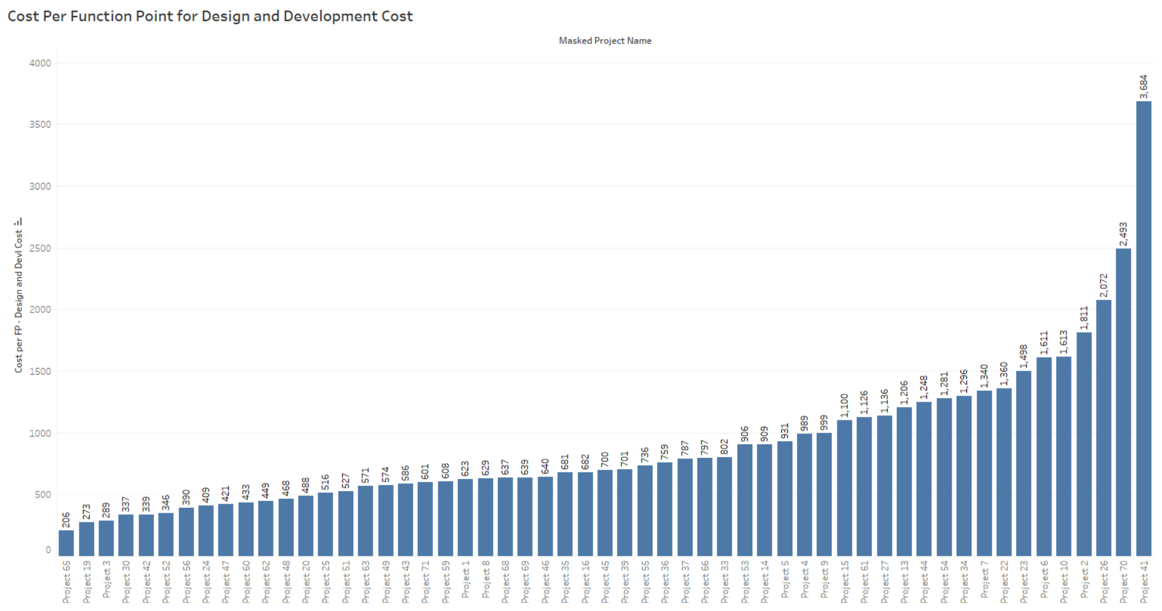
When we examine the cost per function point for the total cost, we obtain Figure 13. The outliers have been removed in this visualization, and the bar graph shows the cost per function point for total project cost sorted in ascending order across all projects. The cost per function point for total project cost ranges from $567 to $4,154.

Figure 13: Bar graph of cost per FP for total cost across all projects sorted in ascending order



When we examine the cost per function point for the design and development cost, we obtain Figure 14. The bar graph shows the cost per function point for total design and development cost sorted in ascending order across all projects. The cost per function point for total design and development cost ranges from $206 to $3,684.

Figure 14: Bar graph of cost per FP for design and development across all projects sorted in ascending order



The summary statistics are shown below for the cost per function point for the design and development cost and total labor cost in Table 15. Table 15 also provides the summary statistics for the function point count and the five components that make up the function point count.

Table 15: Summary statistics for Function points related metrics

Variable	Mean	Std Dev	Minimum	Maximum	N	10th Pctl	90th Pctl
Adjusted Funtion Point Count	294.54	247.36	14.24	1379.40	58	94.00	495.00
Cost per FP - Design and Devl Co	850.65	623.69	6.03	3684.43	58	288.60	1611.23
Cost per FP	1692.89	799.33	566.72	4153.69	58	831.70	2747.89
External Inputs	53.57	58.14	0.00	324.00	58	9.00	131.00
External Interface Files	41.12	35.28	0.00	198.00	58	5.00	69.00
External Outputs	39.38	81.03	0.00	605.00	58	4.00	71.00
External Queries	59.76	72.11	0.00	429.00	58	6.00	155.00
Internal Logical Files	70.28	59.39	0.00	365.00	58	17.00	114.00

I removed the projects that were below the 10th percentile and above the 90th percentile as outliers and used the resulting dataset for my modeling. I modeled the design and development costs separately, resulting in the following models.

$E(Design\ and\ Devl\ Cost) = -23{,}001 + 930.66 * Adjusted\ Function\ Points\ Count$

The above-mentioned model has an R-Square of 0.7068, and the results associated with the model are shown in Appendix 1.

$E(Total\ Labor\ Cost) = 50{,}369 + 1464.22 * Adjusted\ Function\ Points\ Count$

The above-mentioned model has an R-Square of 0.7016, and the results associated with the model are shown in Appendix 2. This is one of the estimates used among the suite of estimates in the estimating tool.

We also considered the following models.

$E(Estimated\ Cost) = \beta_0 + \beta_1 * \#\ of\ ILF + \beta_2 * \#\ of\ EIF$

$E(Estimated\ Cost) = \beta_0 + \beta_1 * \#\ of\ EQ + \beta_2 * \#\ of\ EI + \beta_3 * \#\ of\ EO$

$E(Estimated\ Cost) = \beta_0 + \beta_1 * \#\ of\ EQ + \beta_2 * \#\ of\ EI + \beta_3 * \#\ of\ E\ O + \beta_4 * \#\ of\ ILF + \beta_5 * \#\ of\ EIF$

I used Forward Selection as the model selection method in SAS, and it resulted in the following models. The significance level chosen to enter the model was 0.2, and to stay in the model was 0.1.

The resulting models are listed below.

$E(Estimated\ Cost) = 99147 + 2581.11 * \#\ of\ ILF + 4757.32 * \#\ of\ EIF$

The R-square for the above model is 0.47, and it explains 47% of the variability in the model. Both the variables are statistically significant in the model.

$E(Estimated\ Cost) = 67302 + \ 2924.39\ * \#\ of\ EQ + \ 4582.43 * \#\ of\ EI$

The EO variable did not make it into the model, and the R-Square for this model is 0.71. This model explains 71% of the variability in the model.

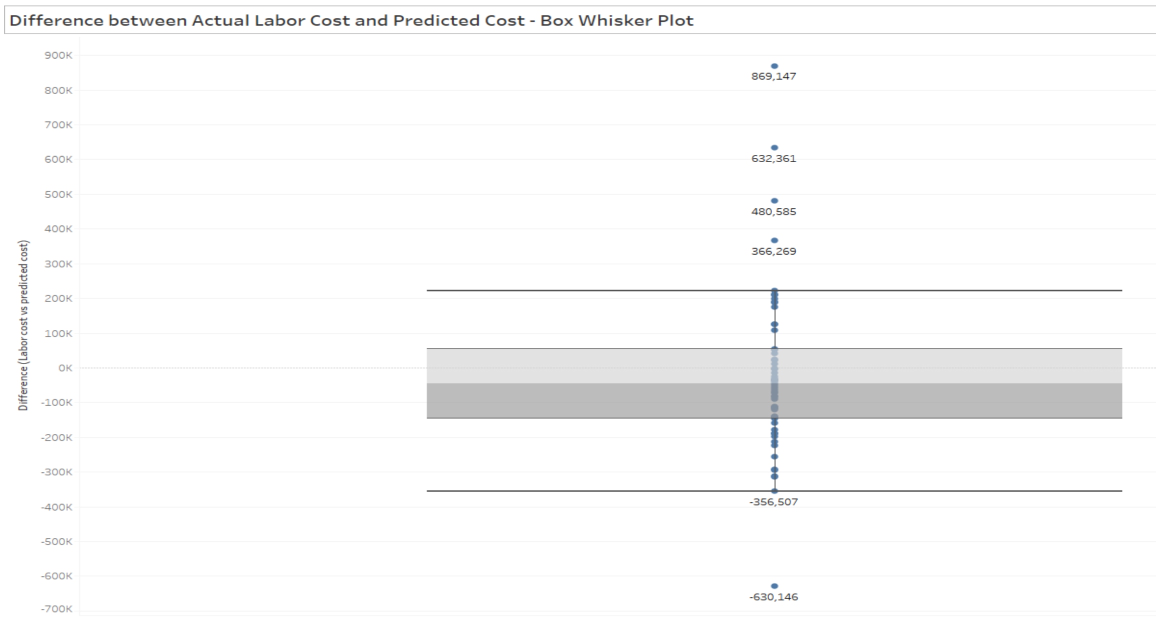
$E(Estimated\ Cost) = 30359 + \ 2466.42\ * \#\ of\ EQ + \ 3991.05 * \#\ of\ EI + \ 2177.66\ * \#\ of\ EIF$

The above mentioned three variables meet the requirements of the model, and other two variables do not enter the model. The R-Square on the model is 0.74.

When we look at all the models, the number of external queries, external inputs, and external interface files contribute to a useful model, and it conveys an important message for the organization. It points to an organization where systems are tightly integrated, and this is reflected in the data.

The difference between the actual labor cost and the predicted cost from the model is shown below in Figure 15 as a box whisker plot.

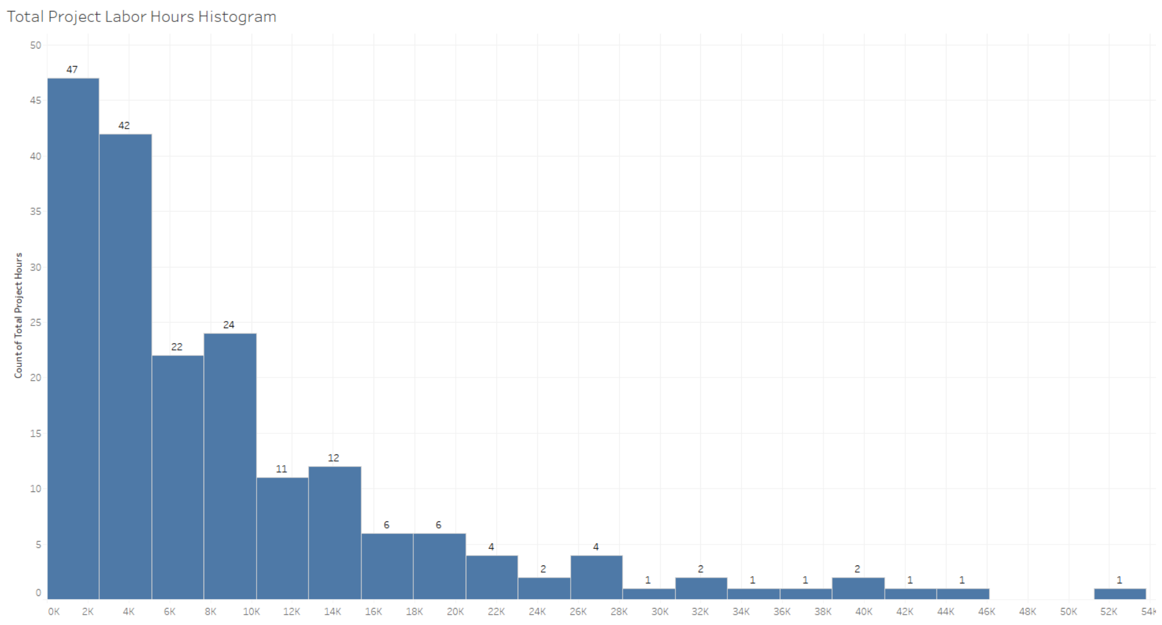
Figure 15: Box Whisker Plot of the difference between Actual Labor Cost and Predicted Cost.



6.1.2 Two Stage Least Squares (2SLS) model results and Decision Tree analysis

The histogram of the total labor hours for all projects in the repository is shown below in Figure 16. There are some clear outliers, as can be observed from the scatter plot. ABC Inc. as an organization was moving towards breaking down bigger projects into smaller projects to have a better handle on the execution of the projects. We had a good base of 116 projects that were under 10,000 total labor hours of which 103 projects were under 7,500 total labor hours. We separate the data by the total labor hours into two sets, one with projects under 7,500 total labor hours and the second with projects under 10,000 total labor hours based on the direction ABC Inc. was moving towards of breaking down bigger projects into multiple smaller projects. We fit separate coefficients for each dataset.

Figure 16: Histogram of total project labor hours



The goal was to predict the total labor hours, and the following explanatory variables were used in the model. A dataset with projects that had total labor hours that were less than 7500 hours was first created.

1. Project type
2. Number of tasks by each phase, design, development, testing, implementation, and planning
3. Number of contracting resources
4. Number of contracting resources considering overall work experience of more than three years
5. Number of employees
6. Percentage time spent on design
7. Percentage time spent on development
8. Percentage time spent on planning
9. Percentage time spent on verification/testing by IS
10. Percentage time spent on meetings
11. Number of defects by category (sev1, sev2, sev3, and sev4)
12. Number of requirements
13. Number of test plans
14. Area doing the project
15. Duration of the project
16. Project type

This research used Forward Selection as the model selection method in SAS, and it generates the following results. The significance level chosen to enter the model was 0.2, and to stay in the model was 0.1.

$$E(Total\ Labor\ Hours) = 785.39 + 404.58 * \#\ of\ contractors + 220.98 * \#\ of\ experienced\ contractors + 130.62 * \#\ of\ employees + 2665.81 * percent\ impl + 21.40 * \#\ of\ development\ tasks + 0.63 * \#\ of\ days - 1452.73 * total\ planning\ percent - 566.87 * proj\ type\ 1 - 697.57 * area\ 1 - 1154 * area\ 2$$

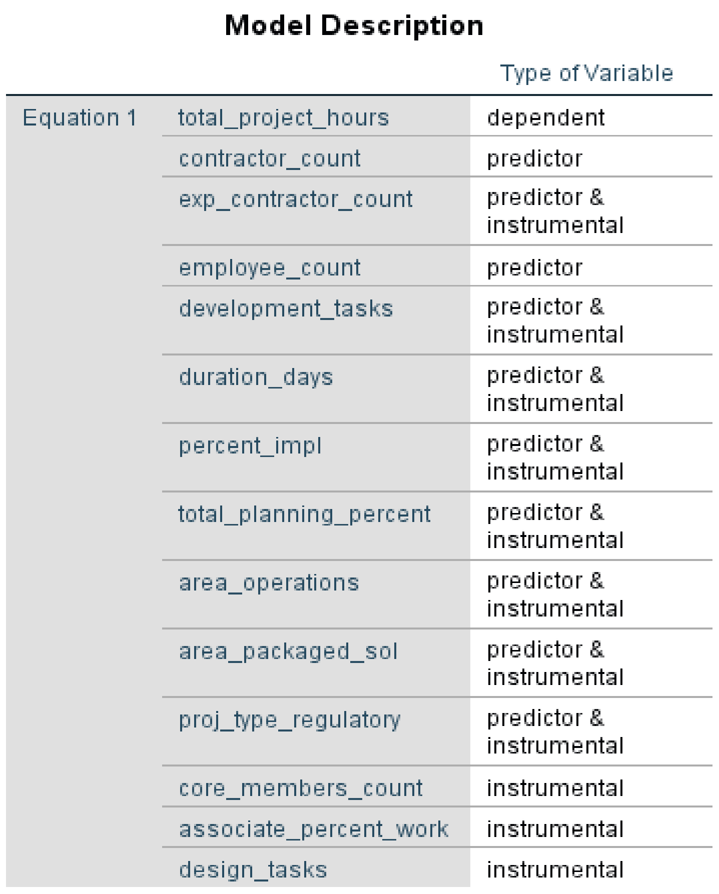
The collinearity diagnostics and Variance Inflation Factors (VIF's) were generated as part of the output while running the OLS model above. The observation of the results did not indicate the presence of multicollinearity. The explanatory variables were not significantly correlated. I then ran the White test on the OLS estimates to test for Heteroscedasticity. The White Test came out to be insignificant. The insignificance of the test indicated homoscedastic errors and showed that these errors were not related to the explanatory variables.

This was followed by running separate OLS models to explore for possible endogeneity of some of the explanatory variables in the model. One such model is shown below that examines # of contractors.

$$E(\#\ of\ contractors) = 1.15 + 0.32 * \#\ of\ core\ members\ count - 3.17 * employee\ percent\ work - 0.34 * \exp contractor\ count + 4.18 * percent\ impl + 0.02 * \#\ of\ design\ tasks$$

The model above had an R-Square of 0.62. We then followed this by running a two stage least squares model. The model description is shown below in Figure 17.

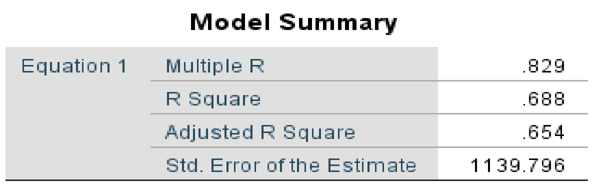
Figure 17: Model description and summary for Two Stage Least Squares model with hours less than 7500 hours.



Model Description

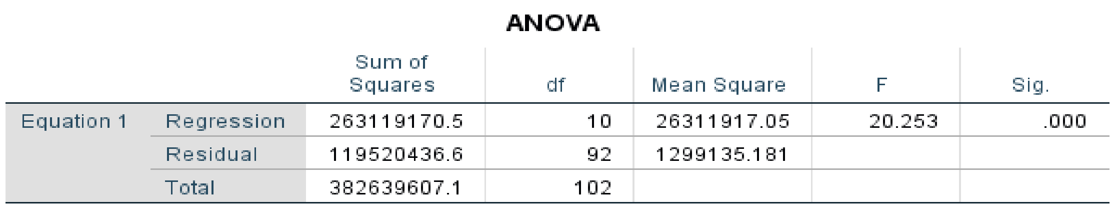
		Type of Variable
Equation 1	total_project_hours	dependent
	contractor_count	predictor
	exp_contractor_count	predictor & instrumental
	employee_count	predictor
	development_tasks	predictor & instrumental
	duration_days	predictor & instrumental
	percent_impl	predictor & instrumental
	total_planning_percent	predictor & instrumental
	area_operations	predictor & instrumental
	area_packaged_sol	predictor & instrumental
	proj_type_regulatory	predictor & instrumental
	core_members_count	instrumental
	associate_percent_work	instrumental
	design_tasks	instrumental

The model summary is shown below



Model Summary

Equation 1	Multiple R	.829
	R Square	.688
	Adjusted R Square	.654
	Std. Error of the Estimate	1139.796



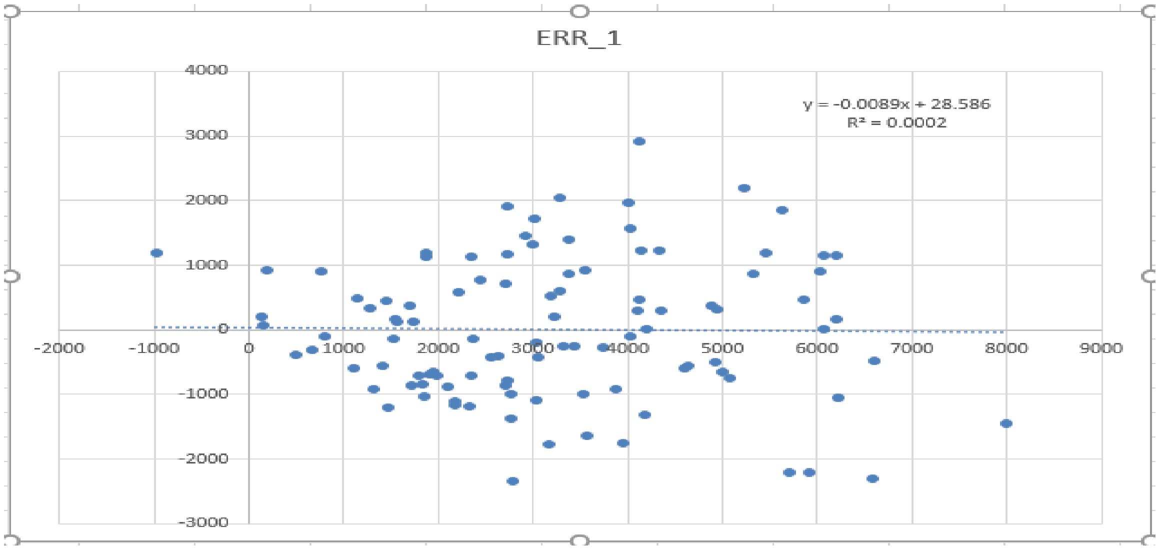
ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Equation 1	Regression	263119170.5	10	26311917.05	20.253	.000
	Residual	119520436.6	92	1299135.181		
	Total	382639607.1	102			

$$E(Total\ Labor\ Hours) = 767.39 + 433.92 * \#\ of\ contractors + 216.31 * \#\ of\ experienced\ contractors + 129.86 * \#\ of\ employees + 2623.1 * percent\ impl + 21.27 * \#\ of\ development\ tasks + 0.621 * \#\ of\ days - 1435.14 * total\ planning\ percent - 567.47 * proj\ type\ 1 - 695.62 * area\ 1 - 1158.65 * area\ 2$$

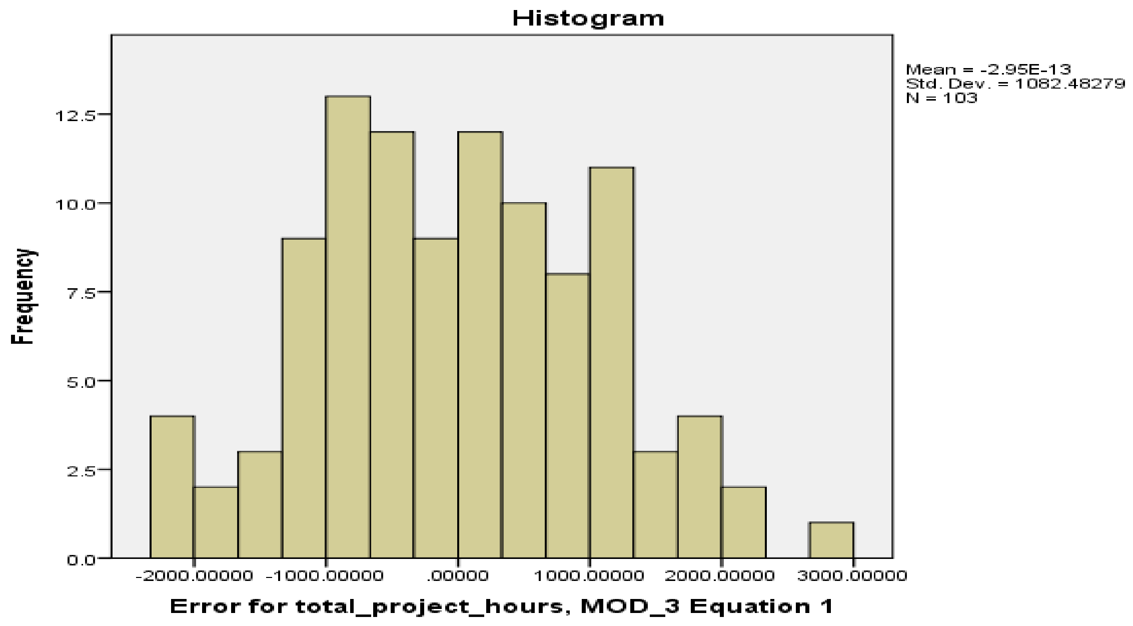
SPSS generated two new variables as output from the run of the two stage least squares model, ERR_1 and FIT_1 for each of the 103 data points. The ERR_1 represents the residuals, and the FIT_1 represents the predicted value of the dependent variable. The first test that we did was to ensure that the residuals from the model (ERR_1) was unrelated to the predicted values of the dependent variable (FIT_1). This was visualized in Excel by creating a scatter plot of ERR_1 with FIT_1 and then fitting a trendline to the data to see if there is a significant linear trend between the residuals and the fit values.

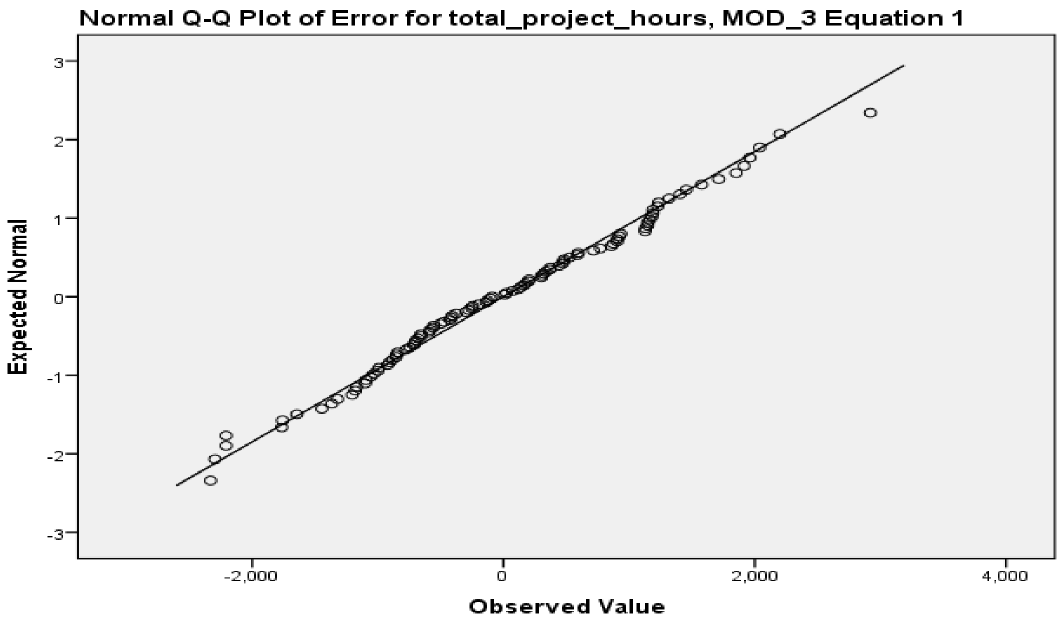
Figure 18: Scatter plot of residuals with predicted values for 2SLS (less than 7550 hours)



The second test was done to ensure the residuals (ERR_1) was normally distributed. This was done in SPSS Statistics, and the results are shown below in Figure 19.

Figure 19: Test on residuals for 2SLS model with hours less than 7500 hours





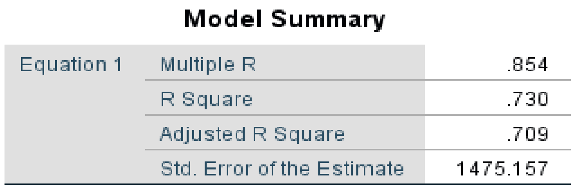
We observe that the residuals (ERR_1) are normally distributed. Both of the diagnostic tests help validate the assumptions of the two stage least squares model as outlined by Kmenta, and it helps to explain the validity of the model.

A dataset with projects that had total labor hours less than 10,000 hours was then created. We went through the same process as above and eventually ended up with the model below from the two stage least squares model. The initial model that was the result of this run is as shown below.

$$E(Total\ Labor\ Hours) = -360.29 + 467.36 * \#\ of\ contractors + 318.46 * \#\ of\ experienced\ contractors + 140.62 * \#\ of\ employees + 3856.74 * percent\ impl + 22.69 * \#\ of\ development\ tasks + 779.41 * proj\ type\ 2 - 660.17 * area\ 1 - 1654.42 * area\ 2.$$

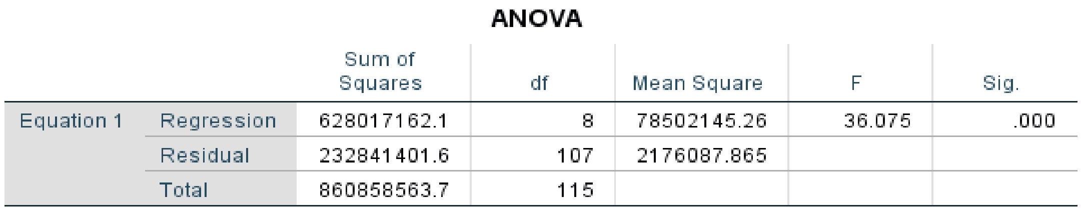
The model summary and description for the two stage least squares model with hours less than 10,000 hours is shown below in Figure 20.

Figure 20: Model description and summary for Two Stage Least Squares model with hours less than 10000 hours.



Model Summary

Equation 1	Multiple R	.854
	R Square	.730
	Adjusted R Square	.709
	Std. Error of the Estimate	1475.157

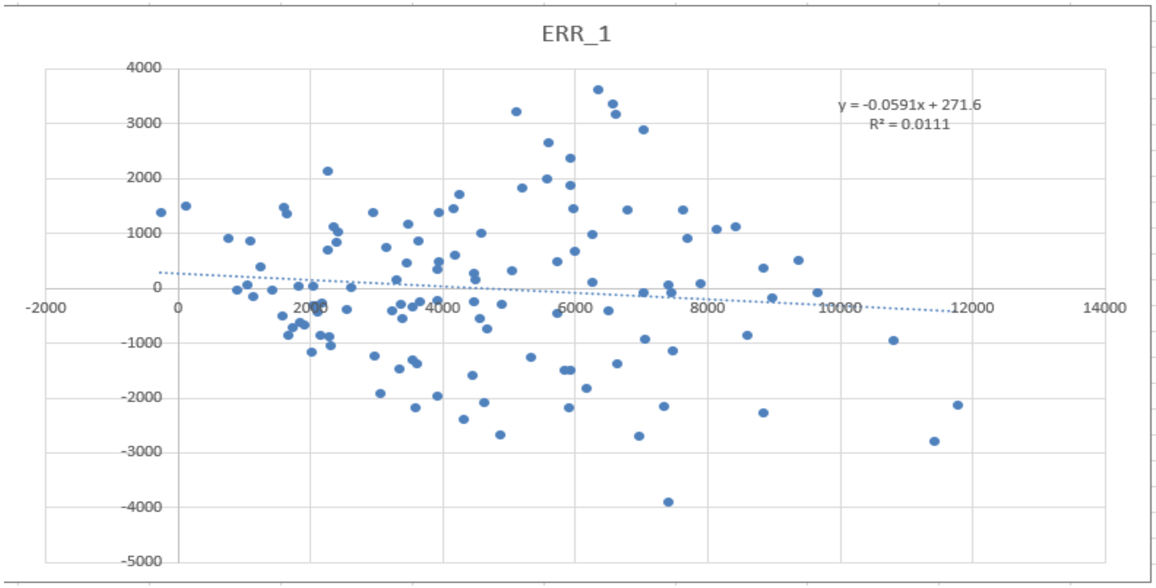


ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Equation 1	Regression	628017162.1	8	78502145.26	36.075	.000
	Residual	232841401.6	107	2176087.865		
	Total	860858563.7	115			

The ERR_1 and FIT_1 for each of the 116 data points were generated. The first test that we did was to ensure that the residuals from the model (ERR_1) was unrelated to the predicted values of the dependent variable (FIT_1). This was visualized in Excel by creating a scatter plot of ERR_1 with FIT_1 and then fitting a trendline to the data to see if there is a significant linear trend between the residuals and the fit values.

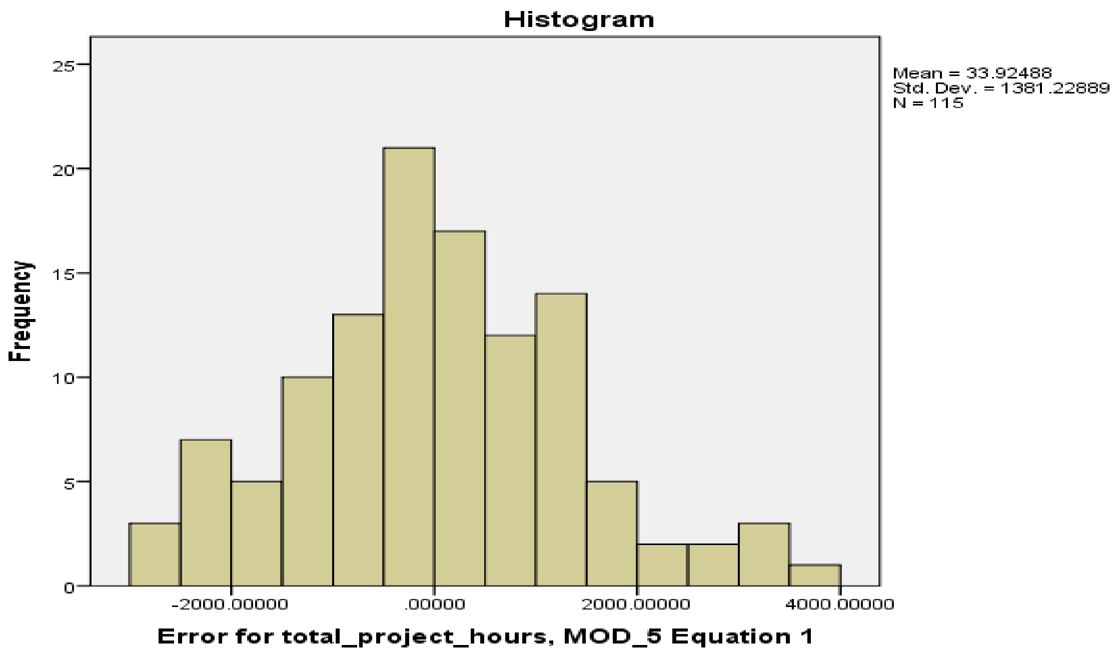
Figure 21: Scatter plot of residuals with predicted values for 2SLS (less than 10000 hours)

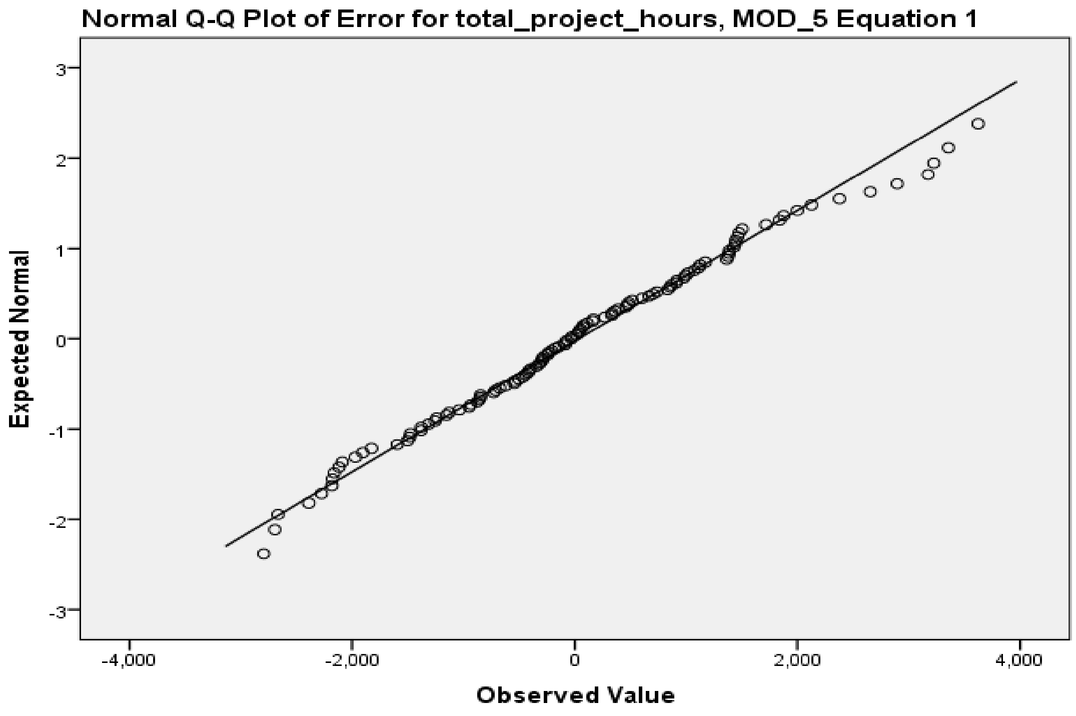


The second test was done to ensure the residuals (ERR_1) was normally distributed. This was done in SPSS Statistics, and the results are shown below in Figure 22.

Figure 22: Test on residuals for 2SLS model with hours less than 10000 hours.

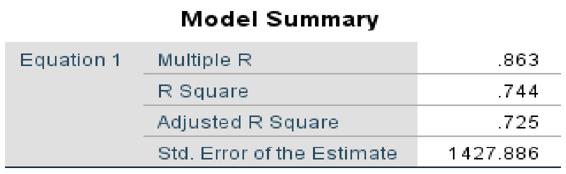
Error for total_project_hours, MOD_5 Equation 1





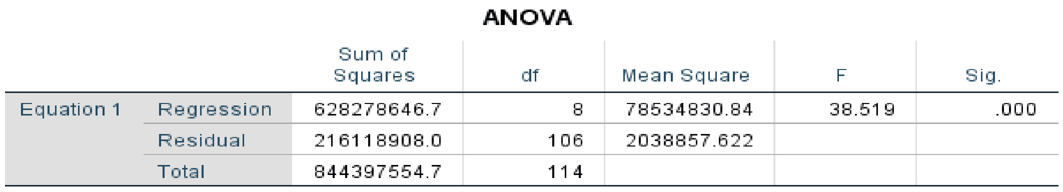
We removed one of the outliers with a residual value of -3901.36 from the dataset and reran the two stage least squares model. The model summary is shown below in Figure 23. This run had an R-Square of .744, and it was a better model as it has a better R-Square, as it explained more of the variability, and it shows that there are smaller differences between the observed and fitted values.

Figure 23: Model description and summary for Two Stage Least Squares model with hours less than 10000 hours less one outlier



Model Summary

Equation 1	Multiple R	.863
	R Square	.744
	Adjusted R Square	.725
	Std. Error of the Estimate	1427.886



ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
Equation 1	Regression	628278646.7	8	78534830.84	38.519	.000
	Residual	216118908.0	106	2038857.622		
	Total	844397554.7	114			

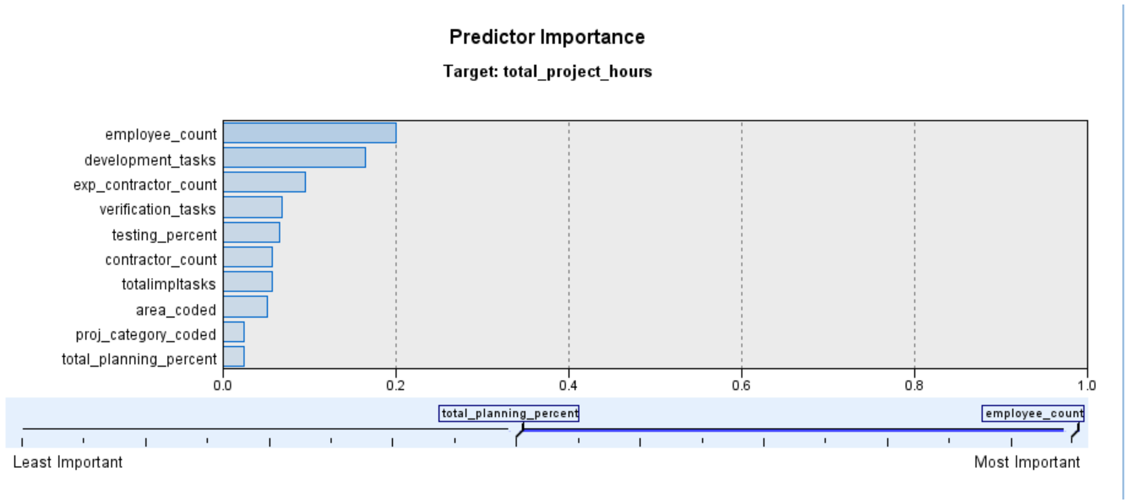
The resulting model was as follows

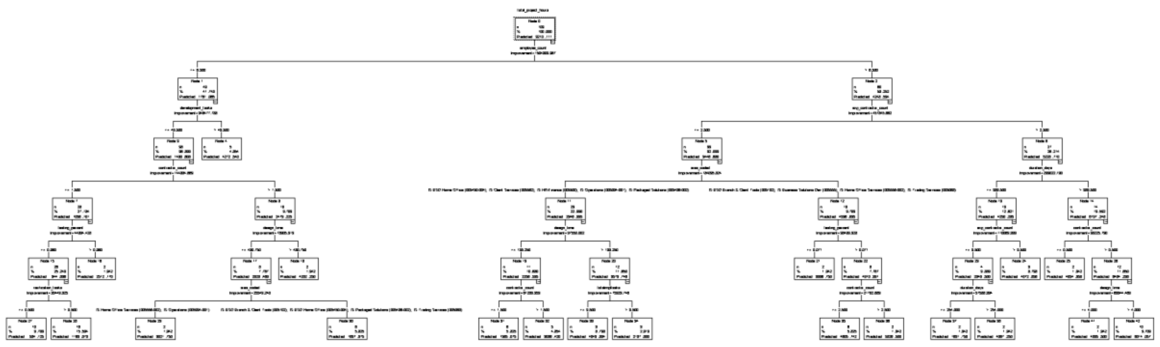
$$E(Total\ Labor\ Hours) = -358.59 + 402.82 * \#\ of\ contractors + 350.50 * \#\ of\ experienced\ contractors + 163.12 * \#\ of\ employees + 3240.79 * percent\ impl + 20.97 * \#\ of\ development\ tasks + 724.007 * proj\ type\ 2 - 332.22 * area\ 1 - 1608.22 * area\ 2.$$

I then ran the decision tree analysis in SPSS modeler using the same dataset to validate our results. The results from the decision tree analysis are shown below in Figure

24. I first fit the data using the Regression Tree (CART) analysis in SPSS Modeler, and the results are summarized below.

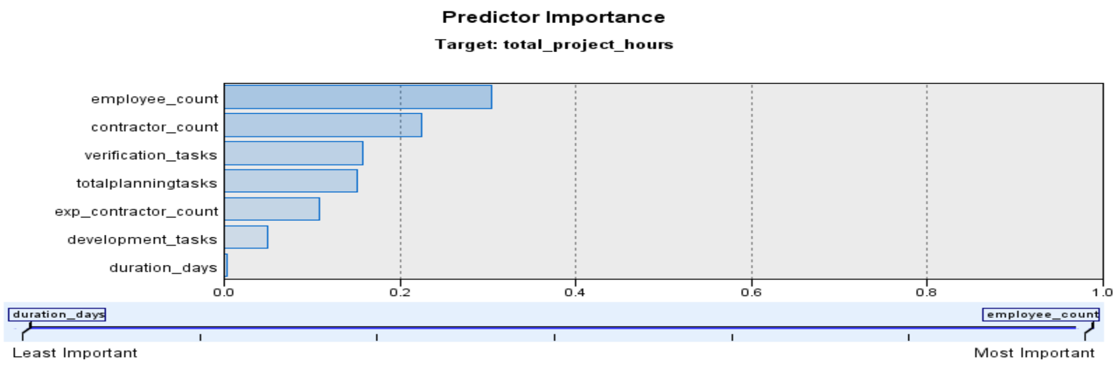
Figure 24: Decision Tree Analysis, Regression Tree (CART)

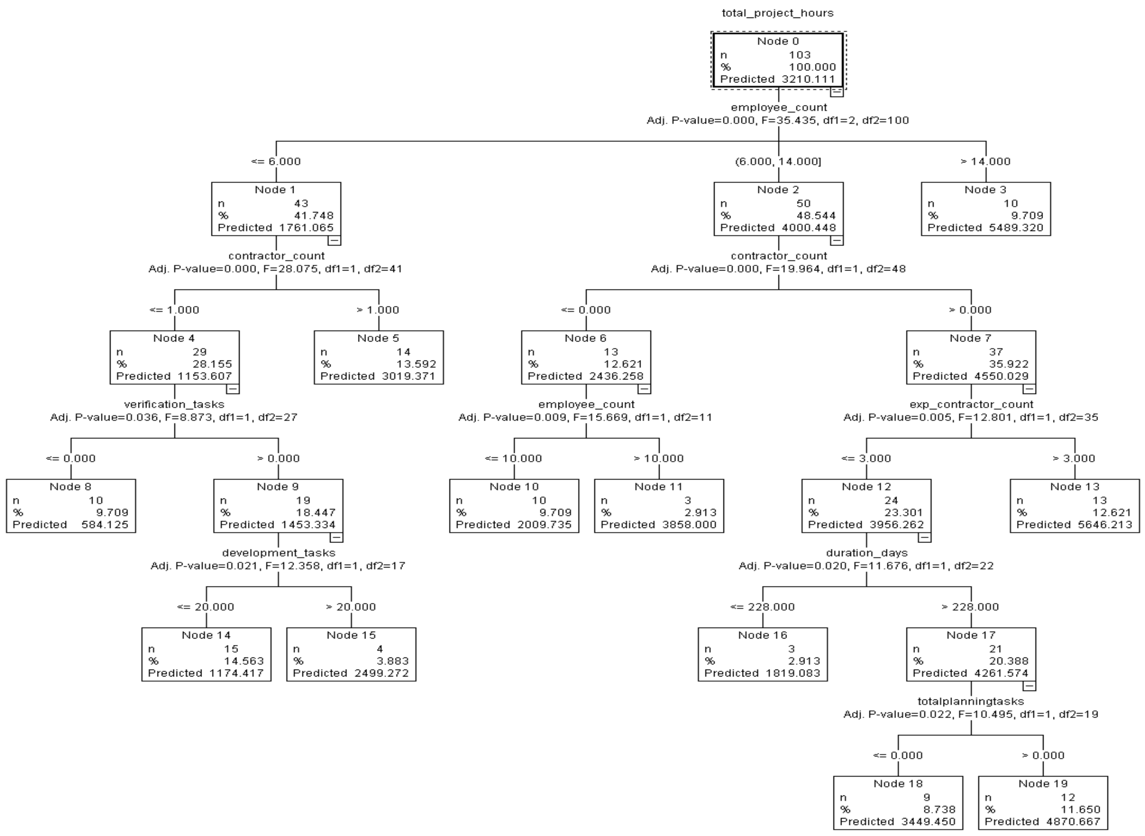




We then fit the data using the CHAID decision tree analysis, and those results are summarized below in Figure 25. The decision tree analysis in conjunction with the analysis from the predictive modeling helped us better understand the most important contributing explanatory variables.

Figure 25: Decision Tree Analysis, Regression Tree (CHAID)



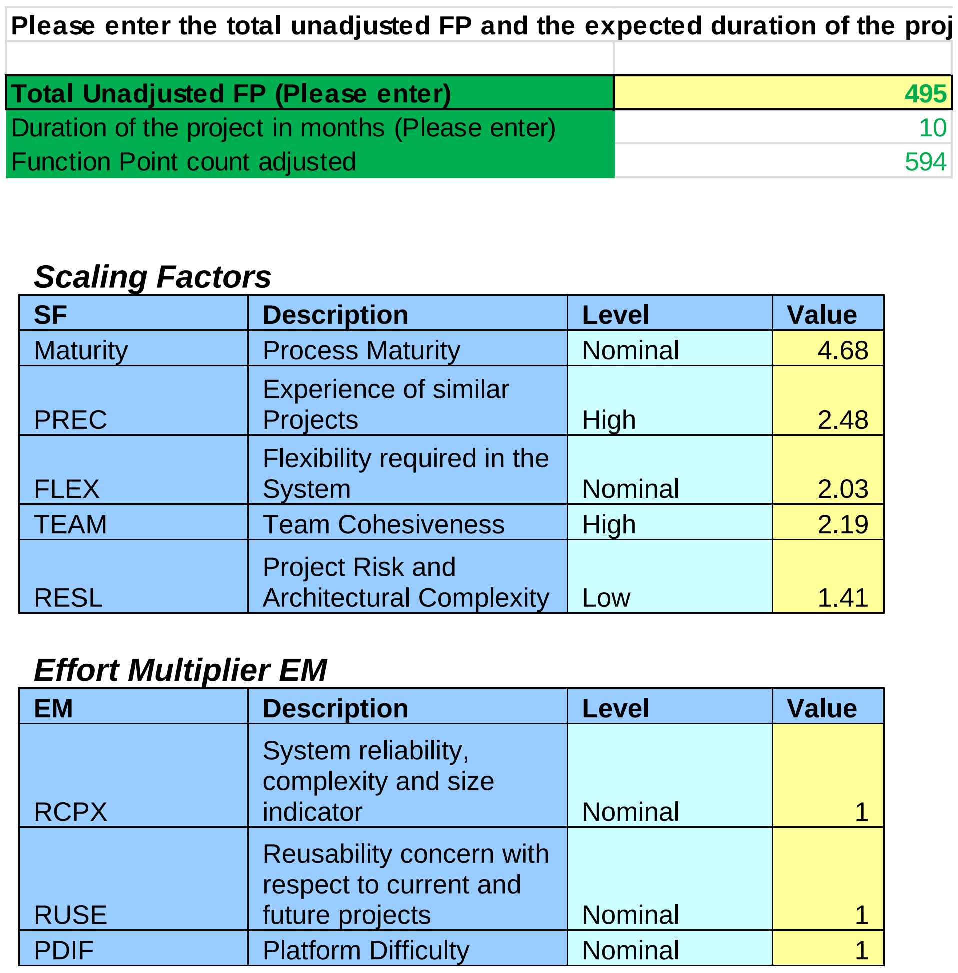

total_project_hours
Node 0 n 103 % 100.000 Predicted 3210.111
employee_count Adj. P-value=0.000, F=35.435, df1=2, df2=100
<= 6.000
(6.000, 14.000]
> 14.000
Node 1 n 43 % 41.748 Predicted 1761.065
Node 2 n 50 % 48.544 Predicted 4000.448
Node 3 n 10 % 9.709 Predicted 5489.320
contractor_count Adj. P-value=0.000, F=28.075, df1=1, df2=41
contractor_count Adj. P-value=0.000, F=19.964, df1=1, df2=48
<= 1.000
> 1.000
<= 0.000
> 0.000
Node 4 n 29 % 28.155 Predicted 1153.607
Node 5 n 14 % 13.592 Predicted 3019.371
Node 6 n 13 % 12.621 Predicted 2436.258
Node 7 n 37 % 35.922 Predicted 4550.029
verification_tasks Adj. P-value=0.036, F=8.873, df1=1, df2=27
employee_count Adj. P-value=0.009, F=15.669, df1=1, df2=11
exp_contractor_count Adj. P-value=0.005, F=12.801, df1=1, df2=35
<= 0.000
> 0.000
<= 10.000
> 10.000
<= 3.000
> 3.000
Node 8 n 10 % 9.709 Predicted 584.125
Node 9 n 19 % 18.447 Predicted 1453.334
Node 10 n 10 % 9.709 Predicted 2009.735
Node 11 n 3 % 2.913 Predicted 3858.000
Node 12 n 24 % 23.301 Predicted 3956.262
Node 13 n 13 % 12.621 Predicted 5646.213
development_tasks Adj. P-value=0.021, F=12.358, df1=1, df2=17
duration_days Adj. P-value=0.020, F=11.676, df1=1, df2=22
<= 20.000
> 20.000
<= 228.000
> 228.000
Node 14 n 15 % 14.563 Predicted 1174.417
Node 15 n 4 % 3.883 Predicted 2499.272
Node 16 n 3 % 2.913 Predicted 1819.083
Node 17 n 21 % 20.388 Predicted 4261.574
totalplanningtasks Adj. P-value=0.022, F=10.495, df1=1, df2=19
<= 0.000
> 0.000
Node 18 n 9 % 8.738 Predicted 3449.450
Node 19 n 12 % 11.650 Predicted 4870.667

6.1.3 Function points related suite of estimates:

6.1.3.1 Incorporating the COCOMO II early design model into the suite of estimates:

I developed an Excel tool to input the function points that gives a suite of estimates, including three estimates from the COCOMO II early design model and the predictive model estimate based on function points. Figure 26 provides an illustration of the tool.

Figure 26: Snapshots of implemented COCOMO II tool



Please enter the total unadjusted FP and the expected duration of the proj

Total Unadjusted FP (Please enter)	**495**
Duration of the project in months (Please enter)	10
Function Point count adjusted	594

Scaling Factors

SF	Description	Level	Value
Maturity	Process Maturity	Nominal	4.68
PREC	Experience of similar Projects	High	2.48
FLEX	Flexibility required in the System	Nominal	2.03
TEAM	Team Cohesiveness	High	2.19
RESL	Project Risk and Architectural Complexity	Low	1.41

Effort Multiplier EM

EM	Description	Level	Value
RCPX	System reliability, complexity and size indicator	Nominal	1
RUSE	Reusability concern with respect to current and future projects	Nominal	1
PDIF	Platform Difficulty	Nominal	1

PERS	Personal capability of team. Like technical capability of Programmers, Designers and testers.	Nominal	1
PREX	Application, Language and tool experience	High	0.87
FCIL	Using Case tools for development etc.	High	0.87
SCED	Schedule Pressure	High	1.14

Constants	**Value**
B	0.91
A	2.94
E	1.0379
C	3.67
D	0.28
F	0.30558
EM	0.862866

Consolidated Size and Effort

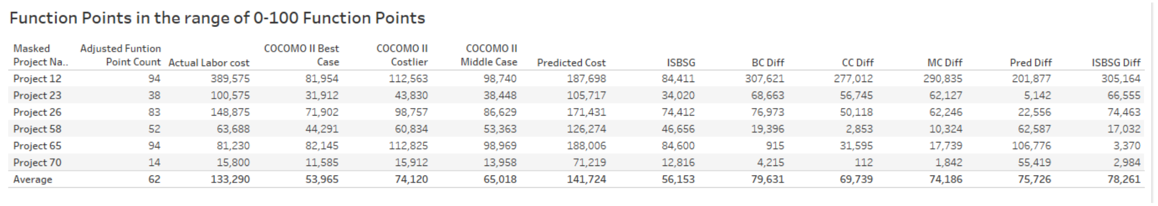
Technology	**Java**
Increase due to lifecycle	**0%**
SLOC per FP	**53**
SLOC	**30170.25**
PM	**87.0855**
Man-days	**1828.7946**
FP from LOC	**569.2500**
Hours per FP	**25.7011**

Total Cost	**$731,517.84**							
TDEV	14.37							
Staff Estimate	6.06							
The project will take about		14.37	months	to complete with abou		6.06	folks on the project.	

6.1.3.1 Comparison of the Function points related suite of estimates

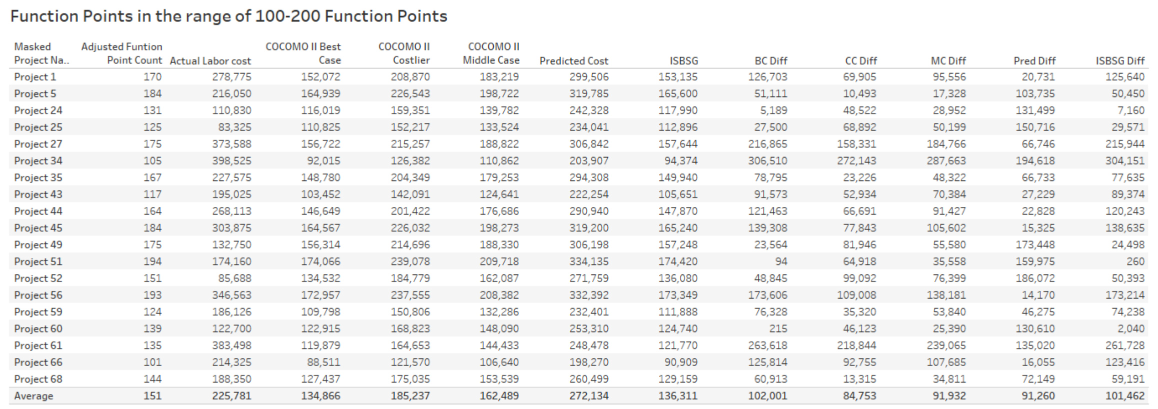
We then calculated the four estimates for each project in the function point repository. We also calculated the ISBSG benchwork estimate for each project. The absolute value of the differences in estimates from the actual labor cost was calculated, and this gives us feedback on which estimate is closest to the actual labor cost in each individual range. The repository was split by function point ranges as shown below in Table 16. The ranges we used were 0-100, 100-200, 200-300, 300-400, 400-500 and above 500 function points.

Table 16: Tableau Dashboard of Function Point Suite of Estimates and differences of the estimates from the actual labor cost



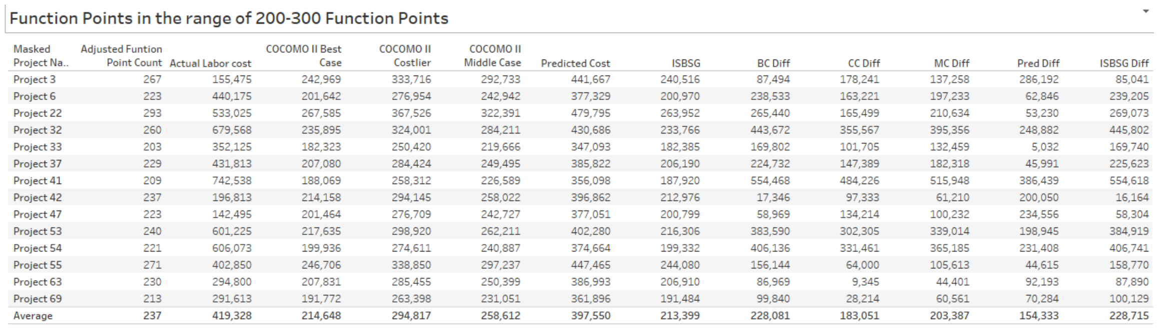
Function Points in the range of 0-100 Function Points

Masked Project Na..	Adjusted Funtion Point Count	Actual Labor cost	COCOMO II Best Case	COCOMO II Costlier	COCOMO II Middle Case	Predicted Cost	ISBSG	BC Diff	CC Diff	MC Diff	Pred Diff	ISBSG Diff
Project 12	94	389,575	81,954	112,563	98,740	187,698	84,411	307,621	277,012	290,835	201,877	305,164
Project 23	38	100,575	31,912	43,830	38,448	105,717	34,020	68,663	56,745	62,127	5,142	66,555
Project 26	83	148,875	71,902	98,757	86,629	171,431	74,412	76,973	50,118	62,246	22,556	74,463
Project 58	52	63,688	44,291	60,834	53,363	126,274	46,656	19,396	2,853	10,324	62,587	17,032
Project 65	94	81,230	82,145	112,825	98,969	188,006	84,600	915	31,595	17,739	106,776	3,370
Project 70	14	15,800	11,585	15,912	13,958	71,219	12,816	4,215	112	1,842	55,419	2,984
Average	62	133,290	53,965	74,120	65,018	141,724	56,153	79,631	69,739	74,186	75,726	78,261



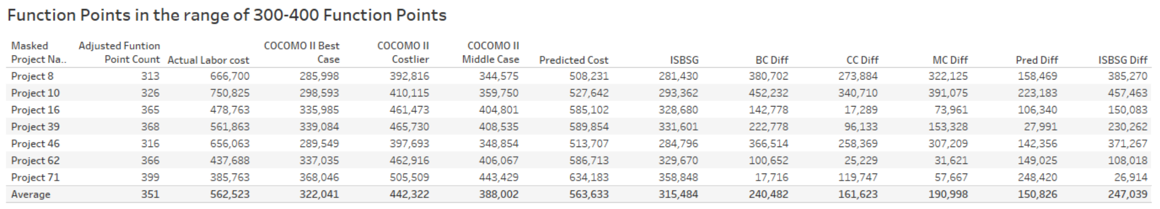
Function Points in the range of 100-200 Function Points

Masked Project Na..	Adjusted Funtion Point Count	Actual Labor cost	COCOMO II Best Case	COCOMO II Costlier	COCOMO II Middle Case	Predicted Cost	ISBSG	BC Diff	CC Diff	MC Diff	Pred Diff	ISBSG Diff
Project 1	170	278,775	152,072	208,870	183,219	299,506	153,135	126,703	69,905	95,556	20,731	125,640
Project 5	184	216,050	164,939	226,543	198,722	319,785	165,600	51,111	10,493	17,328	103,735	50,450
Project 24	131	110,830	116,019	159,351	139,782	242,328	117,990	5,189	48,522	28,952	131,499	7,160
Project 25	125	83,325	110,825	152,217	133,524	234,041	112,896	27,500	68,892	50,199	150,716	29,571
Project 27	175	373,588	156,722	215,257	188,822	306,842	157,644	216,865	158,331	184,766	66,746	215,944
Project 34	105	398,525	92,015	126,382	110,862	203,907	94,374	306,510	272,143	287,663	194,618	304,151
Project 35	167	227,575	148,780	204,349	179,253	294,308	149,940	78,795	23,226	48,322	66,733	77,635
Project 43	117	195,025	103,452	142,091	124,641	222,254	105,651	91,573	52,934	70,384	27,229	89,374
Project 44	164	268,113	146,649	201,422	176,686	290,940	147,870	121,463	66,691	91,427	22,828	120,243
Project 45	184	303,875	164,567	226,032	198,273	319,200	165,240	139,308	77,843	105,602	15,325	138,635
Project 49	175	132,750	156,314	214,696	188,330	306,198	157,248	23,564	81,946	55,580	173,448	24,498
Project 51	194	174,160	174,066	239,078	209,718	334,135	174,420	94	64,918	35,558	159,975	260
Project 52	151	85,688	134,532	184,779	162,087	271,759	136,080	48,845	99,092	76,399	186,072	50,393
Project 56	193	346,563	172,957	237,555	208,382	332,392	173,349	173,606	109,008	138,181	14,170	173,214
Project 59	124	186,126	109,798	150,806	132,286	232,401	111,888	76,328	35,320	53,840	46,275	74,238
Project 60	139	122,700	122,915	168,823	148,090	253,310	124,740	215	46,123	25,390	130,610	2,040
Project 61	135	383,498	119,879	164,653	144,433	248,478	121,770	263,618	218,844	239,065	135,020	261,728
Project 66	101	214,325	88,511	121,570	106,640	198,270	90,909	125,814	92,755	107,685	16,055	123,416
Project 68	144	188,350	127,437	175,035	153,539	260,499	129,159	60,913	13,315	34,811	72,149	59,191
Average	151	225,781	134,866	185,237	162,489	272,134	136,311	102,001	84,753	91,932	91,260	101,462



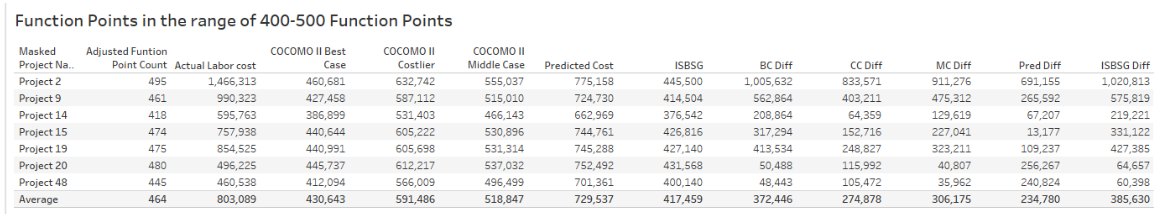
Function Points in the range of 200-300 Function Points

Masked Project Na..	Adjusted Funtion Point Count	Actual Labor cost	COCOMO II Best Case	COCOMO II Costlier	COCOMO II Middle Case	Predicted Cost	ISBSG	BC Diff	CC Diff	MC Diff	Pred Diff	ISBSG Diff
Project 3	267	155,475	242,969	333,716	292,733	441,667	240,516	87,494	178,241	137,258	286,192	85,041
Project 6	223	440,175	201,642	276,954	242,942	377,329	200,970	238,533	163,221	197,233	62,846	239,205
Project 22	293	533,025	267,585	367,526	322,391	479,795	263,952	265,440	165,499	210,634	53,230	269,073
Project 32	260	679,568	235,895	324,001	284,211	430,686	233,766	443,672	355,567	395,356	248,882	445,802
Project 33	203	352,125	182,323	250,420	219,666	347,093	182,385	169,802	101,705	132,459	5,032	169,740
Project 37	229	431,813	207,080	284,424	249,495	385,822	206,190	224,732	147,389	182,318	45,991	225,623
Project 41	209	742,538	188,069	258,312	226,589	356,098	187,920	554,468	484,226	515,948	386,439	554,618
Project 42	237	196,813	214,158	294,145	258,022	396,862	212,976	17,346	97,333	61,210	200,050	16,164
Project 47	223	142,495	201,464	276,709	242,727	377,051	200,799	58,969	134,214	100,232	234,556	58,304
Project 53	240	601,225	217,635	298,920	262,211	402,280	216,306	383,590	302,305	339,014	198,945	384,919
Project 54	221	606,073	199,936	274,611	240,887	374,664	199,332	406,136	331,461	365,185	231,408	406,741
Project 55	271	402,850	246,706	338,850	297,237	447,465	244,080	156,144	64,000	105,613	44,615	158,770
Project 63	230	294,800	207,831	285,455	250,399	386,993	206,910	86,969	9,345	44,401	92,193	87,890
Project 69	213	291,613	191,772	263,398	231,051	361,896	191,484	99,840	28,214	60,561	70,284	100,129
Average	237	419,328	214,648	294,817	258,612	397,550	213,399	228,081	183,051	203,387	154,333	228,715



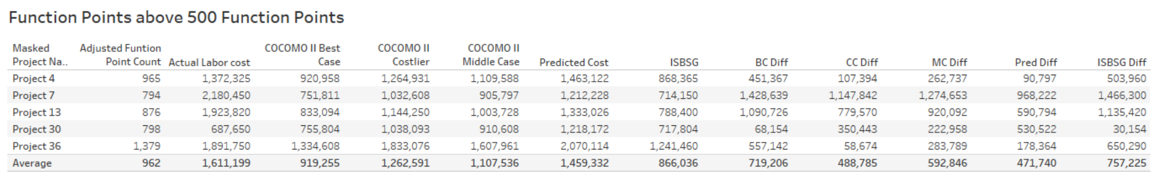
Function Points in the range of 300-400 Function Points

Masked Project Na..	Adjusted Funtion Point Count	Actual Labor cost	COCOMO II Best Case	COCOMO II Costlier	COCOMO II Middle Case	Predicted Cost	ISBSG	BC Diff	CC Diff	MC Diff	Pred Diff	ISBSG Diff
Project 8	313	666,700	285,998	392,816	344,575	508,231	281,430	380,702	273,884	322,125	158,469	385,270
Project 10	326	750,825	298,593	410,115	359,750	527,642	293,362	452,232	340,710	391,075	223,183	457,463
Project 16	365	478,763	335,985	461,473	404,801	585,102	328,680	142,778	17,289	73,961	106,340	150,083
Project 39	368	561,863	339,084	465,730	408,535	589,854	331,601	222,778	96,133	153,328	27,991	230,262
Project 46	316	656,063	289,549	397,693	348,854	513,707	284,796	366,514	258,369	307,209	142,356	371,267
Project 62	366	437,688	337,035	462,916	406,067	586,713	329,670	100,652	25,229	31,621	149,025	108,018
Project 71	399	385,763	368,046	505,509	443,429	634,183	358,848	17,716	119,747	57,667	248,420	26,914
Average	351	562,523	322,041	442,322	388,002	563,633	315,484	240,482	161,623	190,998	150,826	247,039



Function Points in the range of 400-500 Function Points

Masked Project Na..	Adjusted Funtion Point Count	Actual Labor cost	COCOMO II Best Case	COCOMO II Costlier	COCOMO II Middle Case	Predicted Cost	ISBSG	BC Diff	CC Diff	MC Diff	Pred Diff	ISBSG Diff
Project 2	495	1,466,313	460,681	632,742	555,037	775,158	445,500	1,005,632	833,571	911,276	691,155	1,020,813
Project 9	461	990,323	427,458	587,112	515,010	724,730	414,504	562,864	403,211	475,312	265,592	575,819
Project 14	418	595,763	386,899	531,403	466,143	662,969	376,542	208,864	64,359	129,619	67,207	219,221
Project 15	474	757,938	440,644	605,222	530,896	744,761	426,816	317,294	152,716	227,041	13,177	331,122
Project 19	475	854,525	440,991	605,698	531,314	745,288	427,140	413,534	248,827	323,211	109,237	427,385
Project 20	480	496,225	445,737	612,217	537,032	752,492	431,568	50,488	115,992	40,807	256,267	64,657
Project 48	445	460,538	412,094	566,009	496,499	701,361	400,140	48,443	105,472	35,962	240,824	60,398
Average	464	803,089	430,643	591,486	518,847	729,537	417,459	372,446	274,878	306,175	234,780	385,630



Function Points above 500 Function Points

Masked Project Na..	Adjusted Funtion Point Count	Actual Labor cost	COCOMO II Best Case	COCOMO II Costlier	COCOMO II Middle Case	Predicted Cost	ISBSG	BC Diff	CC Diff	MC Diff	Pred Diff	ISBSG Diff
Project 4	965	1,372,325	920,958	1,264,931	1,109,588	1,463,122	868,365	451,367	107,394	262,737	90,797	503,960
Project 7	794	2,180,450	751,811	1,032,608	905,797	1,212,228	714,150	1,428,639	1,147,842	1,274,653	968,222	1,466,300
Project 13	876	1,923,820	833,094	1,144,250	1,003,728	1,333,026	788,400	1,090,726	779,570	920,092	590,794	1,135,420
Project 30	798	687,650	755,804	1,038,093	910,608	1,218,172	717,804	68,154	350,443	222,958	530,522	30,154
Project 36	1,379	1,891,750	1,334,608	1,833,076	1,607,961	2,070,114	1,241,460	557,142	58,674	283,789	178,364	650,290
Average	962	1,611,199	919,255	1,262,591	1,107,536	1,459,332	866,036	719,206	488,785	592,846	471,740	757,225

The results of the analysis is summarized below. On average, the COCOMO II best case estimate is the better estimate in the range of 0-100 function points, and the COCOMO II costlier estimate is the better estimate in the range of 100-200 function points. The predictive model offers the better estimate in all other ranges (200-300, 300-400, 400-500 and above 500 function points).

6.1.4 Implementation of the Estimating Tool:

We examine the performance of the estimating tool by examining its implementation in 20 projects. We selected 20 of the most recent projects executed at the organization and fitted them into multiple design patterns. We were dealing mostly with web development projects and created design patterns for

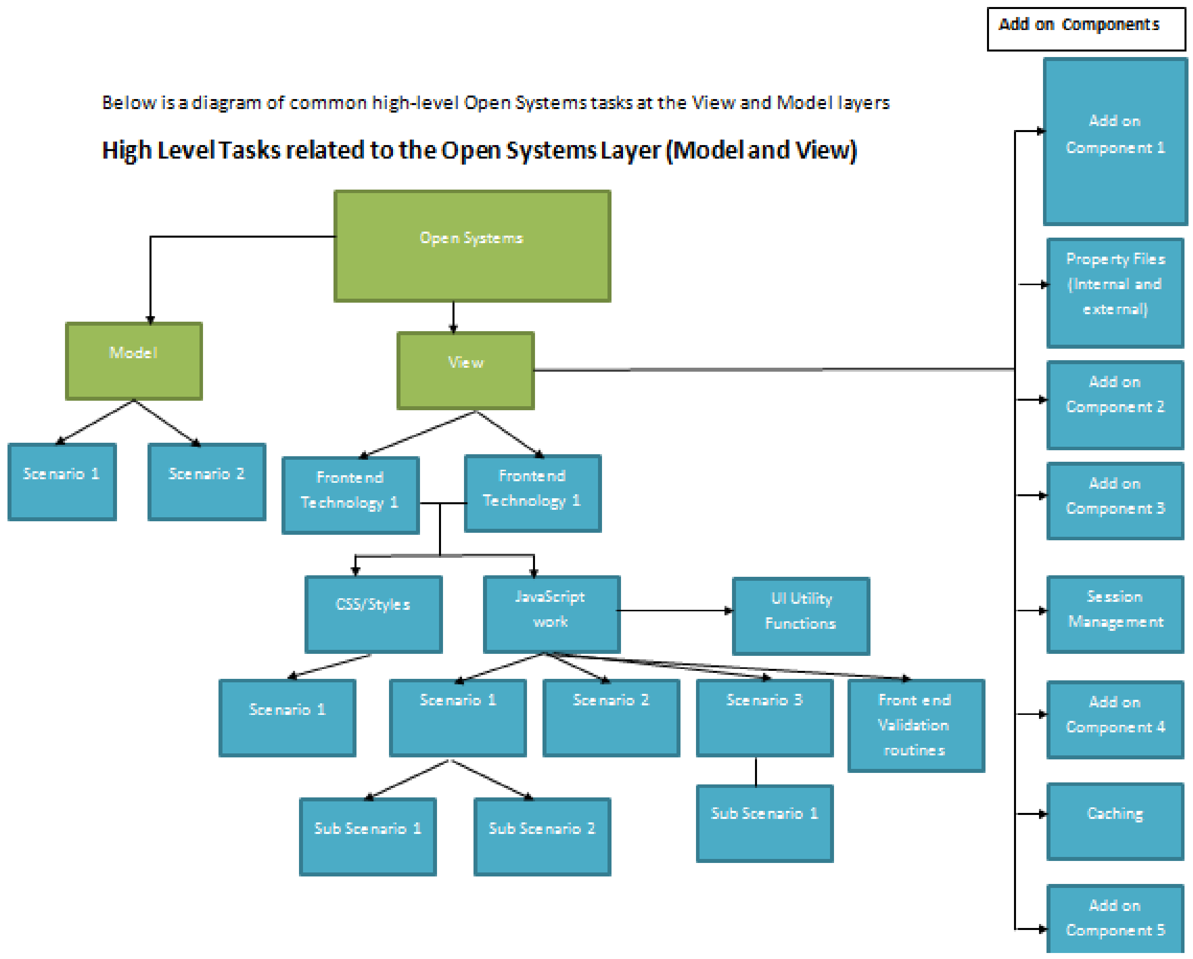
1. Backend Processing
2. Front end processing
3. Document processing
4. Batch processing

The process we followed is as follows:

1. I extracted out the design and development tasks for each project.
2. Read the description of each task and fit it into a scenario that was part of a bigger design pattern.
3. Documented tasks that were repeated in the same project as part of different scenarios. We could be retrieving data from multiple data sources, interfacing with multiple systems, etc.
4. We could have scenarios like retrieval of data, saving of data, etc. For example, I found the retrieval of data was done in different ways in multiple projects. I documented each of the methods to retrieve the data part of the design pattern. The same approach was used for other scenarios.
5. Each project had between 50-500 tasks. I looked at each individual task and fitted it into the design pattern.
6. Once we had the design patterns, I validated the design patterns with the two enterprise architects at the company and got their feedback.

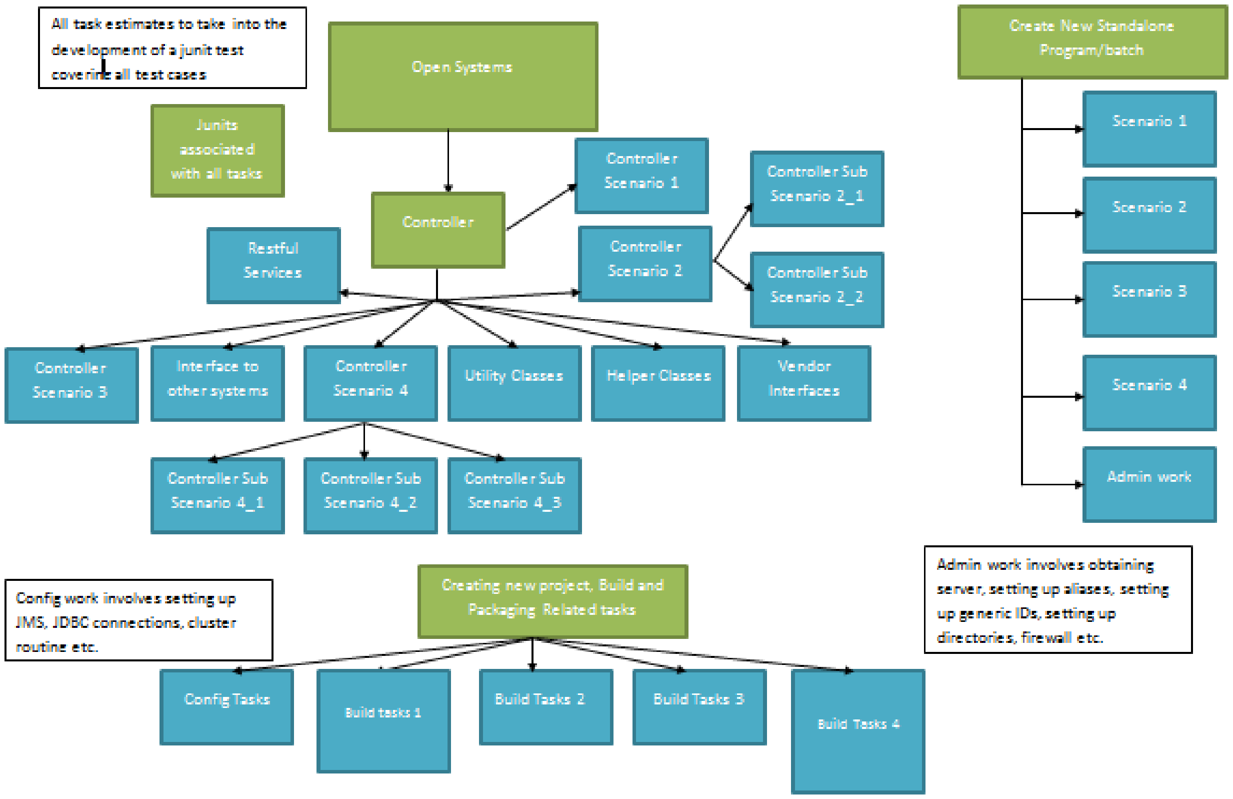
The design patterns that we came up with are listed below. Most of the scenarios have been masked to protect the actual design patterns practiced at the organization.

Figure 27: Masked high level design patterns at ABC Inc.

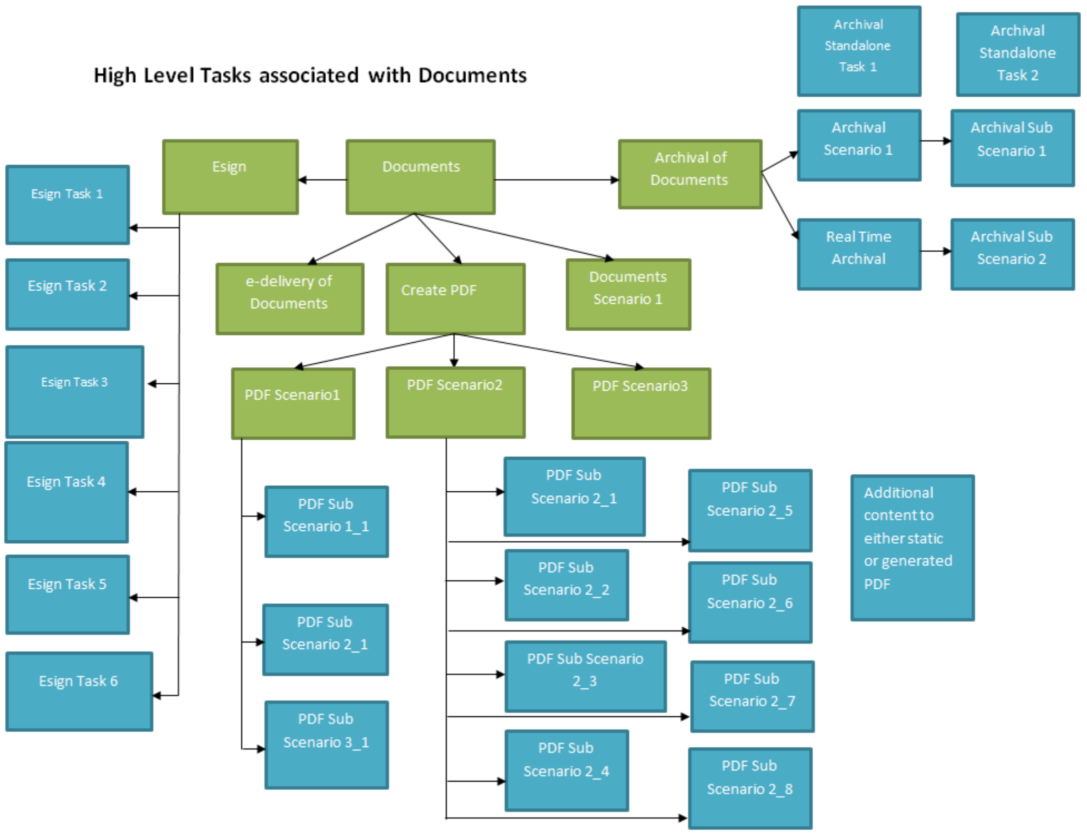


Below is a diagram of common high-level Open Systems tasks at the Controller Layer related to Application Development.

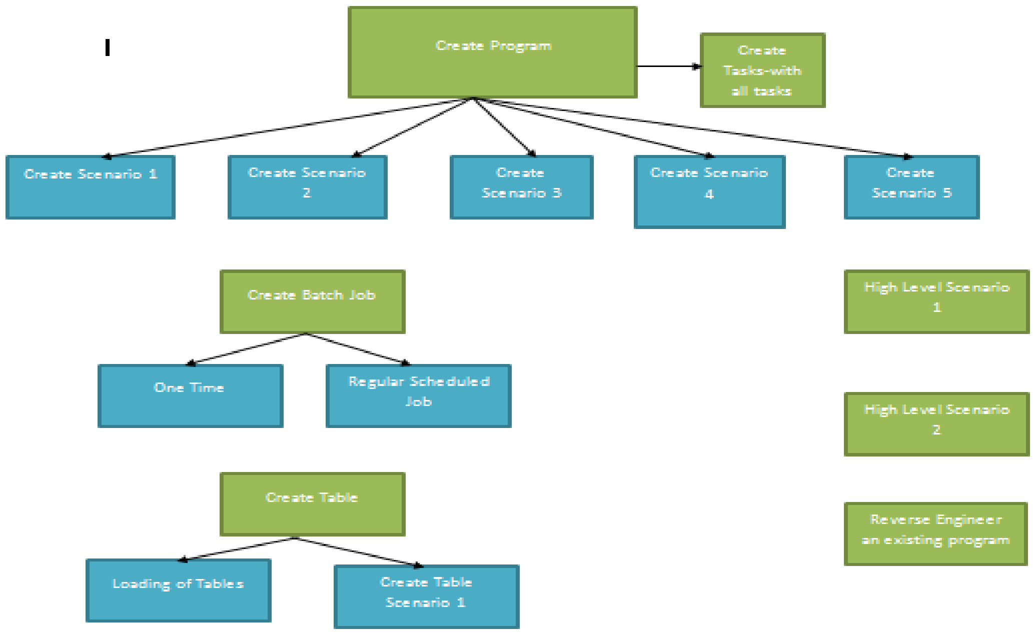
High Level Tasks related to the Open Systems Layer (Controller and other tasks)


All task estimates to take into the development of a junit test covering all test cases
Open Systems
Create New Standalone Program/batch
Scenario 1
Scenario 2
Scenario 3
Scenario 4
Admin work
Junits associated with all tasks
Controller
Controller Scenario 1
Controller Scenario 2
Controller Sub Scenario 2_1
Controller Sub Scenario 2_2
Restful Services
Controller Scenario 3
Interface to other systems
Controller Scenario 4
Utility Classes
Helper Classes
Vendor Interfaces
Controller Sub Scenario 4_1
Controller Sub Scenario 4_2
Controller Sub Scenario 4_3
Config work involves setting up JMS, JDBC connections, cluster routine etc.
Creating new project, Build and Packaging Related tasks
Admin work involves obtaining server, setting up aliases, setting up generic IDs, setting up directories, firewall etc.
Config Tasks
Build tasks 1
Build Tasks 2
Build Tasks 3
Build Tasks 4

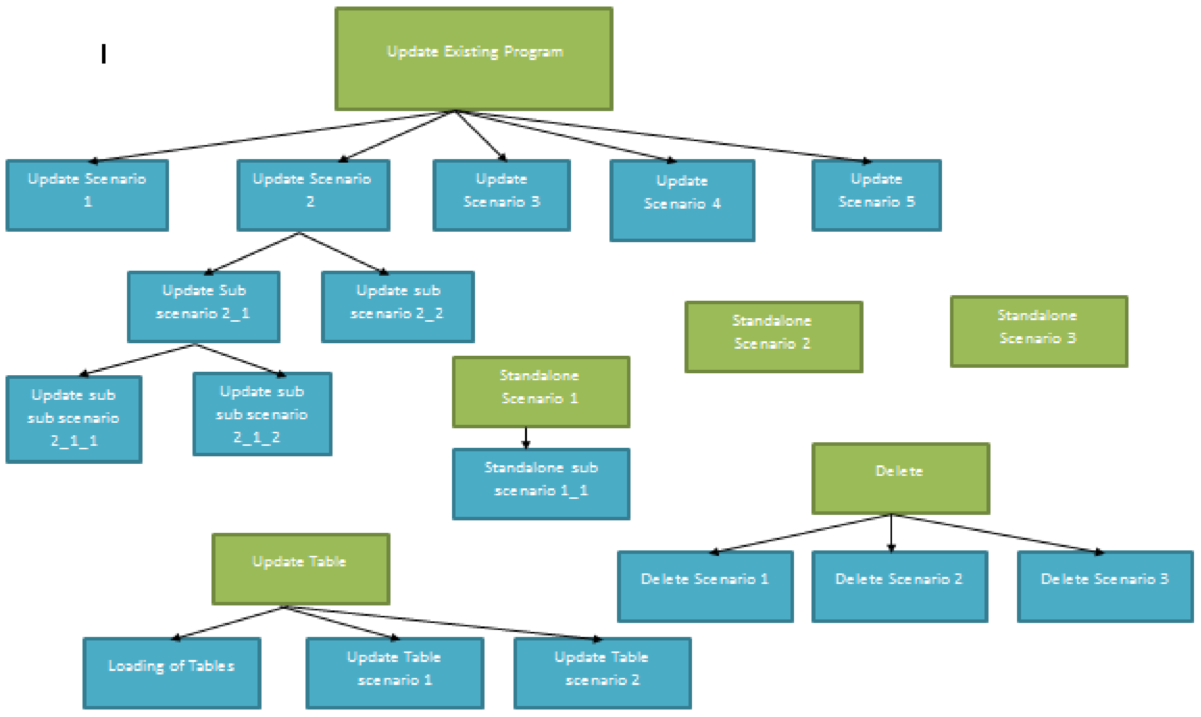
High Level Tasks associated with Documents



High Level Tasks related to all the things that we could create:



High Level Tasks related to all the things that we could Update:



There were two other design patterns like the above design patterns, and they have been submitted to the doctoral committee along with the other documents related to the dissertation.

These design patterns then formed the basis for a survey. The survey was sent out to all subject matter experts in the company.

Please see below a few questions from the survey. The questions have been masked in this snapshot. The actual full survey is being submitted to the doctoral committee, along with the other documents related to the dissertation. Each scenario in the survey was a direct result of the scenario being in the design pattern. We requested each subject matter expert to give estimates for design and development. They were also requested to give estimates considering the complexity of the scenario. We considered three levels for complexity: easy, medium and hard.

Sample Survey:

Time Estimates for Tasks related to Application Development

Please give your estimates on how much time employees would spend on design and development/unit testing when creating these new artifacts. Please include the time for writing junits into your estimate. Please estimate it for an employee doing the work.

Table 17: Sample survey

Description of Task			Design Hours	Development/ Unit Testing Hours	Are there Additional Items that Should Be Added to this Program Level Description?
Retrieval of data. We can retrieve data through four different ways	Retrieval Scenario 1	Easy			
		Medium			
		Hard			
	Retrieval Scenario 2	Easy			
		Medium			
		Hard			
	Retrieval Scenario 3	Easy			
		Medium			
		Hard			

	Retrieval Scenario 4	Easy			
		Medium			
		Hard			
Processing of the retrieved data to include business logic. This could include writing utility functions or helper classes to validate data on the open systems side and process it to be ready for the presentation layer.		Easy			
		Medium			
		Hard			
Saving of data including four scenarios	Save Scenario 1	Easy			
		Medium			
		Hard			
	Save Scenario 2	Easy			
		Medium			
		Hard			
	Save Scenario 3	Easy			
		Medium			
		Hard			
	Save Scenario 4	Easy			
		Medium			
		Hard			
External system service integration.		Easy			
		Medium			
		Hard			

Setting up a new project. Set up the repository. 1. Look into whether the artifact is for internal or external application? 2. And what other applications or artifacts it interacts with and the call volumes approximately. 3. What domain to deploy? 4. Will you need cluster routing?	Easy			
	Medium			
	Hard			

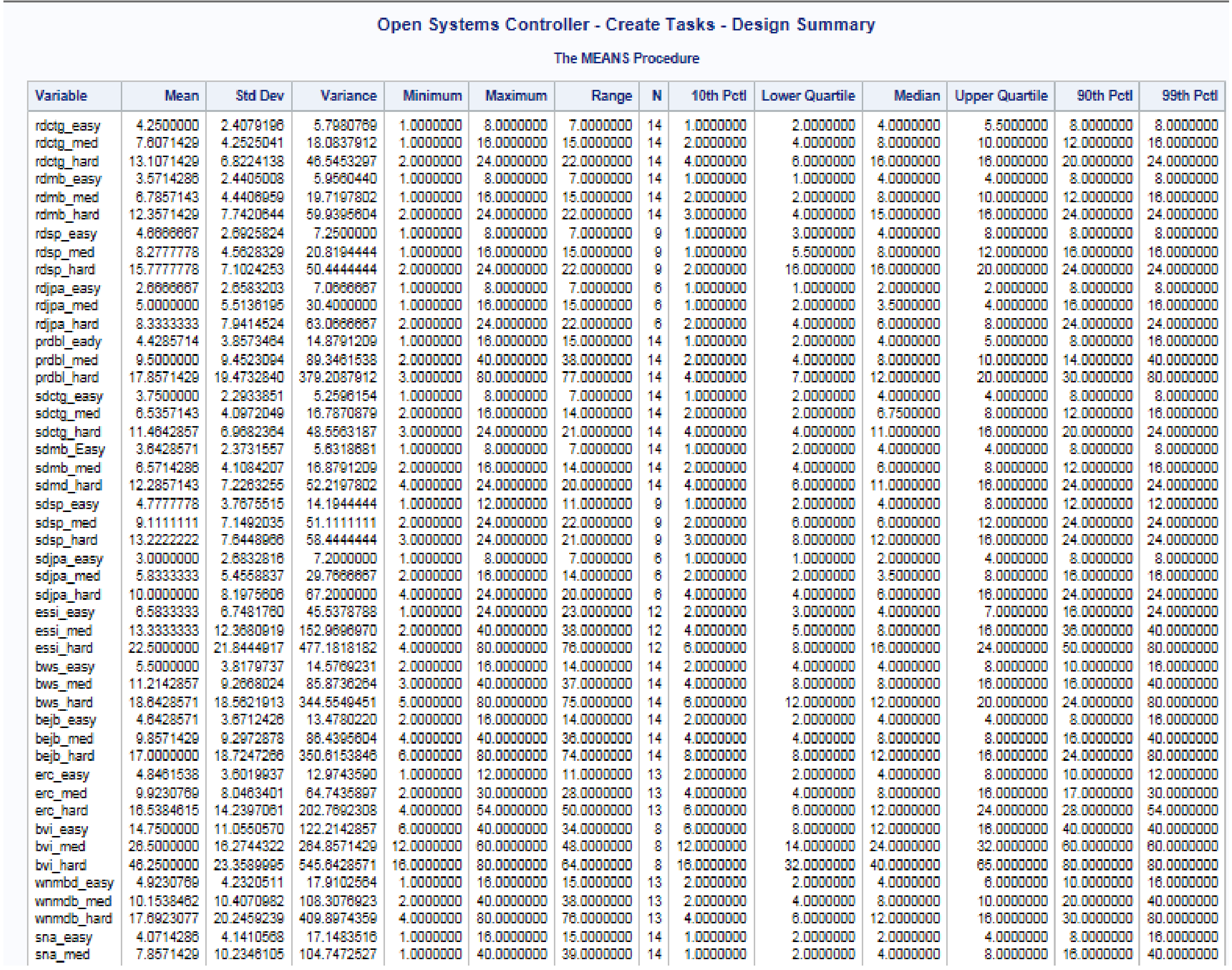
We had 95 scenarios each for design and development. We sent the survey to 30 subject matter experts and got responses back from 28 of them. The survey responses were collected and aggregated in the format shown in Table 18.

Table 18: Snippet of aggregated survey responses

rdctg_easy	rdctg_med	rdctg_hard	rdmb_easy	rdmb_med	rdmb_hard	rdsp_easy	rdsp_med	rdsp_hard	rdjpa_easy	rdjpa_med	rdjpa_hard	prdbl_eady
4	8	16	4	8	16	4	8	16	2	4	8	2
6	7.5	13.5	8	12	24	8	12	24				8
8	12	20	4	8	16	3	6	20	2	3	6	6
4	8	10	6	10	14							3
8	16	24	8	16	24	8	16	24	8	16	24	16
1	2	4	1	2	4	2	4	8				4
4	8	16	4	8	16							4
4	10	20	4	10	20							5
1	1	2	1	1	2	1	1	2	1	1	2	1
4	8	16	1	2	3	4	8	16				2
2	2	4	1	2	4				1	2	4	1
4	8	16	4	8	16	4	8	16				4
8	12	16	2	4	8	8	12	16				4
2	4	6	2	4	6				2	4	6	2

We aggregated the survey responses and processed them in SAS to generate summary statistics for each scenario at the complexity level. Table 19 illustrates these summary statistics for some of the scenarios.

Table 19: Processing of survey results in SAS



Open Systems Controller - Create Tasks - Design Summary

The MEANS Procedure

Variable	Mean	Std Dev	Variance	Minimum	Maximum	Range	N	10th Pctl	Lower Quartile	Median	Upper Quartile	90th Pctl	99th Pctl
rdctg_easy	4.2500000	2.4079196	5.7980769	1.0000000	8.0000000	7.0000000	14	1.0000000	2.0000000	4.0000000	5.5000000	8.0000000	8.0000000
rdctg_med	7.6071429	4.2525041	18.0837912	1.0000000	16.0000000	15.0000000	14	2.0000000	4.0000000	8.0000000	10.0000000	12.0000000	16.0000000
rdctg_hard	13.1071429	6.8224138	46.5453297	2.0000000	24.0000000	22.0000000	14	4.0000000	6.0000000	16.0000000	16.0000000	20.0000000	24.0000000
rdmb_easy	3.5714286	2.4405008	5.9560440	1.0000000	8.0000000	7.0000000	14	1.0000000	1.0000000	4.0000000	4.0000000	8.0000000	8.0000000
rdmb_med	6.7857143	4.4406959	19.7197802	1.0000000	16.0000000	15.0000000	14	2.0000000	2.0000000	8.0000000	10.0000000	12.0000000	16.0000000
rdmb_hard	12.3571429	7.7420644	59.9395604	2.0000000	24.0000000	22.0000000	14	3.0000000	4.0000000	15.0000000	16.0000000	24.0000000	24.0000000
rdsp_easy	4.6666667	2.6925824	7.2500000	1.0000000	8.0000000	7.0000000	9	1.0000000	3.0000000	4.0000000	8.0000000	8.0000000	8.0000000
rdsp_med	8.2777778	4.5628329	20.8194444	1.0000000	16.0000000	15.0000000	9	1.0000000	5.5000000	8.0000000	12.0000000	16.0000000	16.0000000
rdsp_hard	15.7777778	7.1024253	50.4444444	2.0000000	24.0000000	22.0000000	9	2.0000000	16.0000000	16.0000000	20.0000000	24.0000000	24.0000000
rdjpa_easy	2.6666667	2.6583203	7.0666667	1.0000000	8.0000000	7.0000000	6	1.0000000	1.0000000	2.0000000	2.0000000	8.0000000	8.0000000
rdjpa_med	5.0000000	5.5136195	30.4000000	1.0000000	16.0000000	15.0000000	6	1.0000000	2.0000000	3.5000000	4.0000000	16.0000000	16.0000000
rdjpa_hard	8.3333333	7.9414524	63.0666667	2.0000000	24.0000000	22.0000000	6	2.0000000	4.0000000	6.0000000	8.0000000	24.0000000	24.0000000
prdbl_eady	4.4285714	3.8573464	14.8791209	1.0000000	16.0000000	15.0000000	14	1.0000000	2.0000000	4.0000000	5.0000000	8.0000000	16.0000000
prdbl_med	9.5000000	9.4523094	89.3461538	2.0000000	40.0000000	38.0000000	14	2.0000000	4.0000000	8.0000000	10.0000000	14.0000000	40.0000000
prdbl_hard	17.8571429	19.4732840	379.2087912	3.0000000	80.0000000	77.0000000	14	4.0000000	7.0000000	12.0000000	20.0000000	30.0000000	80.0000000
sdctg_easy	3.7500000	2.2933851	5.2596154	1.0000000	8.0000000	7.0000000	14	1.0000000	2.0000000	4.0000000	4.0000000	8.0000000	8.0000000
sdctg_med	6.5357143	4.0972049	16.7870879	2.0000000	16.0000000	14.0000000	14	2.0000000	2.0000000	6.7500000	8.0000000	12.0000000	16.0000000
sdctg_hard	11.4642857	6.9682364	48.5563187	3.0000000	24.0000000	21.0000000	14	4.0000000	4.0000000	11.0000000	16.0000000	20.0000000	24.0000000
sdmb_Easy	3.6428571	2.3731557	5.6318681	1.0000000	8.0000000	7.0000000	14	1.0000000	2.0000000	4.0000000	4.0000000	8.0000000	8.0000000
sdmb_med	6.5714286	4.1084207	16.8791209	2.0000000	16.0000000	14.0000000	14	2.0000000	4.0000000	6.0000000	8.0000000	12.0000000	16.0000000
sdmd_hard	12.2857143	7.2263255	52.2197802	4.0000000	24.0000000	20.0000000	14	4.0000000	6.0000000	11.0000000	16.0000000	24.0000000	24.0000000
sdsp_easy	4.7777778	3.7675515	14.1944444	1.0000000	12.0000000	11.0000000	9	1.0000000	2.0000000	4.0000000	8.0000000	12.0000000	12.0000000
sdsp_med	9.1111111	7.1492035	51.1111111	2.0000000	24.0000000	22.0000000	9	2.0000000	6.0000000	6.0000000	12.0000000	24.0000000	24.0000000
sdsp_hard	13.2222222	7.6448966	58.4444444	3.0000000	24.0000000	21.0000000	9	3.0000000	8.0000000	12.0000000	16.0000000	24.0000000	24.0000000
sdjpa_easy	3.0000000	2.6832816	7.2000000	1.0000000	8.0000000	7.0000000	6	1.0000000	1.0000000	2.0000000	4.0000000	8.0000000	8.0000000
sdjpa_med	5.8333333	5.4558837	29.7666667	2.0000000	16.0000000	14.0000000	6	2.0000000	2.0000000	3.5000000	8.0000000	16.0000000	16.0000000
sdjpa_hard	10.0000000	8.1975606	67.2000000	4.0000000	24.0000000	20.0000000	6	4.0000000	4.0000000	6.0000000	16.0000000	24.0000000	24.0000000
essi_easy	6.5833333	6.7481760	45.5378788	1.0000000	24.0000000	23.0000000	12	2.0000000	3.0000000	4.0000000	7.0000000	16.0000000	24.0000000
essi_med	13.3333333	12.3680919	152.9696970	2.0000000	40.0000000	38.0000000	12	4.0000000	5.0000000	8.0000000	16.0000000	36.0000000	40.0000000
essi_hard	22.5000000	21.8444917	477.1818182	4.0000000	80.0000000	76.0000000	12	6.0000000	8.0000000	16.0000000	24.0000000	50.0000000	80.0000000
bws_easy	5.5000000	3.8179737	14.5769231	2.0000000	16.0000000	14.0000000	14	2.0000000	4.0000000	4.0000000	8.0000000	10.0000000	16.0000000
bws_med	11.2142857	9.2668024	85.8736264	3.0000000	40.0000000	37.0000000	14	4.0000000	8.0000000	8.0000000	16.0000000	16.0000000	40.0000000
bws_hard	18.6428571	18.5621913	344.5549451	5.0000000	80.0000000	75.0000000	14	6.0000000	12.0000000	12.0000000	20.0000000	24.0000000	80.0000000
bejb_easy	4.6428571	3.6712426	13.4780220	2.0000000	16.0000000	14.0000000	14	2.0000000	2.0000000	4.0000000	4.0000000	8.0000000	16.0000000
bejb_med	9.8571429	9.2972878	86.4395604	4.0000000	40.0000000	36.0000000	14	4.0000000	4.0000000	8.0000000	8.0000000	16.0000000	40.0000000
bejb_hard	17.0000000	18.7247266	350.6153846	6.0000000	80.0000000	74.0000000	14	8.0000000	8.0000000	12.0000000	16.0000000	24.0000000	80.0000000
erc_easy	4.8461538	3.6019937	12.9743590	1.0000000	12.0000000	11.0000000	13	2.0000000	2.0000000	4.0000000	8.0000000	10.0000000	12.0000000
erc_med	9.9230769	8.0463401	64.7435897	2.0000000	30.0000000	28.0000000	13	4.0000000	4.0000000	8.0000000	16.0000000	17.0000000	30.0000000
erc_hard	16.5384615	14.2397061	202.7692308	4.0000000	54.0000000	50.0000000	13	6.0000000	6.0000000	12.0000000	24.0000000	28.0000000	54.0000000
bwi_easy	14.7500000	11.0550570	122.2142857	6.0000000	40.0000000	34.0000000	8	6.0000000	8.0000000	12.0000000	16.0000000	40.0000000	40.0000000
bwi_med	26.5000000	16.2744322	264.8571429	12.0000000	60.0000000	48.0000000	8	12.0000000	14.0000000	24.0000000	32.0000000	60.0000000	60.0000000
bwi_hard	46.2500000	23.3589995	545.6428571	16.0000000	80.0000000	64.0000000	8	16.0000000	32.0000000	40.0000000	65.0000000	80.0000000	80.0000000
wnmbd_easy	4.9230769	4.2320511	17.9102564	1.0000000	16.0000000	15.0000000	13	2.0000000	2.0000000	4.0000000	6.0000000	10.0000000	16.0000000
wnmdb_med	10.1538462	10.4070982	108.3076923	2.0000000	40.0000000	38.0000000	13	2.0000000	4.0000000	8.0000000	10.0000000	20.0000000	40.0000000
wnmdb_hard	17.6923077	20.2459239	409.8974359	4.0000000	80.0000000	76.0000000	13	4.0000000	6.0000000	12.0000000	16.0000000	30.0000000	80.0000000
sna_easy	4.0714286	4.1410568	17.1483516	1.0000000	16.0000000	15.0000000	14	1.0000000	2.0000000	2.0000000	4.0000000	8.0000000	16.0000000
sna_med	7.8571429	10.2346105	104.7472527	1.0000000	40.0000000	39.0000000	14	1.0000000	2.0000000	4.0000000	8.0000000	16.0000000	40.0000000

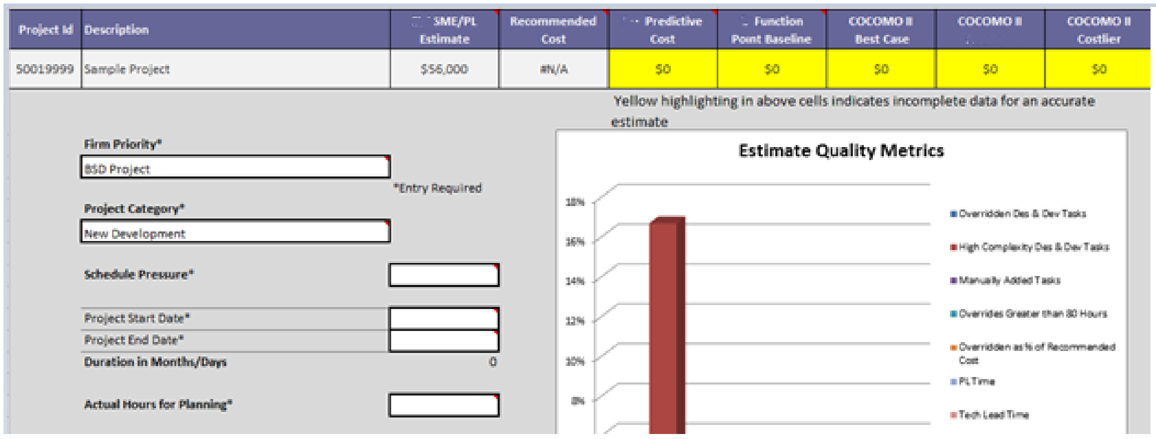
We then went back to the two enterprise architects at the organization to review the results. All of us finally agreed to baseline estimates for each design and development tasks that comprised the scenarios in the design patterns.

Once the results from the survey were processed, we were ready to develop the estimating tool.

6.1.5 Creation of the Estimating Tool

The estimating tool was created in Excel and. incorporated all the requirements mentioned in chapter 4.1.5. I worked with the program management office at the organization and worked with a couple of project leaders to develop and finalize the estimating tool. Figures 28 through 31 illustrate the estimating tool interface and outputs.

Figure 28 - Estimate Comparative tab in Estimating Tool



Project Id	Description	SME/PL Estimate	Recommended Cost	Predictive Cost	Function Point Baseline	COCOMO II Best Case	COCOMO II	COCOMO II Costlier
50019999	Sample Project	$56,000	#N/A	$0	$0	$0	$0	$0

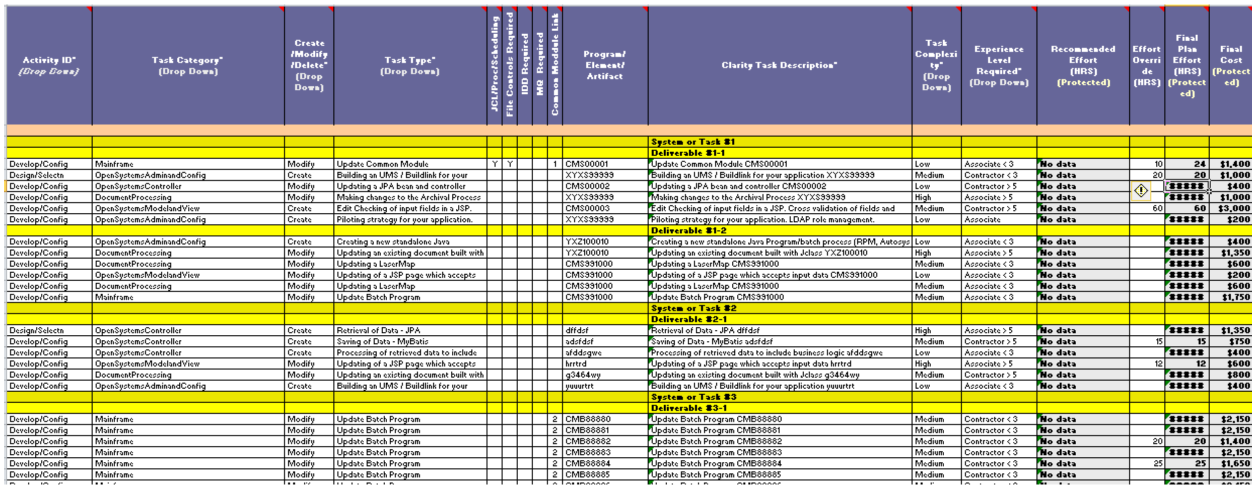
Yellow highlighting in above cells indicates incomplete data for an accurate estimate

Firm Priority*	BSD Project
Project Category*	New Development
Schedule Pressure*	
Project Start Date*	
Project End Date*	
Duration in Months/Days	0
Actual Hours for Planning*	

*Entry Required

Estimate Quality Metrics
18%
16%
14%
12%
10%
8%
Overridden Des & Dev Tasks
High Complexity Des & Dev Tasks
Manually Added Tasks
Overrides Greater than 80 Hours
Overridden as % of Recommended Cost
PL Time
Tech Lead Time

Figure 29: Design and development tasks entered by the SME



Activity ID* (Drop Down)	Task Category* (Drop Down)	Create /Modify /Delete* (Drop Down)	Task Type* (Drop Down)	JCL/Proc/Scheduling	File Controls Required	IDD Required	MQ Required	Common Module Link	Program/ Element/ Artifact	Clarity Task Description*	Task Complexity* (Drop Down)	Experience Level Required* (Drop Down)	Recommended Effort (HRS) (Protected)	Effort Override (HRS)	Final Plan Effort (HRS) (Protected)	Final Cost (Protected)
										System or Task #1						
										Deliverable #1-1						
Develop/Config	Mainframe	Modify	Update Common Module	Y	Y			1	CMS00001	Update Common Module CMS00001	Low	Associate < 3	No data	10	24	$1,400
Design/Selectn	OpenSystemsAdminandConfig	Create	Building an UMS / Buildlink for your						XYXS99999	Building an UMS / Buildlink for your application XYXS99999	Medium	Contractor < 3	No data	20	20	$1,000
Develop/Config	OpenSystemsController	Modify	Updating a JPA bean and controller						CMS00002	Updating a JPA bean and controller CMS00002	Low	Contractor > 5	No data		#####	$400
Develop/Config	DocumentProcessing	Modify	Making changes to the Archival Process						XYXS99999	Making changes to the Archival Process XYXS99999	High	Associate > 5	No data		#####	$1,000
Develop/Config	OpenSystemsModelandView	Create	Edit Checking of input fields in a JSP.						CMS00003	Edit Checking of input fields in a JSP. Cross validation of fields and	Medium	Contractor > 5	No data	60	60	$3,000
Develop/Config	OpenSystemsAdminandConfig	Create	Piloting strategy for your application.						XYXS99999	Piloting strategy for your application. LDAP role management.	Low	Associate	No data		#####	$200
										Deliverable #1-2						
Develop/Config	OpenSystemsAdminandConfig	Create	Creating a new standalone Java						YXZ100010	Creating a new standalone Java Program/batch process (RPM, Autosys	Low	Associate < 3	No data		#####	$400
Develop/Config	DocumentProcessing	Modify	Updating an existing document built with						YXZ100010	Updating an existing document built with Jclass YXZ100010	High	Associate > 5	No data		#####	$1,350
Develop/Config	DocumentProcessing	Modify	Updating a LaserMap						CMS991000	Updating a LaserMap CMS991000	Medium	Associate < 3	No data		#####	$600
Develop/Config	OpenSystemsModelandView	Modify	Updating of a JSP page which accepts						CMS991000	Updating of a JSP page which accepts input data CMS991000	Low	Associate < 3	No data		#####	$200
Develop/Config	DocumentProcessing	Modify	Updating a LaserMap						CMS991000	Updating a LaserMap CMS991000	Medium	Associate < 3	No data		#####	$600
Develop/Config	Mainframe	Modify	Update Batch Program						CMS991000	Update Batch Program CMS991000	Medium	Associate < 3	No data		#####	$1,750
										System or Task #2						
										Deliverable #2-1						
Design/Selectn	OpenSystemsController	Create	Retrieval of Data - JPA						dffdsf	Retrieval of Data - JPA dffdsf	High	Associate > 5	No data		#####	$1,350
Develop/Config	OpenSystemsController	Create	Saving of Data - MyBatis						adsfdsf	Saving of Data - MyBatis adsfdsf	Medium	Contractor > 5	No data	15	15	$750
Develop/Config	OpenSystemsController	Create	Processing of retrieved data to include						afddsgwe	Processing of retrieved data to include business logic afddsgwe	Low	Associate < 3	No data		#####	$400
Develop/Config	OpenSystemsModelandView	Modify	Updating of a JSP page which accepts						hrrtrd	Updating of a JSP page which accepts input data hrrtrd	High	Associate > 5	No data	12	12	$600
Develop/Config	DocumentProcessing	Modify	Updating an existing document built with						g3464wy	Updating an existing document built with Jclass g3464wy	Medium	Contractor > 5	No data		#####	$800
Develop/Config	OpenSystemsAdminandConfig	Create	Building an UMS / Buildlink for your						yuuurtrt	Building an UMS / Buildlink for your application yuuurtrt	Low	Associate < 3	No data		#####	$400
										System or Task #3						
										Deliverable #3-1						
Develop/Config	Mainframe	Modify	Update Batch Program					2	CMB88880	Update Batch Program CMB88880	Medium	Contractor < 3	No data		#####	$2,150
Develop/Config	Mainframe	Modify	Update Batch Program					2	CMB88881	Update Batch Program CMB88881	Medium	Contractor < 3	No data		#####	$2,150
Develop/Config	Mainframe	Modify	Update Batch Program					2	CMB88882	Update Batch Program CMB88882	Medium	Contractor < 3	No data	20	20	$1,400
Develop/Config	Mainframe	Modify	Update Batch Program					2	CMB88883	Update Batch Program CMB88883	Medium	Contractor < 3	No data		#####	$2,150
Develop/Config	Mainframe	Modify	Update Batch Program					2	CMB88884	Update Batch Program CMB88884	Medium	Contractor < 3	No data	25	25	$1,650
Develop/Config	Mainframe	Modify	Update Batch Program					2	CMB88885	Update Batch Program CMB88885	Medium	Contractor < 3	No data		#####	$2,150

Figure 30: Pie charts in the overhead and summaries tab of Estimating Tool

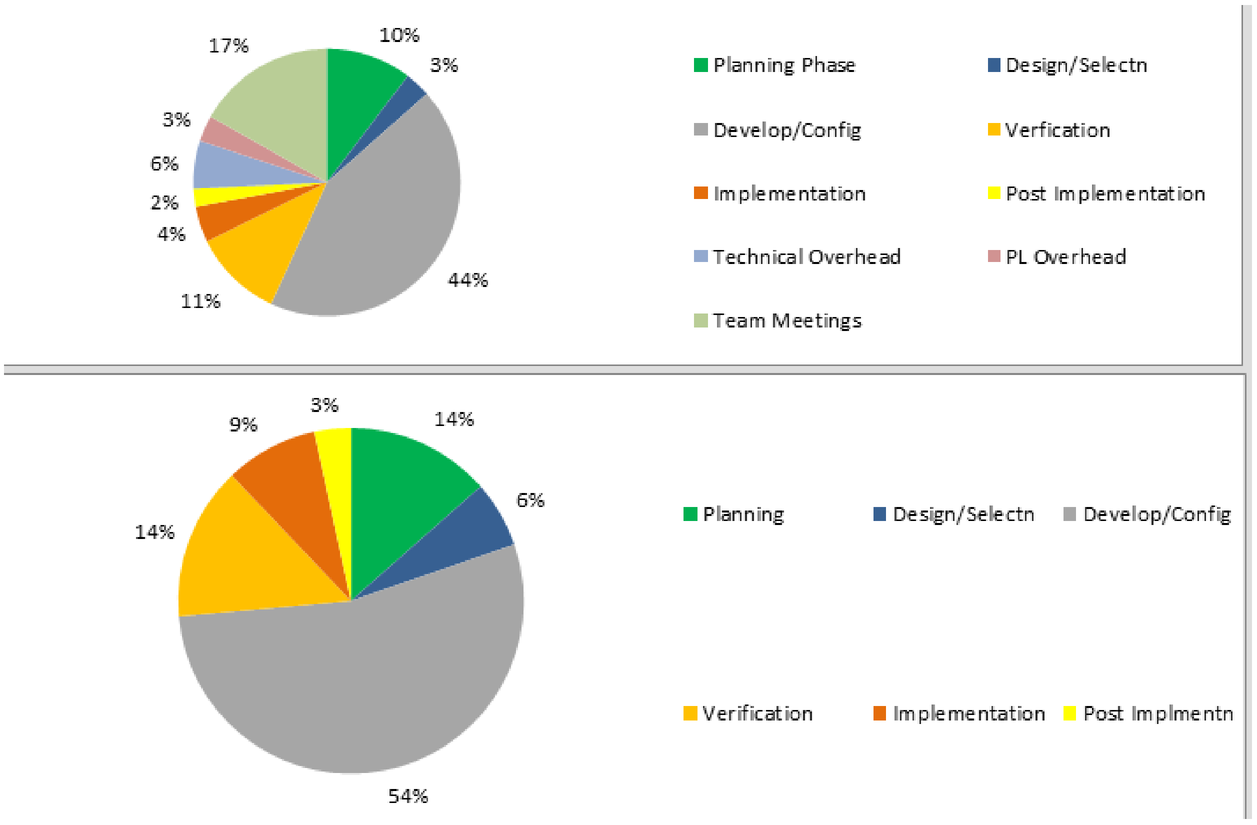
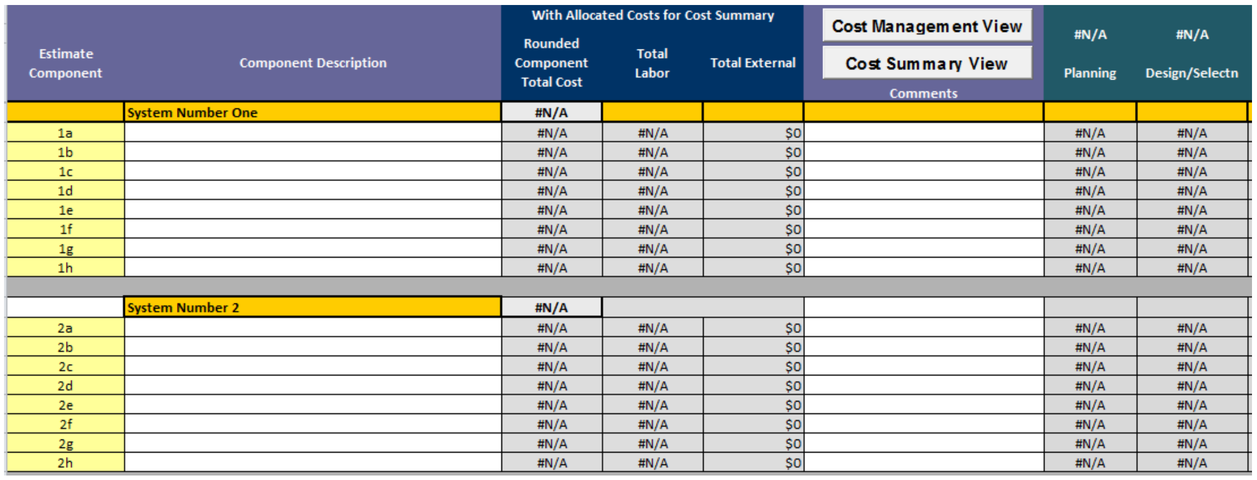


Figure 31: Cost breakdown summary in Estimating Tool

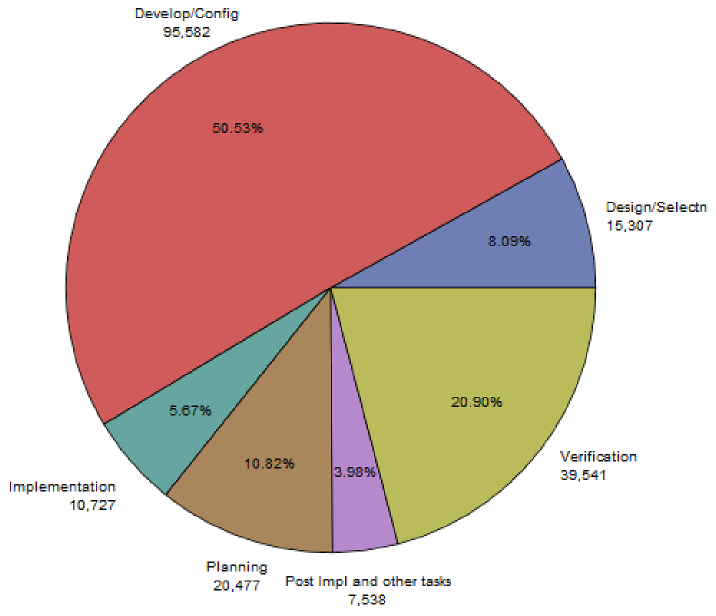


		With Allocated Costs for Cost Summary			Cost Management View	#N/A	#N/A
Estimate Component	Component Description	Rounded Component Total Cost	Total Labor	Total External	Cost Summary View Comments	Planning	Design/Selectn
	System Number One	#N/A					
1a		#N/A	#N/A	$0		#N/A	#N/A
1b		#N/A	#N/A	$0		#N/A	#N/A
1c		#N/A	#N/A	$0		#N/A	#N/A
1d		#N/A	#N/A	$0		#N/A	#N/A
1e		#N/A	#N/A	$0		#N/A	#N/A
1f		#N/A	#N/A	$0		#N/A	#N/A
1g		#N/A	#N/A	$0		#N/A	#N/A
1h		#N/A	#N/A	$0		#N/A	#N/A
	System Number 2	#N/A					
2a		#N/A	#N/A	$0		#N/A	#N/A
2b		#N/A	#N/A	$0		#N/A	#N/A
2c		#N/A	#N/A	$0		#N/A	#N/A
2d		#N/A	#N/A	$0		#N/A	#N/A
2e		#N/A	#N/A	$0		#N/A	#N/A
2f		#N/A	#N/A	$0		#N/A	#N/A
2g		#N/A	#N/A	$0		#N/A	#N/A
2h		#N/A	#N/A	$0		#N/A	#N/A

A copy of the masked estimating tool is attached in the Appendix.

We analyzed the project data for the past 12 months and the breakdown of the overall hours by phases in shown below in Figure 32.

Figure 32: Project data by phases for the past 12 months



The estimating tool was setup to allocate time for verification, implementation, and post implementation based on how much time was estimated for the design and development phases. The numbers we agreed on were

1. Verification is allocated as 25% of development time.
2. Implementation is allocated as 10% of development time.
3. Post Implementation is allocated as 5% of development time.

6.1.6 Summary of the new estimating Process:

The organization came up with a revised process to incorporate the use of the estimating tool in the project life cycle. Every project starts off with a planning phase, followed by the definition of requirements by the business area for the project. This is followed by a system design by the technical team. The function point analysis is completed around this time. At this point, there is sufficient information to create a bottom-up estimate using the new estimating tool. The subject matter experts come up with a system design and translate the design into high level tasks that are then input into the estimating tool. It is often the case where there are multiple subject matter experts involved in a project, and each of them estimate a separate feature of the project.

1. The SME's start on the "Cost Projection Form" tab and set up basic demographic information about the project.
2. They then move onto the "Design and Development Estimates" tab to enter the task level estimate. The following fields are required for each task
    a. Activity Type: Indicates if the task is a design or a development task.
    b. Task Category: This indicates the high-level design pattern to which the task belongs. Some of the options include Open System Controller, Open Systems Model, and View, Document Processing, Mobile, etc.
    c. Create/Modify: This indicates whether the task involves creating a new artifact or modifying an existing artifact.

d. Task Type: The values in this drop-down are derived directly from the design patterns that we came up with. The values in the dropdown are populated based on the entries for the previous three fields.

e. Task Complexity: The options for this field are high, medium, and low.

f. Task Description: This is the description for the task that is used in the time tracking system.

Once you enter the data for the above mentioned six fields, you get a recommended estimate. This estimate initially is based on the baseline for each task that originates from the survey results and discussion with the enterprise architects at the organization.

g. Recommended Estimate: This field is automatically populated when the above mentioned six fields are entered or chosen.

h. Override Estimate: The SME can choose to override the estimate at the task level, and when they do so, the override value takes effect for the task.

i. Override Comments: The SME is required to enter comments when a task is overridden.

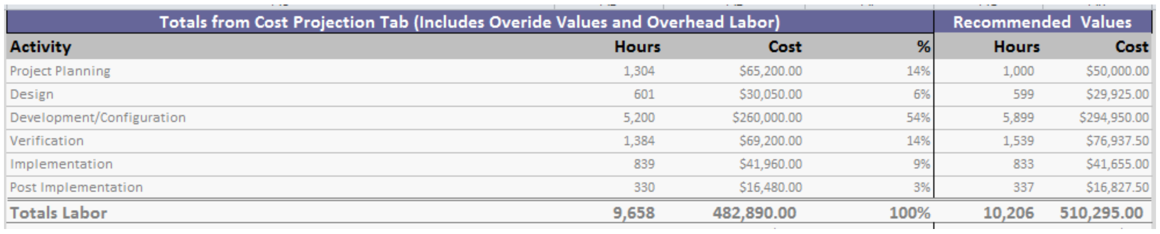
j. Estimate Component: This field is used to break down the cost by features.

k. Requirement: Some project managers like to track the project by requirements. This field gives the capability to enter the requirement number related to the task.

l. Owner: This field indicates the name of the SME doing the estimate and is considered the owner of the task.

As each task is entered, a running total of the overall cost for the project is shown at the top of the design and development estimates tab, as shown below in Figure 33.

Figure 33: Cost projection table in Estimating Tool



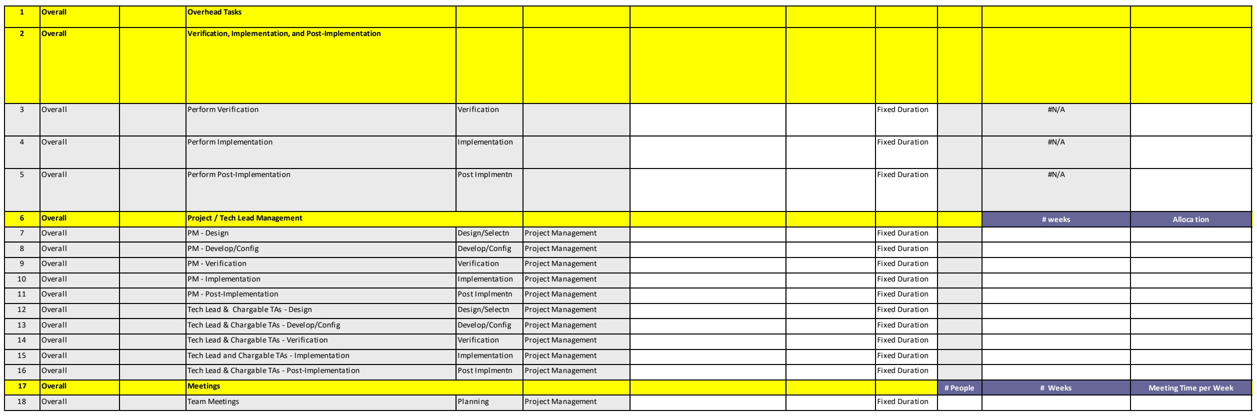
Totals from Cost Projection Tab (Includes Overide Values and Overhead Labor)				Recommended Values	
Activity	**Hours**	**Cost**	**%**	**Hours**	**Cost**
Project Planning	1,304	$65,200.00	14%	1,000	$50,000.00
Design	601	$30,050.00	6%	599	$29,925.00
Development/Configuration	5,200	$260,000.00	54%	5,899	$294,950.00
Verification	1,384	$69,200.00	14%	1,539	$76,937.50
Implementation	839	$41,960.00	9%	833	$41,655.00
Post Implementation	330	$16,480.00	3%	337	$16,827.50
Totals Labor	**9,658**	**482,890.00**	**100%**	**10,206**	**510,295.00**

At any given point of time, we can see a comparison of the SME estimate (this includes the overridden cost for each task) and the recommended cost (this does not include the overrides).

m. There are times when a task does not fit into one of the design patterns. In this case, there is always the option to "enter a new task" and manually put in the estimate for that task. When we see multiple application development teams requesting for a new task that is similar, we add it to the design pattern of the organization and incorporate it into the estimating tool.

n. The tool also gives the capability to override the verification, implementation, and post-implementation time in the "Overhead and Summaries" tab. You also have the capability to enter the project management time, tech lead time and meeting time in the "Overhead and Summaries" tab as shown below in Figure 34.

Figure 34: Overhead and Summaries tab in Estimating Tool

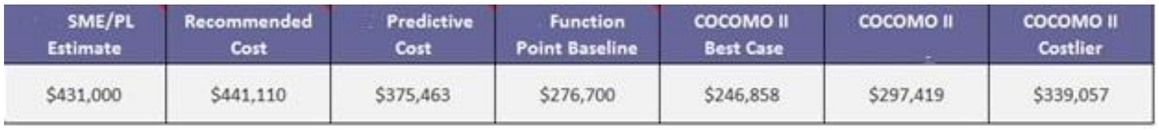


1	**Overall**		**Overhead Tasks**								
2	**Overall**		**Verification, Implementation, and Post-Implementation**								
3	Overall		Perform Verification	Verification				Fixed Duration		#N/A	
4	Overall		Perform Implementation	Implementation				Fixed Duration		#N/A	
5	Overall		Perform Post-Implementation	Post Implmentn				Fixed Duration		#N/A	
6	**Overall**		**Project / Tech Lead Management**							**# weeks**	**Alloca tion**
7	Overall		PM - Design	Design/Selectn	Project Management			Fixed Duration			
8	Overall		PM - Develop/Config	Develop/Config	Project Management			Fixed Duration			
9	Overall		PM - Verification	Verification	Project Management			Fixed Duration			
10	Overall		PM - Implementation	Implementation	Project Management			Fixed Duration			
11	Overall		PM - Post-Implementation	Post Implmentn	Project Management			Fixed Duration			
12	Overall		Tech Lead & Chargable TAs - Design	Design/Selectn	Project Management			Fixed Duration			
13	Overall		Tech Lead & Chargable TAs - Develop/Config	Develop/Config	Project Management			Fixed Duration			
14	Overall		Tech Lead & Chargable TAs - Verification	Verification	Project Management			Fixed Duration			
15	Overall		Tech Lead and Chargable TAs - Implementation	Implementation	Project Management			Fixed Duration			
16	Overall		Tech Lead & Chargable TAs - Post-Implementation	Post Implmentn	Project Management			Fixed Duration			
17	**Overall**		**Meetings**						**# People**	**# Weeks**	**Meeting Time per Week**
18	Overall		Team Meetings	Planning	Project Management			Fixed Duration			

o. At this point, the subject matter expert works with the project manager and other managers in the organization to explain the estimate.

p. The project manager now moves on to the "Estimate Comparative" tab and enters the adjusted function point number. You now have access to six of the seven estimates in the suite of estimates. The first two estimates are directly from the data that is entered in the "Design and Development Estimates" tab. The last four estimates are based off the function point's count. The suite of seven estimates is shown below in Figure 35

Figure 35: Suite of seven estimates

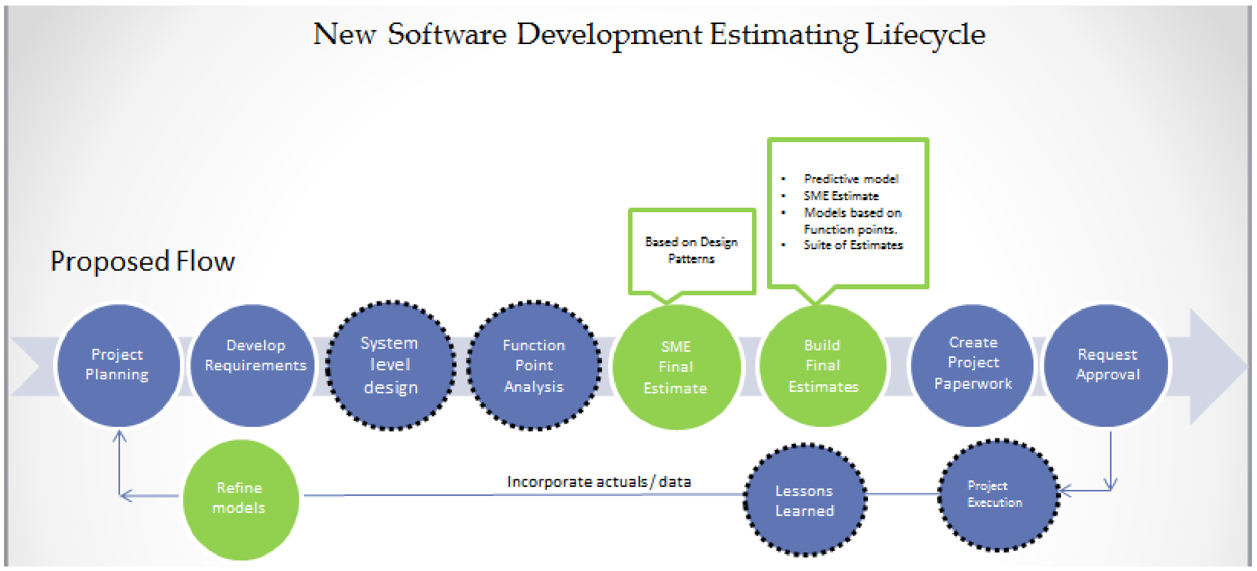


SME/PL Estimate	Recommended Cost	Predictive Cost	Function Point Baseline	COCOMO II Best Case	COCOMO II	COCOMO II Costlier
$431,000	$441,110	$375,463	$276,700	$246,858	$297,419	$339,057

q. There are other fields that are required to be entered on the estimates comparative tab, and this populates the predictive cost for the organization. This is based on the historical data at the organization.

A representation of the new estimating lifecycle is outlined below in Figure 36.

Figure 36: New software development estimating lifecycle



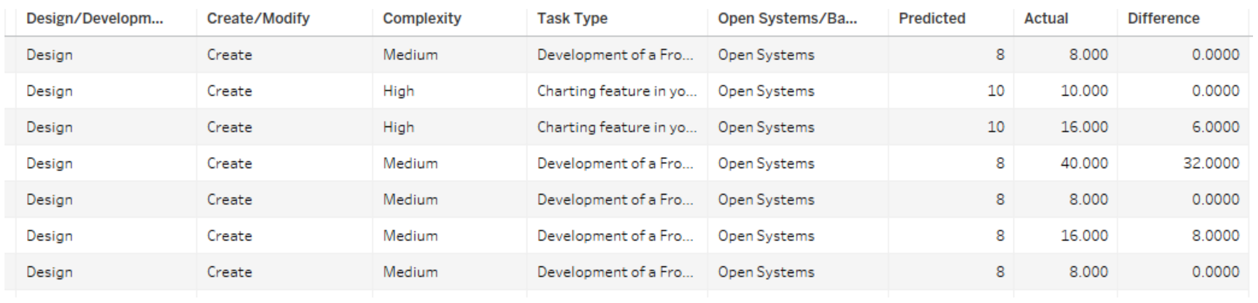
6.2 Assessing the Quality of the Estimate from the Estimating Tool

The estimating tool has been used to estimate close to 25 projects thus far. Five of those projects have completed execution. The projects ranged in size from 451 hours to 3818 hours to complete the design and development phase of the project. This section of the chapter details the processing of the data to evaluate the estimating tool.

Processing of the actual data

Previously there was no framework to process the actual data from the execution of the project and compare it with the initial estimation data. A framework has now been created where this data can be processed seamlessly, and the data can be fed into a Tableau dashboard. All we need to do is update an Excel worksheet into a shared location on the network when a new project has data to be harvested, and Tableau will extract the information nightly. A hidden field was created in the estimating tool to capture the characteristics of the task, and that information was uploaded into the time tracking tool. When we extract the actual data, we now have the capability to pull down the hidden field as well. We then parse out this hidden field to create the framework for reporting in Tableau, as shown in Figure 37 below.

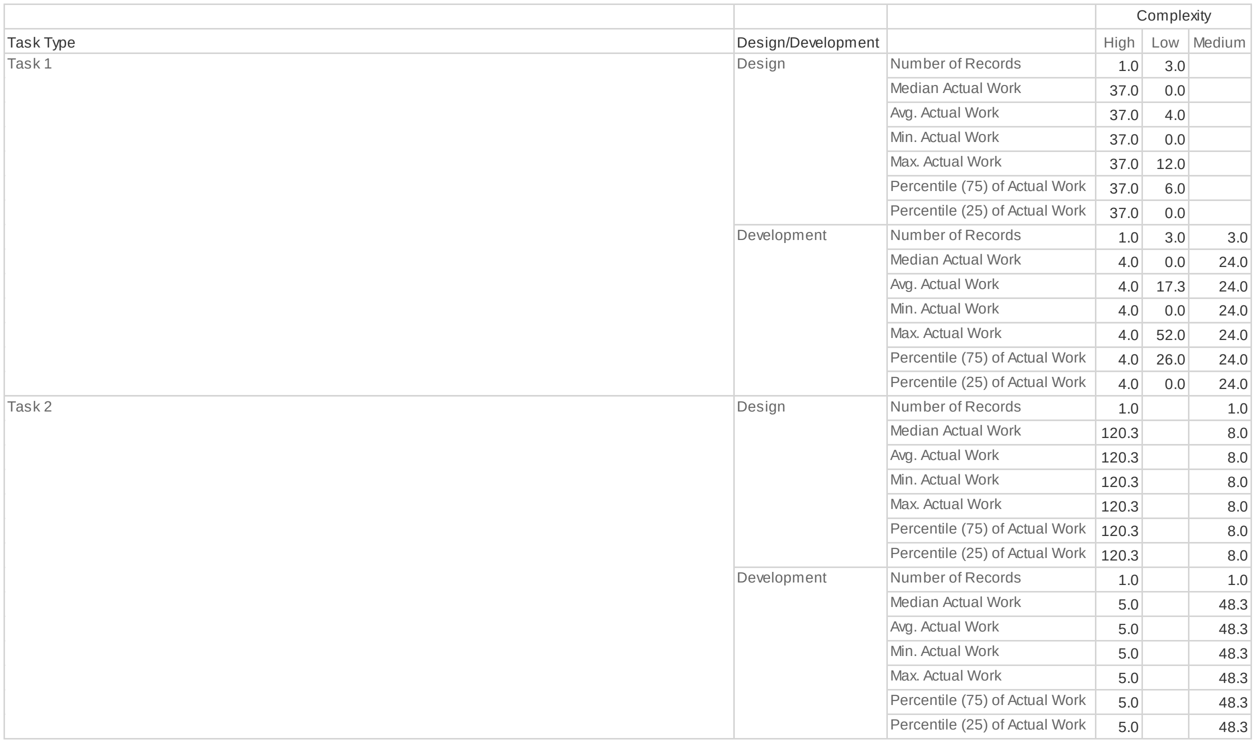
Figure 37: Tableau reporting for processing data from finished projects



Design/Developm...	Create/Modify	Complexity	Task Type	Open Systems/Ba...	Predicted	Actual	Difference
Design	Create	Medium	Development of a Fro...	Open Systems	8	8.000	0.0000
Design	Create	High	Charting feature in yo...	Open Systems	10	10.000	0.0000
Design	Create	High	Charting feature in yo...	Open Systems	10	16.000	6.0000
Design	Create	Medium	Development of a Fro...	Open Systems	8	40.000	32.0000
Design	Create	Medium	Development of a Fro...	Open Systems	8	8.000	0.0000
Design	Create	Medium	Development of a Fro...	Open Systems	8	16.000	8.0000
Design	Create	Medium	Development of a Fro...	Open Systems	8	8.000	0.0000

This framework also helps us append data from multiple projects, and it creates the basis for refining the estimate at the task level on an ongoing basis. We have created a Tableau dashboard for the same, as shown below in Figure 38.

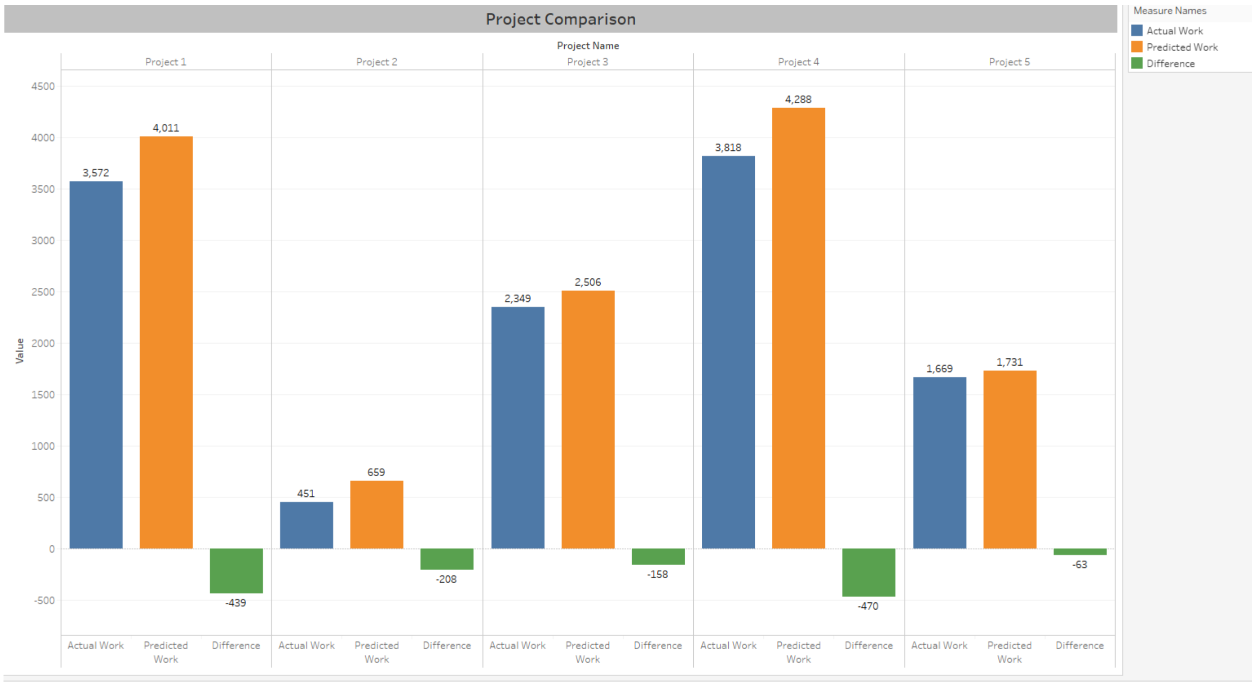
Figure 38: Tableau Analysis for analyzing task level data



			Complexity		
Task Type	Design/Development		High	Low	Medium
Task 1	Design	Number of Records	1.0	3.0	
		Median Actual Work	37.0	0.0	
		Avg. Actual Work	37.0	4.0	
		Min. Actual Work	37.0	0.0	
		Max. Actual Work	37.0	12.0	
		Percentile (75) of Actual Work	37.0	6.0	
		Percentile (25) of Actual Work	37.0	0.0	
	Development	Number of Records	1.0	3.0	3.0
		Median Actual Work	4.0	0.0	24.0
		Avg. Actual Work	4.0	17.3	24.0
		Min. Actual Work	4.0	0.0	24.0
		Max. Actual Work	4.0	52.0	24.0
		Percentile (75) of Actual Work	4.0	26.0	24.0
		Percentile (25) of Actual Work	4.0	0.0	24.0
Task 2	Design	Number of Records	1.0		1.0
		Median Actual Work	120.3		8.0
		Avg. Actual Work	120.3		8.0
		Min. Actual Work	120.3		8.0
		Max. Actual Work	120.3		8.0
		Percentile (75) of Actual Work	120.3		8.0
		Percentile (25) of Actual Work	120.3		8.0
	Development	Number of Records	1.0		1.0
		Median Actual Work	5.0		48.3
		Avg. Actual Work	5.0		48.3
		Min. Actual Work	5.0		48.3
		Max. Actual Work	5.0		48.3
		Percentile (75) of Actual Work	5.0		48.3
		Percentile (25) of Actual Work	5.0		48.3

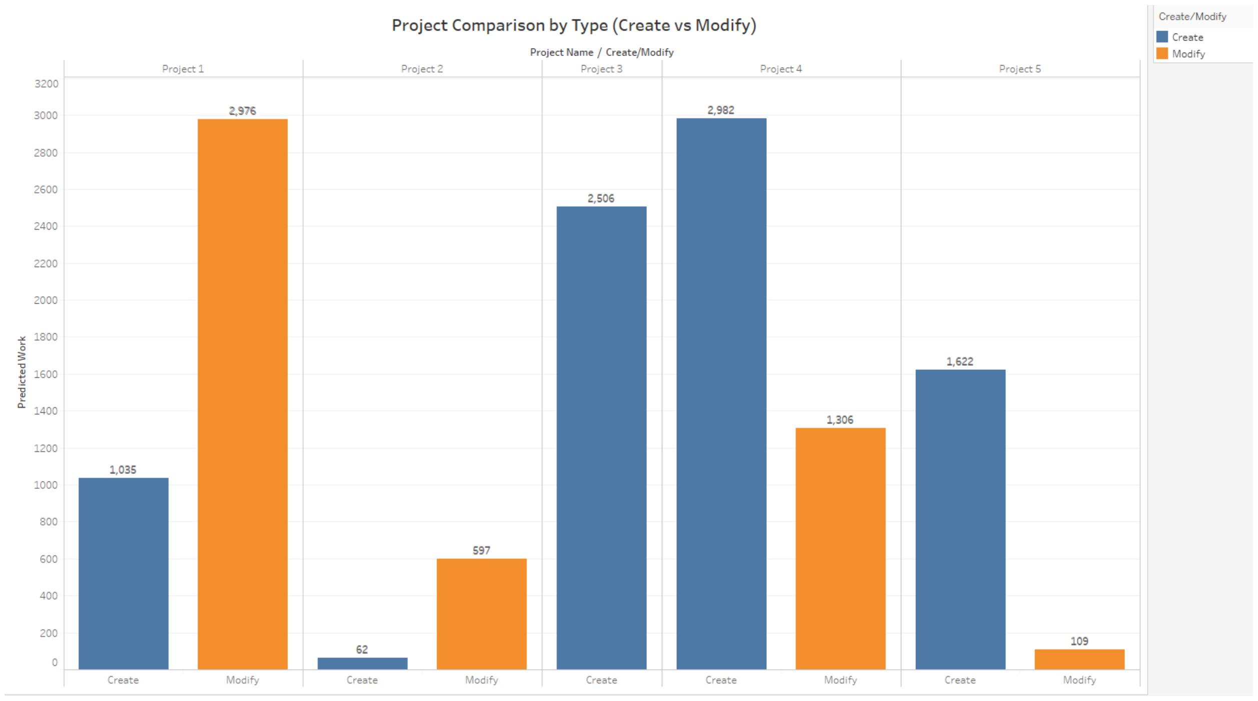
A comparative analysis of the five projects is as shown below in Figure 39. All five projects came in a little under the estimated cost for the design and development phase.

Figure 39: Comparison of actual hours and estimate from Estimating Tool



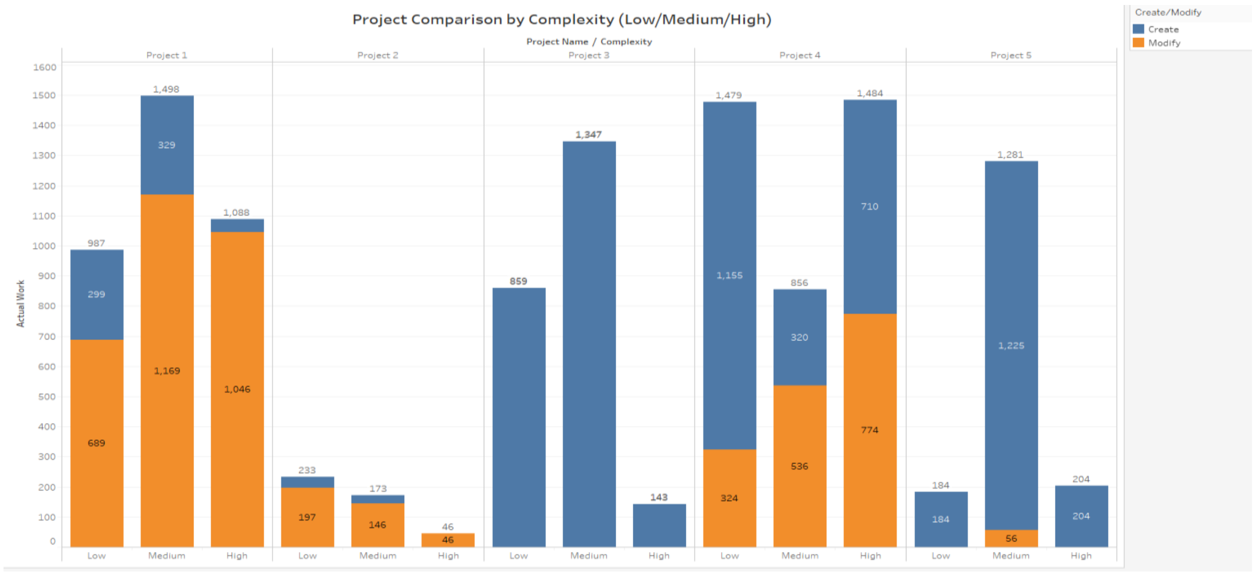
The project comparison by new development versus modification of existing artifacts is listed below. Each application development project will often involve the creation of new artifacts or modification to existing artifacts. One project was all new development. All other projects were a mix of new development and modifications to existing artifacts. A comparison based on this metric is shown below in Figure 40.

Figure 40: Project comparison by type (Create vs. Modify)



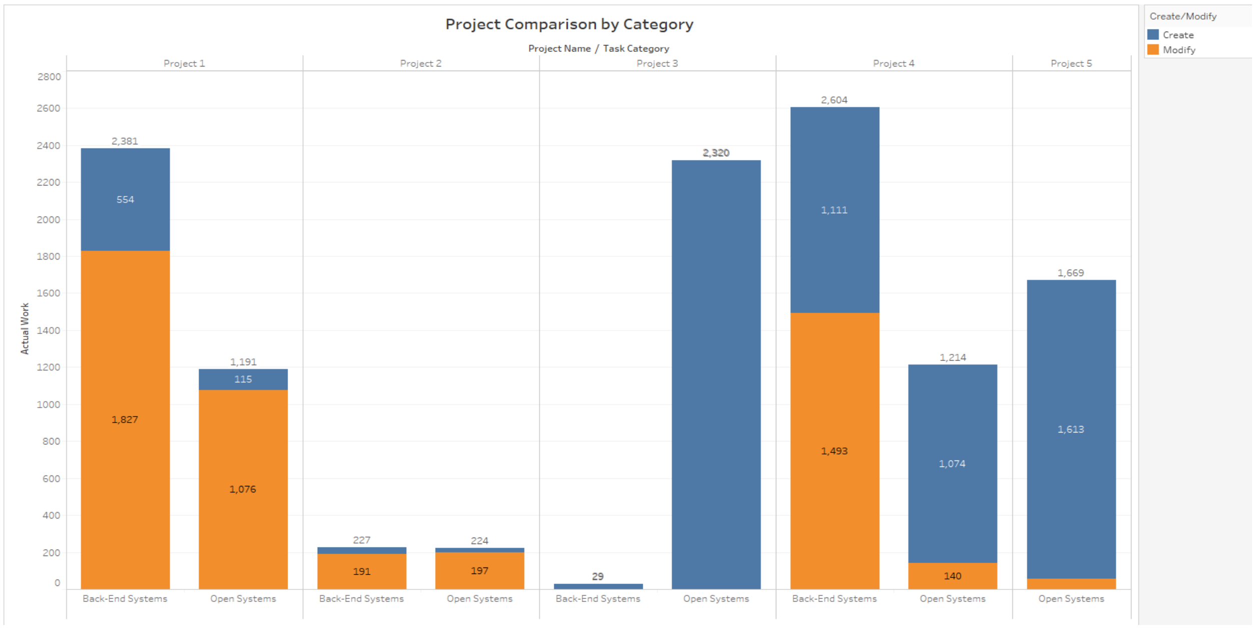
Each task in the estimating tool is categorized as easy, medium, or hard to help the project manager assign the right type of resource to the task. This categorization also helps the project managers and upper management in making better decisions managing for risk in the execution phase of the project. The comparison of the projects based on complexity levels for tasks is shown below in Figure 41.

Figure 41: Project comparison by complexity



There are two main categories under which the tasks in the estimating tool are categorized. These are open systems and back-end systems. The comparison of the projects based on this categorization is shown below in Figure 42.

Figure 42: Project comparison by category (Open Systems vs. Back-end Systems)

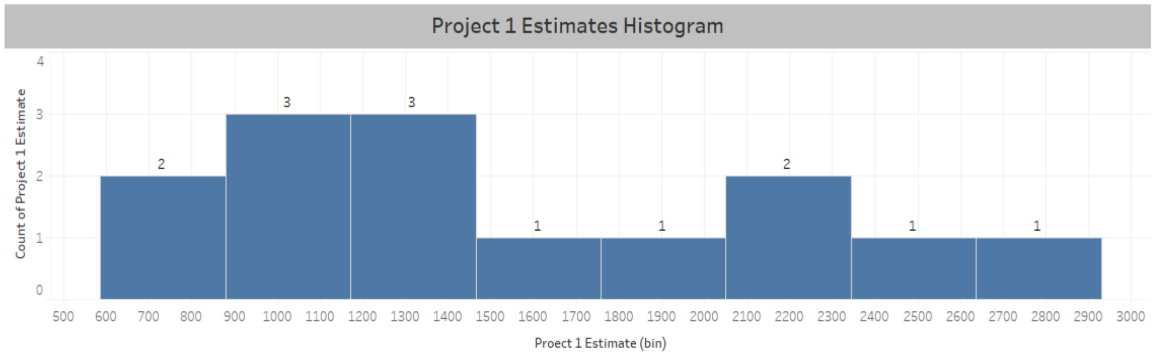


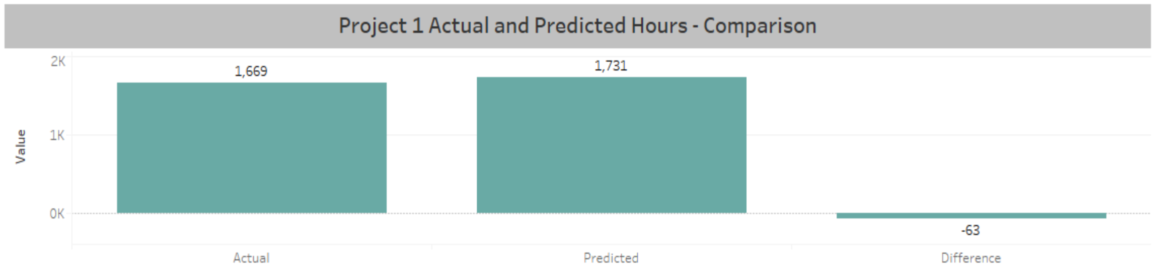
Analysis of the quality of the estimate:

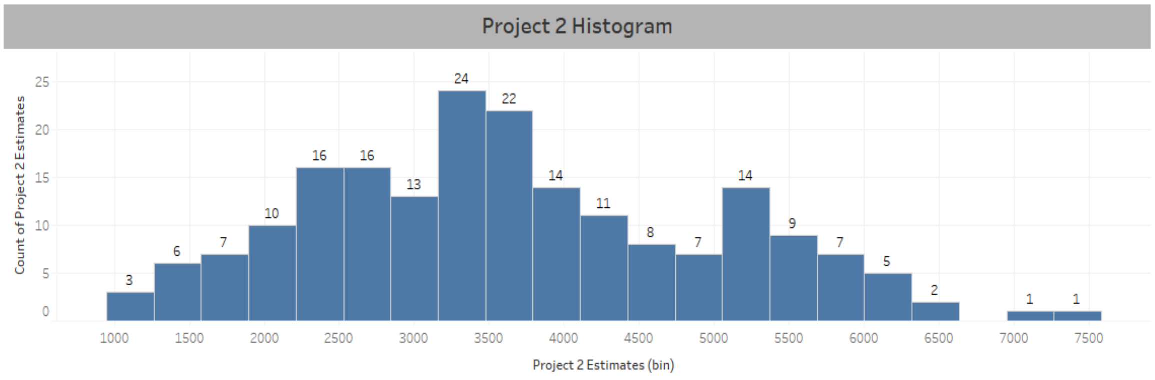
The array of estimates for each project was generated using the Java program, as outlined in section 4.2. Each project had between 14 to 196 estimates. A histogram of estimates for each project was generated using Tableau. The original estimate and actual cost from the project execution were then mapped back on the histogram to get a measure of the quality of the estimate. It is observed that the approved estimate of the project is close to the median and the actual cost is a little less than the approved estimate. The variability that existed in the estimating process prior to the rollout of the estimating tool is apparent when we look at the spread of the estimates. The biggest advantage from the usage of the estimating tool is that it has brought consistency to the process across all the application development teams within the organization.

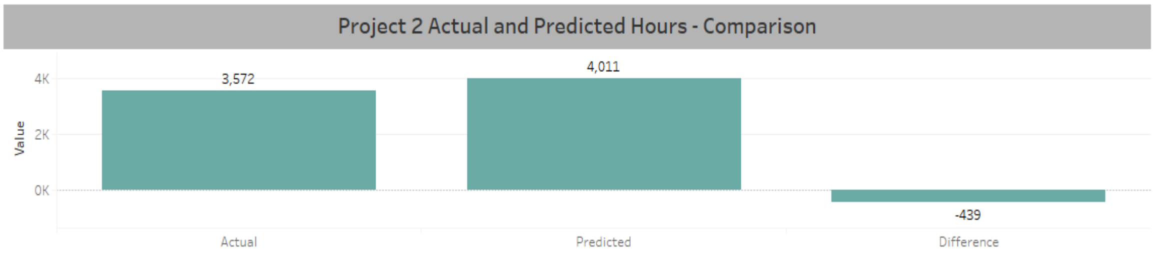
The histograms and the actual labor hours in comparison to the initial estimate for each of the five projects are shown below in Figure 43.

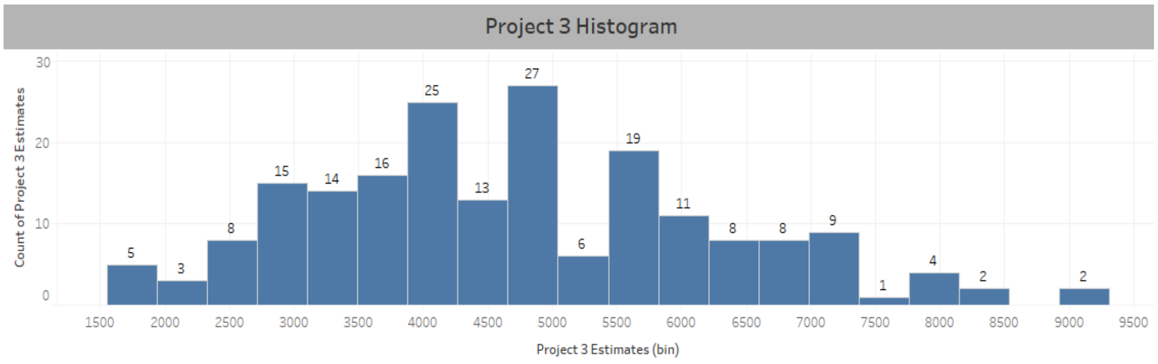
Figure 43: Histograms assessing the quality of the estimate for five projects

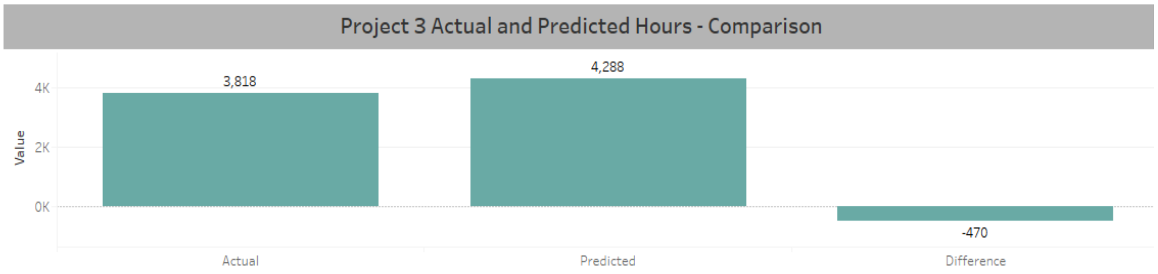


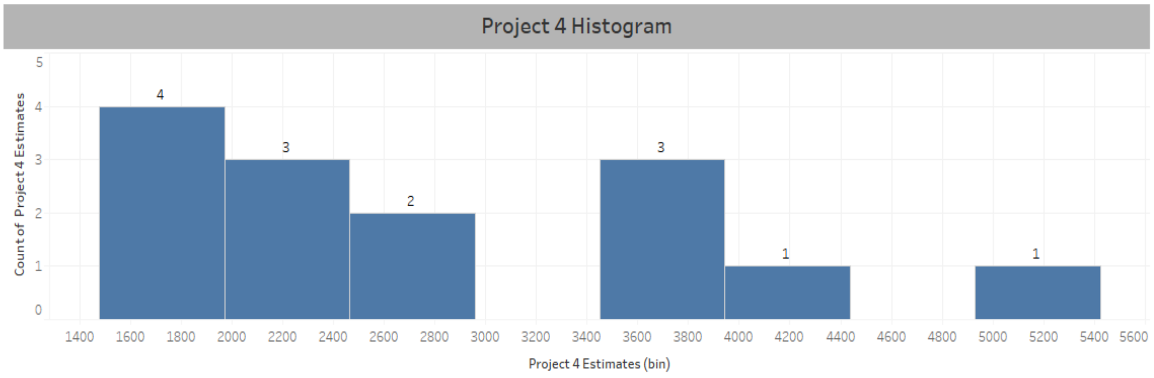


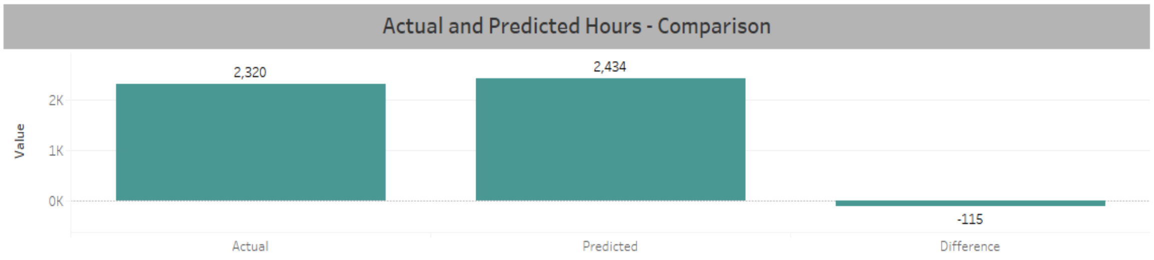


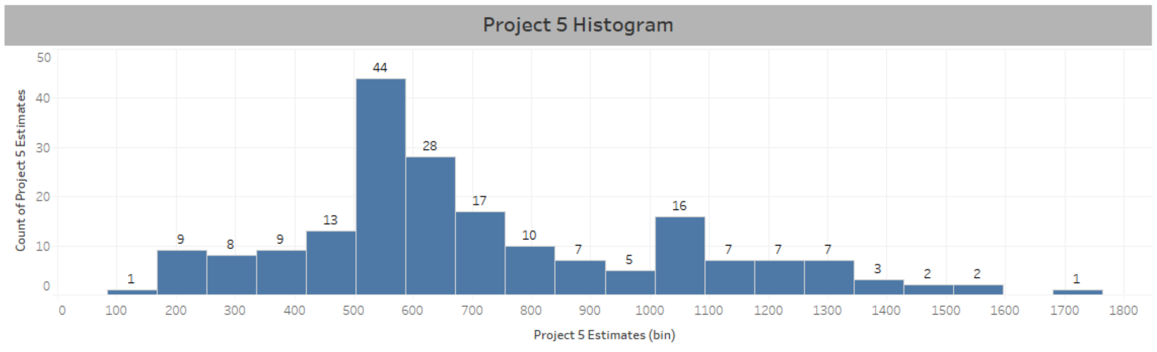


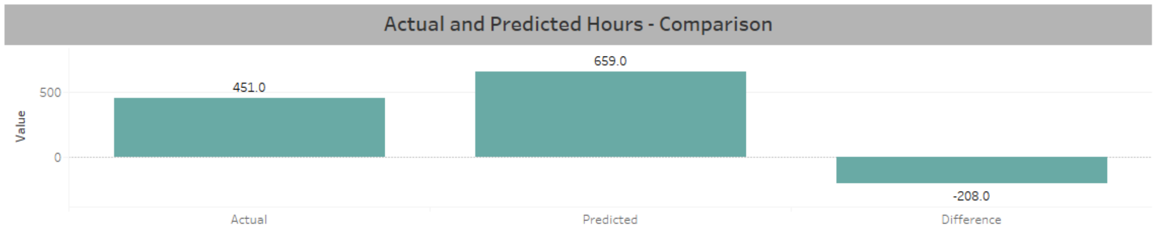

Project 3 Histogram
Count of Project 3 Estimates
30
20
10
0
5
3
8
15
14
16
25
13
27
6
19
11
8
8
9
1
4
2
2
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
7500
8000
8500
9000
9500
Project 3 Estimates (bin)


Project 3 Actual and Predicted Hours - Comparison
Value
4K
2K
0K
3,818
4,288
-470
Actual
Predicted
Difference


Project 4 Histogram
Count of Project 4 Estimates
5
4
3
2
1
0
4
3
2
3
1
1
1400
1600
1800
2000
2200
2400
2600
2800
3000
3200
3400
3600
3800
4000
4200
4400
4600
4800
5000
5200
5400
5600
Project 4 Estimates (bin)


Actual and Predicted Hours - Comparison
Value
2K
1K
0K
2,320
2,434
-115
Actual
Predicted
Difference



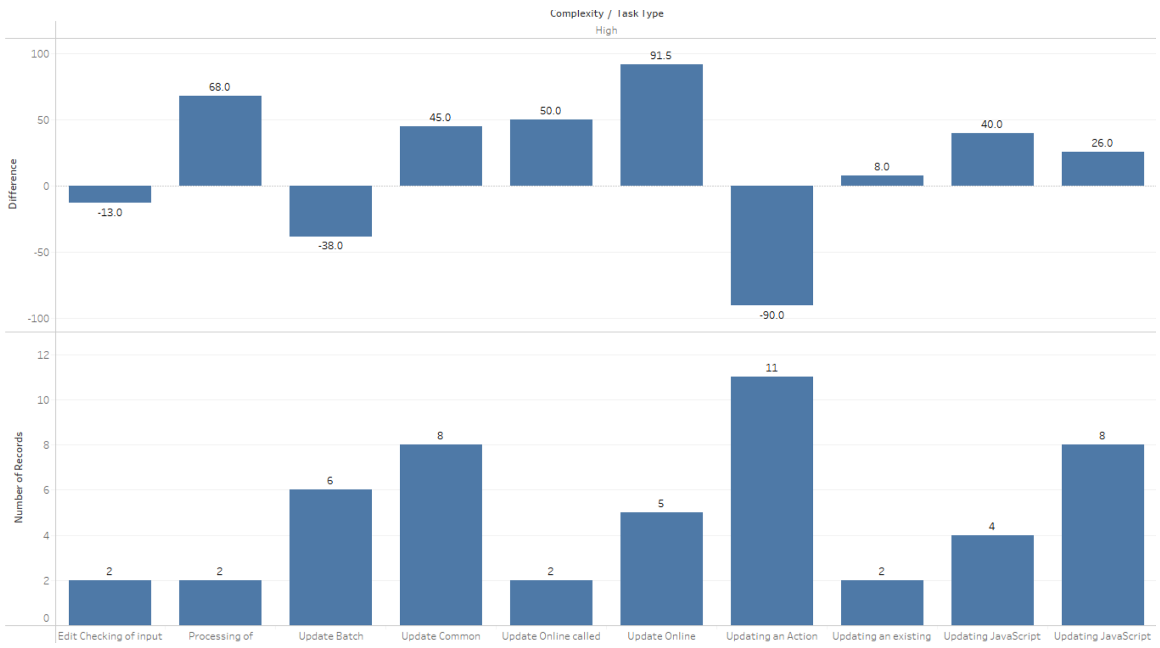


I also analyzed the data from the perspective of how resources were used in the project, what type of resources were used, and how many systems did the project touch. A Tableau dashboard was created for this view as well. An interesting observation that lined up with my initial findings is that there was one project in this set of five projects that touched multiple systems, and it had a pool of contractors that were predominantly less experienced in the systems of ABC Inc. This project cost more and went over the overall estimated cost when the verification and implementation phases were completed. We had actual data processed for only five projects, and ongoing data collection is needed to continue to evaluate the approach. One possible hypothesis that can be tested going forward as we collect more data is outlined below.

Hypothesis $H1$: Projects tend to go over the estimated cost when less experienced contractors are used in the execution of the project and when they touch multiple systems due to poor quality in the design and development phase of the project.

The hypothesis can be tested against the number of defects found in the verification phase, and this is something that can be pursued for future research. Individual dashboards were also setup for each of the projects. A framework has been established for extracting the data and setting up a dashboard for a new project should only take a few clicks going forward. The view below in Figure 44 shows at a high level how many types of tasks constituted the project and the aggregated stats around the processing times associated with those tasks.

Figure 44: Statistics on processing times at the task level



The task data across all projects is also summarized in Tableau, as shown in Table 20. The sample below is shown for ten task types, and the task names have been masked.

Table 20: Summary of task level data

		Complexity		
Task Type		High	Low	Medium
Task 1	Avg. Actual Work	4.0	17.3	24.0
	Number of Records	1.0	3.0	3.0
	Avg. Predicted Work	40.0	12.0	24.0
Task 2	Avg. Actual Work	5.0		48.3
	Number of Records	1.0		1.0
	Avg. Predicted Work	40.0		40.0
Task 3	Avg. Actual Work	44.0	16.0	32.0
	Number of Records	1.0	5.0	11.0
	Avg. Predicted Work	64.0	16.0	36.4
Task 4	Avg. Actual Work			4.0
	Number of Records			1.0
	Avg. Predicted Work			4.0
Task 5	Avg. Actual Work	62.0	7.7	
	Number of Records	2.0	3.0	
	Avg. Predicted Work	60.0	8.0	
Task 6	Avg. Actual Work	0.0	0.0	0.0
	Number of Records	2.0	3.0	4.0
	Avg. Predicted Work	80.0	6.7	29.0
Task 7	Avg. Actual Work	0.0	37.2	39.3
	Number of Records	1.0	28.0	9.0
	Avg. Predicted Work	121.0	36.8	65.9
Task 8	Avg. Actual Work	19.0	37.0	
	Number of Records	1.0	1.0	
	Avg. Predicted Work	90.0	38.0	
Task 9	Avg. Actual Work	0.0	36.0	
	Number of Records	1.0	1.0	
	Avg. Predicted Work	80.0	34.0	
Task 10	Avg. Actual Work	31.5	82.0	14.0
	Number of Records	1.0	1.0	1.0
	Avg. Predicted Work	122.0	20.0	51.0

6.3 Optimization Model Results and Experiments

This section presents a case study that shows how estimates that are developed at the task and scenario level are used in an Agile setting to optimize the use of limited resources. Our scenario accounts for economies of scale that can result from managing the skills and domain expertise of resources that are part of multiple projects. This model can be used in an environment of multiple small size projects to manage a pool of resources given that we always have a backlog of tasks to complete. This scenario will be a case study providing a proof of concept to optimize executing multiple small projects at the same time. The small projects are rooted in the design patterns of ABC Inc. and will cater to the local context.

In the Agile methodology, a few concepts are becoming more popular. One of them is the concept of using Full Stack Developers as part of the team. Traditionally, most application development teams were comprised of specialists in a domain or technology. We had UX (User Experience) specialists, developers who specialized in back-end server-side coding, testers who specialized in writing automated tests, and in the testing of the application, database specialists, etc. A full stack developer is someone who conceptually specializes in all the above-mentioned technologies. The expectation is that they can handle front end, server side, databases, and testing while at the same team fulfilling the role of a subject matter expert (SME). A team comprised of several full stack developers is called a self-driven team. A company that is rooted in the traditional application development is going to start off with resources having 1-2 specialized skills. It will be important for a company to build a resource pool that specializes in multiple skill sets to make the transition to be a self-driven team.

It is in this context that the concept of an efficiency score is introduced. It is quite impractical in the real world, particularly in a company that has always done traditional application development to assume everyone is going to excel at all the skills. The efficiency score will be based on a few factors, such as:

1. Total experience of an individual in a skill.
2. The date when the skill was last used by an individual.
3. The self-rating for the skill by the individual.
4. Rating of the individual on the skill by the SME or by all the other team members

It will be a weighted score based on the above factors. The core basis of the efficiency score is that an individual with the higher efficiency score for a skill will complete that story faster with a better quality. On the other hand, an individual with a lower efficiency score for a skill will take more time to complete the task, and it might not have the best quality. The work might result in more defects that will need to be resolved during the validation phase.

A sample prototype for calculating the efficiency score is shown below in Table 21.

Table 21: Sample efficiency score prototype

Skills	Years of Experience	Last time skill used	Self Rating	Peer Rating	Efficiency Score
Spring Batch	3-5 years	More than three year	Med	Low	0.53
React JS	3-5 years	More than three year	Beginner	Beginner	0.38
Java	1-3 years	More than five years l	Low	Med	0.42
Hadoop	No Experience	Never	Beginner	None/New to Firm	0.05

The whole concept of becoming self-driven teams with full stack developers is directly related to the concept of an efficiency score. The model considers the efficiency score for a skill and pay rate of an individual while assigning the appropriate resource to a skill that is needed for a story. The core premise of a self-driven team is that the efficiency scores of individuals for each skill will keep going up with experience. The model facilitates the analysis of strategic personnel assignment and shows how it can increase efficiency scores in the long term. Eventually, the increase in efficiency scores will result in increased profitability. This research shows how the work that originally required 14 individuals can in fact be completed by seven individuals when each of the individuals possesses multiple skills. This gets the organization to start thinking in terms of development plans that can be tailored around encouraging individuals to learn new skills. We assume that the efficiency score of an individual will go up over time and with experience. An individual who is an expert in particular skills over an extended period will always be a specialist in that skill and can learn new skills as part of the self-driven team. It will take some time for that individual to be a specialist in the newer skills, and we need to account for that as we assign resources to stories.

The concept of cross functional teams is also becoming more popular in the context of self-driven teams. Again traditionally, companies encouraged people to develop domain expertise in an area. This could be Finance, Marketing, Human Resources, Legal, etc. These individuals typically would be subject matter experts who specialize in an area of expertise. In a cross functional team, individuals from different functional areas come together as a self-driven team, and this typically results in everyone become more familiar about other areas outside of their core functional area of expertise. The domain expertise of individuals in a cross functional team can also be looked at from the perspective of skills, and our model looks at subject matter expertise in a particular domain as a skill that can be viewed from the perspective of an efficiency score. We assume that subject matter expertise in a domain can be acquired over time and that developers can transition to acquire this skill over time. This happens in the real world in application development. The Agile methodology can aid in fast tracking the development of subject matter experts within an organization.

Workload availability is to be seen in conjunction with upskilling of the resources in the context of the optimization model. My approach in the initial sprints is to allocate a percentage of time for cross training amongst the resources. The workload availability of resources will be adjusted accordingly in the initial sprints. The resources with a high efficiency scores in a skill will be transitioning knowledge to resources with low efficiency scores in that particular skill. The vision is that every individual has goals in terms of where they need to be on the skills matrix, and the team has a goal of a desirable skills matrix state that they need to achieve in a set number of sprints. There are techniques like pair programming or mobbing that can be used in this context. Two programmers share the same computer and work on a single task in pair programming while one person who is proficient in a skill is sharing the knowledge with a group of programming in mobbing.

A hypothesized implication of using an efficiency score in the model is as follows.

H_1: A team comprised of resources with higher efficiency scores in multiple skills will result in more stories being allocated, thereby resulting in overall better objective value.

It is often the case in traditional software development that key resources are stretched thin between projects. They are often allocated only for a percentage of time to work on a certain project, and it is reality that they have to spend time working on different projects at the same time. Resources often work on other support related initiatives in addition to working on a project. One need to account for time spent in administrative tasks, training time, etc. The amount of time a resource is available for the project varies, and it is in this context that a workload availability of an individual resource becomes a critical factor in planning a project schedule. In addition, we must consider the time a team consciously spends in cross training and upskilling. The time spent in cross training is accounted for by adjusting the workload availability of resources. My model considers the workload availability of each individual resource for an individual sprint and ensures that the workload capacity of an individual for a sprint is not exceeded.

The following two hypotheses pertain to the effect of workload availability at the resource level for a sprint in the model and the effect of workload availability in conjunction with efficiency score in the model.

H_2: A team with higher workload resource availability will have a more efficient schedule with more stories being taken up for allocation, and it will result in overall better objective value.

H_{2a}: Workload resource availability in conjunction with higher efficiency score will result in better outcomes in terms of schedule efficiency and overall better objective value.

In software development, it is often the case where there is a difference in the payrate for employees and contractors. It is also the case that more experienced resources and resources who are proficient in multiple skills are paid differently. A subject matter expert is paid differently as compared to a starting developer. The model considers this factor and will try to allocate the person with the best payrate while considering other factors like workload availability to do the task and the efficiency score of the individual on the task.

The hypothesized relationship between the overall utility of the project and the available resources can be stated as follows.

H_3: The overall objective value associated with the project is maximized when we can find the optimal pool of resources for the project with higher efficiency scores, higher workload availability at the most competitive hourly rate.

6.3.1 Current process of executing projects

This following is a list of steps that are followed in the current process:

1. The first step is to break down the project into high level tasks.
2. The next step is to create a work breakdown structure (WBS), where each high-level task is broken down into manageable chunks. The estimating tool that this research developed in essence creates a WBS for the design and development phases of your project. The WBS contains tasks at a level of detail which anyone working on it can understand and execute.
3. The project manager then looks at the resource availability for each resource who is assigned to the project and builds a schedule based on the available information.
   a. At ABC Inc., resources on these small efforts are typically shared between multiple efforts. The resource availability in some cases was 20%. This means a resource will take five days to complete a task that typically takes a full-time resource one day to complete.
   b. The SMEs are typically assigned to multiple projects, and they are available for 10-20 percent of the time. Developers who are looking for feedback from SMEs typically have to wait, and this process is repeated over and over again.
   c. One such cycle can be summarized as follows
      i. Task from WBS ready to be assigned to chosen developer.
      ii. Developer starts work
      iii. Developer need feedback from SME.
      iv. Typically, there is a wait each time input is needed from the SME.

v. Developer completes the task.

vi. SME needs to review the work, and this again results in more wait time.

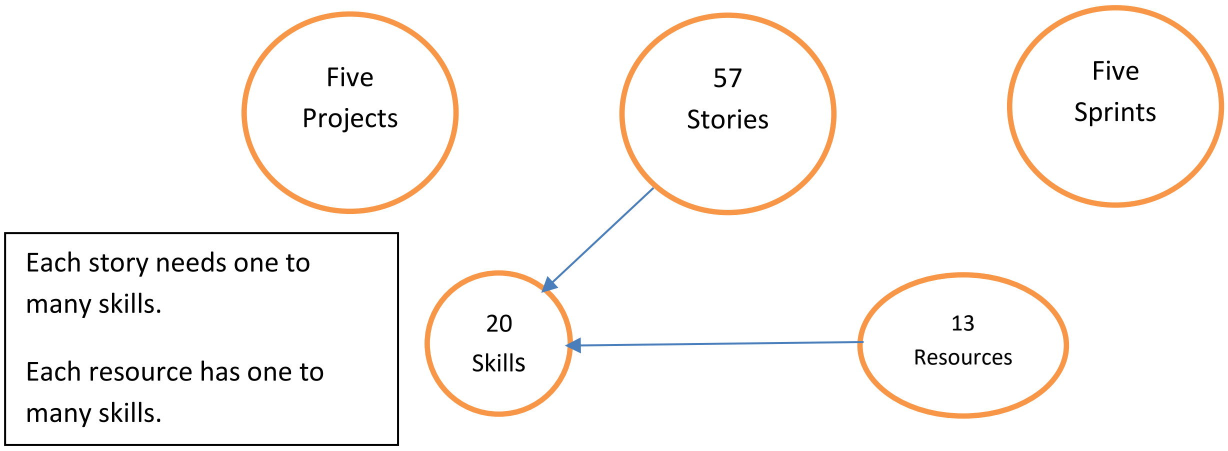
d. There are other resources like the UX specialist, Design thinking specialist, Legal analysts who are critical resources and not assigned full time to the project. There are tasks that can be executed only by these resources and are often on the critical path. This often results in additional wait time in the execution schedule.

4. Once the schedule is built, the project execution starts.

6.3.2 Base Case

The base case analysis refers to the use case data that was described above. I had data for five projects that were executed using a hybrid approach where tenets of both Waterfall and Agile were used. These five projects had a total of 57 stories that needed to be completed in five sprints. A total of 20 skills were needed to complete all stories. Each story typically required 1-4 skills. We had a total of 13 resources that were available for the projects, and many of them were typically shared resources for the execution of these projects.

Figure 45: High level representation of use case data


Five Projects
57 Stories
Five Sprints
Each story needs one to many skills.
Each resource has one to many skills.
20 Skills
13 Resources

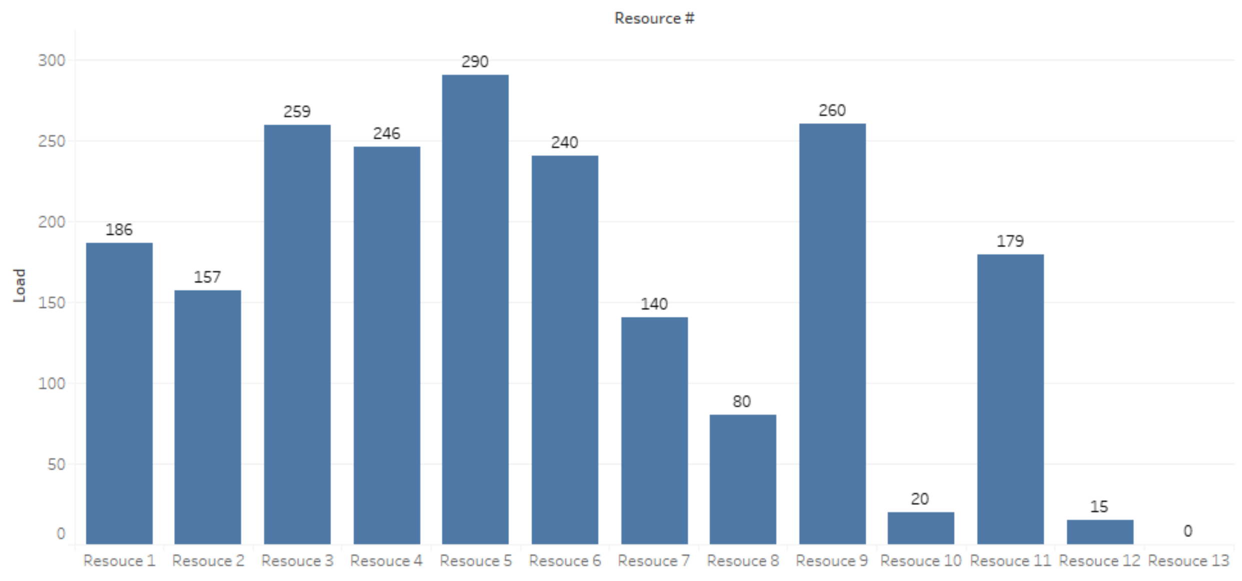
The estimated total number of hours across the 57 stories was 3522 hours. The resources that were initially needed to execute the projects were

a. Multiple Information System SME's
b. Multiple Business side SME's
c. Design Thinking Specialist
d. Information Systems Project Manager
e. Business Side Project Manager
f. Developers with Front End skills
g. Developers with Back End server-side skills
h. Legal Analyst
i. QA Testers

The data was setup for the base case with 13 resources. The assumption was made that an individual had skills that pertained to their core competency. A business area SME had only skills that pertained to their area of expertise like making the business case, validating business requirements, etc., a front-end developer had only skills that pertained to the presentation layer and so on. The efficiency score for these individuals for the skills they had was assumed to be perfect and were given a 1.0 for all the relevant skills, and they were assigned a score of 0.01 on all other skills. The workload was adjusted as per the availability. None of the resources were available full-time for the effort. This was reflected in the data.

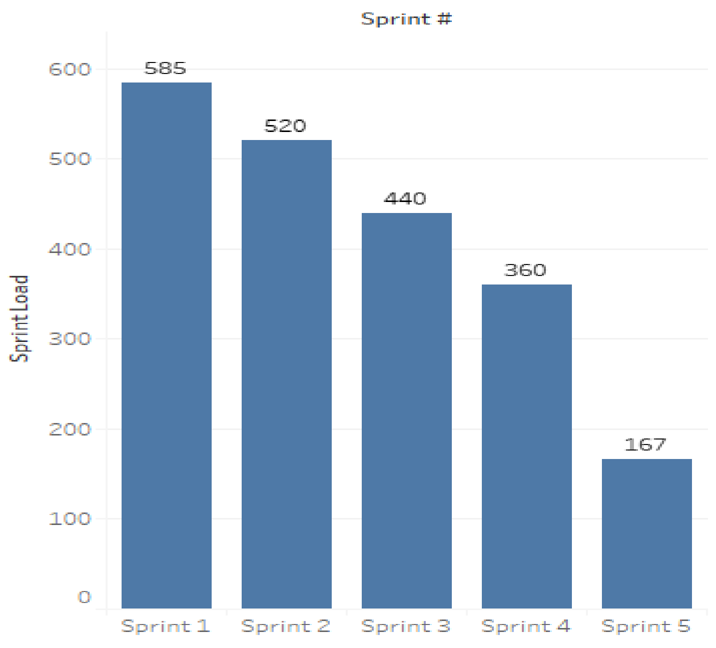
The planning horizon was five sprints. The model for the base case was executed on an Intel(R) Core (TM) i5-8250U CPU @1.60GHz, 1801 Mhz, 4 Core(s), 8 Logical Processors machine with 12 GB RAM. The total computational time (root+branch+cut) was 8.63 seconds, and the solution was optional with an optimality gap of 0.33% for the run. We observed that only 36 stories could be allocated given the resource constraints. It was the workload constraint that mostly prevented the other stories from being taken into consideration. The resource load across the five sprints is shown below. We observed that the resource whose workload availability was higher were utilized more. The resource utilization is as shown below in Figure 46.

Figure 46: Resource utilization for the base case execution



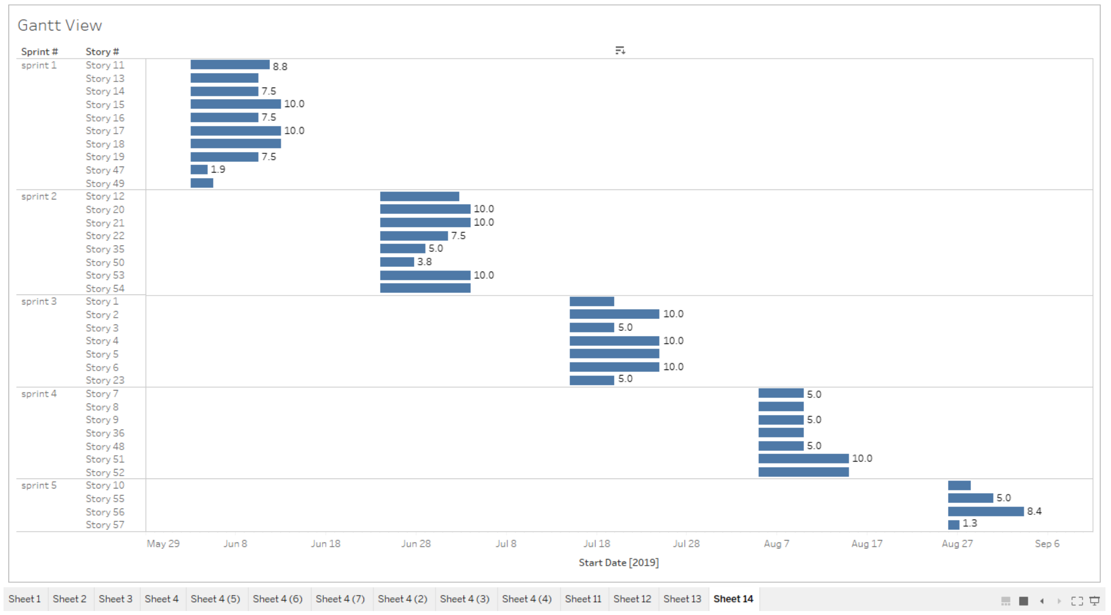
The sprint load across the five sprints is as shown below in Figure 47. More stories are allocated to the initial sprints as compared to the later sprints.

Figure 47: Sprint load across the five sprints



The allocation of stories to the five sprints is as shown below.

Figure 47a: Allocation of stories to the five sprints in base case



The optimal objective value of maximum discounted total return was $135,653.13. The main takeaway here is that 21 stories could not be assigned due to the constraint that we assume all stories, including all the skills have to be completed in one sprint. Many of the resources were available only for part of the sprint, and the workload capacity constraint played a big part in these stories not being assigned. The optimization model is able to precisely prescribe the number of unassigned stories and which stories should not be assigned.

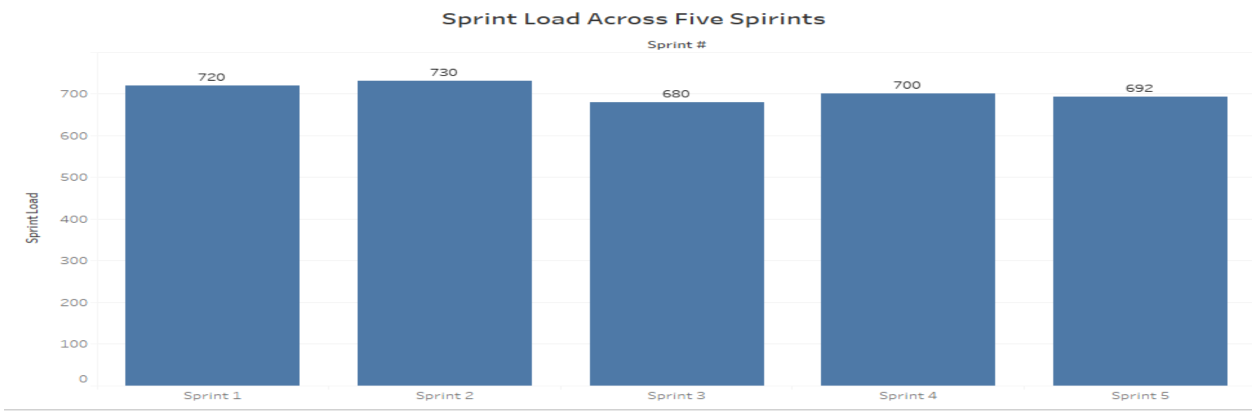
6.3.3 Scenario 1: Bringing more efficiencies by cross training resources on multiple skills

The whole concept of becoming self-driven teams with full stack developers can be viewed from the context of efficiency score and cross functional teams. In this scenario, we only have one skill called subject matter expertise, and everyone on the team could have the skills with varying levels of efficiency scores. It is shown that the work that originally required 14 individuals can in-fact be completed by seven individuals when each of the individuals possessing multiple skills. This is particularly relevant in an organization that is starting to move from the traditional Waterfall method to the Agile methodology. In this scenario, we need the following type of resources.

1. Subject Matter Expert
2. Developer
3. Project Manager

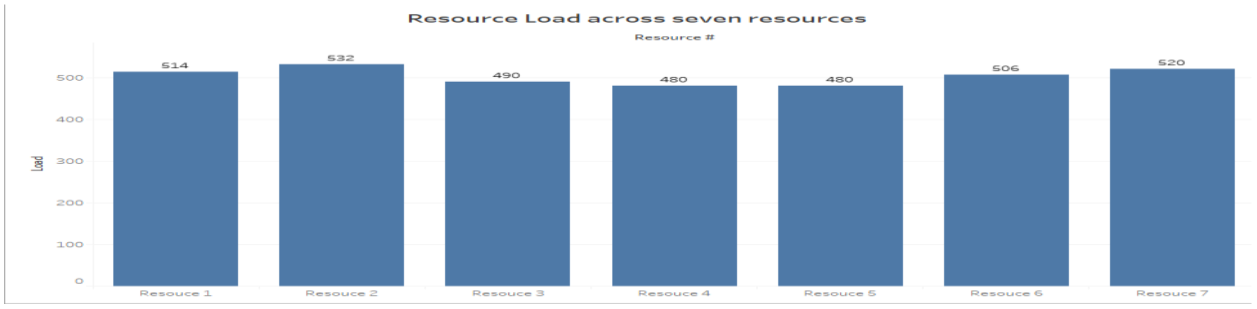
The assumption here is one resource can acquire multiple skills and become proficient in them over time. For example, a developer can become a subject matter expert over time and can acquire testing skills over time as well. The sprint load across five sprints is shown below in figure 48. We see that the load is split pretty evenly across the five sprints.

Figure 48: Sprint load across five sprints for the first scenario



The resource load of seven resources across five sprints is shown below in Figure 49. We also see that all the seven resources are used evenly across the five sprints.

Figure 49: Resource load utilization for the first scenario



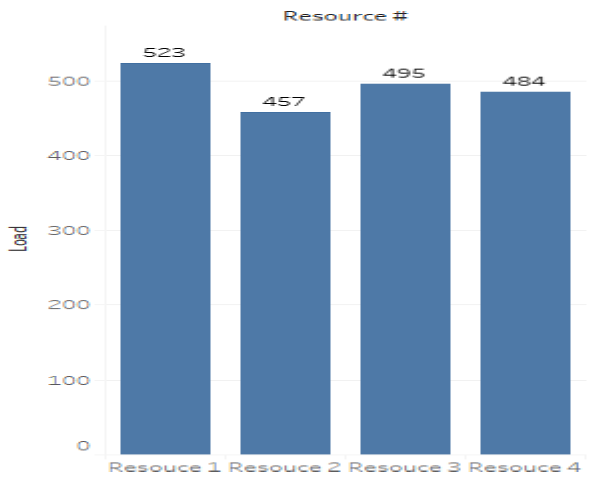
The maximum discounted return is shown below. The objective value in this scenario is $199,494. All the stories are assigned and executed in this scenario. The model for scenario 1 was executed on an Intel(R) Core (TM) i5-8250U CPU @1.60GHz, 1801 Mhz, 4 Core(s), 8 Logical Processors machine with 12 GB RAM. The total computational time (root+branch+cut) was 60.33 seconds, and the solution was optional with an optimality gap of 0.01% for the run.

6.3.4 Scenario 2: Combining all development tasks separately.

An additional observation is that every task is typically comprised of storyboarding, design, development, testing and validation, implementation, and finally feedback. Storyboarding can be thought as analogous to the requirements phase in a Waterfall project, but on a much smaller scale and the scope of it pertains to that of a single sprint. We typically have a sprint planning meeting where you discuss the requirements and needs for all stories in that sprint and follow it up with a high-level design approach. It is then followed up by assigning estimates for each story, and then we are ready to start working on the stories in the sprint. There is development, testing and implementation tasks within each sprint. In the context of our model and its usage, it is imperative that the product owner or scrum master consider the skill levels of everyone for all skills that are going to be required for the next few sprints as they plan out an epic. The goal of a self-driven team is often that individuals are improving on the efficiency score associated with all skills that are needed to execute a project.

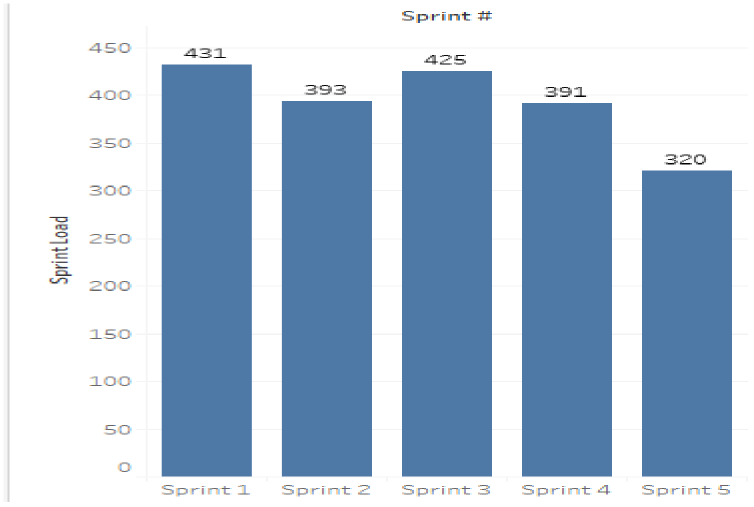
One alternative way to look at this is to finish the storyboarding for all stories upfront and then have a group of development stories across all five projects. I extracted out all the stories that were related to development or coding. We had a total of 26 such stories for an estimated 1960 hours. The planning for this scenario had four resources who possessed all the skills pertaining to coding and subject matter expertise. The value of this scenario is that it matches the right resource for each skill within each story based on the efficiency score, workload constraints, and pay rate of the individual. It also shows the value of having a team of resources who are proficient in multiple skills. The resource load across the four sprints is shown below in Figure 50. One can observe that the load is spread more evenly across the four resources.

Figure 50: Resource load for scenario two



The sprint load across the five sprints is shown below in Figure 51.

Figure 51: Sprint load for scenario two

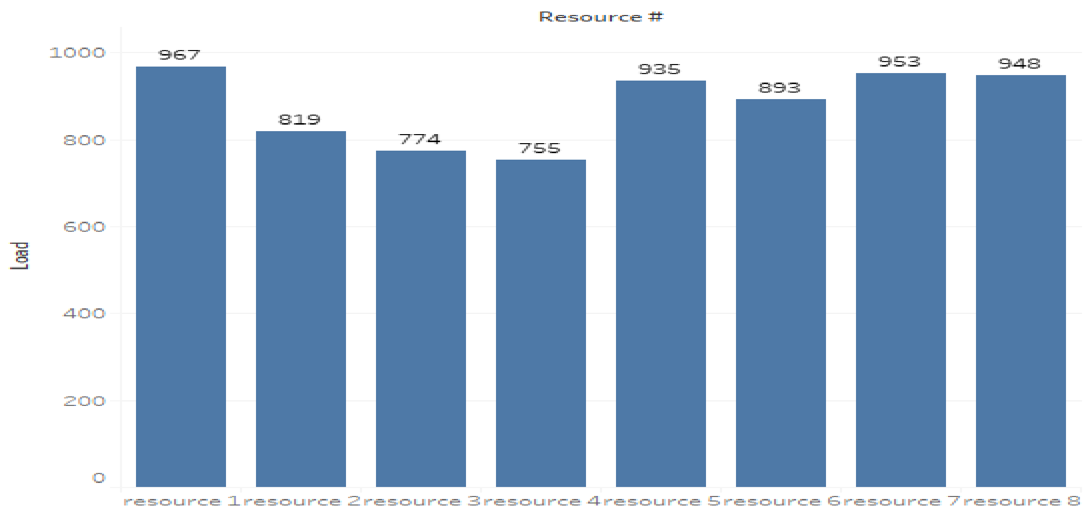


The objective value for this run was $115,370. The takeaway from this scenario is that it provides an alternative way to think about how to group together stories and execute them. All planning for the epic (deliverable that encompasses multiple sprints) can be done separately, followed by the execution of all development related stories. Eventually all testing related stories across all five projects can be executed. This scenario can be considered when one does not have set dates for each project and all five projects can be delivered together. The model for scenario 2 was again executed on an Intel(R) Core (TM) i5-8250U CPU @1.60GHz, 1801 Mhz, 4 Core(s), 8 Logical Processors machine with 12 GB RAM. The total computational time (root+branch+cut) was 5.58 seconds, and the solution was optional with an optimality gap of 0.06% for the run.

6.3.5 Medium size case with 10 projects and 114 stories

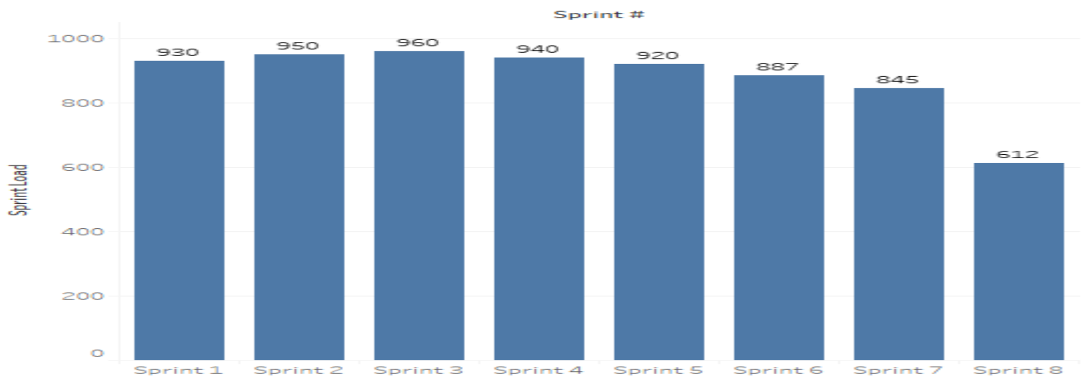
There was a total of five projects and 57 stories in the original data that was available. I used that as the base case. In this experiment, five new projects that were very similar to the original five projects were added for consideration. The total number of stories that were part of the planning process was 114 stories. This is a hypothetical scenario, and it gives us an idea on how to use the model for a larger planning horizon and plan accordingly. The total estimate for all 114 stories was 7182 hours. I considered using eight resources and tried to fit in all of the 114 stories into eight sprints for this scenario. The resource load across the eight sprints is as shown below in Figure 52.

Figure 52: Resource load for medium size case (Ten Projects)



All 114 stories are assigned, and the workload is distributed across the seven sprints, as shown below in Figure 53.

Figure 53: Sprint load for medium size case



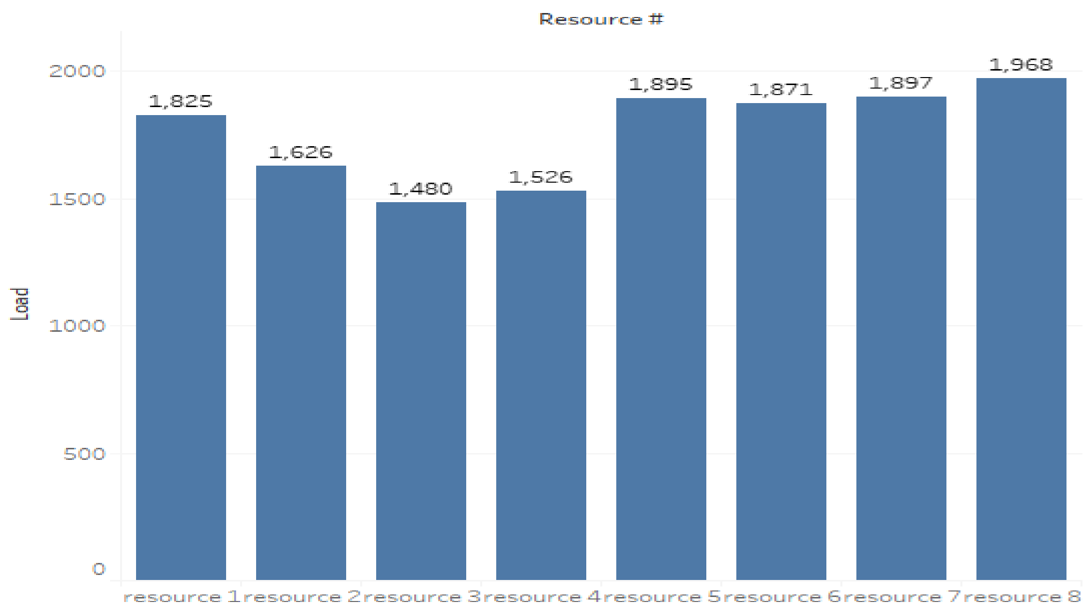
The Cplex time limit for this run was set at 10 minutes, and the optimality gap was 0.24%. The objective value was $ 423045.01 for all of 114 stories that were assigned. The model for the medium size case again was executed on an Intel(R) Core (TM) i5-8250U CPU @1.60GHz, 1801 Mhz, 4 Core(s), 8 Logical Processors machine with 12 GB RAM.

6.3.6 Large size case with 20 projects and 228 stories

In this experiment, fifteen new projects that were very similar to the original five projects were added for consideration. The total number of stories that were part of the planning process was 228 stories. This again is a hypothetical scenario, and it gives us an idea on how to use the model for a very long planning horizon. It is very unlikely that a project manager in the real world will need to plan for 228 stories at the same time. In the real world, most companies work on the sprint plus one model where you have detailed stories breakdown for two sprints, the current sprint, and the next sprint in line. I considered using eight resources and tried to fit in as many stories as possible into 16 sprints for this scenario. The Cplex time limit for this run was set at 10 minutes, and the optimality gap was 0.94%. I adjusted the data, the efficiency score and pay rate to ensure all the 228 stories were assigned and executed. The workload was kept constant at 130 hours per sprint per resource. The objective value is $ 795984.13 for all 228 stories that were assigned across 20 projects. The model for the large size case again was executed on an Intel(R) Core (TM) i5-8250U CPU @1.60GHz, 1801 Mhz, 4 Core(s), 8 Logical Processors machine with 12 GB RAM.

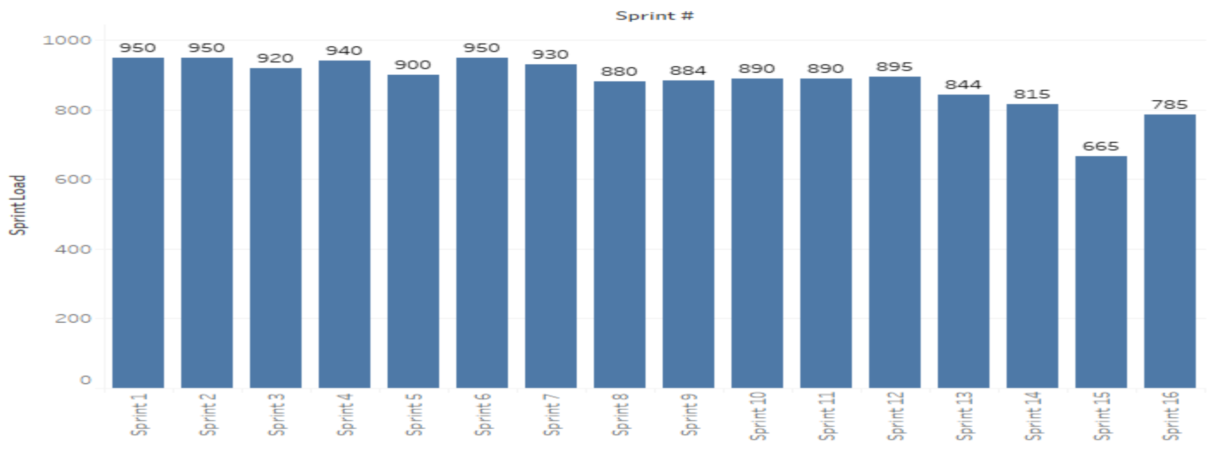
The resource allocation across the 16 sprints is shown below in Figure 54.

Figure 54: Resource load for large size case (20 Projects)



The workload is split across the 16 sprints, as shown below. The workload is pretty evenly split across the 16 sprints as shown below in Figure 55.

Figure 55: Sprint Load for large size case (20 projects)

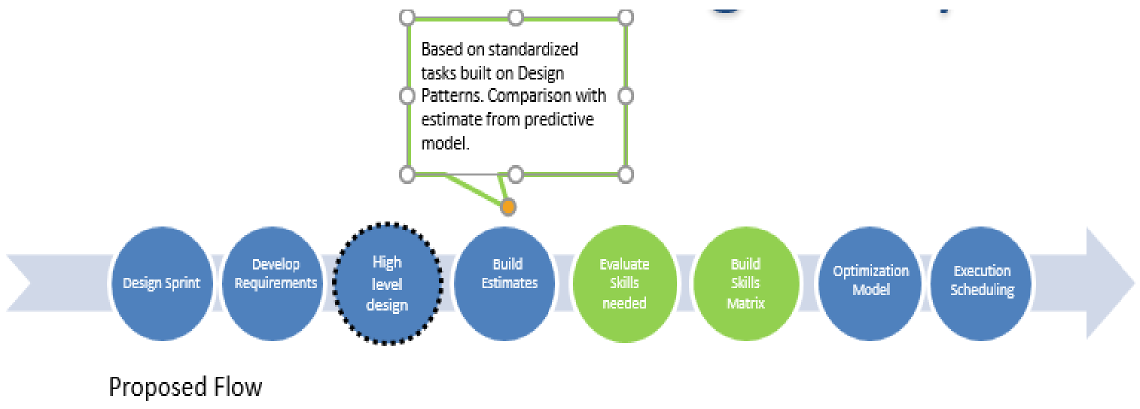


6.3.7 How to use the model at an organization

The utility of the model is increased when there is an equivalent dashboard to look at historical data from a resource skill perspective. As part of the historical data that was given for developing the estimating tool, I had access to eight years of project data. I developed a query tool that could potentially be used to query on task names, program names, artifact names, etc., and this gives us a starting point to try to match resources based on skills that are needed for stories. As mentioned before, this model has the maximum utility for an organization that is making a transition from the traditional Waterfall method of project management to the Agile methodology for delivering projects.

The proposed flow below in Figure 56 shows when skills are evaluated, when the skills matrix is built, and when the optimization model would be used in the project life cycle.

Figure 56: Proposed new flow incorporating Optimization Model



The steps that need to be taken by a project team are outlined below.

1. The project team, under the leadership of the scrum master and product owner, decomposes the overall deliverable into a set of sprints that are each composed of a set of stories. The project team find if there are any set end dates that have to be met.
2. Each story is then decomposed into a set of skills that are needed for the story to be completed. We identify the stories that need to be completed together, the stories that are incompatible and have to be done independently and the finally the stories with precedence relationships.
3. At this point, we have a high-level idea about the set of skills that are needed for the overall deliverable, and an attempt can be made to identify the best possible set of resources with the matching skill set. A team with a good blend of experience and resources all across the hierarchy would be ideal to minimize overall costs.

4. The efficiency score of each resource as it pertains to each skill needed for the project is identified at the outset. At this point, development goals (both short-term and long-term) could be set for each resource to acquire new skills in the next few sprints and epics.
5. The workload availability of each resource can be identified at this point. This is particularly important if certain resources are shared across projects and efforts.
6. The data file can be created based on the above factors, and it should give the project manager a proposed schedule based on model constraints. The project manager can tweak the data file iteratively to understand various what-if scenarios based on which resource is available and what skill sets they bring to the table.
7. The above process can be repeated each time we are ready to start a new epic, and this should help lay out a tentative schedule.

The goal at the end of 3-4 epics should be to have a cross-functional team of full-stack developers that are proficient in multiple skills and can execute projects in the most optimal way. The model will have utility until the team reaches a point where everyone has the highest possible efficiency score for all skills. That very rarely happens in the real world and so we will always have the need for such a model.

6.3.8 Computational Experiments

The goal of the computational experiments was to conduct sensitivity analyses to examine how varying levels of the efficiency score, workload, and payrate affect the overall objective value and the number of stories allocated. We wrote a Java program to vary the efficiency score, workload availability, and payrate as listed below to generate 27 combinations of the variables for the computational experiments. We then ran the model in batch mode three times, once for each sized set of projects, with each batch including these 27 combinations. The experiments were run on a machine with an Intel Quad Core 3.0 GHz CPU with 8 GB RAM.

- Vary Size
  - Low : Five Projects
  - Medium: Ten Projects
  - Large: Twenty Projects
- Vary Efficiency Score for each size above
  - Low : Reduce Efficiency Score by 30%
  - Medium: Reduce Efficiency Score by 15%
  - High: At an optimal level, all stories are allocated
- Vary Workload Availability for each size above
  - Low : Reduce Workload availability by 30%
  - Medium: Reduce Workload availability by 15%
  - High: At an optimal level, all stories are allocated

- Vary Payrate for each size above
    - Low : Reduce Pay Rate by 30%
    - Medium: Reduce Pay Rate by 15%
    - High : At an optimal level, all stories are allocated

Each of the batch runs generated a text file. The output from the text file for the batch run of 27 files for the small size (five projects) is shown in Table 15. We generated similar files for the medium (10 projects) and large (20 projects) batch runs. A snippet of the data from the batch run for the small size grouping is shown below in Table 22.

Table 22: Computational experiment results from the batch run

Size	WL	EF	PR	# of projects	# of Stories	# of spirints	# of skills	# of resou	#together	#incompa	Cplex Stat	Cplex Time	Actual Time
1	3	3	3	5	57	4	20	8	1	5	102	2495365.953	
1	3	3	1	5	57	4	20	8	1	5	102	2495372.484	6.531
1	3	3	2	5	57	4	20	8	1	5	102	2495377.375	4.891
1	3	1	3	5	57	4	20	8	1	5	102	2495394.515	17.14
1	3	1	1	5	57	4	20	8	1	5	11	2498996.859	3602.344
1	3	1	2	5	57	4	20	8	1	5	102	2499067.343	70.484
1	3	2	3	5	57	4	20	8	1	5	102	2499084.5	17.157
1	3	2	1	5	57	4	20	8	1	5	102	2500731.5	1647
1	3	2	2	5	57	4	20	8	1	5	102	2500936.046	204.546
1	1	3	3	5	57	4	20	8	1	5	11	2504537.218	3601.172
1	1	3	1	5	57	4	20	8	1	5	11	2508161	3623.782
1	1	3	2	5	57	4	20	8	1	5	11	2512091.265	3930.265
1	1	1	3	5	57	4	20	8	1	5	102	2512093.39	2.125
1	1	1	1	5	57	4	20	8	1	5	102	2512095.312	1.922
1	1	1	2	5	57	4	20	8	1	5	102	2512097.343	2.031
1	1	2	3	5	57	4	20	8	1	5	102	2512099	1.657
1	1	2	1	5	57	4	20	8	1	5	102	2512100.5	1.5
1	1	2	2	5	57	4	20	8	1	5	102	2512102.734	2.234
1	2	3	3	5	57	4	20	8	1	5	102	2512379.5	276.766
1	2	3	1	5	57	4	20	8	1	5	102	2513528.89	1149.39

Objective Value	MIP GAP	RL1	RL2	RL3	RL4	RL5	RL6	RL7	RL8	SL1	SL2	SL3	SL4	Cost	Value	# of storie
217149.9844	0.000991	517	407	405	390	500	459	354	490	940	890	920	772	184177.8	401327.8	57
272505.2114	0.000327	499	417	405	370	520	484	340	487	900	920	870	832	128915.1	401420.3	57
246956.8417	0.000993	500	420	395	380	505	467	375	480	910	930	830	852	154506.9	401463.7	57
147824.707	0.001	360	275	320	320	362	320	185	248	650	660	580	500	176547.5	324372.2	40
203726.629	0.001827	360	286	320	320	359	360	310	340	670	670	670	645	138189.1	341915.7	45
177085.6399	0.000763	360	290	320	320	357	358	225	345	660	670	610	635	159483.9	336569.5	44
183938.7764	0.000565	438	342	400	360	440	315	212	375	820	760	775	527	174928.1	358866.9	49
242503.8313	0.000532	440	347	400	392	440	427	401	435	830	810	820	822	140888	383391.9	54
214649.0956	0.001	440	357	400	370	430	421	357	440	820	820	780	795	164755	379404.1	52
203373.9521	0.004227	362	274	320	320	362	345	295	339	660	640	657	660	136276.5	339650.5	44
245217.2145	0.005571	362	299	320	320	360	352	310	359	670	680	670	662	97350.09	342567.3	46
225871.4254	0.004189	361	296	320	320	361	343	320	334	660	670	660	665	115924.9	341796.4	45
47248.15427	0.000978	180	80	55	0	120	50	0	55	330	150	60	0	38240.87	85489.03	12
60600.28501	0.000967	232	105	65	0	140	95	0	105	330	272	80	60	37036.97	97637.26	17
54390.68114	0.000851	217	210	65	0	115	60	0	20	280	200	100	107	40119.87	94510.55	15
75216.19759	0.000589	255	120	185	70	157	65	0	75	460	287	140	40	54887.94	130104.1	18
92296.5819	0.000848	255	115	195	60	212	75	0	70	460	235	247	40	40705.34	133001.9	20
84895.6905	0.000805	277	245	190	65	90	65	0	50	460	220	187	115	48038.47	132934.2	20
210194.0273	0.000772	425	354	376	365	440	440	390	425	790	810	800	815	169150.5	379344.5	52
260874.0764	0.000612	440	349	377	361	425	440	398	425	790	810	810	805	118554.9	379428.9	52

The results from the batch runs are summarized below. Figure 57 visualizes the results of the sensitivity analysis for the overall objective value. Each line represents the objective value for each unique combination of size, efficiency score, workload, and payrate. We observe higher objective values when the efficiency score is high, and the work load availability is high. The highest objective value is observed when the efficiency score is high, workload availability is high, and payrate is low. The lowest objective value is observed when the efficiency score is low, workload availability is low, and payrate is high. Figure 58 visualizes the results of the sensitivity analysis for the number of stories allocated. Each line represents the number of stories allocated for each unique combination of size, efficiency score, workload, and payrate. We observe that a greater number of stories is allocated when the efficiency score is high, and the work load availability is high. The greatest number of stories is allocated when the efficiency score is high, workload availability is high, and payrate is low. The lowest number of stories is allocated when the efficiency score is low, workload availability is low, and payrate is high.

Figure 57: Result of computational experiments – Objective value analysis

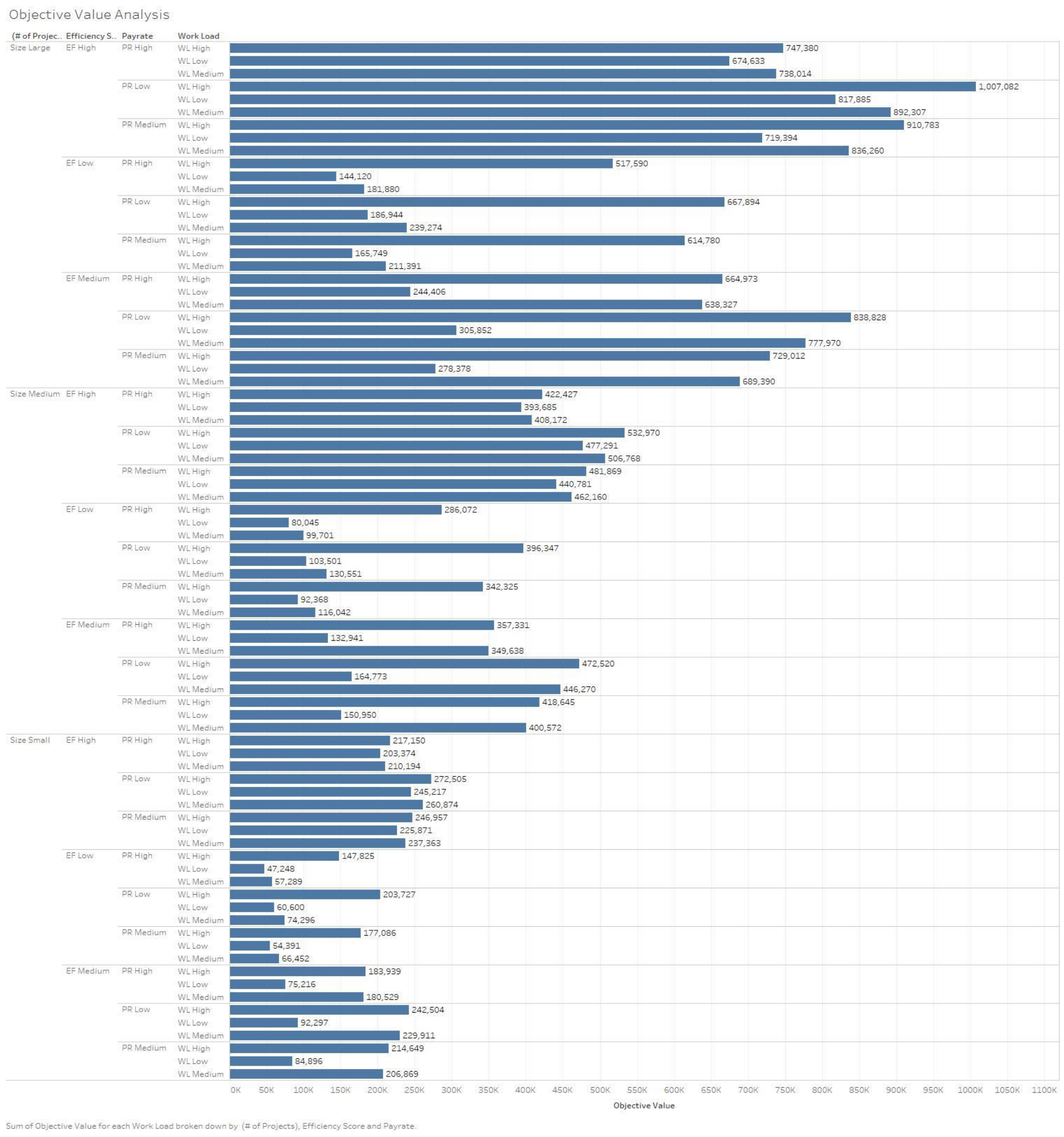
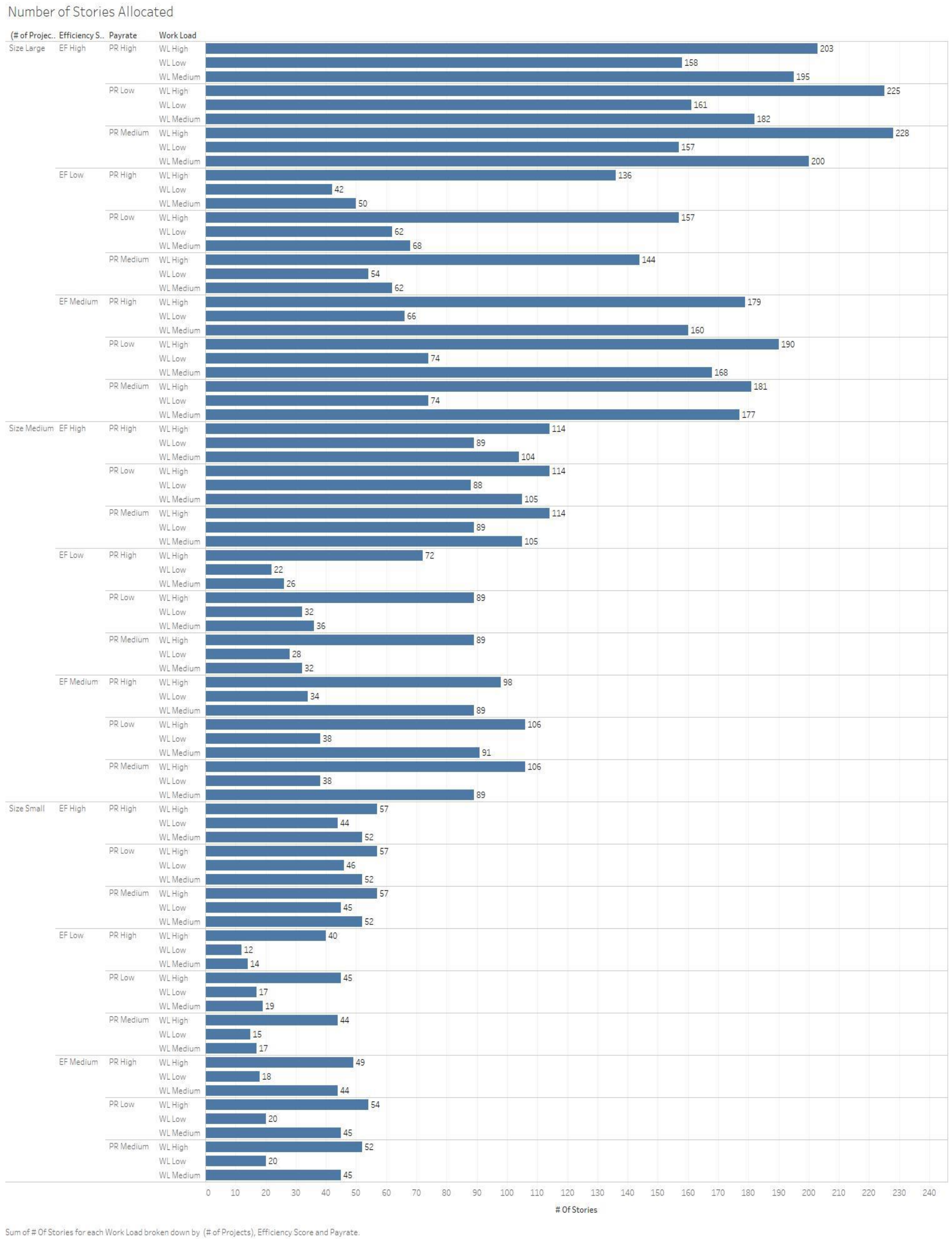


Figure 58: Result of computational experiments – Number of stories allocated



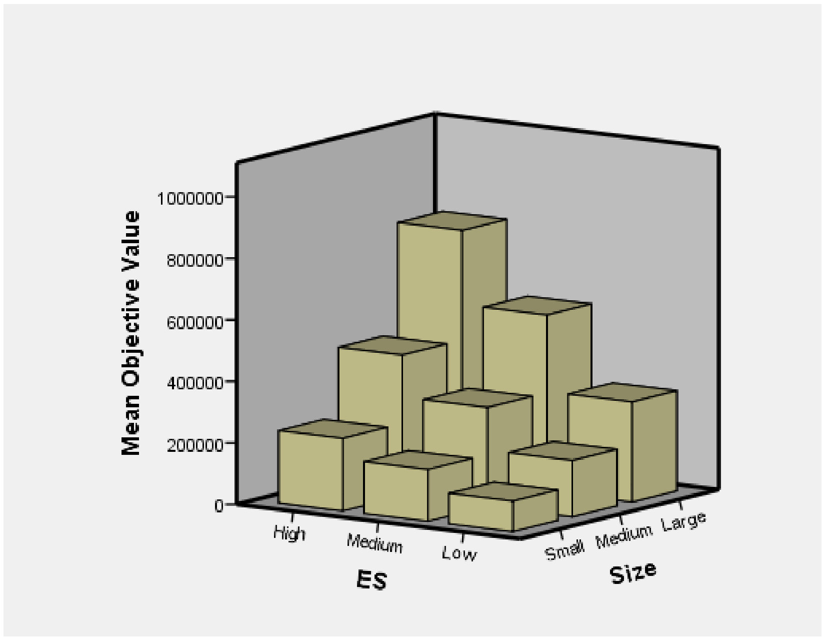
All three batch files were then combined to generate one combined file with the following fields, as shown in Table 23 to do a sensitivity analysis on the overall objective value and the number of stories allocated to varying levels of efficiency score, workload, and payrate. We look at each of these attributes individually to garner insights.

Table 23: Sensitivity Analysis dataset

Size	WL	EF	PR	Objective Value	# of stories Allocated
Small	High	High	High	217149.9844	57
Small	High	High	Low	272505.2114	57
Small	High	High	Medium	246956.8417	57
Small	High	Low	High	147824.707	40
Small	High	Low	Low	203726.629	45
Small	High	Low	Medium	177085.6399	44
Small	High	Medium	High	183938.7764	49
Small	High	Medium	Low	242503.8313	54
Small	High	Medium	Medium	214649.0956	52
Small	Low	High	High	203373.9521	44
Small	Low	High	Low	245217.2145	46
Small	Low	High	Medium	225871.4254	45
Small	Low	Low	High	47248.15427	12
Small	Low	Low	Low	60600.28501	17
Small	Low	Low	Medium	54390.68114	15
Small	Low	Medium	High	75216.19759	18
Small	Low	Medium	Low	92296.5819	20
Small	Low	Medium	Medium	84895.6905	20
Small	Medium	High	High	210194.0273	52
Small	Medium	High	Low	260874.0764	52
Small	Medium	High	Medium	237363.4832	52
Small	Medium	Low	High	57288.8107	14
Small	Medium	Low	Low	74295.84362	19
Small	Medium	Low	Medium	66451.76148	17
Small	Medium	Medium	High	180529.2469	44
Small	Medium	Medium	Low	229910.7105	45
Small	Medium	Medium	Medium	206869.2513	45
Medium	High	High	High	422427.184	114
Medium	High	High	Low	532970.3436	114
Medium	High	High	Medium	481869.0677	114
Medium	High	Low	High	286071.9711	72
Medium	High	Low	Low	396347.3062	89
Medium	High	Low	Medium	342325.2425	89
Medium	High	Medium	High	357331.0604	98

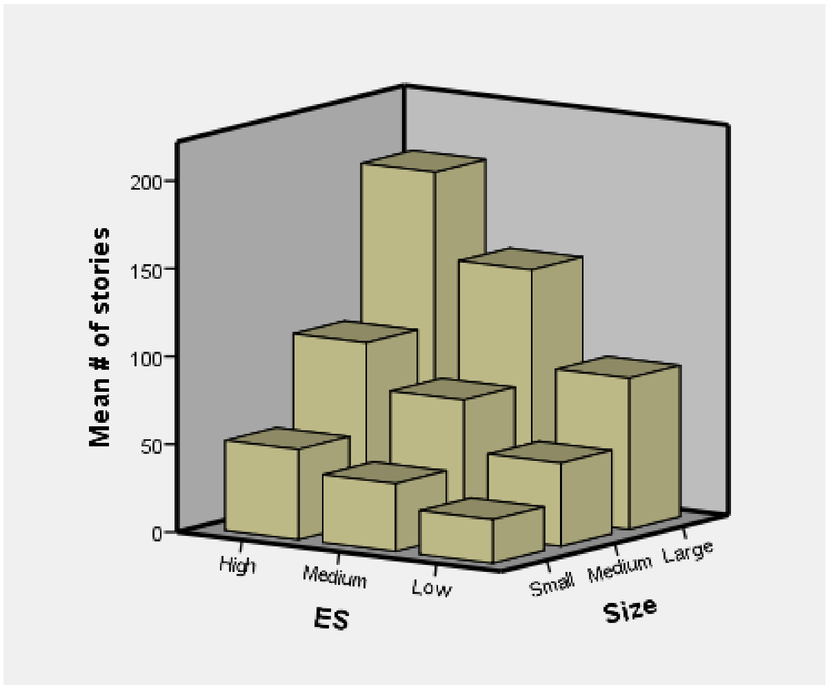
The results of the sensitivity analysis of the overall objective value to varying levels of efficiency scores for different combinations of projects in terms of size (small: five projects, medium: ten projects, large: twenty projects) is shown in shown in Figure 59. The small size represents the run with five projects while the medium size represents the run with ten projects, and finally the large run represents the run with twenty projects. We observe that the higher efficiency scores result in higher overall objective values across all three project size groupings. Similarly, lower efficiency scores result in lower overall objective values across all three project size groupings. We also observe that the sensitivity of optimal objective value with respective to efficiency score increases when project group size increases.

Figure 59: Sensitivity analysis of objective value to varying efficiency scores and sizes



The results of the sensitivity analysis of number of stories allocated to varying levels of efficiency scores for different combinations of projects in terms of size (small: five projects, medium: ten projects, large: twenty projects) is shown in Figure 60. We observe that the higher efficiency scores result in higher number of stories being allocated across all three project size groupings. Similarly, lower efficiency scores results in lower number of stories being allocated across all three project size groupings. We also observe that the sensitivity of the number of stories allocated with respective to efficiency score increases when project group size increases.

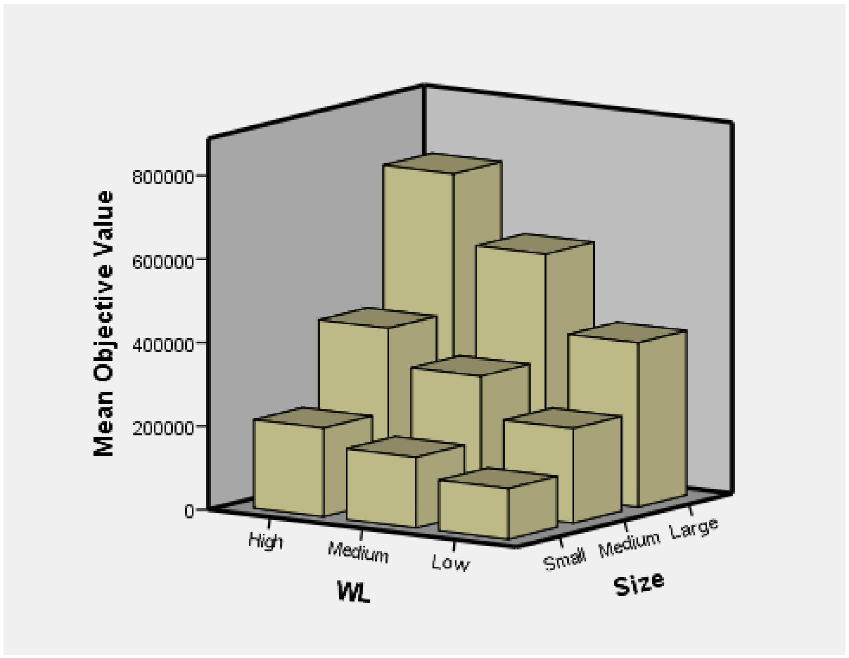
Figure 60: Sensitivity analysis of number of stories allocated to varying efficiency scores and sizes



These findings support our first hypothesis that a team comprised of resources with higher efficiency scores in multiple skills will result in more stories being assigned, thereby resulting in overall better objective values.

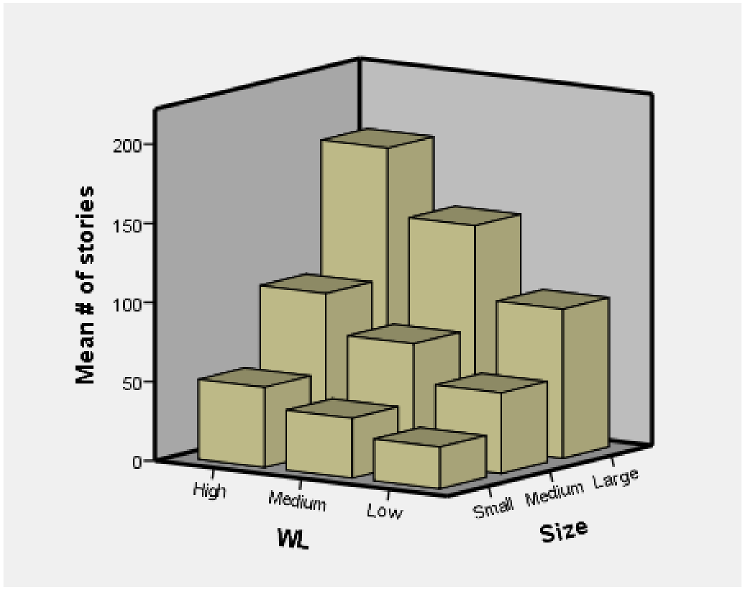
The results of the sensitivity analysis of overall objective value to varying levels of workload availability for different combinations of projects in terms of size (small: five projects, medium: ten projects, large: twenty projects) is shown in shown in Figure 61. We observe that the higher workload availability results in higher overall objective values across all three project size groupings. Similarly, lower workload availability results in lower overall objective values across all three project size groupings.

Figure 61: Sensitivity analysis of objective value to varying workload availability and sizes



The results of the sensitivity analysis of number of stories allocated to varying levels of workload availability for different combinations of projects in terms of size (small: five projects, medium: ten projects, large: twenty projects) is shown in in Figure 62. We observe that the higher workload availability results in higher number of stories being allocated across all three project size groupings. Similarly, lower workload availability results in lower number of stories being allocated across all three project size groupings.

Figure 62: Sensitivity analysis of number of stories allocated to varying workload availability and sizes

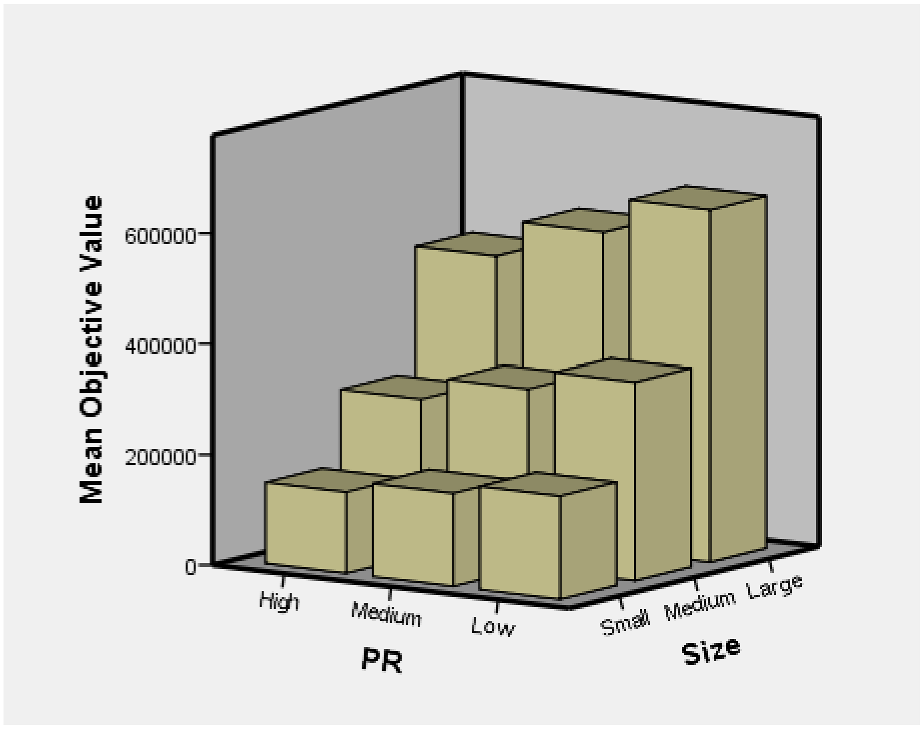


This observation supports our second hypothesis that a team with higher workload resource availability will have a more efficient schedule with more stories being taken up for allocation, and it will result in overall better objective value.

The results of the sensitivity analysis of objective value and number of stories allocated to varying levels of efficiency scores and workload availability as illustrated above support our hypothesis 2a that workload resource availability in conjunction with higher efficiency score will result in better outcomes in terms of overall better objective value and schedule efficiency in terms of number of stories allocated.

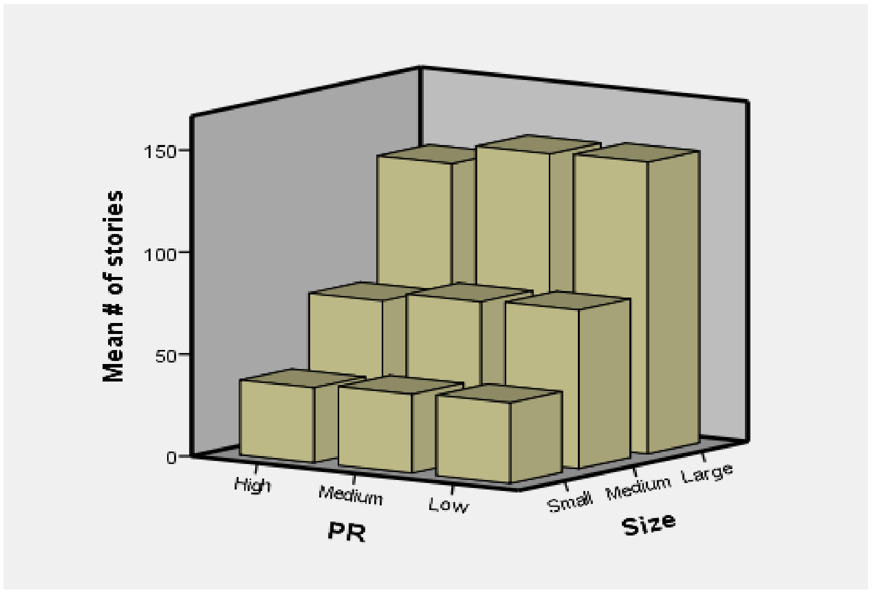
The results of the sensitivity analysis of overall objective value to varying levels of payrates for different combinations of projects in terms of size (small: five projects, medium: ten projects, large: twenty projects) is shown in shown in Figure 63. We observe that decreasing the payrate results in results in higher objective value across all three project size groupings.

Figure 63: Sensitivity analysis of objective value to varying payrates and sizes



The results of the sensitivity analysis of number of stories allocated to varying levels of payrates for different combinations of projects in terms of size (small: five projects, medium: ten projects, large: twenty projects) is shown in Figure 64. We observe that the increasing or decreasing the payrate does not directly result in more or less number of stories being allocated across all three project size groupings.

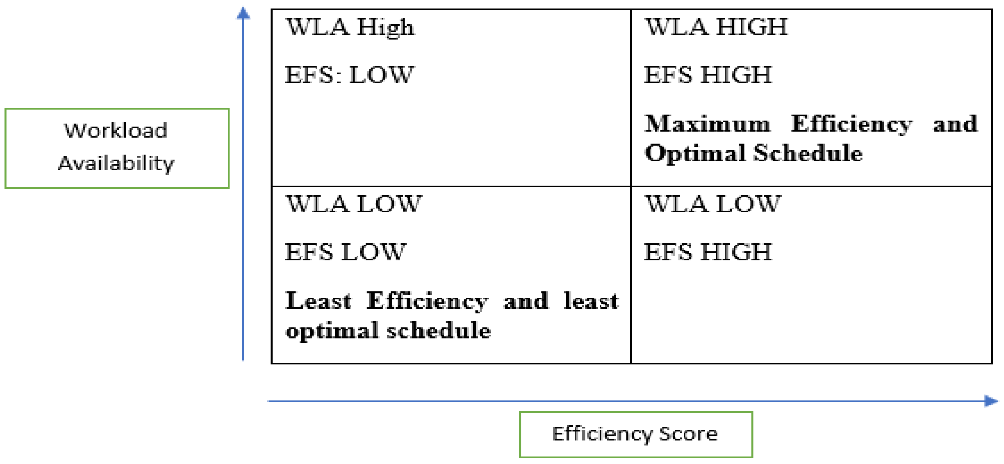
Figure 64: Sensitivity analysis of number of stories allocated to varying payrates and sizes



The results of the sensitivity analysis of objective value and number of stories allocated to varying levels of payrates, as illustrated above are mixed and not obvious. This results in our third hypothesis not being supported.

The results can be summarized, as shown in Figure 65.

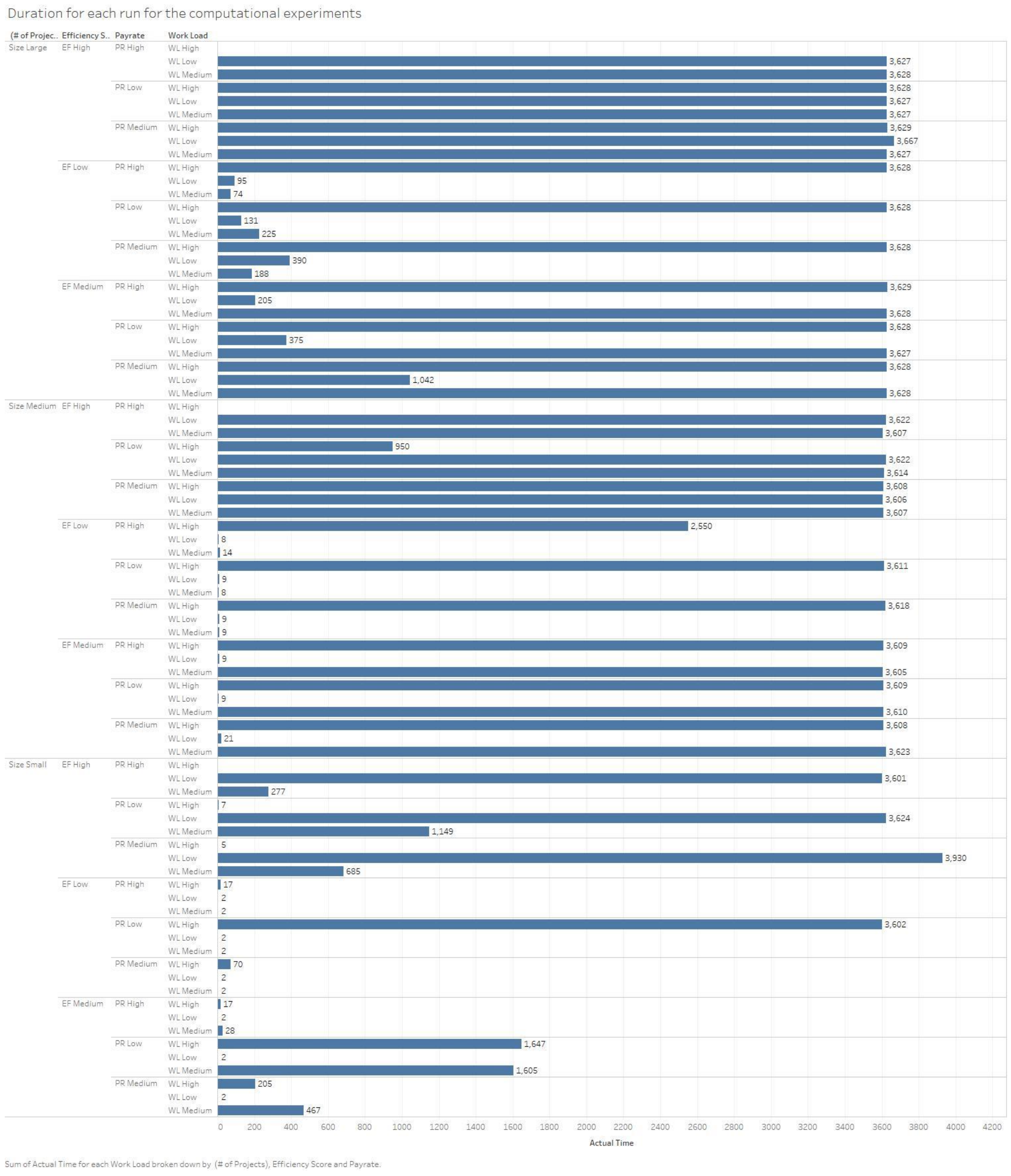
Figure 65: Computational Experiments results summary



The effect of the first three hypotheses being supported by the results of the computational experiments can be summarized as follows. A team should strive to have a dedicated team of resources who are part of a cross functional team and are proficient in multiple skills to have the most efficient schedules. This will result in a greater number of stories being allocated and overall better return on investment (ROI) in terms of objective value. The resources should have the highest workload availability and highest efficiency scores for each of the skills that are required as part of delivering projects.

Finally, the duration for each of the runs is summarized below. We had a time limit of 10 minutes computing time to stop the run.

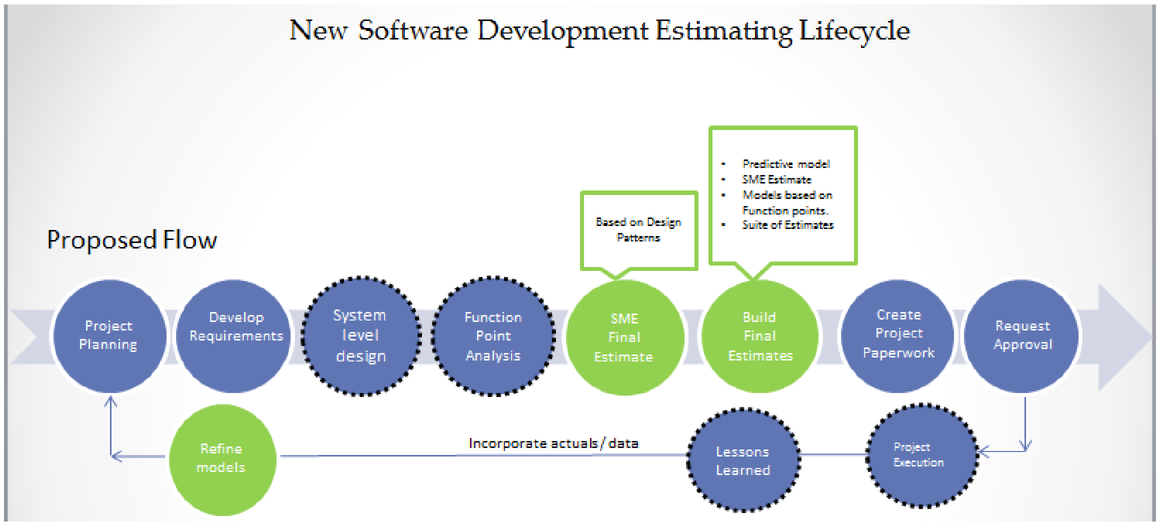
Figure 66: Duration for all runs in the batch process



Chapter 7: Conclusion and Future Work

7.1 Summary

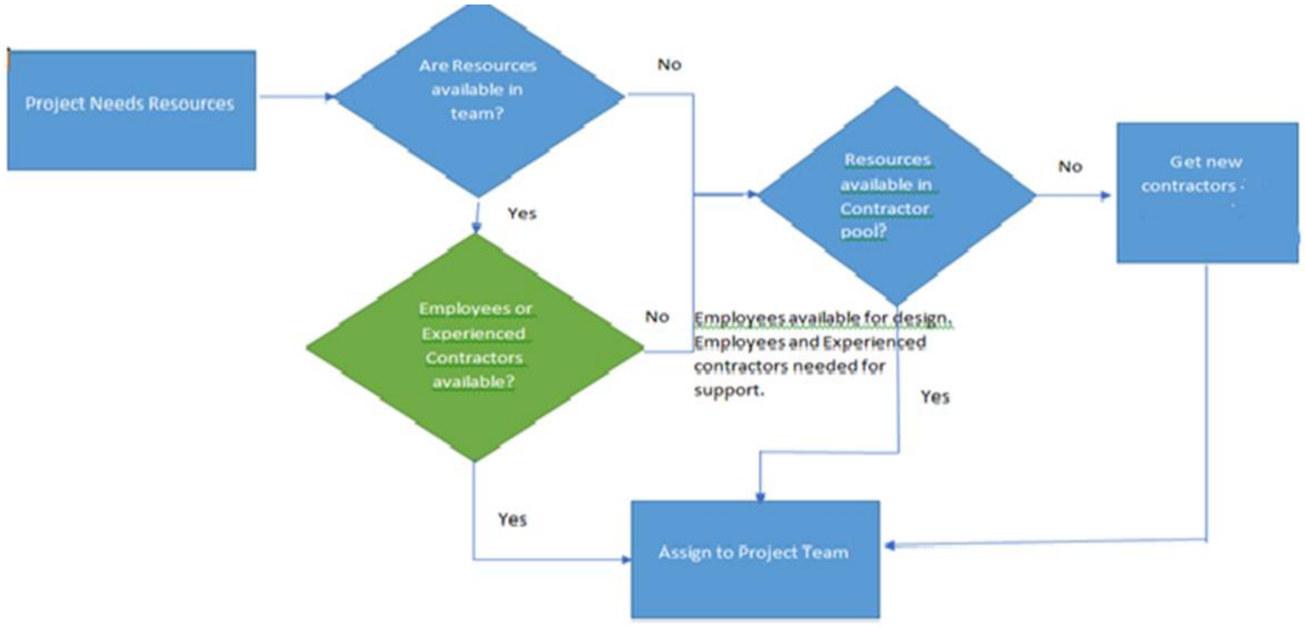
This research develops a new integrated predictive-prescriptive analytical approach to improve Software Estimating and Agile project management, based on Design Patterns that are unique to an organization. There was a quote given by an Enterprise Architect at ABC Inc., who was one of my mentors at the organization that helped me with the development of the baselines for the estimating tool. "*To me patterns are just an approach to solving a problem that can be applied to multiple domains. I don't need a fancy name to describe it, as long as I can describe the principles behind it*". He also said, "*In a nutshell, patterns are everywhere.*" This research developed a systematic and structured approach to analyze the data and identify those design patterns that were unique to ABC Inc. It was an eye-opening experience as we could fit in literally thousands of individual tasks from multiple projects into a finite set of less than 10 unique design patterns. It is a painstaking process to analyze thousands of lines of task level data and fit them into patterns, but the result of the effort was a clear understanding of how the organization executed application development projects. This research resulted in the development of a new estimating tool for ABC Inc. and has been implemented for the 28 projects over the past 15 months. We had five projects that completed execution, and the results from those projects were analyzed to assess the quality of estimation of the tool. The new tool has reduced the variability in the estimating process and has been adopted well by the technical subject matter experts and the leadership at the organization. The estimating process has been incorporated into the formal project life cycle at the organization, and that resulted in the new software development estimating lifecycle as shown in Figure 36 above and shown again below.



This research builds out the development of the estimating model in generic terms so that the process can be repeated by other organizations to develop their own version of the estimating tool. This tool will bring consistency to the estimating process and will reduce the variability in the process. It will significantly help reduce the likelihood for projects to exceed the budget.

The second part of this research dealt with better understanding labor costs associated with the project using a two-stage least squares model. The results of the predictive model enabled us to take a deeper look into how resources were allocated to projects. The model results pointed to the fact that new contractors were assigned to the project team, and this resulted in overall increased labor costs to the project as compared to assigning experienced contractors or employees to the project. A deeper look into the onboarding of contractors pointed to the workflow outlined below in Figure 67.

Figure 67: Assignment of resources to a project


Project Needs Resources
Are Resources available in team?
No
Yes
Resources available in Contractor pool?
No
Get new contractors
Employees or Experienced Contractors available?
No
Employees available for design. Employees and Experienced contractors needed for support.
Yes
Yes
Assign to Project Team

There was a task force at ABC Inc. that was already looking into how to improve the process of onboarding contractors at the organization. Our research findings were shared with the members of the task force. This has resulted in a revised process whereby contractors are familiarized with the development environment at the organization, and they work on small support tasks before being assigned to a project. We expect a better process flow and reduced labor costs over the long term, but we did not have the opportunity to analyze the impact of this revised flow. The aggregation of the various disparate data sources and descriptive analysis of the data pointed a lot of interesting trends at the organization. This research has given ABC Inc. a deeper understanding of how resources are used across the organization on different types of efforts including projects, support, enhancements, etc. More importantly, it has given the organization a deeper understanding of how temporary resources are used to augment employees across the organization in all types of efforts.

The third part of this research produced two ensembles of estimating models. The first ensemble brought together two estimates that were rooted in the new estimating model based on design patterns and one estimate that was based on the two-stage least squares predictive model. The second ensemble relied on the company continuing to use function points to size projects, and it produced four estimates. The first ensemble was based on the bottom-up estimating approach, and the second ensemble used the top-down approach to size projects. The biggest takeaway for ABC Inc. was the opportunity for the organization to combine executive judgment with data science to reach consensus on the final estimate for a project. This process has set the stage for executive management to have discussions with project managers and technical leaders, and it has helped reach consensus on a more balanced estimate. This research also set the foundation by assessing the quality of the estimate for each project.

This research also took a deeper look into the function point repository from a historical perspective. We created a couple of predictive models using the function points data, and this has helped ABC Inc. get a deeper understanding of the underlying data. This research has provided ABC Inc. with a predictive model to estimate a project based on function points. It also points to the fact that systems are tightly integrated at the organization as borne out by the results of the second predictive model based on the individual components that make up the overall function points.

The final part of this research was to augment the use of the estimating model to build a data-driven resource allocation framework that was rooted in a prescriptive analytics (optimization) model that consider multiple skills and domain expertise of resources. This research provides thought leadership for companies considering a move from Waterfall to an Agile process. It provides a metric to measure skills development across multiple sprints, and also sets the foundation for cross training of skills among resources, and it helps the organization move towards self-driven teams. The perceived shortage of SMEs in organizations like ABC Inc. is addressed when this model is used, and it should not be an issue going forward.

The prescriptive analytics model based Agile planning approach has the following benefits.

1. Balance the effects of workload availability with skills development of resources within the project execution
2. Lays the foundation for a self-driven and cross functional team that is rooted in a foundation of skills development/sharing of skills
3. Efficiency score and workload availability are key metrics to be considered by organizations.

Predictive Analytics in the estimating process generates credible and quality estimates, which serve as inputs to the Prescriptive Analytics model that optimize the data-driven planning and resource allocation decisions in the Agile environment.

7.2 Limitations

This research built the new estimating tool but did not take up the continuous refinement of the task level estimates to adjust the baseline estimates in the estimating tool. We set up a Tableau dashboard to process the data at the task level and have set a solid foundation for the continuous refinement of the task level estimates. The continuous refinement of estimates at the task level will add more value to the estimating tool.

The estimating tool was used for more than 25 projects, but only five projects have completed execution. The remaining projects are in progress. It would have been helpful if we had a bigger base of projects that completed execution to better assess its financial benefit. We had access to data from the quality perspective or the testing area, but we could not relate that data back to the base project data due to variety of internal factors at ABC Inc. It will be good to spend some time after this research to fix the issues with this data and take a deeper dive into this data. This research has set up the foundation to collect good quality data in this space.

The organization is starting to execute a few projects in Agile. We took data from five small projects that were executed in a Waterfall methodology and mocked it up in an Agile context in consultation with the business users to form the basis for the optimization model. The optimization model and the usage of an efficiency score to track skills development is currently being considered on a pilot basis in a few Agile projects, and it will take some time to observe and analyze the outcomes.

7.3 Future Work

There is an opportunity to use the actual data from projects that use the new estimating tool to create a Bayesian updating based approach to enhance the quality of estimates. This research created the foundation to aggregate the data in a Tableau dashboard for each individual task category, which facilitate the development of the Bayesian models in the future.

There will be a larger set of projects that are going to complete execution in the next 6-12 months, and there is an opportunity to assess the economic impact of the estimating tool. This research laid the foundation of assessing the quality of the estimate for each project. The next step would be to process the data after the development of the estimating tool and compare it to the era prior to the development of the estimating tool. This will provide a more accurate assessment of the economic benefit of the estimating tool.

There is an opportunity to extend the estimating tool to introduce risk at the task level and develop more sophisticated models and methods that explicitly cope with uncertainty. For example, one may start with a Monte Carlo Simulation to predict the likelihood of the project meeting its delivery date. This will provide useful information for project managers, and it can be the next logical step to extend the estimating tool.

There is additional opportunity to address other objective functions building on the established framework. We can maximize the number of stories allocated within the planning horizon. It is also possible to consider the minimum completion time for all stories in the planning horizon, that is, the minimize the makespan of some selected stories. There is also good opportunity to implement the prescriptive analytics model on a pilot Agile project, and it will be helpful to get feedback and refine the model based on the feedback. This research proposed a framework to use the efficiency score as the basis to lay the foundation for skills development in an organization. An Agile Dojo is a physical space dedicated to accelerated learning for teams, and ABC Inc. is starting to encourage teams to learn in this new environment. It will be great to see this framework being implemented in an such an environment which encourages cross sharing among team members. ABC Inc. is investing and moving in this direction. We should soon have data available in this space. The optimization model needs to be made production-ready by getting it to accept input from an Excel sheet and write output to an Excel sheet. The interface for that can be written in Python or Java, and that will be needed for organizations to use the model on a regular basis.

References

Abdel-Hamid, A. N., & Abdel-Kader, M. A. (2011). *Process increments: An agile approach to software process improvement.* Paper presented at the Proceedings - 2011 Agile Conference, Agile 2011.

About ISBSG. Retrieved from http://isbsg.org/about-isbsg/

Albrecht, A. J., & Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering*(6), 639-648.

Baresi, L., Morasca, S., & Paolini, P. (2003). *Estimating the design effort of web applications.* Paper presented at the Software Metrics Symposium, 2003. Proceedings. Ninth International.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Jeffries, R. (2001). Manifesto for agile software development.

Bernard, J. M. (2015). An Application of Data Analytics to Outcomes of Missouri Motor Vehicle Crashes.

Bibi, S., & Stamelos, I. (2006) Selecting the appropriate machine learning techniques for the prediction of software development costs. In*: Vol. 204. IFIP International Federation for Information Processing* (pp. 533-540).

Bibi, S., Stamelos, I., & Angelis, L. (2008). Combining probabilistic models for explanatory productivity estimation. *Information and Software Technology, 50*(7-8), 656-669. doi:10.1016/j.infsof.2007.06.004

Bilgaiyan, S., Sagnika, S., Mishra, S., & Das, M. (2017). A systematic review on software cost estimation in Agile Software Development. *Journal of Engineering Science and Technology Review, 10*(4), 51-64. doi:10.25103/jestr.104.08

Boschetti, M. A., Golfarelli, M., Rizzi, S., & Turricchia, E. (2014). A Lagrangian heuristic for sprint planning in agile software development. *Computers and Operations Research, 43*(1), 116-128. doi:10.1016/j.cor.2013.09.007

Čeke, D., & Milašinović, B. (2015). Early effort estimation in web application development. *Journal of Systems and Software, 103*, 219-237. doi:10.1016/j.jss.2015.02.006

Chemuturi, M. (2009). *Software estimation best practices, tools & techniques: A complete guide for software project estimators*: J. Ross Publishing.

Cocomo, I. (2000). Model Definition Manual. *Copyright Center for Software Engineering, USC*.

Costagliola, G., Di Martino, S., Ferrucci, F., Gravino, C., Tortora, G., & Vitiello, G. (2006). A COSMIC-FFP approach to predict web application development effort. *J. Web Eng, 5*(2), 93-120.

Costagliola, G., Ferrucci, F., Tortora, G., & Vitiello, G. (2005). Class point: An approach for the size estimation of object-oriented systems. *IEEE Transactions on Software Engineering, 31*(1), 52-74. doi:10.1109/TSE.2005.5

CPrime. (2013). Scrum-FAQ.

Cunha, J. C., Cruz, S., Costa, M., Rodrigues, A. R., & Vieira, M. (2012). *Implementing software effort estimation in a medium-sized company.* Paper presented at the Proceedings - 2011 34th IEEE Software Engineering Workshop, SEW 2011.

Cusumano, M. A. (2007). Extreme programming compared with Microsoft-style iterative development. *Communications of the ACM, 50*(10), 15-18. doi:10.1145/1290958.1290979

Da Silva, T. S., Martin, A., Maurer, F., & Silveira, M. (2011). *User-centered design and agile methods: A systematic review.* Paper presented at the Proceedings - 2011 Agile Conference, Agile 2011.

Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2012). Data mining techniques for software effort estimation: A comparative study. *IEEE Transactions on Software Engineering, 38*(2), 375-397. doi:10.1109/TSE.2011.55

Di Martino, S., Ferrucci, F., Gravino, C., & Mendes, E. (2007). *Comparing size measures for predicting Web application development effort: A case study.* Paper presented at the Proceedings - 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007.

Dolado, J. J. (2000). A validation of the component-based method for software size estimation. *IEEE Transactions on Software Engineering, 26*(10), 1006-1021. doi:10.1109/32.879821

Dong, X., Yang, Q. S., Wang, Q., Zhai, J., & Ruhe, G. (2011). *Value-risk trade-off analysis for iteration planning in eXtreme Programming.* Paper presented at the Proceedings - Asia-Pacific Software Engineering Conference, APSEC.

Dragicevic, S., Celar, S., & Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software, 127*, 109-119. doi:10.1016/j.jss.2017.01.027

Evans, J. R., & Lindner, C. H. (2012). Business analytics: the next frontier for decision sciences. *Decision Line, 43*(2), 4-6.

Ferrucci, F., Gravino, C., & Di Martino, S. (2008). *A case study using web objects and COSMIC for effort estimation of web applications.* Paper presented at the EUROMICRO 2008 - Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2008.

Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., & Mendes, E. (2010). *Investigating tabu search for web effort estimation.* Paper presented at the Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on.

Forsyth, D. K., & Burt, C. D. B. (2008). Allocating time to future tasks: The effect of task segmentation on planning fallacy bias. *Memory and Cognition, 36*(4), 791-798. doi:10.3758/MC.36.4.791

Gartner. (2014). *Gartner Says Optimizing Application Development and Maintenance Can Cut Costs by More Than 50 Percent* Retrieved from STAMFORD, Conn: https://www.gartner.com/newsroom/id/2711017

Grapenthin, S., Poggel, S., Book, M., & Gruhn, V. (2015). Improving task breakdown comprehensiveness in agile projects with an Interaction Room. *Information and Software Technology, 67*, 254-264. doi:10.1016/j.infsof.2015.07.008

Gray, A., & MacDonell, S. G. (1997). A comparison of techniques for developing predictive models of software metrics.

Hakuta, M., Tone, F., & Ohminami, M. (1997). A software size estimation model and its evaluation. *Journal of Systems and Software, 37*(3), 253-263.

Halkjelsvik, T., & Jørgensen, M. (2012). From Origami to software development: A review of studies on judgment-based predictions of performance time. *Psychological Bulletin, 138*(2), 238-271. doi:10.1037/a0025996

Hannay, J. E., Benestad, H. C., & Strand, K. (2017). Earned Business Value: See That You Deliver Value to Your Customer. *IEEE Software, 34*(4), 58-70. doi:10.1109/MS.2017.105

Hayata, T., & Han, J. (2011). *A hybrid model for IT project with Scrum.* Paper presented at the Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on.

Hill, J., Thomas, L. C., & Allen, D. E. (2000). Experts' estimates of task durations in software development projects. *International Journal of Project Management, 18*(1), 13-21. doi:10.1016/S0263-7863(98)00062-3

Hoda, R., Salleh, N., Grundy, J., & Tee, H. M. (2017). Systematic literature reviews in agile software development: A tertiary study. *Information and Software Technology, 85*, 60-70. doi:https://doi.org/10.1016/j.infsof.2017.01.007

Hsu, C. J., Rodas, N. U., Huang, C. Y., & Peng, K. L. (2010). *A study of improving the accuracy of software effort estimation using linearly weighted combinations.* Paper presented at the Proceedings - International Computer Software and Applications Conference.

Idri, A., Amazal, F. A., & Abran, A. (2015). Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software Technology, 58*, 206-230. doi:10.1016/j.infsof.2014.07.013

Idri, A., Hosni, M., & Abran, A. (2016a). Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles. *Applied Soft Computing Journal, 49*, 990-1019. doi:10.1016/j.asoc.2016.08.012

Idri, A., Hosni, M., & Abran, A. (2016b). Systematic literature review of ensemble effort estimation. *Journal of Systems and Software, 118*, 151-175. doi:10.1016/j.jss.2016.05.016

Jenkins, A. M., Naumann, J. D., & Wetherbe, J. C. (1984). Empirical investigation of systems development practices and results. *Information and Management, 7*(2), 73-82. doi:10.1016/0378-7206(84)90012-0

Jones, T. C. (2007). *Estimating software costs*: McGraw-Hill, Inc.

Jørgensen, M. (2004a). A review of studies on expert estimation of software development effort. *Journal of Systems and Software, 70*(1-2), 37-60. doi:10.1016/S0164-1212(02)00156-5

Jørgensen, M. (2004b). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology, 46*(1), 3-16. doi:10.1016/S0950-5849(03)00093-4

Jørgensen, M. (2007). Forecasting of software development work effort: Evidence on expert judgement and formal models. *International Journal of Forecasting, 23*(3), 449-462. doi:10.1016/j.ijforecast.2007.05.008

Jørgensen, M. (2014). What We Do and Don't Know about Software Development Effort Estimation. *IEEE Software, 31*(2), 37-40. doi:10.1109/MS.2014.49

Jørgensen, M., Boehm, B., & Rifkin, S. (2009). Software development effort estimation: Formal models or expert judgment? *IEEE Software, 26*(2), 14-19. doi:10.1109/MS.2009.47

Jørgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering, 33*(1), 33-53. doi:10.1109/TSE.2007.256943

Kanmani, S., Kathiravan, J., Senthil Kumar, S., & Shanmugam, M. (2007). *Neural network based effort estimation using class points for OO systems.* Paper presented at the Proceedings - International Conference on Computing: Theory and Applications, ICCTA 2007.

Kmenta, J. *Elements of Econometrics. 1971 (2nd edition 2011)* (Vol. 391).

Kocaguneli, E., Menzies, T., & Keung, J. W. (2012). On the value of ensemble effort estimation. *IEEE Transactions on Software Engineering, 38*(6), 1403-1416. doi:10.1109/TSE.2011.111

Ktata, O., & Lévesque, G. (2010). *Designing and implementing a measurement program for scrum teams: What do agile developers really need and want?* Paper presented at the ACM International Conference Proceeding Series.

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., . . . Hanser, E. (2017). *Hybrid software and system development in practice: waterfall, scrum, and beyond.* Paper presented at the Proceedings of the 2017 International Conference on Software and System Process.

Lederer, A. L., & Prasad, J. (1992). Nine Management Guidelines for Better Cost Estimating. *Communications of the ACM, 35*(2), 51-59. doi:10.1145/129630.129632

Lederer, A. L., & Prasad, J. (1995). Causes of inaccurate software development cost estimates. *The Journal of Systems and Software, 31*(2), 125-134. doi:10.1016/0164-1212(94)00092-2

Logue, K., & McDaid, K. (2008). *Agile release planning: Dealing with uncertainty in development time and business value.* Paper presented at the Proceedings - Fifteenth IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS 2008.

Longstreet, D. (2002). Fundamentals of function point analysis. *Longstreet Consulting, Inc*.

Low, G. C., & Jeffery, D. R. (1990). Function points in the estimation and evaluation of the software process. *IEEE Transactions on Software Engineering, 16*(1), 64-71.

MacDonell, S. G. (2003). Software source code sizing using fuzzy logic modeling. *Information and Software Technology, 45*(7 SPEC.), 389-404. doi:10.1016/S0950-5849(03)00011-9

MacDonell, S. G., & Shepperd, M. J. (2003a). Combining techniques to optimize effort predictions in software project management. *Journal of Systems and Software, 66*(2), 91-98. doi:10.1016/S0164-1212(02)00067-5

MacDonell, S. G., & Shepperd, M. J. (2003b). *Using prior-phase effort records for re-estimation during software projects.* Paper presented at the Proceedings - International Software Metrics Symposium.

Mair, C., & Shepperd, M. (2005). *The consistency of empirical comparisons of regression and analogy-based software project cost prediction.* Paper presented at the 2005 International Symposium on Empirical Software Engineering, ISESE 2005.

Mendes, E., Abutalib, M., & Counsell, S. (2012). *Applying knowledge elicitation to improve web effort estimation: A case study.* Paper presented at the Proceedings - International Computer Software and Applications Conference.

Mendes, E., & Counsell, S. (2000). *Web development effort estimation using analogy.* Paper presented at the Proceedings of the Australian Software Engineering Conference, ASWEC.

Mendes, E., Mosley, N., & Counsell, S. (2003). *Early Web size measures and effort prediction for Web costimation.* Paper presented at the Proceedings - International Software Metrics Symposium.

Mendes, E., Mosley, N., & Counsell, S. (2005). Investigating Web size metrics for early Web cost estimation. *Journal of Systems and Software, 77*(2), 157-172. doi:10.1016/j.jss.2004.08.034

Mendes, E., Watson, I., Triggs, C., Mosley, N., & Counsell, S. (2002). *A comparison of development effort estimation techniques for Web hypermedia applications.* Paper presented at the Proceedings - International Software Metrics Symposium.

Mittas, N., & Angelis, L. (2010). LSEbA: Least squares regression and estimation by analogy in a semi-parametric model for Software Cost Estimation. *Empirical Software Engineering, 15*(5), 523-555. doi:10.1007/s10664-010-9128-6

Miyazaki, Y., Terakado, M., Ozaki, K., & Nozaki, H. (1994). Robust regression for developing software estimation models. *The Journal of Systems and Software, 27*(1), 3-16. doi:10.1016/0164-1212(94)90110-4

Moløkken-Østvold, K., Haugen, N. C., & Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software, 81*(12), 2106-2117. doi:10.1016/j.jss.2008.03.058

Moløkken-Østvold, K., Jørgensen, M., Tanilkan, S. S., Gallis, H., Lien, A. C., & Hove, S. E. (2004). *A survey on software estimation in the norwegian industry*. Paper presented at the Proceedings - International Software Metrics Symposium.

Moløkken, K., & Jørgensen, M. (2003). *A review of software surveys on software effort estimation.* Paper presented at the Proceedings - 2003 International Symposium on Empirical Software Engineering, ISESE 2003.

Mortenson, M. J., Doherty, N. F., & Robinson, S. (2015). Operational research from Taylorism to Terabytes: A research agenda for the analytics age. *European Journal of Operational Research, 241*(3), 583-595.

Myrtveit, I., Stensrud, E., & Shepperd, M. (2005). Reliability and validity in comparative studies of software prediction models. *IEEE Transactions on Software Engineering, 31*(5), 380-391. doi:10.1109/TSE.2005.58

Nagler, J. (1999). Notes on Simultaneous Equations and Two Stage Least Squares Estimates

Nemhauser, G. L., & Wolsey, L. A. (1988). Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin, 20*, 8-12.

Parlati, G., Larenza, R., & Caronia, L. . (2011). *Measuring software 4 dummies*. Paper presented at the PMI® Global Congress 2011, Dallas Texas. https://www.pmi.org/learning/library/software-measuring-function-point-methodology-6201

Pendharkar, P. C. (2004). An exploratory study of object-oriented software component size determinants and the application of regression tree forecasting models. *Information and Management, 42*(1), 61-73. doi:10.1016/j.im.2003.12.004

Pfleeger, S. L., Jeffery, R., Curtis, B., & Kitchenham, B. (1997). Status report on software measurement. *IEEE Software, 14*(2), 33-43. doi:10.1109/52.582973

Raz, T., & Elnathan, D. (1999). Activity based costing for projects. *International Journal of Project Management, 17*(1), 61-67. doi:10.1016/S0263-7863(97)00073-2

Regolin, E. N., De Souza, G. A., Pozo, A. R. T., & Vergilio, S. R. (2003). *Exploring machine learning techniques for software size estimation.* Paper presented at the Proceedings - International Conference of the Chilean Computer Science Society, SCCC.

Reifer, D. J. (2000). Web development: Estimating quick-to-market software. *IEEE Software*, *17*(6), 57-64. doi:10.1109/52.895169

Ruhe, G., & Saliu, M. O. (2005). The art and science of software release planning. *IEEE Software*, *22*(6), 47-53.

Ruhe, M., Jeffery, R., & Wieczorek, I. (2003a). *Cost estimation for web applications.* Paper presented at the Software Engineering, 2003. Proceedings. 25th International Conference on.

Ruhe, M., Jeffery, R., & Wieczorek, I. (2003b). *Using web objects for estimating software development effort for web applications.* Paper presented at the Software Metrics Symposium, 2003. Proceedings. Ninth International.

Sehra, S. K., Brar, Y. S., Kaur, N., & Sehra, S. S. (2017). Research patterns and trends in software effort estimation. *Information and Software Technology, 91*, 1-21.

Seo, Y. S., & Bae, D. H. (2013). On the value of outlier elimination on software effort estimation research. *Empirical Software Engineering, 18*(4), 659-698. doi:10.1007/s10664-012-9207-y

Shepperd, M. (2007). *Software project economics: A roadmap.* Paper presented at the FoSE 2007: Future of Software Engineering.

Sliger, M., & Broderick, S. (2008). *The software project manager's bridge to agility*: Addison-Wesley Professional.

Symons, C. R. (1991). *Software sizing and estimating: Mk II FPA (function point analysis)*: John Wiley & Sons, Inc.

Szke, A. (2011). Conceptual scheduling model and optimized release scheduling for agile environments. *Information and Software Technology, 53*(6), 574-591. doi:10.1016/j.infsof.2011.01.008

Szoke, Á. (2009) Decision support for iteration scheduling in agile environments. In*: Vol. 32 LNBIP. Lecture Notes in Business Information Processing* (pp. 156-170).

Szoke, Á. (2010) Optimized feature distribution in distributed agile environments. In*: Vol. 6156 LNCS. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 62-76).

Turhan, B., & Mendes, E. (2014). *A comparison of cross-versus single-company effort prediction models for web projects.* Paper presented at the Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014.

Usman, M., Börstler, J., & Petersen, K. (2017). An Effort Estimation Taxonomy for Agile Software Development. *International Journal of Software Engineering and Knowledge Engineering*, *27*(4), 641-674. doi:10.1142/S0218194017500243

Usman, M., Mendes, E., & Börstler, J. (2015). *Effort estimation in Agile software development: A survey on the state of the practice.* Paper presented at the ACM International Conference Proceeding Series.

Usman, M., Mendes, E., Weidt, F., & Britto, R. (2014). *Effort estimation in Agile Software Development: A systematic literature review.* Paper presented at the ACM International Conference Proceeding Series.

Van Valkenhoef, G., Tervonen, T., De Brock, B., & Postmus, D. (2011). Quantitative release planning in extreme programming. *Information and Software Technology, 53*(11), 1227-1235. doi:10.1016/j.infsof.2011.05.007

Verner, J., & Tate, G. (1992). A software size model. *IEEE Transactions on Software Engineering, 18*(4), 265-278. doi:10.1109/32.129216

Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology, 54*(1), 41-59. doi:10.1016/j.infsof.2011.09.002

West, D., Gilpin, M., Grant, T., & Anderson, A. (2011). Water-scrum-fall is the reality of agile for most organizations today. *Forrester Research, 26*.

Wijayasiriwardhane, T., & Lai, R. (2008). *A method for measuring the size of a component-based system specification.* Paper presented at the Proceedings - International Conference on Quality Software.

Wu, D., Li, J., & Liang, Y. (2013). Linear combination of multiple case-based reasoning with optimized weight for software effort estimation. *Journal of Supercomputing, 64*(3), 898-918. doi:10.1007/s11227-010-0525-9

Zhou, Y., Yang, Y., Xu, B., Leung, H., & Zhou, X. (2014). Source code size estimation approaches for object-oriented systems from UML class diagrams: A comparative study. *Information and Software Technology, 56*(2), 220-237. doi:10.1016/j.infsof.2013.09.003