On the Viability of Quantitative Assessment Methods in Software Engineering and Software Services

A Dissertation

Presented to

the Faculty of Daniel Felix Ritchie School of Engineering and Computer Science

University of Denver

in Partial Fulfillment

of the Requirements for the Degree

of Doctor of Philosophy

by

Joseph D. Lucente

June 2015

Advisor: Anneliese Andrews

Author: Joseph D. Lucente
Title: On the Viability of Quantitative Assessment Methods in Software Engineering and Software Services
Advisor: Anneliese Andrews
Degree Date: June 2015

Abstract

IT help desk operations are expensive. Costs associated with IT operations present challenges to profit goals. Help desk managers need a way to plan staffing levels so that labor costs are minimized while problems are resolved efficiently. An incident prediction method is needed for planning staffing levels. The potential value of a solution to this problem is important to an IT service provider since software failures are inevitable and their timing is difficult to predict. In this research, a cost model for help desk operations is developed. The cost model relates predicted incidents to labor costs using real help desk data. Incidents are predicted using software reliability growth models. Cluster analysis is used to group products with similar help desk incident characteristics. Principal Components Analysis is used to determine one product per cluster for the prediction of incidents for all members of the cluster. Incident prediction accuracy is demonstrated using cluster representatives, and is done so successfully for all clusters with accuracy comparable to making predictions for each product in the portfolio. Linear regression is used with cost data for the resolution of incidents to relate incident predictions to help desk labor costs. Following a series of four pilot studies, the cost model is validated by successfully demonstrating cost prediction accuracy for one month prediction intervals over a 22 month period.

Table of Contents

List of Tables . . . vi
List of Figures . . . viii

1 Introduction **1**
1.1 Problem Statement . . . 1
1.2 Goals and Objectives . . . 8
1.3 Rationale for Pilot Studies . . . 9
1.4 Organization . . . 10

2 Background **11**
2.1 Background Research Methodology . . . 11
2.2 Evolution of Empirical Software Engineering Methods . . . 12
2.2.1 Information Technology Process Improvements . . . 13
2.3 Analytical Methods for IT Process Improvements . . . 14
2.3.1 Principal Components Analysis . . . 14
2.3.1.1 Software Project Characteristics . . . 15
2.3.1.2 Code Decay Analysis . . . 16
2.3.1.3 Information Technology . . . 17
2.3.2 Software Reliability Growth Models . . . 18
2.3.2.1 Software Reliability Model Types . . . 20
2.3.2.2 Reliability Model Assumptions . . . 20
2.3.2.3 Evolution of Software Reliability Models . . . 22
2.3.2.4 Reliability Models and IT . . . 25
2.3.3 Cost Models . . . 25
2.3.3.1 Cost of Software Testing . . . 27
2.3.3.2 Help Desk Process Improvements . . . 28
2.3.3.3 Help Desk Cost Models . . . 29
2.3.3.4 Cluster Analysis . . . 29
2.3.3.5 How Cluster Analysis has been Used in Existing Work . . . 30

3 Pilot Studies **32**
3.1 Pilot 1: Application of Principal Components Analysis . . . 32
3.1.1 Summary . . . 32
3.1.2 Purpose . . . 32
3.1.3 Scope . . . 33

3.1.4 Approach . . . 34
3.1.4.1 Overview . . . 34
3.1.4.2 Incident Data . . . 34
3.1.4.3 Principal Components Analysis . . . 35
3.1.4.4 Results and Interpretation . . . 37
3.1.4.5 Process . . . 39
3.1.4.6 Case Study . . . 41
3.1.5 Lessons Learned and Conclusions . . . 52
3.2 Pilot 2: Extension of PCA to Survey Data Analysis[1] . . . 53
3.2.1 Summary . . . 53
3.2.2 Purpose . . . 54
3.2.3 Scope . . . 56
3.2.4 Approach . . . 56
3.2.4.1 PCA Results . . . 59
3.2.4.2 Test 'All' Results . . . 59
3.2.4.3 Test A, A1, A2, A3 Results . . . 61
3.2.4.4 Test B, B1, B2, B3 Results . . . 63
3.2.4.5 Test C, C1, C2, C3 Results . . . 65
3.2.4.6 Summary of common relationships . . . 66
3.2.5 Lessons Learned and Conclusions . . . 68
3.3 Pilot 3: Software Reliability Growth Models . . . 70
3.3.1 Summary . . . 70
3.3.2 Purpose . . . 70
3.3.3 Scope . . . 71
3.3.4 Approach . . . 71
3.3.5 Lessons Learned and Conclusions . . . 81
3.4 Pilot 4: PCA and SRGMs with Extended Product Set . . . 82
3.4.1 Summary . . . 82
3.4.2 Purpose . . . 82
3.4.3 Scope . . . 83
3.4.4 Approach . . . 83
3.4.5 Lessons Learned and Conclusions . . . 87

4 Large Scale Approach **93**
4.1 Summary . . . 93
4.2 Purpose . . . 95
4.3 Scope . . . 95
4.4 Case Study . . . 96
4.4.1 Help Desk Operations . . . 96
4.4.2 Data . . . 96
4.5 Approach . . . 99

[1]This pilot study is included with reference to contributions by Dr. Julia Varnell-Sarjeant in her PhD thesis on the subject of software reuse in embedded vs. non-embedded systems [121]. This survey is also used by Varnell-Sarjeant et al. [122]. Dr. Varnell-Sarjeant designed and conducted the survey.

4.5.1 Cluster Analysis . . . 99
4.5.1.1 Representative Product per Cluster . . . 102
4.5.1.2 Incident Estimates Representative Product per Cluster . . 106
4.5.1.3 Incident Estimation Using Cluster Representatives 110
4.5.1.4 Scalar Method . . . 112
4.5.1.5 Linear Function Method . . . 117
4.5.1.6 Estimate of Incidents for All Products . . . 119
4.5.1.7 Comparison of approaches to estimate future incidents . . 120
4.5.1.8 Cost Data Analysis . . . 122
4.6 Validation . . . 126
4.6.1 Incident Estimation . . . 126
4.6.2 Effort estimation . . . 131
4.7 Lessons Learned and Conclusions . . . 136
4.7.1 Threats to validity . . . 136
4.7.2 Conclusions . . . 137

5 Future Work **140**
5.1 Error Analysis . . . 140
5.2 Product Portfolio Upgrades and Removals . . . 140
5.3 Cost Estimation Model with Refined Effort Data . . . 141
5.4 Generalization to Other Domains . . . 141
5.5 Prediction of Incident Severity . . . 142

Bibliography **142**

Appendix A Help Desk Operations **161**

Appendix B Case Study Data **166**

Appendix C The Survey[2] **171**

Appendix D Published Papers **175**

[2]This survey is included with reference to contributions by Dr. Julia Varnell-Sarjeant in her PhD thesis on the subject of software reuse in embedded vs. non-embedded systems [121]. This survey is also used by Dr. Varnell-Sarjeant et al. [122]. Dr. Varnell-Sarjeant developed the survey and collected the data. This survey is included to assist in understanding Pilot Study 3.2.

List of Tables

3.1 Desirable attribute pair relationships 40
3.2 Product Statistics 41
3.3 Factor loadings (Product A) 43
3.4 Recommendations for Preferred Help Desk Resolution Methodologies Discovered - Product A 43
3.5 Recommendations for Non-preferred Help Desk Resolution Methodologies Discovered - Product A 44
3.6 Observed attribute relationships pairs with desirable attribute pairs overlaid - Product A 47
3.7 Factor loadings (Product B) 48
3.8 Observed attribute relationships pairs with desirable attribute pairs overlaid - Product B 50
3.9 Recommendations for Preferred Help Desk Resolution Methodologies Discovered - Product B 51
3.10 Recommendations for Non-preferred Help Desk Resolution Methodologies Discovered - Product B 51
3.11 PCAs performed 57
3.12 Principal component relationships in survey responses 57
3.13 PCA duplicate relationship summary 68
3.14 Desktop software products 73
3.15 SRGMs investigated 74
3.16 Prediction error at ten week prediction time frame for converging models not rejected at $R = 0.95$ 75
3.17 Predicted total number of incidents for product A488 (R=0.95) 76
3.18 Predicted total number of incidents for product A39 (R=0.95) 77
3.19 Predicted total number of incidents for product A459 (R=0.95) 77
3.20 Results of applying model selection approach using $R = 0.95$ 79
3.21 Results of applying model selection approach using $R = 0.99$ 79
3.22 Incident prediction relative error at five prediction intervals, $R^2 \geq 0.95$ . . 87
3.23 Incident prediction relative error at five prediction intervals, $R^2 \geq 0.99$ . . 88
3.24 Incident prediction relative error summary for five prediction intervals . . . 88
3.25 Median incident prediction accuracy and interquartile range (IQR) 91

4.1 Monthly labor hours for products selected in case study 98

4.2 Cluster product scores based on number of shared and unique principal components within cluster . . . 103
4.3 PCA loadings for scoring example: product p_1 . . . 104
4.4 PCA loadings for scoring example: product p_2 . . . 105
4.5 PCA loadings for scoring example: product p_3 . . . 105
4.6 PCA loadings for scoring example: product p_4 . . . 105
4.7 Product similarity scoring example for cluster c . . . 107
4.8 Cluster representatives - product characteristics . . . 107
4.9 Incident estimations for cluster representatives, $R^2 \geq 0.95$ . . . 109
4.10 Remaining incidents for cluster representatives, $R^2 \geq 0.95$ . . . 109
4.11 Incident estimations for cluster representatives, $R^2 \geq 0.99$ . . . 109
4.12 Remaining incidents for cluster representatives, $R^2 \geq 0.99$ . . . 110
4.13 Actual incidents for predictions by cluster representatives, $R^2 \geq 0.95$ . . . 110
4.14 Relative error for predictions by cluster representatives, $R^2 \geq 0.95$ . . . 111
4.15 Scaled incident prediction relative error for one month prediction using Scalar Method (September 2012) . . . 112
4.16 Incident prediction relative error for one month prediction using Linear Function Method (September 2012) . . . 118
4.17 Incident prediction relative error for one month prediction: Scalar and Linear function methods (September 2012) . . . 119
4.18 Incident prediction relative error for one month prediction: All products method (September 2012) . . . 120
4.19 Cost prediction relative error for one month prediction (September 2012) . 125
4.20 Summary of one month incident estimation relative error at $R^2 \geq 0.95$ . . . 129
4.21 Summary of one month incident estimation relative error at $R^2 \geq 0.99$ . . . 129
4.22 Incident estimation method comparison by one month median prediction accuracy, IQR and outlier range . . . 130
4.23 Comparison of estimation efficiency . . . 130
4.24 Summary of one month labor cost estimation relative error by incident prediction method . . . 132
4.25 One month cumulative cost estimations: Scalar method $R^2 \geq 0.99$ . . . 133
4.26 One month cumulative cost estimations: Linear function method $R^2 \geq 0.99$ 134

A.1 Incident Attributes . . . 165

B.1 Installation and incident data for products selected in case study . . . 168
B.2 Comparison of selected product descriptive statistics to full help desk product portfolio . . . 169

List of Figures

3.1 Attribute Relationships - Product A (part 1) 42
3.2 Attribute Relationships - Product A (part 2) 44
3.3 Attribute Relationships - Product B (part 1) 49
3.4 Attribute Relationships - Product B (part 2) 49
3.5 Flowchart for approach 78
3.6 SRGM prediction behavior as a function of prediction time frame for product A123 using the Gompertz model 79
3.8 Determination of remaining incidents from model plot 86

4.3 Cluster DV representative 114
4.4 Cluster EF representative 114
4.5 Cluster EI representative 114
4.6 Cluster EW representative 114
4.8 One month incident estimation relative error: comparison between three estimation methods: $R^2 \geq 0.95$ 128
4.9 One month incident estimation relative error: comparison between three estimation methods: $R^2 \geq 0.99$ 128
4.10 Help desk one month incident estimation relative error by incident estimation method 131
4.11 Help desk one month labor cost estimation relative error by incident estimation method 132

A.1 Help desk incident resolution labor categories 162

B.1 Product installations and incident quantities: 156 Products 167

1 Introduction

1.1 Problem Statement

Today's modern information systems are built on the foundation of complex software systems. Software engineering plays a fundamental role in the design and maintenance of software systems. The primary goals of software engineering are to deliver high quality products that work reliably and have predictable costs in their design and maintenance. The accurate estimation of costs associated with software development, test, deployment and maintenance in order to achieve reliability goals is important to most organizations. Quality must be assessed at all phases of software development, not just in the testing process and after the product is deployed. Assessing quality throughout all phases leading up to delivery helps meet reliability goals. Essentially, effort expended during development and test phases to ensure the quality of a product in its operational environment is an investment whose return is quantified by the avoidance of maintenance costs. The goal of these investments is to mitigate risks associated with the quality of the product when it is deployed and running in its intended operational environment. Since no software product is perfect, unavoidable effort is necessary to manage anomalies and failures that occur after deployment. Knowing the magnitude of labor costs for troubleshooting and problem resolution activity would be of great value to managers. Not knowing how to determine these costs before an internally developed product is released, or a Commercial Off-the Shelf (COTS) product is deployed, is a problem in organizations who invest in software and rely on it for profit. As such, cost management associated with product maintenance should not stop at deployment time and should be considered when purchasing a product. Processes must be in place to protect the investment of software costs through an understanding of its total cost of ownership over its useful life. Underestimating these costs can result in negative effects on the quality of the software and insufficient resources to fix problems. Additionally, a company who experiences reliability issues with software products in which capital investments are made risks reputation issues and potential loss of competitiveness. Conversely, missed opportunities to fund other business investments can occur by overestimating software maintenance costs [19]. In short, organizations must have techniques to help predict product maintenance costs. This thesis contributes novel ways to help predict maintenance costs of deployed software products.

In order to assess the role of software engineering in terms of product quality, reliability and cost, scientific methods must be used on real-world data gathered through focused observation of deployed systems, or through components designed to gather data through instrumentation. In some cases, system data may be collected for purposes other than the empirical evaluation of some aspect of a system. For example, problem reports submitted to an IT help desk may be managed in a knowledge repository so that common problems can be mapped to known solutions. The same data can be used to discover relationships between the attributes of problem reports to discover process improvement opportunities. On the other hand, information related to some phenomenon of interest may be obtained from human subjects through a survey designed to produce data for use with a selected empirical method, for a specific goal. Empirical methods are applicable for either type of data and are an important aspect of software engineering, without which one cannot account for real-world phenomena.

The usefulness of empirical methods can be viewed in terms of the scope of their applicability to various domains. No single technique for cost estimation or process improvement provides an exhaustive methodology for achieving improvement goals. Some techniques address the performance aspect of systems while others target financial gains in their approach. The wide variety of goals established in process improvement drive an equally broad scope of potential solutions. In the discovery of a sufficiently robust set of techniques and the systems for which success is realized through their application, a framework of methods can be constructed. Design goals in its assembly are met through the delivery of a framework which is adaptable, scalable, cost effective, and one through which alternative solutions can be evaluated for their contribution to process improvements in IT operations. This thesis makes an additional contribution to software engineering by demonstrating the portability of analytical methods used in the discovery of IT process improvement opportunities to the analysis of survey data not constrained to IT. This extension expresses the value of the techniques and their overall generalizability.

Recent advancements in the state of the art of software cost estimation indicate a trend toward integrating fine grain metrics with more coarse grain metrics such as those found in legacy software capability maturity models. In the work of Kauland and Sharma [64], the granularity of an estimation metric is presented in terms of the level at which it uses the attributes of actual code artifacts. For example, fine grain parameters such as code quality, code revision frequency and test coverage of existing code are combined to form a cost index for use in estimations of project costs for similar future efforts. Conversely, coarse grain estimation parameters are derived from a pyramid model whereby higher labor costs are attributed to fewer, highly-compensated managers represented at the apex of the pyramid, and much larger numbers of individual contributors are represented at the base of the pyramid. Increasing levels of compensation, and decreasing numbers of individuals are represented in the middle layer of the pyramid. With this approach, producing labor estimates consists of estimating the number of team members and their financial compensation.

More specific to maintenance in the prediction of software costs, Nguyen [80] proposes an eight step process by which improvements to the COCOMO [15] estimation model is effected through incorporation of parameters related to the size of the code to be maintained, as well as the number of added, deleted and modified lines of code. The last three metrics are aggregated into a single parameter which expresses the amount by which the code is impacted through the three types of code changes. The proposed technique is applied to a selected set of estimation models. The estimation models, now adjusted using the aggregate parameter described above, are evaluated against industry datasets in terms of their maintenance cost prediction performance.

A fault-driven approach to the reduction of effort in projects with partially overlapping development phases is proposed by Henningson et al. [46]. The authors propose a method by which faults found in a software project can be used to identify similar faults in projects with overlapping development schedules. The process is qualitative and has a general business focus. The authors do not include empirical evidence of the success of the approach.

The focus of this thesis is on cost estimation and process improvement techniques applicable to the resolution of software product failures, and, for specific techniques, their generalizability to the analysis of survey data is presented. The proposed methods use actual defect data from an IT help desk. In the analysis of survey data, the results of a set of questions on the subject of software reuse obtained from software practitioners is used. In summary, a literature review of the state of the art of software cost estimation techniques and the identification of process improvements indicates opportunities to contribute research in the area of defect-driven models.

Business processes and profitability are impacted by unplanned interruptions experienced in the end user computing environment. Excess project costs can be attributed to loss of productivity when business-critical software applications fail. Costs can be directly attributed to the length of time required to resolve problems. In a company whose profitability relies on the production and sale of goods, overhead processes such as maintaining the IT infrastructure must serve the production environment with minimal failure. Business must focus on core strategic direction while the IT teams keep software and hardware error-free. Interruptions to productivity such as IT-related failures are therefore disruptive and costly. The dynamic phenomena of a helpdesk environment in an industrial setting result in opportunities to gain practical knowledge from historical information. Specifically, information from the resolution of problematic software can be investigated for primary causal relationships between attributes in help desk problem records, known as incidents in this thesis. Principal Components Analysis (PCA) facilitates the identification of similarly varying attributes in a dataset. From the relationships discovered, recommendations may be formulated for operational optimization and as corrective action to avoid future problems.

Process enhancements are targeted for an investigation of desktop software failures in an industrial setting with over 100,000 employees. The company's core business is comprised of the manufacture and sale of large hardware and software systems, and the provision of systems engineering expertise in technology-based solutions. Over 4000 software applications facilitate productivity among the employees. Most products are commercial off the shelf (COTS).

The company has an internal organization whose responsibility is to manage the IT infrastructure through a centralized help desk. The IT service portfolio provided by the help desk includes break/fix desktop hardware and software support, and desktop software and hardware provisioning. Routine assistance such as password resets and general IT-related questions are managed through the help desk.

The staffing structure at the help desk is comprised of tiers. The first tier deals with initial contact. Staff at this tier provide solutions mainly through script-based assistance. A separate product-focused tier is trained in a variety of domain-specific solutions. The initial tier is shared with a process-focused team. Individuals maintain throughput by escalating incidents for which scripted solutions do not provide a fix. Incidents are laterally transferred to product-focused groups when resolution cannot be attained at the initial tier. Incidents which exceed defined levels of urgency receive the attention of a manager whose role is to expedite resolution. This process is known as incident escalation.

Metrics are used to establish time-based service targets and to alert management for prioritization. These metrics capture real-time occurrences of service target breaches. A fixed number of attributes is associated with each incident record. These include a unique identifier, textual and numeric fields to describe the problem and solution and record information such as the amount of time required to resolve the incident, the number of lateral transfers and whether or not the incident required escalation. Attributes exist to record whether or not the help desk technician finds a match to any one of various historical record types such as other incidents and known problems. The match criteria is subjective; matches may be made on the basis of problem similarity, solution similarity or both. Additional fields which help define service targets contain levels of impact and urgency of the problem.

Incident trends are discovered and investigated through conventional methods such as identifying the top ten incident-prone software products each month and developing corrective actions based on the most frequently occurring resolution categories from mandatory fields in each incident record. Of particular interest is aggregate service target breaches over time, and quantifiable impacts to productivity due to service target breaches. An effective mechanism for analyzing data behavior among sets of incidents does not exist. Management would benefit from knowing incident attribute behavior with respect to preferred resolution outcomes. Principal Components Analysis (PCA) is investigated in this case study to address the analytical gap.

A Information Technology (IT) help desk is a resource designed to provide knowledge and resources to enable individuals to be productive in the organization to which the IT services are delivered. The purpose of a help desk is primarily to resolve problems and provide information about products such as software and computers. In some cases, the individuals served by an organization's help desk are employees of the organization, and in other cases the customers are external. Organizations typically provide IT help desk services to their customers through methods such as websites and toll-free telephone numbers. The adoption of newer technologies for communication with help desks such as mobile applications and instant messaging is becoming popular. All of these communication methods are used in a support model known as *remote assistance*. Some organizations offer services for problem resolution on physical devices. This type of service offering is referred to as *desk side support*. IT help desks in today's environment operate under a formal management framework. Generically, this approach is referred to as IT Service Management (ITSM). Implementation of this management framework requires alignment to service strategy, service design, service operation and the measurement of services to effect process improvement and attain designated levels of maturity. ITSM is often implemented under the Information Technology Infrastructure Library (ITIL). The UK Office of Government Commerce publishes standards in the ITIL library that provide guidance in the implementation of ITIL-based ITSM [22]. A motivating force behind help desk process formalization is an increasing focus on integrating an organization's business processes such as customer engagement and cost management with its IT support model [26]. Some contractual obligations between business partners specify operation under some form of ITSM while others specify the adoption of ITIL standards for consummate partnership.

First Research Industry Profiles presents a quarterly update of business trends in a wide variety of industries. In the May 2014 issue under the categories of *call centers* and *service industries*, centers around the world account for $150 billion USD in annual revenue. Additionally, the call center industry is projected to realize more than $300 billion in annual revenue by 2018 through industry expansion. The US telephone call center industry is comprised of 4,200 companies with a total annual revenue of $16 billion. The growth of this industry could expand current annual revenue to more than $300 billion by 2018 [117].

The IT help desk services investigated in this thesis are a assembled through ITIL standards. The goal of the help desk is to minimize the total cost of the IT services while maximizing business value. The help desk services approximately 100,000 employees in a large, multinational company whose core products and services are delivered to aerospace customers. Cost management is executed within the ITIL framework. Budgets for capital expenditures such as hardware and software, as well as operational expenditures for labor, are established on a yearly basis and reviewed by management on a monthly basis. Financial performance to budgeted expenditures is scrutinized for cost-savings opportunities. The approaches to labor cost estimation comes with challenges of not knowing what products will require break/fix assistance, and the number of help desk technicians required to resolve software anomalies, hardware failures and other service outages. New processes and new or upgraded software usually result in an increase in help desk incident volume. These challenges are typical of help desk operations overall [3]. Current methods for estimating operational labor costs are limited to an analysis of historical failure trends, knowledge of planned software product releases by vendors, and projections of growth or down-sizing of the company's labor force. The company is committed to implementing standard processes for efficiency improvements and seeks novel ways to achieve them. In this thesis we investigate PCA and cost models as innovative techniques to help achieve the company's goals of increasing help desk efficiency.

1.2 Goals and Objectives

Goals established for research in this thesis are formalized in the following research questions:

RQ1 Can PCA be used to assess help desk operations and suggest process improvements?

RQ2 Can SRGM be used to predict future incidents so as to facilitate staffing predictions?

RQ3 Can we develop a cost model for incidents?

The applications of PCA, SRGM and cost models address problem resolution of desktop software failures. While the help desk provides a variety of services, we focus on resolution of anomalies and failures of desktop software. Problems submitted to the help desk for services such as password resets, requests for new hardware and questions about how to work with software applications are excluded from this research.

Finally, we are interested to what degree our PCA approach can be generalized to other domains. We formulate the following additional research question:

RQ4 Can our approach to PCA be generalized to determine factors that influence success in reuse projects?

1.3 Rationale for Pilot Studies

Four pilot studies are conducted as part of this thesis. At the conclusion of the pilot studies, a large scale study is conducted. Pilot studies 1 through 4 established a foundation for several analytical approaches, the development of incident prediction techniques, and validation of a help desk cost estimation model. In Pilot study 1, PCA was successfully applied to incident attribute values from help desk incidents produced by a small set of products selected from a much larger portfolio of products, to determine the statistical relationships between attributes. Attribute pairs with stronger covariance were selected to reduce the overall number of attributes to be considered for developing recommendations for operational improvements at the help desk. Pilot study 1 was scoped to an IT help desk environment. Pilot study 2 extended the approach used in the first pilot, to survey data related to the reuse of artifacts in embedded and non-embedded software systems. The success with which PCA was used in Pilot study 2 demonstrated the technique used to analyze help desk incidents can also be used to relate survey question responses. Pilot study 2 is conducted in an industrial setting that is quite different from the help desk environment in the first pilot. Pilot study 3 develops a product reliability estimation technique using SRGMs to predict incidents. This pilot study is conducted in the same industrial setting as with the pilot study 1. Although the approach is successful based on incident prediction accuracy, the study is limited to a much smaller number of products than what is used in the industrial setting. Pilot study 4 addresses this gap by applying incident estimation techniques to a larger portfolio of 156 products vs. the 18 products used in pilots 1 and 3.

1.4 Organization

This thesis is organized as follows: Chapter 2 introduces the selected background research methodology, includes a discussion of how empirical software engineering methods have evolved, and concludes with a documentation of existing work related to our research goals and in the context of PCA, SRGMs and cost models. The approach to applying our techniques is described in Chapter 3, including results. The material in Chapter 3 is presented as four pilot studies. Chapter 4 brings together the concepts and techniques covered in the four pilot studies described in Chapter 3 through presentation of a large scale approach. Chapter 5 concludes with a discussion of future work inspired by the research in this thesis.

2 Background

2.1 Background Research Methodology

A systematic and auditable methodology is selected in our review of related work in this thesis [68]. Specifically, our goals are

- To learn the range of empirical methods used in the study of IT process improvements
- To determine any gaps in current research related to IT process improvements
- To establish a background in order to appropriately position the methods proposed herein, with respect to existing, related work in IT process improvements and their generalizability to other domains.

This literature review seeks relevant work in fulfillment of the objectives stated above. The review protocol consists of searches within IEEE Xplore, ACM Digital Library, and specific journals and conference proceedings. Concepts from published work already integrated into the development of methods and their preliminary results discussed within this thesis are included in this background section, as are related works cited within these publications. The protocol is based mainly on key word searches and author searches. Search results are reviewed at the title and abstract level, and accepted as candidates for consideration if the theme or subject embodies a topic in this thesis and fulfills the background research goals. The initial set of papers is read thoroughly and accepted or rejected based again on thesis topic applicability and potential for meeting the background research goals.

2.2 Evolution of Empirical Software Engineering Methods

Studies conducted on the application of empirical software engineering methods indicate three types of change over the past three decades: 1) The quantity of published papers using empirical software engineering methods has increased; 2) The quality of data in published work has improved; 3) The scope of software system types on which empirical methods have been applied has broadened. Zannier et al. [138] conducted a quantitative study of the success of empirical methods in software engineering over the period of 1975 to 2005. Statistically significant results from a study of randomly selected samples of papers accepted through the International Conference on Software Engineering (ICSE) over a three decade period demonstrate the quantity of papers increased over the period of study. These results confirm growth in interest in empirical software engineering methods. The authors also included a statistical analysis of the quality of papers from the same population sample, based on a clearly articulated definition of research quality used in several previous surveys of empirical software engineering research. In this component of their investigation, statistical evidence did not support a hypothesis of growth in the quality of empirical software engineering research. In 2013, however, Bosu and MacDonell studied the quality of data in empirical software engineering research and observed improvements in data quality [17]. The scope of software system types investigated using empirical methods was noted to have increased [138], but modern technologies such as cloud computing and embedded systems are deserving of deeper empirical research [125]. Much like the vision presented by Sjoberg, Dyba and Jorgensen in 2007, in the many sub-disciplines of software engineering, empirical research methods should facilitate the development of scientific knowledge about the applicability of various SE technologies to different individuals engaged in different activities and systems. As increasingly sophisticated methods are developed, they should be applied in novel ways to advance our understanding of software processes [100]. The research goals in this thesis aim to benefit the software engineering and IT help desk communities through the use of such methods in the interest of increasing the quality of software processes.

2.2.1 Information Technology Process Improvements

An investigation into published work related to process improvements in information technology (IT) is essential in establishing a research direction for this thesis. We begin with an investigation of work that is related to IT process improvement in general. Published work related to process improvements is investigated in other areas with a strategy to identify techniques and approaches that may be useful in IT Help Desk operations. We then concentrate on content that applies directly to IT disciplines and centralizes application of Principal Components Analysis (PCA) and Software Reliability Growth Models (SRGMs), to data obtained from sources in various IT domains.

An alignment of IT services to a business model is the focus of a study by Bose, et al. [16]. Continuous process improvement based on a set of consistent metrics and an analytical engine for processing anomalous events and process shifts are the key components of a model-based framework for continuous process improvement in the delivery of IT services. The framework is implemented in an industrial setting. While our case study is also in an industrial setting, our approach differs from that of Bose, et al. in the phase in which our techniques are applied. We derive a cost model from help desk data and predict labor. In contrast, Bose, et al. effect process improvement in real-time events.

Three types of outbound call scheduling algorithms are introduced by Gulati and Malcom [43] in the interest of increasing unpaid balance collection productivity in large call centers. A simulation-based methodology in which the model mimics the outbound call process by accessing lists returned by each of the scheduling algorithms provides a way to assess the system performance against established goals. Three call scheduling algorithms are introduced. The main difference between each algorithm is the manner in which call lists are managed throughout the day. In the first algorithm, outbound calls are sequenced according to the amount of the customer's outstanding balance due. Customers with higher outstanding balances are called first. If an attempt to contact a customer is unsuccessful, the outcome is noted and the call is scheduled for retry after all scheduled calls are attempted. In the second algorithm, calls are sequenced according balance due, but a re-sequence is performed at the beginning of each day. The third algorithm differed from the second in that the rescheduling occurs hourly. The second and third algorithms outperformed the first one in terms of successfully contacting the right person to discuss collection arrangements.

2.3 Analytical Methods for IT Process Improvements

Three analytical methods are included in this thesis proposal. In the sections below each method is discussed in terms of related work and its role in software engineering. Where applicable, related work specific to IT help desk operations is included with a focus on achieving improvements through use of these methods.

1. Principal Components Analysis (PCA)
2. Software Reliability Growth Models (SRGM)
3. Cost Models

2.3.1 Principal Components Analysis

Principal Components Analysis (PCA) was invented by Karl Pearson in 1901 [87]. It can be used to reduce the number of variables in a data set based on how much each variable contributes to the overall variance in the data. It has been used heavily in a variety of applications over the past four decades. In each of these examples, goals for variable reduction in systems with a range of complexity are met. Early examples include the reduction of search space in text-base searching [70], variable reduction in multi-variate datasets [33], data compression in image processing [12] and speech analysis [40]. PCA gained interest in the artificial intelligence community in the 1990's. Examples include the integration of artificial neural networks for variable reduction [52], applications to neural networks for process fault diagnosis [101] and the application of PCA to multi-layer perceptron learning algorithms to improve convergence speed with the back propagation algorithm [106].

2.3.1.1 Software Project Characteristics

PCA has been adopted as a popular method in software engineering to achieve reduction in large variable sets and to group variables into similarly-behaving subsets [83,84,111, 112]. The predictors of software development project success factors based on project similarities in [127] use PCA to group projects with similarly varying characteristics. Project attributes and success indicators are evaluated in a database of projects where PCA is successfully used in the identification of key characteristics to determine project outcome.

The outcome of projects with shared characteristics is investigated through the prioritization of success factors as presented by Wohlin and Andrews [21]. In their work, PCA is used primarily to reduce a potentially large set of variables to a smaller set whose behavior is consistent with an overall set of success variables determined for a database of software projects. In keeping with an objective to find project characteristics that influence success, PCA is used to determine factors which vary together. By grouping co-varying characteristics with overall success variables, the objective is met. We build on this success by using a similar approach with PCA. In this case study the variables of our PCA are incident attributes. We group similarly behaving subsets of incident attributes for a product group as a means of identifying the product which has the greatest similarity to the other products in the group. We designate this product as the representative of the group and use it to address incident prediction using product clusters rather than the full product portfolio. Through this approach, the overall level of effort associated with this approach is reduced.

A method for selecting a software architecture based on attributes of product quality is presented as an empirical study by Svahnberg and Wohlin [108]. The method quantifies the perceived level of support provided by different software architectures such as layered and Model-View-Controller, on the basis of quality attributes such as functionality and reliability. A set of questions was asked of a selected group of participants to understand their perception of support level by software architecture type. Among several techniques used to analyze the survey responses, PCA was used to group how participants answered the survey questions. The authors note how PCA is sensitive to small numerical differences in data, and confirmed such sensitivity in their results. Although PCA successfully produced groupings of responders in terms of their perceived levels of architectural support, the authors concluded with a recommendation to use the squared distance to the mean value for each data point, as opposed to PCA, to identify groups of responders. This research contributes a method for making a managerial decision among a set of architecture candidates for best fit to a project. Additionally, it illustrates the sensitivity of PCA in data with small differences. Svahnberg and Wohlin contribute a method of making a managerial decision among a set of architecture candidates for best fit to a software project. This establishes validity in the selection of PCA in our case study in the designation of a product which best fits the behavior among the set of products in a cluster. Our case study differs in the object of selection. Svahnberg and Wohlin select a software project where we select a software product.

2.3.1.2 Code Decay Analysis

Wohlin et al. [83] use PCA in an investigation of software code decay across four successive releases of a large (800,000 LOC) industrial software product consisting of 130 components. Using a red/yellow/green coding scale, software components were assigned one of three classifications in terms of their fault proneness. The characteristics of each of the three categories of fault-proneness were of interest in this research. Twenty eight attributes of the software components were investigated. Examples of attributes are lines of code and average number of fixes. The authors were interested in knowing which of the 28 attributes in the database of software components and their attribute values varied the most. PCA was used to determine the rank order of attributes in terms of variance, such that a selected threshold of total variance allowed the authors to select only the attributes above the threshold to be considered in their subsequent analysis. By grouping the attributes according to similarities in their variance, conclusions could be drawn regarding the causes for which certain components are healthy and which are problematic [83].

2.3.1.3 Information Technology

In many applications of PCA, a reduction of the number variables is a preprocessing step to facilitate computational tractability. This is the case with Liu and Liu [73] in the development of a back propagation neural network designed to forecast call center traffic. PCA is used to identify m consecutive half-hour periods over which call volume varies in n groups. A reduction from 48 time periods to nine time periods was realized through PCA. These become inputs to a back propagation neural network used to forecast call volume. The authors' technique is evaluated against an existing call volume forecasting tool and was found to outperform it.

Another example of PCA applied specifically to the IT domain is found in the work by Tannahill, et al. [116]. The authors introduce PCA as a statistical tool in the reduction of variables in big data analysis. Their work is among the first to associate PCA with data analytics through recognition of the growing volume of data available in modern information systems. Their application of PCA to Big Data analytics permits large-scale reduction in variables through elimination of a large number of variables which contribute little to the variance of the data. Although our research is not considered big data analysis, the work of Tannahill, et al. is important to potential scalability of our solution. As mentioned in our introduction, we focus on one of three categories of incidents related to a software products. Should the cost model proposed in this case study be extended to incidents in all three categories, the scale of the problem could easily approach that of big data.

PCA with a focus on desktop computer security is addressed in research conducted by Kim et al [67]. Security anomalies such as attempts to access information by unauthorized agents are detected using a Back-propagation Neural Network (BNN). Similar to an approach taken by Liu and Liu [73], PCA is combined with a BNN to enhance computer intrusion detection efficiency. Working with an intrusion detection database consisting of 41 variables, the authors use Genetic Algorithms (GA) to obtain the optimal set of features for PCA as well as the best parameters values for topology of the BNN. Unlike Liu and Liu, however, their use of PCA is not limited to a simple reduction in variables.

PCA lends itself to Statistical Process Control (SPC) due to the large number of variables involved in many industrial processes. For example, temperature, fuel flow rate, fuel pressure, and a number of other environmental factors must be controlled during preparations for a commercial airplane to land safely. Timely judgments must be made based on the interaction of these variables. A primary goal in SPC is to detect conditions which approach established and sometimes adaptive limits. Zhou and Gou [142] work within a system consisting of many processes and build a process monitoring model which reduces a multi-dimensional system of variables to a low-dimensional set. The reduced variable set is obtained using PCA.

2.3.2 Software Reliability Growth Models

We turn our attention to Software Reliability Growth Models (SRGMs) as the second of three analytical methods used in this research. Reliability models have been used successfully to predict problems with software applications. The dynamic selection of Software Reliability Growth Models (SRGMs) has been demonstrated by assessing model performance in the selection of a preferred model [105] [7]. SRGMs include a parameter that estimates the total number of defects in a software product. Knowing the value of this parameter and the number of defects already reported allows us to estimate the number of remaining defects for a software application. Knowledge of the number of residual defects assists in quantifying information related to product quality. From the research conducted by Stringfellow, Andrews and Andersson [7, 105], we have confidence that SRGMs can predict the number of residual defects and assist in quantifying information related to product quality during software testing. This establishes a pivotal element of this research and exposes a fundamental difference between SRGMs applied to software testing vs. products in their operational life cycle. We formalize this by investigating if desktop software product reliability data obtained from help desk incidents be used to predict future incident volume? We answer this question by demonstrating accurate predictions as part of our cost model.Similar to products delivered through software development, the quality and reliability of desktop software applications can be assessed through the number of problems reported as incidents to an IT help desk.

The typical life cycle of a software product under development includes requirements, interface specifications, design reviews and code inspections. Software development organizations usually have a team whose mission is to evaluate the quality of a product, usually with respect to its requirements. The development team produces code and executes unit tests. When unit testing is judged to be successful, the quality team executes system integration testing. Defects detected during the unit and system testing are recorded in a defect database. A history of product defects can then be derived for application to reliability models.

In this thesis we investigate incidents submitted to an IT help desk for products already deployed to computers in an industrial environment. Unlike the defect reporting process used with internally-developed software, our approach focuses on products actively used in their intended operational context. Our goal in this case study is to apply the SRGMs to a model selection process. We select a model based on its prediction performance to estimate the total number of incidents in software applications. Knowing how many incidents to expect assists in quantifying information about product quality, loss of employee productivity, and projected levels of help desk staffing to prepare for problem resolution.

2.3.2.1 Software Reliability Model Types

A software reliability model is a mathematical representation of a random process from which software quality, or some quantity that represents software quality, is expressed as a function of time [53]. Two types of models have been used to assess the quality of software in terms of its reliability: static and dynamic models. Both models types have been used to estimate the number of faults (defects) in software artifacts. Static models are based on software metrics to measure product quality. Examples of software metrics are lines of code, number of function calls and complexity metrics. Dynamic models make use of historical failure data such as defect data collected during a test phase or over a specified time period of a deployed product running in its normal operational profile. Dynamic models applied to software in its test phase can be used to make release decisions. The same type of models applied to a post-release phase can be used to estimate the number of remaining defects for purposes of planning support staff or projecting operational maintenance costs [62]. Dynamic models are usually time-based. For example, the time between failures is commonly used in the dynamic reliability models discussed by Musa et al. [79]. The time component of dynamic models is usually based on a set of times t_i for $i > 1$, at some fixed interval such as each day or each week. Organizational practices will usually dictate the reporting interval based, for example, on the interval of meetings at which defects are discussed or the time interval over which defects are collected through a defect management system.

2.3.2.2 Reliability Model Assumptions

SRGMs are dynamic models that make use of historical failure data such as defect data collected during a test phase or over a specified time period of a deployed product running in its normal operational profile. Dynamic models applied to software in its test phase can be used to make release decisions. The same type of models applied to a post-release phase can be used to estimate the number of remaining defects for purposes of planning support staff or projecting operational maintenance costs. Our case study goes beyond the approach taken by Kan [62] by developing reliability models from defects generated by products in their operational life cycle. Kan predicts post-release operational defects using defect data from product testing. Dynamic models are usually time-based. For example, the time between failures is commonly used in the dynamic reliability models discussed by Musa et al. [79]. The time component of dynamic models is usually based on a set of times t_i for $i > 1$, at some fixed interval such as each day or each week. Organizational practices will usually dictate the reporting interval based, for example, on the interval of meetings at which defects are discussed or the time interval over which defects are collected through a defect management system. Our case study follows the Musa model in this regard by establishing time intervals over which incident data are collected and calculating cumulative incidents at one week intervals. Date and time stamps on incident records facilitated data collection in this regard.

Time can be measured as execution time or calendar time. An assumption made with dynamic software reliability models is that the product for which reliability is being estimated operates in its operational profile. This means that in pre-release testing, the test cases are designed according to the probability of their occurrence during the intended operational profile. Similarly, models used for the assessment of quality during post-release evaluation assume the software is operating under its intended profile. In each of these cases, a number of factors influence the quantity and type of failures that can occur such as the execution environment [37]. Reliability model assumptions are addressed in our case study by investigating 1) if help desk incidents can be used to build reliability models, 2) if calendar time can be used in the reliability models. Since the models we use are constructed from incidents from products in their operational life cycle, we meet the assumption of software operating under its intended profile.

2.3.2.3 Evolution of Software Reliability Models

Over three decades of research beginning with formative works by Iannino, Musa, Okumoto and Littlewood [53] in 1983 have resulted in a body of knowledge of software reliability growth models [133] [135] [134] [79] [137] [78] [119] [34] [136] [129] [130]. These works concentrated on foundational topics such as the application of SRGMs to software reliability theory [79] and selection criteria for reliability models [53]. Later works concentrate on specific attributes of SRGMs. Gokhale and Trivedi propose a log-logistic model applicable to increasing or decreasing failure rates to address the assumptions of constant, monotonic increasing or monotonic decreasing failure occurrence rates in finite-failure, non-homogeneous Poisson process (NHPP) models [39]. A summary of research in SRGMS through the late 1990's is presented in the works of Everett et al. [35], where real-world implementation of software reliability management is discussed in the context of software reliability management practices, tools designed for reliability management and the applicability of reliability models to safety-critical systems [35]. Similar to the approach taken by Gokhale and Trivedi [39], assertions that failure rates that fit a lognormal distribution [77] motivate the development of a model based on lognormal execution time [77] with supportive empirical evidence of its success with two sets of failure data. Pant and Jeske explore software reliability predictions applicable to distributed software [86]. Assumptions of independence among successive software failures by SRGMS studied through the late 1990s are addressed by Goseva-Popstojanova and Trivedi through the introduction of a software reliability modeling framework based on Markov renewal processes [41]. In early 2000 an adjustment to the Yamada S-shaped model [133] was proposed through the integration of a testing effort function to produce the Delayed S-shape model [48]. Conclusive empirical evidence of performance improvements to the existing Goel-Okumoto model is demonstrated by Keiller and Mazzuchi in 2000 by introducing a technique based on failure windows [66]. Further refinement of existing SRGMS is proposed in the work of Kuo, et al. where consideration for both testing effort and fault detection rates produces a novel framework for the construction of SRGMs [69]. Further improvements in SRGMs are proposed by Donovan and Murphy in 2001 [32]. An approach to the estimation of software reliability through grouping of failure data into clusters of homogeneous failure intensities is adopted by Tian in 2002 [118]. Huang et al [50] and Peng et all [88] account for real-world violation of assumptions of immediate correction of defects and mutual independence between defects by proposing new SRGMs and their success in defect prediction capability. These same two assumptions and the realistic likelihood of their violation is also addressed by Jalote and Murphy in 2004 through the introduction of a new SRGM [59]. In related work, Shu et al propose a new SRGM which accounts for the lag between defect correction and detection, thereby addressing the reality of assumption violations associated with immediate defect repair [95, 96]. Differences in failure removal phenomena between software under test and software operating in a production environment motivate Huang, et al to propose a new SRGM which accounts for these differences [51]. The differences between Software test environments and production environments is also investigated by Jeske et al through the use of test data. A calibration technique for adjusting the failure rate estimate obtained from analyzing test data offers a practical way to extend SRGMs to account for operational differences in the execution of post-release software. Operational difference between software test and production environments are also investigated by Zhao et al [140] and Li et al [72]. The observed differences motivate the authors to assess the differences in fault prediction capabilities between two SRGMs and recommend a preferred model for use with failure data obtained from a production environment. The impacts of open source software projects on defect prediction capability of SRGMs is the focus of research by Tamura and Yamada [113, 114], Ullah and Morisio [120], Singh and Maurya [98], and Syed-Mohamad and McBride [109, 110]. The Analytical Hierarchy Process (AHP) is proposed as a software reliability assessment method for the development of software for distributed system. Reliability-driven software release decisions are assisted by SRGMS in the approach proposed by Bhawnani et al in 2005 [14] and Stringfellow et al. [105]. As with most applications of SRGMs in the literature, their approach focuses on software testing unlike the perspective of help desk incidents in our case study. Huang et all [49] address challenges of non-constant test effort in exponential and s-shaped SRGMs from which the reliability, residual number of defects, failure rate, and optimal release time are determined. A logistic testing-effort function is incorporated into both s-shaped and exponential models in a demonstration of their effectiveness in defect estimation. [24, 25]. A sharp increase in the production of consumer electronics and concern for sound methods for estimating embedded software reliability motivate Almering et al to conduct an empirical investigation into the applicability of SRGMs to this growing class of devices. The authors confirmed the models studied can assist managers in decisions related to the market readiness of consumer devices [5].

SRGM estimation parameters are derived using a genetic algorithm (GA) in research conducted by Aljahdali El-Telbany [4]. Application of the models with GA-driven parameters were compared to those with model parameters derived through non-linear regression methods. The GA-driven models outperformed the non-linear regression models in terms of defect prediction performance. In another extension of advanced techniques to SRGMs in the interest of improving fault prediction performance, Li et al. [71] combine SRGMs in a selective neural network model. The average performance accuracy in terms of fault prediction is improved compared to the individual models from which the network of SRGMs is comprised. Hsu and Huang investigate combinations of SRGMs using weighted models to build the combination [47].

Change points, defined as the times or time intervals at which defect occurrences in the test process change notably, are modeled in the proposal of an SRGM framework to address model assumptions which are violated in the presence of change points. Owing to the effects of change points on the accuracy of reliability model fault prediction, Inoue and Yamada demonstrate a two dimensional model, with testing time and testing effort as the two dimensions, on actual software defect data [55, 141]. Rana et al. [91] provide an insightful comparison between the two common SRGM parameter estimation techniques - Maximum Likelihood Estimator (MLE) and Method of Least Squares (MLS). The authors investigate both techniques in the context of estimation of model parameters and remaining defect prediction accuracy using relative error as a metric. Their evaluation of both techniques on data from literature suggests MLS is a good estimator for curve-fitting data to observed failure data while MLE outperforms MLS as an estimator for reliable predictions. Finally, the applicability of SRGMs to mobile applications is investigated. Meskini et al apply several commonly used SRGM to smart phone applications and report their failure to curve-fit defect data for all models. The authors attribute the failures to differences between desktop and smart phone applications [76].

2.3.2.4 Reliability Models and IT

Research conducted by Condon and Cukier [24] related to the applicability of SRGMs to security incidents is perhaps the most closely related work to the topic of this thesis, respective of the limited scope of incident types in the security domain. Their work considers security events as incidents, whereas in this thesis an incident is an informational record reported to an IT help desk. This difference distinguishes our case study from research conducted by Condon and Cukier.

2.3.3 Cost Models

The identification of principal, influential factors coupled with cost modeling techniques has obvious benefits for software engineering processes. In this paper we apply similar variable reduction techniques to IT help desk processes through the investigation of historical resolution data. We discover relationships between data attributes in the interest of proposing cost-saving corrective action to mitigate undesirable resolution outcomes. Additionally, we observe desirable incident resolution behavior as confirmation of successful help desk operations.

Dinh and Leonard articulate the motivation for establishing a strong foundation for designing and maintaining service systems. The service sector dominates many global economies and demands the creation of value. Using a call center as an example of a Service Value Creation Network, the three main economic entities are defined: 1) the service client, such as a retail business, 2) the service provider such as a help desk or call center, and 3) the service target which is the customer. A conceptual framework is developed under the key concepts of the role of people and shared information technology. The authors contribute to service science by formalizing key components in creating economic value from a service provider. [28].

Minimizing software failures in an operational environment is important to control cost impacts and to maintain organizational productivity. User-facing behavior such as unscheduled downtime, slow performance and anomalous behavior in software systems affects the perception of a product in terms of potential maintenance costs to an organization and impacts to employee productivity. Critical business applications and systems with human life at stake demand reliability and continuous availability of services. Software down time causes revenue loss [103]. While no marketable product is entirely defect-free, software quality can benefit from reliability models which assist with the estimation of remaining defects in future phases of the product life cycle. Similar benefits can be realized through the prediction of problems associated with desktop software applications installed on computers in an IT environment. Where a large number of desktop products is in use in an organization, an IT help desk can expect a significant workload of problem tickets associated with software failures and anomalous operational behavior. The effort required to resolve these problems has costs associated with help desk technician labor and impacts to employee productivity. Trends in problems and failures with certain software applications drive costs to maintain the products. When employees experience failures with applications on their computers, they expend unplanned effort during their engagement with the help desk to report problems. Employees often work real-time with a technician as the problem is investigated and resolved. For complex problems, additional costs are incurred through the escalation of issues to help desk subject matter experts. These scenarios usually require help desk management intervention to ensure the incident is routed to the appropriate team for investigation. In some cases employees must find work-arounds to remain productive while problems are being investigated, or even tolerate complete loss of productivity if no alternate ways to remain productive exist. The latter is commonly associated with failures experienced while using expensive, domain-specific applications for which alternate products are unlikely to exist. In the worst scenario contractual deadlines for deliverables can be missed due to application failures, resulting in financial penalties which impact revenue and the perception of operational quality of an organization.

Knowledge of future help desk incident volume can be useful to plan for resources who can assist with the resolution of problems. Methods to predict help desk workloads can assist managers with staffing strategies such as short-term adjustments to the number of help desk technicians to mitigate incident backlog, documentation of quick-access solution models for efficient problem resolution, and pre-notification to subject matter experts for solution planning. In organizations with a self-service problem resolution model, known problems can be documented with easily accessible resolution instructions available to employees without direct interaction with the help desk.

2.3.3.1 Cost of Software Testing

Software processes with an economic focus are of interest in the management of development and test activity. Cost benefits in a test environment are investigated in [29] where test case prioritization techniques for regression testing are empirically evaluated. A focus on cost benefit trade-off is investigated in the modeling of component costs in regression testing in [30, 31]. The accuracy of a model for predicting selective regression testing cost-effectiveness in [45] is evaluated empirically. The economics of regression testing specific to cost-benefit trade-offs in regression testing are empirically evaluated in [74]. The authors demonstrate techniques that may be of concern to managers in industrial project cost estimation. Do, Rothermel and Malishevsky focus on economic aspects of pre-release phases of software development. While this research shares similar goals related to software costs, our focus is on products in their operational phase. This is an important distinction in that our cost model applies to products whose operational lifespan is usually unknown at the time of release. Should the lifespan of a product go beyond an incident prediction period, our model can be reapplied using more recent incident data for further cost projections.

2.3.3.2 Help Desk Process Improvements

Help desk process improvements are the focus of research conducted by Shu and Guang [97] where quality improvement is modeled using data mining techniques. Increases in processing throughput are demonstrated using real-time service metrics such as average speed of answer. The authors address the problem of help desk efficiency in terms of service quality and measure efficiency using real-time metrics. Our research addresses efficiency improvements differently, through non real-time techniques. Our approach does not rely on synchronous events, and in fact uses historical events to our advantage. Novel approximations methods for call arrival processes in service centers are proposed by Steckley [102] and confirmed through a computational evaluation using call center data. Similarly, Ibrahim [54] exposes the challenges of building realistic statistical tools for analyzing call arrival processes. In this case study our strategy for help desk process improvements models costs, unlike the approach based on call center arrival used by [54]. Further research in call center operations is presented through the introduction of a novel call type classification technology by Tang [115]. By using actual transcribed conversations from an IT help desk at the University of Colorado in Boulder, CO, Tang applies four text-based call-type classification algorithms to determine the error compared to manually classifying the calls using the call-type taxonomy. An algorithm based on Support Vector Machine (SVM) outperforms other approaches. Tang demonstrates help desk improvements through real-time processes, where our case study 1) relies on historical data, and 2) delivers a managerial solution rather than solving an operational problem. Help desk automation through the development of problem resolution expertise is discussed in [75] in the context of high staff turnover and resulting loss of experience at call centers. Case-Based Reasoning (CBR) is proposed as a departure from rule-based methodologies. The application of CBR to historical help desk data is shown to evolve knowledge in problem resolution. An approach to identifying solutions by matching symptoms to an archival database of problems and their solutions is proposed as a method for the automatic diagnosis of software problems reported by users. While McCarthy clearly focuses on help desk process improvements, specifically with levels of experience, our research addresses efficiency in the managerial aspect of help desk management rather than in live help desk operational scenarios.

2.3.3.3 Help Desk Cost Models

Several cost models specific to call centers and help desks exist in the literature. Hampshire and Massey first propose a stochastic model based on queuing theory to address call center call arrival and servicing, and then extend the model with an economic component to develop an approximation algorithm for a profitable, optimally scheduled staff [44]. Fanaeepour et al. survey existing call center structures and propose a service model based on Generalized Stochastic Petri Net (GSPN). The model is evaluated with a goal to minimize the workload of servers to accommodate a self-service element [36].

2.3.3.4 Cluster Analysis

Cluster analysis is a generic name associated with a set of methods which classify multivariate data on a selected basis for categorization. The resulting clusters of data can assist in identifying a set of characteristics of the patterns of the clusters that are present. The overall goal in clustering is to divide components into groups so that 1) members in a group are as similar to each other as possible (*intra-group dissimilarity*), and 2) the groups are as dissimilar as possible (*intergroup similarity*). In terms of statistical variance, *intragroup variance* should be small, and *intergroup variance* should be large [58]. Two main clustering techniques, *hierarchical* and *non-hierarchical*, are found in the literature. A commonly used hierarchical approach referred to as the Minimum Spanning Tree Method (MSTM), starts with n clusters for a dataset with n elements, and merges data points to achieve a targeted number of clusters. MSTM is used widely in practice. Some examples can be found in the works of Sun, et al [107] and Wu, et al [131] with web images. The iterative approach characteristic of MSTM is also used in the work of [123], where specific concepts addressed by MSTM are formalized. As with the MSTM method described by Jain [58], a distance metric is selected in the approach by Sun, et al [107]. Specifically, Sun formalizes definitions for *well separated clusters*, *connected clusters* and *relaxed well separated clusters*. Additional work with a cluster analysis using a hierarchical approach is found in the work by Grygorash, Zhou and Jorgensen [42]. The authors demonstrate the performance of two different variants of the minimum spanning tree method described by Jain [58] through color separation in image processing.

2.3.3.5 How Cluster Analysis has been Used in Existing Work

A survey of the fundamental concepts and techniques associated with cluster analysis by Jain, et al [56] discusses the motivation for cluster analysis and the large scope of disciplines in which data practitioners use it. Specifically, cluster analysis techniques have been used to characterize workloads in performance analysis [58] [90], in information technology with hypertext-based solutions [18] [139], and, more recently, in big data analysis in cloud computing environments [132]. Specific to software engineering, Singh et al [99] investigate software component coupling, the level of dependence between software components, by seeking the distribution pattern between 145 variables in a software artifact in terms of six attributes. Additional examples of the use of cluster analysis in software engineering include an investigation into design improvements with component interfaces [2], and the evaluation of software engineering methodologies [23]. Data mining techniques include cluster analysis, as observed in the works of Busse [20] and De L Torre [27]. Through a cluster analysis of operational execution profiles, Podgurski et al propose a novel approach to the estimation of software reliability [89]. Numerous examples of cluster analysis can be found in telecommunications [6] [94] [13] [60].

For research related to large numbers of desktop software products with widely varying distributions in a company, and large variations in their associated help desk incidents, cluster analysis is helpful in classifying products with similar distributions and incident volume [58]. Having identified clusters, small samples from each cluster can be used to represent the population of the clusters to study, for example, the principal components responsible for co-variations between attributes of help desk incidents. We use the MSTM method in this case study to establish a hierarchy of clusters from which a set of eleven is selected. One product from each cluster is used to represent all products in the cluster.

3 Pilot Studies

3.1 Pilot 1: Application of Principal Components Analysis

3.1.1 Summary

In this first pilot study we investigate process improvements for help desk operations. Most help desk managers keep records on incidents and how they are resolved. Rarely are these analyzed with respect to recommendations for improvements. At the same time, improvements have the potential to decrease costs of help desk operations and increase effectiveness. Incident reports from two desktop software products (one incident prone, the other not incident prone) used at a major, multinational corporation are analyzed in this pilot study. Several recommendations for improvements of help desk operations were identified.

3.1.2 Purpose

The purpose of this pilot study is to investigate a technique for analyzing help desk incident data behavior to identify operational process improvements. Managers at help desks would benefit from knowing incident attribute behavior with respect to preferred resolution outcomes. For example, if an incident submitted to a help desk is identified as impacting the productivity of a large number of employees in an organization, help desk managers would be motivated to facilitate a quick resolution of the incident through prioritization of resources. Overall, managers prefer incidents with wide impact to be resolved quickly. An analysis of high impact problems submitted to the help desk should indicate a trend toward prioritization of incidents. Verifying this trend confirms help desk operations meet the business objective of incident prioritization for high-impact problems. However, if high-impact incidents tend not to be prioritized, the analysis is successful in discovering an opportunity for improvement. In practice this type of analysis is seldom conducted. Principal Components Analysis (PCA) is investigated in this pilot study to address the analytical gap. We attempt to analyze data from an industrial IT help desk to answer the following questions related to this pilot study:

RQ1- 1: Can an incident report database for desktop software products be used to discover primary relationships between incident report attributes?

RQ1- 2: Can process improvements be formulated from attribute relationships discovered in the principal components of an incident report database?

RQ1- 3: Can preferred attribute relationships which are not observed in the data be useful toward identifying unnecessary resolution processes?

3.1.3 Scope

This pilot study is scoped to an analysis of desktop software failures in an industrial setting with over 100,000 employees. The core business of the organization is comprised of the manufacture and sale of large hardware and software systems, and the provision of systems engineering expertise in technology-based solutions. Thousands of software applications facilitate productivity among the employees. Most products are commercial off the shelf (COTS). In this study, two such products are investigated. One is incident-prone and the other is not. Using this approach we establish the foundation of one aspect of this research so that it may be extended in scope to other techniques and to a larger set of products.

3.1.4 Approach

3.1.4.1 Overview

We use a strategy which combines multiple sources of evidence with goals to effect change in help desk processes. This work is a cross between *case study* and *action research* [93]. We present this work as an introduction to an iterative case study in which we demonstrate help desk improvement methods. This iteration concludes with opportunities for future research in the implementation of recommended improvements and the evaluation of the effectiveness of the improvements. For simplicity, we refer to this work as a case study while acknowledging the hybrid approach described above.

3.1.4.2 Incident Data

The data used in this pilot study is described in detail in Appendix A: Help Desk Operations. In our approach, we extract a dataset D of incident history over time T for software product P by querying the help desk database for incident records associated with the 156 selected according to the process in Appendix B. Incidents in D which are unresolved or noted as canceled may not have values for the attributes which are of interest in our analysis, so they are discarded. Values from n incident attributes are extracted from the full set of attributes A for P. Descriptors for the incident attributes selected in this case study are shown in Table A.1. All selected attributes have numerical values for all incident records. For attributes with binary values where $1 = true$ and $0 = false$, blanks are implicitly false and are thus replaced with zeros to maintain computational tractability. The 17 attributes in our analysis were selected based on their quantitative data, rather than text fields with personnel information. The resulting D for our PCA consists of N incident records with n attributes. This format is consistent with [61]. A description of the 17 incident attributes is included in Appendix A: Help Desk Operations.

3.1.4.3 Principal Components Analysis

We use the R statistical tool to perform PCA. The *stats* package includes the *prcomp* method with options to center and scale the input data, in our case *D*. The *prcomp* method centers data for each attribute by first determining the arithmetic mean of the set of values for an attribute, then subtracting the mean from each value. Data scaling is achieved by calculating the standard deviation of the values for an attribute, then dividing each value by the standard deviation of the attribute data. All attributes are scaled using this standardization method. We confirmed the functionality of the built-in *center* and *scale* options on our datasets by mean-centering and scaling through manual calculations, and then applying *prcomp* to the scaled datasets without the *center* and *scale* options. We then compared the PCA results using the manual scaling methods to those obtained through use of the *center* and *scale* options applied to raw (unscaled) data, and confirmed identical results. We include a mixture of continuous, rank order (ordinal) and categorical (1/0) variables in our analysis, as shown in Table A.1. A discussion of the suitability of PCA with a mixture of variable types is included in [61]. PCA is justified as suitable with both heterogeneous and non-heterogeneous data types by referring to the overall objective of PCA - to determine a subset of variables responsible for most of the variation in the dataset. The discussion in [61] recognizes challenges when interpreting linear functions of continuous variables vs. that of categorical variables but does not exclude ordinal and categorical variables from the suitability of PCA. Not all authors concur with this position [81]. We refer to our hybrid case study approach which also includes an exploratory component regarding the use of binary data with PCA. A future iteration could accommodate techniques designed specifically for the treatment of categorical data with PCA. In this case study we adopt the use of continuous, rank order and categorical data with PCA as is successfully demonstrated in [21, 127]. As noted in Table A.1, there are differences in scales between some of the attributes in our resulting dataset. Further analysis of the dataset in this case study shows large differences in the standard deviation between many of the attributes. This motivates us to standardize our data as described in [61] by mean-centering the data for each attribute, then dividing by the respective standard deviation prior to applying PCA. For a dataset with *n* attributes, a PCA of the standardized data produces an *n* by *n* matrix *M* of factor loadings. The set of rows r_1 through r_n of *M* hold the loadings of each factor (principal component) such that each factor is interpreted as a linear combination of the standardized attributes in the original dataset *D*. Each column c_1 through c_n is comprised of a vector of *n* attribute loadings, one for each attribute. To facilitate analysis of *M*, the columns are labeled PC_1 through PC_n and the rows are labeled a_1 through a_n. The principal components are ordered as PC_1 through PC_n, such that PC_1 accounts for the most variance in the dataset and PC_n accounts for the least variance.

$$Var(PC_1) \geq Var(PC_2) \geq ... \geq Var(PC_n) \geq 0 \tag{3.1.1}$$

Each row r_i in *M* corresponds to attribute a_i in the original dataset. Each column c_j in *M* contains the vector of loadings for attributes 1 through *n*, for PC_j. A decision can be made regarding a cutoff threshold for the number of principal components to be accepted and rejected in the analysis. Components with higher loadings are of interest in this analysis. Using a component selection method in [1], the Kaiser criterion is applied by selecting components with an eigenvalue (of the correlation matrix) greater than one. A threshold *k* is established by this method, where $1 \leq k \leq n$. Application of *k* returns PC_1 through PC_k which become the target of our analysis for attribute relationships and process improvements. From the *k* principal components derived using the threshold method described above, a loading from each attribute is assigned to no more than one principal component by the following method. PCA conventions justify selection of the attribute loading for principal component PC_j based on a loading threshold of 0.32 [1]. For each attribute a_i, the principal component with the largest absolute value greater than 0.32 in *k* attribute loadings is selected. In other words, an attribute is viewed to belong to the principal component where the attribute has the highest positive loading [127] greater than 0.32. Completion of this selection method for attributes a_1 through a_n results in $n' \leq n$ flagged loadings, each of which is associated with one principal component. For $0 < k \leq n$, each principal component will contain zero or more flagged attribute loadings but not more than $n-k+1$. With our flagged attributes in the set of *k* principal components, we are positioned to extract attribute relationships. We observe for each principal component PC_j there are $0 \leq m_j \leq n-k+1$ flagged attributes. For $m_j > 1$ we form all possible unique pairwise relationships between flagged attributes. We note that the number of unique attribute pairs pr_j for m_j flagged attributes is

$$pr_j = \sum_{q=1}^{m_j}(q-1) \tag{3.1.2}$$

Because we permit $m_j = 1$, we maintain interest in all factors which have exactly one flagged attribute and address these singleton instances through separate analysis.

3.1.4.4 Results and Interpretation

With our flagged attribute pairs and any singletons identified, we may proceed to the interpretation phase of our approach. Before interpretation, we assign roles to the attributes such that each attribute is an *input* or an *output*. The role assignment strategy assists us in defining preferred attribute pair relationship behavior. Table A.1 is annotated with the assigned roles for all attributes. Domain knowledge gained through the engagement of help desk managers in our case study confirms the following incident attribute pair relationships model preferred incident resolution behavior. The following notation is used to describe attribute pair behavioral relationships:

$$a_u\{\uparrow \mid \downarrow\} \Rightarrow a_v\{\uparrow \mid \downarrow\} \tag{3.1.3}$$

The notation is interpreted as "in attribute pair (a_u, a_v) for $u \neq v$, as a_u $\{$*increases* $\uparrow$ *or decreases* $\downarrow\}$, a_v $\{$*increases* $\uparrow$ *or decreases* $\downarrow\}$. The attribute relationship

$$A1 \uparrow \ \Rightarrow A14 \downarrow \tag{3.1.4}$$

means "as *incident urgency* increases, *minutes to respond to incidents* decreases." Attribute relationships are interpreted in the context of the behavioral tendency of all N incidents. All attribute pairs for each factor are annotated in this manner. For factors with exactly one flagged attribute such as

$$A7 \uparrow \tag{3.1.5}$$

behavior in the incident set is interpreted as "*the number of matches to any other incident* varies independently from other attributes."

We have a priori knowledge of preferred incident resolution outcomes from operational familiarity with IT service management. These outcomes are validated through interviews with two managers of the company's help desk. Some examples of desirable relationships between the inputs and outputs defined in Table A.1 include (3.1.4), and

- $A3 \uparrow \ \ \Rightarrow A15 \downarrow$ as *incident priority* increases, *hours to resolve incidents* decreases.
- $A5 \uparrow \ \ \Rightarrow A4 \uparrow$ as *incidents marked as resolvable during initial contact with the help desk* increase, resulting *resolution through initial contact* also increases.

Table 3.1 shows a set of desirable attribute relationships between inputs and outputs. In this case study, the discovery of desirable incident resolution behavior confirms help desk processes are beneficial for the products under investigation. Similarly, undesirable attribute relationships which are not observed through PCA imply help desk operations are working, or at least avoid detrimental processes. We are equally interested in knowing about desired behavior which is not apparent in our principal components, and relationships which do not follow desired behavior, as these form the basis for corrective action and operational improvements. From our observations of incident resolution behavior, we construct recommendations for help desk managers to improve help desk functionality.

3.1.4.5 Process

We summarize our process as follows:

1. Classify each attribute as an input or an output.
2. Define attribute input/output relationships which are desirable.
3. Perform a PCA on the incident record set.
4. Identify and then compare attribute relationships in each principal component with the set of desired relationships.
5. Categorize attribute pairs and singletons. Category IDs precede their description:
   (a) (C1) observed input/output relationships which follow desirable behavior
   (b) (C2) observed input/output relationships which do not follow desirable behavior
   (c) (C3) non-observed input/output relationships which follow desirable behavior
   (d) (C4) non-observed input/output relationships which follow undesirable behavior
   (e) (C5) observed relationships (not constrained to input/output types) which confirm
      i. (C5.1) consequential desired behavior
      ii. (C5.2) consequential undesired behavior
      iii. (C5.3) consequential behavior that is neither desired or undesired
6. Form recommendations based on categorized observed and unobserved attribute pair relationships.

Table 3.1: Desirable attribute pair relationships

(AP1)	A1 $\uparrow$	incident urgency	$\Rightarrow$	A4 $\uparrow$	FCR compliant
(AP2)	A1 $\uparrow$	incident urgency	$\Rightarrow$	A12 $\downarrow$	total transfers
(AP3)	A1 $\uparrow$	incident urgency	$\Rightarrow$	A14 $\downarrow$	minutes to respond
(AP4)	A1 $\uparrow$	incident urgency	$\Rightarrow$	A15 $\downarrow$	hours to resolve
(AP5)	A1 $\uparrow$	incident urgency	$\Rightarrow$	A16 $\uparrow$	RTP compliant
(AP6)	A2 $\uparrow$	incident impact	$\Rightarrow$	A4 $\uparrow$	FCR compliant
(AP7)	A2 $\uparrow$	incident impact	$\Rightarrow$	A12 $\downarrow$	total transfers
(AP8)	A2 $\uparrow$	incident impact	$\Rightarrow$	A14 $\downarrow$	minutes to respond
(AP9)	A2 $\uparrow$	incident impact	$\Rightarrow$	A15 $\downarrow$	hours to resolve
(AP10)	A2 $\uparrow$	incident impact	$\Rightarrow$	A16 $\uparrow$	RTP compliant
(AP11)	A3 $\uparrow$	incident priority	$\Rightarrow$	A4 $\uparrow$	FCR compliant
(AP12)	A3 $\uparrow$	incident priority	$\Rightarrow$	A12 $\downarrow$	total transfers
(AP13)	A3 $\uparrow$	incident priority	$\Rightarrow$	A14 $\downarrow$	minutes to respond
(AP14)	A3 $\uparrow$	incident priority	$\Rightarrow$	A15 $\downarrow$	hours to resolve
(AP15)	A3 $\uparrow$	incident priority	$\Rightarrow$	A16 $\uparrow$	RTP compliant
(AP16)	A5 $\uparrow$	FCR resolvable	$\Rightarrow$	A4 $\uparrow$	FCR compliant
(AP17)	A5 $\uparrow$	FCR resolvable	$\Rightarrow$	A12 $\downarrow$	total transfers
(AP18)	A5 $\uparrow$	FCR resolvable	$\Rightarrow$	A13 $\downarrow$	incident escalated
(AP19)	A5 $\uparrow$	FCR resolvable	$\Rightarrow$	A14 $\downarrow$	minutes to respond
(AP20)	A5 $\uparrow$	FCR resolvable	$\Rightarrow$	A15 $\downarrow$	hours to resolve
(AP21)	A5 $\uparrow$	FCR resolvable	$\Rightarrow$	A16 $\uparrow$	RTP compliant
(AP22)	A6 $\uparrow$	incident match	$\Rightarrow$	A4 $\uparrow$	FCR compliant
(AP23)	A6 $\uparrow$	incident match	$\Rightarrow$	A12 $\downarrow$	total transfers
(AP24)	A6 $\uparrow$	incident match	$\Rightarrow$	A13 $\downarrow$	incident escalated
(AP25)	A6 $\uparrow$	incident match	$\Rightarrow$	A15 $\downarrow$	hours to resolve
(AP26)	A17 $\uparrow$	RTP eligible	$\Rightarrow$	A4 $\uparrow$	FCR resolvable
(AP27)	A17 $\uparrow$	RTP eligible	$\Rightarrow$	A12 $\downarrow$	total transfers
(AP28)	A17 $\uparrow$	RTP eligible	$\Rightarrow$	A13 $\downarrow$	incident escalated
(AP29)	A17 $\uparrow$	RTP eligible	$\Rightarrow$	A14 $\downarrow$	minutes to respond
(AP30)	A17 $\uparrow$	RTP eligible	$\Rightarrow$	A15 $\downarrow$	hours to resolve
(AP31)	A17 $\uparrow$	RTP eligible	$\Rightarrow$	A16 $\uparrow$	RTP compliant

Relationships in categories (C1) and (C5.1) confirm preferred help desk resolution methodologies. We are equally interested in relationships in (C2) and (C5.2). These expose opportunities to make recommendations for improvements. The breakdown of relationships in (C5) is provided to categorize output/output relationships which make sense based on our knowledge of help desk operations, yet provide a basis for improvements. In other words, some relationships are a consequence of how the help desk operates, but demonstrate undesirable behavior for which improvement opportunities may exist. Some output/output relationships are categorized as (C5.3) in cases where behavior between attributes is observed to have no desirable or undesirable consequences on resolution outcome. We discuss relationships in categories (C3) as missed opportunities, and in (C4) as avoidance of undesirable resolution outcomes. We complete our analysis by providing recommended help desk process changes.

3.1.4.6 Case Study

Help desk data for two desktop software products P_a and P_b were selected to obtain D_a and D_b, respectively. The time interval T over which the historical data exists is the same for both products. P_a and P_b were selected based on their scope of distribution, incident volume, and proneness to incidents as summarized in Table 3.2 . Product A is a widely-

Table 3.2: Product Statistics

Product	# Users	% Users	# Incidents	Time period T
P_a	$> 100,000$	97%	2706	1/2008 - 12/2012
P_b	$> 60,000$	49%	343	1/2008 - 12/2012

distributed web browser plug-in for viewing videos. Employees use Product A extensively for watching corporate training videos and viewing multimedia content that is embedded in websites provided for various lines of business. The wide distribution, high level of use and frequency of product updates influence the incident volume for this product. In contrast, Product B is less widely distributed and is used infrequently. Product B is a cost-effective solution for view-only access to graphical content created with a more expensive graphics package. Differences in incident-proneness motivate selection of these two products in our analysis rather than the actual intended use of the software.

For P_b we flag the $m_b = 16$ out of seventeen attributes which meet our acceptance criteria for loading. Recall that the number of unique attribute pairs pr_j for $m_j > 1$ for

Figure 3.1: Attribute Relationships - Product A (part 1)

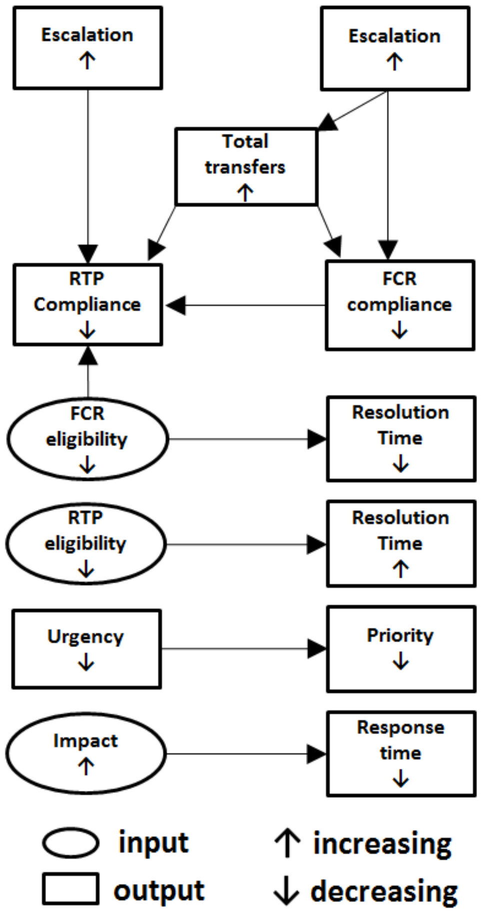


Table 3.3: Factor loadings (Product A)

Attr	PC1	PC2	PC3	PC4	PC5	PC6	PC7
A1	0.282	-0.196	**-0.542**	0.159	-0.073	-0.087	0.080
A2	0.077	-0.067	-0.159	0.234	**0.376**	0.333	-0.314
A3	0.292	-0.196	**-0.544**	0.171	-0.035	-0.053	0.052
A4	**-0.422**	0.017	-0.032	-0.009	-0.042	-0.037	-0.026
A5	-0.202	-0.057	-0.249	**-0.557**	-0.044	-0.071	-0.018
A6	0.058	**0.651**	-0.213	0.015	-0.018	0.091	-0.039
A7	0.103	**0.376**	-0.085	0.045	0.180	-0.200	-0.284
A8	0.002	0.361	-0.151	-0.078	-0.332	**0.676**	0.216
A9	-0.011	-0.008	0.002	0.006	-0.408	-0.041	**-0.822**
A10	-0.004	**0.401**	-0.131	0.065	0.144	-0.373	-0.017
A11	0.055	0.219	-0.005	-0.098	-0.046	**-0.429**	0.242
A12	**0.371**	0.001	0.066	-0.307	0.073	0.026	-0.014
A13	**-0.416**	0.010	-0.115	0.253	0.003	-0.006	0.021
A14	0.040	-0.039	-0.024	0.081	**-0.703**	-0.194	0.087
A15	0.273	-0.003	0.104	**0.401**	0.034	0.0570	-0.105
A16	**-0.408**	-0.032	-0.265	0.005	0.073	-0.003	-0.004
A17	-0.203	-0.094	-0.361	**-0.483**	0.100	0.007	-0.090

Table 3.4: Recommendations for Preferred Help Desk Resolution Methodologies Discovered - Product A

Category	Relationship	Recommendation
C1	A2 $\uparrow$ incident impact $\Rightarrow$ A14 $\downarrow$ time to respond	none - process works
C5.1	A1 $\downarrow$ urgency $\Rightarrow$ A3 $\downarrow$ priority	none - process works

PC_j is found through (3.1.2). For P_a only one flagged attribute (A9) for PC7 exists. We note in Table 3.7 for P_b, PC3 and PC6 each have exactly one flagged attribute.

First, we analyze P_a, the more incident-prone product, for all attribute pairs detected in our PCA. We present a graphical representation of observed attribute pairs in Figures 3.1 and 3.2, and compare those to the desired attribute pairs listed in Table 3.1. The same desired attribute pairs in Table 3.1 are presented in as shaded cells in Table 3.6 with double arrows indicating the preferred directional tendency of the related attributes. When overlaid with the observed attribute pairs shown as single arrows (also indicating the directional tendency of the attributes), we see which observed attribute pairs coincide with desired pairs. For P_a we see exactly one observed attribute pair A2 $\uparrow$ (incident impact) $\Rightarrow$ A14 $\downarrow$ (minutes to respond) which coincides with a preferred pair, inclusive of directional tendency. The observed pair is identified as (AP8) in Table 3.1. Using Equation 3.1.2, the observed

Figure 3.2: Attribute Relationships - Product A (part 2)

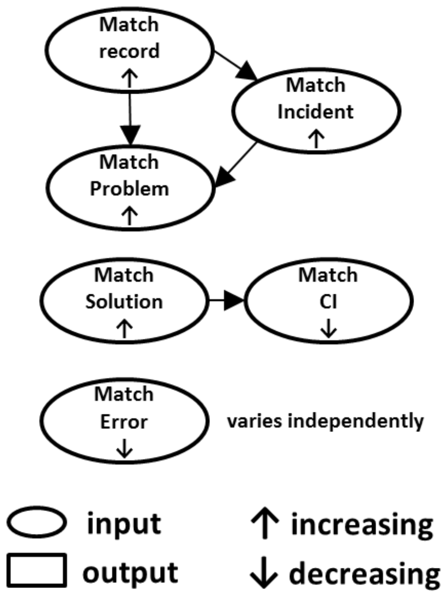


Table 3.5: Recommendations for Non-preferred Help Desk Resolution Methodologies Discovered - Product A

Category	Relationship	Recommendation
C2	A15 $\uparrow$ resolution time $\Rightarrow$ A17 $\downarrow$ RTP eligible	define an escalation time threshold $t_{escl} < t_{rtp}$ to preserve RTP compliance
C5.2	A4 $\downarrow$ FCR compliant $\Rightarrow$ A12 $\uparrow$ incident transfers	define an escalation time threshold $t_{escl} < t_{rtp}$ to minimize incident transfers
	A4 $\downarrow$ FCR compliant $\Rightarrow$ A16 $\downarrow$ RTP compliant	
	A5 $\downarrow$ FCR eligible $\Rightarrow$ A17 $\downarrow$ RTP eligible	
	A12 $\uparrow$ incident transfers $\Rightarrow$ A13 $\downarrow$ incident escalated	
	A12 $\uparrow$ incident transfers $\Rightarrow$ A16 $\downarrow$ RTP compliant	
	A13 $\downarrow$ incident escalated $\Rightarrow$ A16 $\downarrow$ RTP compliant	
	A4 $\downarrow$ FCR compliant $\Rightarrow$ A13 $\downarrow$ incident escalated	

set of attribute pairs and singletons are analyzed for recommendations. The resulting recommendations are shown in Table 3.4 and Table 3.5 for P_a.

We refer to our graphical representation of attribute pair relationships in Figure 3.1 to further assist our interpretation of resolution behavior and the construction of recommended process changes. A comparison of the observed attribute pair relationships found from the flagged component loadings for P_a in Table 3.3 to desired attribute relationships in Table 3.1 shows us attribute relationships which confirm desired incident resolution at the help desk. These are categorized as (C1). Relationships of type (C1) are limited to input/output combinations based on attribute roles defined in Table A.1. (C5.1) relationships are of type input/input or output/output in our approach summary, and are referred to as consequential relationships. Attribute combinations which are contrary to desired behavior are categorized as (C2) (input/output) and (C5.2) (input/input or output/output). No recommendations for process adjustments are made for (C1) and (C5.1) since these relationships imply intended behavior. For (C2) and (C5.2), we construct recommendations with intentions of changing undesirable attribute relationships into desirable ones.

We investigate desired relationships from Table 3.1 which do not show up in our component loadings. These are categorized as (C3). The absence of desirable resolution behavior forms the basis of recommendations for practices which target improvements at the help desk. We observe only one (C1) relationship in P_a which is desirable (AP8). This relationship shows incidents with higher potential impact to the desktop computing environment tend to get the attention of help desk technicians faster, through decreased time for the help desk to respond. While this is noted as a desired practice, most incidents are given their designations for impact, urgency and resulting prioritization after the help desk technician responds to and initiates documentation of the incident. Only incidents submitted through the web portal have attribute values established for these parameters prior to the help desk responding to the incident submission. This key difference between the origin of selected levels of impact and urgency (by the submitter for web submissions and by the help desk for telephone contact) suggests the (AP8) relationship should be interpreted in the context of the incident submission method. Selection of inappropriate levels of urgency and impact by the end user might be made in the interest of getting faster service rather than representing the actual urgency of an incident. The single (C5.1) output/output observed relationship in P_a is listed in Table 3.4. For this we make no recommendation for change since we find the consequence of decreased incident priority from decreased incident urgency to be acceptable. No other desirable relationships in Table 3.1 are evident in our principal component loadings for P_a. These missing, yet desirable relationships are categorized as (C3). The absence of relationships with incident urgency (AP1 through AP5), priority (AP11 through AP15) and all incident impact relationships (AP6 through AP10), with the exception of (AP8) tell us these parametric inputs do not vary strongly with favorable outcomes such as FCR and RTP compliance. Stated differently, efforts to categorize incidents as *urgent* and of *higher impact* do not really help meet service goals of incident resolution during a single, initial contact with the help desk. Similarly, time-based targets established for restoring productivity through incident resolution are not strongly tied to *urgency*, *impact*, and *priority*. Two input attributes, *FCR resolvable* and *RTP eligible* show no strong variance with resolution efficiency such as minimizing wait time, incident transfers, and time to resolve the problem. Incident matching, an effort which is intended to facilitate resolution through reference to problem or solution similarity, incident transfers, and time to resolve the problem, show no strong variance with these outcomes. In fact, attribute relationships with strong loadings in P_a for *incident match* vary mostly with matches to other archival record types such as known errors and problems. Unobserved desirable behavior for P_a motivates us to draw attention to RQ3 in which we seek to identify help desk operational emphasis which may be ineffective in the overall goal to quickly resolve problems. Based on our analysis of P_a, we propose the following recommendation to address missing desirable relationships.

- Efforts to promote incident matching should be evaluated for their cost vs. benefits since they do not seem to influence preferred resolution outcomes.

Table 3.6: Observed attribute relationships pairs with desirable attribute pairs overlaid - Product A

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1 ⇑ ↓	-		↓	⇑*					
A2 ⇑		-		⇑*					
A3 ⇑			-	⇑*					
A4				-					
A5 ⇑				⇑*	-				
A6 ⇑ ↑				⇑*		-	↑		
A7							-		
A8								-	
A9 ↓									↓
A10									
A11									
A12									
A13									
A14									
A15									
A16									
A17 ⇑				⇑*					
	A10	A11	A12	A13	A14	A15	A16	A17	
A1 ⇑			⇓*		⇓*	⇓*	⇑*		
A2 ⇑ ↑			⇓*		↓⇓*	⇓*	⇑*		
A3 ⇑			⇓*		⇓*	⇓*	⇑*		
A4 ↓			↑	↓			↓		
A5 ⇑ ↓			⇓*	⇓*	⇓*	⇓*↑	⇑*	↓	
A6 ⇑ ↑	↑		⇓*	⇓*		⇓*			
A7 ↑	↑								
A8 ↑		↓							
A9									
A10	-								
A11		-							
A12 ↑			-	↓			↓		
A13 ↓				-			↓		
A14					-				
A15 ↑						-		↓	
A16							-		
A17⇑			⇓*	⇓*	⇓*	⇓*	⇑*	-	

Notes:
1. Cells with asterisks identify desired pairs for respective row/col attributes
2. Double arrows show directional tendency of desired attribute pairs
3. Single arrows show directional tendency of observed attribute pairs
4. Observed pairs overlaid with desired pairs show comparison of desired vs. observed resolution behavior.

Table 3.7: Factor loadings (Product B)

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
A1	0.235	**0.514**	-0.333	0.022	0.003	0.105	0.082	-0.014
A2	0.206	0.111	-0.122	0.266	**-0.426**	-0.109	-0.025	0.067
A3	0.280	**0.505**	-0.319	0.095	-0.022	0.071	0.076	-0.004
A4	**-0.478**	0.021	-0.062	-0.006	-0.210	0.138	0.223	0.016
A5	-0.287	0.214	-0.199	**-0.507**	-0.049	-0.016	0.103	-0.018
A6	-0.053	0.389	**0.570**	0.011	-0.120	0.022	-0.009	0.010
A7	0.004	0.265	0.407	0.095	0.237	-0.201	**0.447**	0.115
A8	-0.075	0.2780	0.389	-0.075	-0.388	0.220	**-0.437**	-0.096
A9	-0.074	-0.012	-0.030	0.000	-0.193	0.102	0.285	**0.765**
A10	-0.071	-0.015	-0.026	0.024	-0.240	0.008	0.497	**-0.614**
A11	0.094	0.119	0.171	-0.024	**0.522**	0.079	0.139	-0.020
A12	**0.455**	-0.102	0.084	-0.018	-0.019	-0.262	-0.151	-0.001
A13	-0.200	-0.038	-0.064	**0.617**	-0.090	0.073	-0.021	0.025
A14	-0.118	0.123	-0.138	-0.283	0.176	0.281	-0.261	0.050
A15	0.192	-0.123	0.068	**-0.325**	-0.312	-0.209	0.191	0.072
A16	**-0.380**	0.193	-0.113	0.261	0.209	-0.284	-0.195	-0.015
A17	-0.209	0.174	-0.094	-0.113	-0.060	**-0.755**	-0.153	0.035

Relationships categorized as (C4), undesirable ones which do not show up in our PCA, are simply noted as such. (C5.3) relationships are treated similarly, as they have no particular desirable or undesirable consequences on help desk processes.

We turn our attention to the observed set of attribute pairs for P_b, the less incident-prone product. After performing a PCA on the incident dataset for P_b, we apply Equation 3.1.2 to the component loadings shown in Table 3.7 to obtain the attribute pairs shown in Table 3.8. We apply the same categorization schema used for P_a and produce graphical representations of attribute relationships in Figures 3.3 and 3.4 to assist our analysis. In Table 3.8 we note the absence of observed attribute pairs which coincide with preferred attribute combinations through the presentation of overlaid results. Referring to the category codes defined in the summary of our approach described earlier, and used in the analysis of P_b, we have no observed relationships categorized as (C1). For (C5.1) we observe one consequential, desired output/output relationship which confirms the value of incident escalation in the reduction of resolution time. There is coincidence between observed and preferred pairs with P_b, but the direction in which the attributes show tendency does not fully coincide. These observed pairs are categorized as (C2) and (C5.2) and are the undesirable relationships for which we construct recommendations for process adjustments as shown in Table 3.10.

Figure 3.3: Attribute Relationships - Product B (part 1)

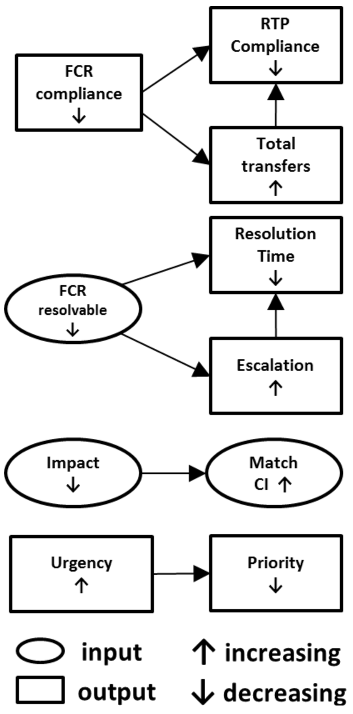


Figure 3.4: Attribute Relationships - Product B (part 2)

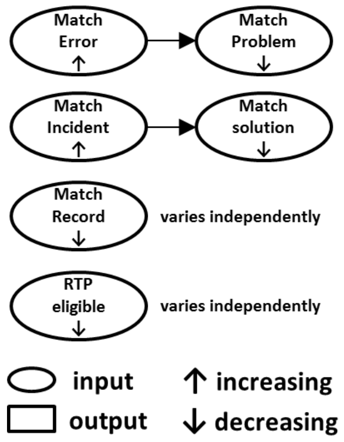


Table 3.8: Observed attribute relationships pairs with desirable attribute pairs overlaid - Product B

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1 ⇑ ↑	-		↑	⇑*					
A2 ⇑		-		⇑*					
A3 ⇑			-	⇑*					
A4				-					
A5 ⇑				⇑*	-				
A6 ⇑ ↑				⇑*		-			
A7 ↑							-	↓	
A8								-	
A9									
A10									
A11									
A12									
A13									
A14									
A15									
A16									
A17 ⇓				⇑*					

	A10	A11	A12	A13	A14	A15	A16	A17
A1 ⇑			⇓*		⇓*	⇓*	⇑*	
A2 ⇑ ↓		↑	⇓*		⇓*	⇓*	⇑*	
A3 ⇑			⇓*		⇓*	⇓*	⇑*	
A4 ↓			↑				↓	
A5⇑↓			⇓*	⇓*↑	⇓*	⇓* ↓	⇑*	
A6⇑↑			⇓*	⇓*		⇓*		
A7								
A8								
A9 ↑	↓							
A10	-							
A11		-						
A12 ↑			-				↓	
A13 ↑				-		↓		
A14					-			
A15						-		
A16							-	
A17⇑↓			⇓*	⇓*	⇓*	⇓*	⇑*	-

Notes:
1. Refer to notes at bottom of Table 3.6

Table 3.9: Recommendations for Preferred Help Desk Resolution Methodologies Discovered - Product B

Cat.	Relationship	Recommendation
C5.1	A13 $\uparrow$ urgency $\Rightarrow$ A15 $\downarrow$ resolution time	none - processes are working

Table 3.10: Recommendations for Non-preferred Help Desk Resolution Methodologies Discovered - Product B

Cat.	Relationship	Recommendation
C2	A5 $\downarrow$ FCR resolvable $\Rightarrow$ A13 $\uparrow$ incident escalated	define an escalation time threshold $t_{escl} < t_{rtp}$ to preserve RTP compliance
	A5 $\downarrow$ FCR resolvable $\Rightarrow$ A15 $\downarrow$ resolution time	
C5.2	A4 $\downarrow$ FCR compliant $\Rightarrow$ A16 $\downarrow$ RTP compliant	define an escalation time threshold $t_{escl} < t_{rtp}$ to preserve RTP compliance
	A12 $\uparrow$ total transfers $\Rightarrow$ A16 $\downarrow$ RTP compliant	
	A4 $\downarrow$ FCR compliant $\Rightarrow$ A12 $\uparrow$ total transfers	define an escalation time threshold $t_{escl} < t_{rtp}$ to minimize incident transfers

For category (C2) we observe two relationships which have (A5) (resolvable through FCR) in common. A negative tendency for FCR resolvability varies with a positive tendency toward incident escalation, and a reduction in incident resolution time. As discussed earlier, relatively few of the incidents in our dataset are marked as not resolvable through FCR, yet sufficient variance exists between those incidents to demonstrate a trend toward 1) incidents being escalated (A13) and 2) shorter resolution times (A15). An interesting triangulation of attribute relationships within these two pairs would suggest managerial use of flagging certain incidents as not resolvable through FCR (an input in our analysis) may in fact be an effective way to achieve shorter resolution times through incident escalation.

Relationships produced from our PCA using P_b categorized as (C5.2), consequential undesirable resolution behavior, deserve similar analysis. In two out of three relationships, (A16) RTP compliance varies negatively as resolution through FCR (A4) decreases and total transfers (A12) increases. Again, a three-way relationship is discovered which in this case confirms excessive transferring of incidents to lateral groups at the help desk negatively influences FCR and RTP compliance. As with P_a, a recommendation to define a time threshold by which escalation is invoked to preserve FCR and RTP compliance is recommended. Similar to conclusions reached in our analysis of P_a, a recommendation for a time-based threshold for escalation would minimize incident transfers in favor of engaging a priority manager. As with P_b, the missing, desirable relationships (C3) may be compared to those in (C2) to assess what is not being achieved at the help desk vs. what managers would like to see achieved. For example, we observe two output/output involving attributes related to incident matching which do not relate to attaining FCR compliance, and minimization of incident transfers, escalations, and resolution time. These are (AP22) through (AP25) in Table 3.1. This was also noted in P_a. In this analysis we performed a separate PCA for P_a and P_b, and made help desk process recommendations through independent product analyses. Although we observed differences in attribute relationship pairs in the two products, we were able to observe some similarities in attribute behavior. Our analysis of P_a and P_b produces similar help desk process recommendations driven by different attribute relationships. The outcome of our analysis of both products shows us escalation in the incident life cycle during which resolution has not occurred may help achieve service goals. We observed triangular attribute relationship structures in both products which provided additional insight into help desk processes. These types of relationship structures establish opportunities for further research.

3.1.5 Lessons Learned and Conclusions

In our research we have demonstrated success in using PCA to analyze data for process improvements at an IT help desk in a large corporation. Through our approach to finding recommendations for improvements in help desk operations, we have shown ways to discover primary relationships between incident attributes using PCA. Combinations of help desk incident attributes were identified using best practices in PCA. Pairwise relationships were formed using attribute combinations and compared to knowledge of preferred resolution behavior gained from interaction and experience with help desk operations. The validity of our approach and interpretations were confirmed by help desk managers. Our conclusions led us to formulate recommendations for process improvements.

Having demonstrated success with the application of PCA to help desk incidents in Pilot 1, extension of the technique to other types of data was selected as the next step in this research. The generalizability to survey data analysis is described in Section 3.2, Pilot 2: Extension of PCA to Survey Data Analysis.

3.2 Pilot 2: Extension of PCA to Survey Data Analysis[1]

3.2.1 Summary

Pilot study 2 demonstrates the generalizability of the techniques we used with IT help desk incidents in Pilot study 1. Specifically, PCA is used in this next pilot to discover statistical relationships between survey responses. An example of a survey response relationship between two TRUE/FALSE questions is "respondents who answered TRUE to survey question number one responded FALSE to question number two." Knowing relationships between survey responses offers insight into the data that is not easily identified through descriptive statistics. Understanding the strength of the relationships is what PCA contributes to survey data analysis. PCA identifies relationship strength (or weakness) between datasets in terms of the level of their statistical covariance (or non-covariance). PCA was used in pilot study 1 to identify the relationship strength between incident attributes. The stronger relationships expressed greater covariance (negative or positive) and were retained in favor of weaker relationships with little covariance. In Pilot study 2, survey response

[1]This pilot study is included with reference to contributions by Dr. Julia Varnell-Sarjeant in her PhD thesis on the subject of software reuse in embedded vs. non-embedded systems [121]. This survey is also used by Varnell-Sarjeant et al. [122]. Dr. Varnell-Sarjeant designed and conducted the survey.

relationships with stronger covariance are retained to help express the outcome of the survey. Response relationships with weak covariance are interpreted as being weakly related and are considered to a lesser degree in the expression of the survey results. Essentially this approach is a survey data reduction technique, just as the approach in Pilot 1 was an incident attribute reduction technique.

Pilot study 2 was a collaborative effort, with contributions made by the following authors.

- Dr. Julia Varnell-Sarjeant designed the survey on the subject of artifact reuse with embedded and non-embedded systems, conducted semi-structured interviews of practitioners, and collected all survey data. Additionally, Dr. Varnell-Sarjeant performed a literature search for related work on the subject of reuse with embedded and non-embedded systems. Dr. Varnell-Sarjeant was the primary contributor to the interpretation of the analysis of the survey results and conclusions drawn from the outcome of the survey.
- Dr. Andreas Stefik performed the statistical analysis of the survey results.
- Joseph Lucente contributed to the background section through a literature search on the subject of PCA and performed PCA on the survey results.
- Dr. Anneliese Andrews facilitated the interpretation of the PCA results with Dr. Varnell-Sarjeant and Joseph Lucente.
- Dr. Anneliese Andrews provided oversight in an advisory role during all phases of the research.

3.2.2 Purpose

The purpose of Pilot study 2 is to demonstrate the generalizability of the technique in Pilot study 1 to survey data. Case study generalizability addresses concerns for external validity by demonstrating a technique that is used in a specific scope or context can be extended to a different scope of study [128]. Pilot study 1 is conducted in an IT help desk environment whereas Pilot study 2 is conducted with data from a survey on reuse strategies.

The approach taken in Pilot study 2 is designed to understand reuse strategies in the context of embedded vs. nonembedded systems. A search of existing literature left several questions that needed to be answered in order to understand and implement successful reuse. First was whether industry practitioners share the same reuse experience as the research for embedded and nonembedded systems. Second is the question of what artifacts can be reused for successful outcomes. Third is the whether there is a difference between embedded systems and nonembedded systems in outcomes. Thus, the following research questions were established for Pilot study 2:

RQ2-1 Do embedded systems use different development approaches than nonembedded systems? This question tries to identify what development approaches are being used on embedded systems vs nonembedded systems projects and how effective they are with regard to reuse. We also want to discover whether reuse strategies are used in combination. Reuse in various development approaches have been studied singly, but are there additional benefits of using the development methods in combination? The null hypothesis assumes that the same development approaches are used whether the system is embedded or nonembedded.

RQ2-2 Do embedded systems reuse different artifacts than nonembedded systems? What artifacts are being reused on what types of projects? Are there artifacts more commonly reused in embedded systems than in nonembedded systems? The null hypothesis assumes that embedded and nonembedded systems reuse the same artifacts at the same level with comparable outcomes.

RQ2-3 Do reuse outcomes vary between embedded systems and nonembedded systems? This question is central to our research. It is designed to determine whether there is a difference in reuse effectiveness or preferred development approaches based on project type. Positive outcomes are measured by: fewer labor hours, fewer defects, less test time, fewer items to be tested, and less risk. While many aspects of software reuse have been studied deeply, our literature review did not surface this specific comparison. The null hypothesis assumes that there is no significant difference in the effectiveness of reuse whether the project is an embedded or a nonembedded system.

3.2.3 Scope

The data analyzed in Pilot study 2 comes from a survey conducted in a large aerospace company, on the subject of system artifact reuse strategies [122]. The primary techniques used to analyze the results of the survey were MANOVA and PCA. The analysis with MANOVA was conducted by two of the authors. The PCA was performed by the author of this thesis and interpreted by the other authors. Therefore, the scope of Pilot study 2 is limited to the use of PCA to interpret the survey results.

3.2.4 Approach

To augment our statistical analysis, we performed multiple PCA to the survey response data of the survey questions. The intent was to determine whether there were survey questions that tended to vary together, particularly questions related to project attributes and outcomes. Survey response variables are derived from the four categories of survey questions: 1) system type, 2) development approaches, 3) reuse artifacts, and 4) outcomes. A goal-based strategy for variable selection is used. In each PCA we select a subset of variables in the interest of discovering the relationships between the variables, independent of all excluded variables. Table 3.11 identifies the variables included in each PCA that was performed. Test 'All' focuses on the relationships between responses for all survey question categories and therefore includes all variables. In test 'A' we are interested in the relationships between system type and reuse artifacts, independent of reuse approaches and outcomes. Tests ‘A1’ and ‘A2’ refine test ‘A’ by including the subset survey responses which indicate embedded system type only (test ‘A1’) and non-embedded system type (test ‘A2’), and the artifacts used for reuse. In test ‘A3’ we are interested in the relationships between the selection of artifacts, independent of system type, reuse approaches and outcomes. A similar approach is taken for variable selection in the remaining tests in Table 3.11.

Table 3.11: PCAs performed

PCA ID	Variables
All	All survey responses
A	System type, artifacts
A1	Embedded system type, artifacts
A2	Non-embedded system type, artifacts
A3	Artifacts only (from both system types)
B	System type, development approaches
B1	Embedded system type, development approaches
B2	Non-embedded system type, development approaches
B3	Development approaches only (from both system types)
C	System type, outcome
C1	Embedded system type, outcome
C2	Non-embedded system type, outcome
C3	Outcomes only (from both system types)

Table 3.12: Principal component relationships in survey responses

ID	Relationship
All_1	A reduction in the number of items for reuse varies independently.
All_2a	An increase in the number of tested products varies with a decrease in labor savings and less reduction in test time.
All_2b	Less labor savings through reuse varies with less reduction in test time
All_3	Greater use of a component based approach varies with less reuse of drawings.
All_4a	Greater use of a heritage/ad hoc approach varies with less reuse of tested clusters and greater risk reduction.
All_4b	Less reuse of tested clusters varies with more risk reduction.
All_5a	Greater use of a model based approach varies with less reuse of code and more reuse of models.
All_5b	Less reuse of code varies with greater reuse of models.
All_6	The level of reuse of hardware varies independently.
All_7	Embedded system type varies with less use of COTS/GOTS approach.
All_8	Greater use of a product line approach varies with more reduction in defects.
A_1a	Greater reuse of requirements varies with more reuse of architecture and tested products.
A_1b	Greater reuse of architecture varies with more reuse of tested products.
A_2a	Greater reuse of code varies with less reuse of drawings and hardware.

Table 3.12: Principal component relationships in survey responses (cont.)

ID	Relationship
A_2b	Less reuse of drawings varies with less reuse of hardware.
A_3	Less reuse of models varies with less reuse of use cases.
A_4	Non-embedded system type varies with less reuse of tested clusters.
A1_1a	Less reuse of requirements varies with less reuse of architecture and use cases.
A1_1b	Less reuse of architecture varies with less reuse of use cases.
A1_2	Greater reuse of code varies with less reuse of drawings.
A1_3a	Less reuse of models varies with more reuse of hardware and tested products.
A1_3b	Greater reuse of hardware varies with more reuse of tested products.
A2_1a	Greater reuse of requirements varies with more reuse of architecture and tested products.
A2_1b	Greater reuse of architecture varies with more reuse of tested products.
A2_2a	Less reuse of code varies with more reuse of drawings and hardware.
A2_2b	Greater reuse of drawings varies with more reuse of hardware.
A2_3a	Less reuse of models varies with less reuse of use cases and tested clusters.
A2_3b	Less reuse of use cases varies with less reuse of tested clusters.
A3_1a	Greater reuse of requirements varies with more reuse of architecture and tested products.
A3_1b	Greater reuse of architecture varies with more reuse of tested products.
A3_2a	Less reuse of code varies with more reuse of drawings and hardware.
A3_2b	Greater reuse of drawings varies with more reuse of hardware.
A3_3	Less reuse of models varies with less reuse of use cases.
B_1	Greater use of a component based approach varies with more use of model based approach.
B_2	Greater use of a product line based approach varies with less use COTS/GOTS based approach.
B_3	Non-embedded system type varies with less use of heritage/ad hoc approach.
B1_1	Less use of a component based approach varies with less use of COTS/GOTS approach.
B1_2	Greater use of a product line based approach varies with less use heritage/ad hoc approach.
B1_3	The level of use of the model based development approach varies independently.
B2_1	Less use of a component based approach varies with less use of model based approach.
B2_2	The level of use of a COTS/GOTS based development approach varies independently.
B2_3	Less use of a product line approach varies with less use of heritage/ad hoc approach.
B3_1	Less use of a component based approach varies with less use of model based approach.
B3_2	The level of use of a COTS/GOTS based development approach varies independently.
B3_3	Greater use of product line approach varies with greater use of heritage/ad hoc approach.
C_1a	Greater labor savings varies with greater reduction in test time and items to be tested.
C_1b	Greater reduction in test time varies with greater reduction in items to be tested.
C_2	System type varies independently.
C_3	Greater defect reduction varies with more risk reduction.
C1_1a	Less labor savings through reuse varies with less reduction in test time and items to be tested.
C1_1b	Less reduction in test time varies with less reduction in items to be tested.
C1_2	The level of defect reduction varies independently.

Table 3.12: Principal component relationships in survey responses (cont.)

ID	Relationship
C1_3	The level of risk reduction varies independently.
C2_1	Greater reduction in test time varies with a greater reduction in items to be tested.
C2_2	The level of labor savings varies independently.
C2_3	Greater defect reduction varies with greater risk reduction.
C3_1	Greater reduction in test time varies with a greater reduction in items to be tested.
C3_2	The level of risk reduction varies independently.
C3_3	Greater labor savings varies with less defect reduction.

3.2.4.1 PCA Results

The results from our PCA of reuse survey responses are discussed for each of the tests identified in Table 3.11. In each discussion we refer to specific relationships found in the principal components. Each relationship is uniquely identified in Table 3.12. The relationship identifier is constructed from the PCA ID in Table 3.11 and the specific principal component in which the relationship is identified. For example, relationship 'All_1' in Table 3.12 describes the relationship found in principal component 1 in test 'All'. In some principal components there are relationships between three variables. For those we identify the relationships, for example, as 'All_2a' and 'All_2b'. The relationships are determined from the loadings produce for each principal component. The loadings identified through the PCA interpretation process describe in our approach are presented in bold. We include the loading tables produced for each PCA listed in Table 3.11 in Appendix B. In the loading tables, PCA relationships are determined by the responses with higher relative loadings, as indicated in boldfaced font.

3.2.4.2 Test 'All' Results

We begin with a discussion of the relationships discovered through inclusion of all survey responses in test 'All'. Eleven relationships identified in Table 3.12 are noted from the eight principal components resulting from test 'All'. We refer to the identifiers in Table

3.12 when discussing specific survey response relationships. In relationship All_1 we discover the outcome of reduction in the number of defects through reuse is independent of both embedded and non-embedded systems, the development approach that is taken and the artifacts selected for reuse. In other words, even when a reduction in defects is noted in the survey responses as an outcome of reuse, it is not influenced by system type, artifacts or approach. In relationships All_2a and All_2b, a three-way relationship between the reuse artifact of tested products and the outcomes of labor savings and test time is evident. We see in All_2a that as more tested products are selected for reuse artifacts, the less labor is saved and the less reduction in test time is realized. This makes sense since greater effort is associated with more tested products, resulting in an impact to labor savings and test time. Relationship All_3 shows us the more component-based development approach is taken, the less drawings are used for reuse artifacts. This is an example of a relationship that is not likely to be discovered through a simple inspection of survey results and demonstrates the usefulness of PCA. In the context of reuse with embedded and non-embedded systems, this relationship suggests components are selected for reuse on the basis of their maturity, i.e., components consist of implemented designs such that their documentation in engineering drawings is not needed. An interesting three-way relationship between heritage ad/hoc (a development approach), tested clusters (a reuse artifact), and risk reduction (an outcome of reuse) is observed in relationships All_4a and All_4b. We see that the more a heritage/ad hoc development approach is taken, the less tested clusters are reused, and the greater a reduction in risk is realized. When interpreted conversely, this relationship suggests greater use of tested clusters lessens risk reduction, as would less use of a heritage ad hoc approach. These relationships could be useful to practitioners when deciding employ reuse, and specifically for consideration in adopting a heritage/ad hoc approach while avoiding the reuse of tested clusters in the interest of risk reduction. Relationship All_5a shows that taking a model based development approach suggests less reuse of code and confirms the trivial case of greater reuse of models. Additionally an implication that new code is generated from models is evident in All_5b. Relationship All_6 shows that in both embedded and non-embedded systems, there is a lack of dependency between reuse of hardware and any other factors including outcome. All_7 shows us reuse with embedded systems tends toward less reuse of a COTS/GOTS approach. Finally, in All_8 we observe using a product line development approach tends to reduce defects in both embedded and non-embedded systems.

3.2.4.3 Test A, A1, A2, A3 Results

Tests A, A1, A2 and A3 each include survey responses related to the selection of artifacts for reuse. We have results with and without system type in tests A and A3 respectively. Tests A1 and A2 differ by including responses limited to the non-embedded system type (Test A1) and the embedded system type (Test A2). The tests are summarized in Table 3.11. We refer to Table 3.12 when discussing specific survey response relationships.

We proceed with analyzing the results of test A in which only system type and artifacts (with both embedded and non-embedded systems) are included in the PCA. Relationships A_1a and A_1b present associations between three artifacts for reuse: requirements, architecture and tested products. We see that greater reuse of requirements goes hand in hand with reuse of architecture and tested products, when both system types are considered. A_2a and A_2b present additional associations between artifacts: The more code is reused, the less likely hardware and drawings are reused. A_3 indicates less reuse of use cases leads to less reuse of models. A_4 relates non-embedded system types to less reuse of tested clusters.

In test A1 we investigate the relationship between artifacts and system type limited to embedded systems. Relationships A1_1a and A1_1b indicate through a three way relationship among requirements, architecture and use cases that if one tends to reuse use cases, one is also unlikely to reuse architecture and requirements. The A1_2 relation between more use of code and less use of drawings makes sense because engineering drawings have already been implemented and code is therefore less likely to be reused. A reduction in the need to reuse models is evident in relationships A1_3a and A1_3b through more reuse of hardware and tested products. There is no relationship between reduction of defects and reduction of risks in embedded systems.

Test A2 addresses the relationship between artifacts and the non-embedded system type. Relationships A2_1a and A2_1b confirm the results obtained earlier in A_1a and A_1b from Test A, ie that greater reuse of requirements goes hand in hand with reuse of architecture and tested products. Test A2, however, is specific to non-embedded systems so we observe these relationships again, but in the context of only non-embedded systems. Relationships A2_2a and A2_2b indicate less reuse of code is coupled with more reuse of drawings and hardware. These associations suggest the need to redesign components when code is not used or is not available. In relationships A2_3a and A2_3b we observe less reuse of use cases goes hand in hand with less reuse of models and tested clusters. If we observe types of artifacts aligned to life cycle phases, less reuse of use cases in early life cycle phases might lead to less reuse of more developed artifacts such as models and tested clusters.

In test A3 we look at survey responses related to artifacts only, regardless of system type. The same relationships between reuse of requirements, architecture and tested products found in relationships A3_1a and A3_1b in this test are found in test A where survey responses for system type are included in the PCA. This indicates these relationships may exist independent of system type. In A3_2a and A3_2b we also find a repeat of relationships, this time with Test A2. In both tests A2 for reuse artifacts in which only the non-embedded system type is considered, and A3 in where artifacts are considered regardless of system type, we see that less use of code goes hand in hand with more reuse of drawings and hardware. Through comparison of test A2 and A3, these same relationships seem to be independent of system type. A3_3, the last relationship in test A3, confirms what is found in relationships A_3 and A2_3a in tests A and A2 respectively. We again observe that reusing fewer use cases suggests reusing fewer models. Relationship A2_3a with the embedded system type differs from A_3 and A2_3a with both system types. From this comparison we observe system type may not influence the relationship between the reuse of models and use cases.

3.2.4.4 Test B, B1, B2, B3 Results

We turn out attention to an analysis of Tests B, B1, B2 and B3 which each include survey responses related to the selection of development approaches. We have results with and without system type in tests B and B3 respectively. Tests B1 and B2 differ by including responses limited to the non-embedded system type (Test B1) and the embedded system type (Test B2).

We start with test B where a PCA is performed on survey responses with development approaches only, for both system types. We observe the first two relationships B_1 and B_2 in which two pairs of development approaches are related. We see in B_1 that a component based approach seems to go hand in hand with a model based approach, where in B_2 selection of a product line based approach relates to less reuse of a COTS/GOTS approach. B_3 includes the non-embedded system type which suggests a tendency to use less heritage/ad hoc development.

In test B1 development approaches are investigated limited to the embedded system type. For this test we see where embedded system types speak against using both the component based and COTS/GOTS approaches, identified in relationship B1_1. We also see in B1_2 that selection of a development approach based on product line tends toward less use of a heritage ad/hoc approach. Relationship B1_3 indicates less use of a model based approach varies independent of other development approaches when analyzed among the embedded system type.

Test B2 differs from B1 discussed above only through its limitation in this case to non-embedded systems. As with B1, the first relationship associates the product line approach with another development approach, this time with the model based approach. Here we see where the non-embedded system types speak against using the use of a model based approach in relationship B2_1. Relationship B2_2 indicates no association between a COTS/GOTS approach and other development approaches for the non-embedded system type. Relationship B2_3 exposes a major difference between embedded and non-embedded systems. Specifically, in test B2 we see that less use of a product line based approach tends toward less heritage/ad hoc based development in relationship B2_3, where in B1_2 the tendency is toward more use of heritage ad hoc development.

We conclude our analysis of the PCA results of development approaches with test B3. This test is comprised of only survey responses for development approaches with both embedded and non embedded system considered. The first relationship in test B3 identified as B3_1, associates less use of a component base development approach with less use of a model based approach. We observed this same relationship in test B2 where survey responses related to development approaches are analyzed in the context of non-embedded systems. While the relationship is the same, its presence now with both system types suggests this relationship is not necessarily influenced by system type. Similarly, relationship B3_2 where there is no association between a COTS/GOTS approach and other development approaches is the same as B2_2. Again, with the difference being the non-embedded system type in test B2 and both system types in B3, we observe this relationship may not be influenced by system type. Relationship B3_3 exposes another difference between embedded and non-embedded systems. In B3_3 which analyzes development approaches independent of system type, we see that use of a product line approach tends toward more use of heritage/ad hoc development. This relationship differs from other tests in that with relationship B1_2, selection of a product line development approach tends toward less reuse of heritage/ad hoc. In contrast, relationship B3_3 shows more use of heritage/ad hoc with a product line approach. Test B1 analyzes development approaches inclusive of survey responses with both system types, where test B3 excludes system type. We therefore observe the embedded system type influences the relationship between product line and heritage ad/hoc development approaches. Secondly, relationships B3_3 and B2_3 are opposite. Relationships B3_3 and B2_3 are complimentary, and therefore illustrate a contrast with B1_2. We now proceed with an analysis of the final set of tests performed.

3.2.4.5 Test C, C1, C2, C3 Results

Our analysis of the next set of tests focuses on combinations of system type and outcomes as shown in Table 3.11. Test C consists of survey responses with both system types and outcomes. This test produces relationship C_1a and C_1b in which the outcomes of labor savings, test time reduction and the number of items to be tested are associated together as indicated in the survey responses. Specifically, more labor savings relates to a greater reduction in test time and the number of items to be tested. This confirms what we strive for in practice. Interestingly, however, we see in relationship C_2 that outcomes are not influenced by reuse in non-embedded systems. Relationship C_3 indicates another confirmation of desired behavior in which fewer defects reduce risk.

We analyze the results of test C1 where outcomes are included only for the embedded system type. We see in relationships C1_1a and C1_1b an association between less labor savings, less reduction in test time, and less reduction in the number of items to be tested. Stated differently, for the embedded system type, if more items need to be tested there is less reduction in test time and therefore less labor saved. Relationships C1_2 and C1_3 tell us for embedded systems, when reuse does little to reduce defects or risk, this does not correlate to any of the outcomes of defect reduction, a decrease in test time and the number of items to be tested, and a reduction of risk.

In contrast to C1, test C2 addresses outcomes for the non-embedded system type only. In relationship C1_1a and C1_1b we observe for non embedded systems, when fewer items need to be tested, test time is reduced. This relationship is consistent with our expectations. Similar to relationships C1_2 and C1_3, we see in C2_3 that labor savings is not influenced by non-embedded systems. C2_3 shows us yet another difference between embedded and non-embedded systems. In this case we observe a relationship between defect reduction and risk. With non-embedded systems, more defect reduction relates to more reduction in risks. In contrast, risk reduction with embedded systems is not related to other outcomes.

Our final analysis looks at outcomes only. In test C3 there are three relationships. In C31 we again see a relationship between a reduction in test time and fewer items to be tested. We saw this relationship in test C2 for non-embedded systems and in test C for both system types. In addition to confirming desired behavior, we conclude the benefits of less test time with fewer items to be tested is evident irrespective of system type. Test C3 also produces relationship C3_2, a disassociation between risk reduction and outcomes as also seen in C1_3. With the exclusion of system type in test C3 and the limitation of embedded system type only in test C1, we observe that system type may not influence risk reduction. Finally, relationship C3_3 shows us a somewhat counterintuitive relationship between the outcomes of greater labor savings with less defect reduction. Stated differently, when outcomes are analyzed independently, the less defects are reduced the more labor is saved. This is counterintuitive from the perspective of saving time and money in the presence of more defects. However one could argue that there is a cost to reducing defects and a cost avoidance when not spending effort on defect reduction. Perhaps some defects may not be worth removing.

3.2.4.6 Summary of common relationships

In a different aspect of the analysis of our results, we inspect all PCA relationships in terms of their frequency of occurrence and discuss those which appear in two or more tests. We discuss these relationships next.

In Table 3.13 four relationships are common with three tests and eight are common with two tests. Three relationships have tests A, A2 and A3 in common (architecture to tested products, models to use cases, and requirements to architecture and tested products). These are tests which analyze survey results for responses containing both system type, artifacts, and the non-embedded system type with artifacts. None of these three tests includes only the embedded system type. This indicates differences between embedded and non-embedded systems, with relationships between architecture, tested products and requirements, and separately, between models and use cases. A fourth relationship in Table 3.13 is shared with tests C, C2 and C3 (test time to items to be tested). These three tests address system type and outcome but none includes only the embedded system type. Thus we also find a difference between embedded and non-embedded systems in test C and C3 in terms of reduction in test time and items to be tested. One relationship in Table 3.13 is shared among tests A and A1 (code to drawings and hardware). These tests differ only in the restriction to the embedded system type in A1. Since the relationship between code, drawings and hardware appears in test A, which includes both system types, and in test A1, which includes only the embedded system type, we conclude system type does not influence the relationship between code, drawings and hardware. Similarly, there are two relationships that are each common to tests A2 and A3 (drawings to hardware and code to drawings and hardware). These two test differ only in the restriction to the non-embedded system type in A2. From this we conclude that system type does not influence choices in drawings, hardware and code as artifacts in reuse. Tests A and C1 share one relationship (labor savings to test time). Since tests A and C1 are disjoint (test A includes system type and artifacts, and test C includes only outcomes with the embedded system type), we cannot make any meaningful conclusions about any factors that influence labor savings and test time from these tests. Three relationships are shared among test B, B1, B2 and B3, which each address system type and development approach. One relationship is shared by B and B1. B analyzes system type with reuse approaches while B1 limits survey inputs to the embedded system type along with reuse approaches. Since the relationship is found in test B, where both system types are included, and in B1 where only the embedded system type is included, we conclude that system type is not necessarily related to product line or COTS/GOTS development approaches. Tests B2 and B3 differ in the restriction to the non-embedded system type in B2. From this we conclude system type is not necessarily associated with the component based and model based development approaches. Tests C1 and C2 which share one relationship (defect reduction to risk reduction) differ by the exclusion of the non-embedded system type in C1 and the exclusion of the embedded system type in C2. Since C1 and C2 differ by system type, we conclude the relationship between defect reduction and risk reduction is independent of system type.

Table 3.13: PCA duplicate relationship summary

Relationship	Tests
Greater reuse of architecture varies with more reuse of tested products	A, A2, A3
Less reuse of models varies with less reuse of use cases	A, A2, A3
Greater reuse of requirements varies with more reuse of architecture and tested products	A, A2, A3
Greater reuse of code varies with less reuse of drawings and hardware	A , A1
Greater reuse of drawings varies with more reuse of hardware	A2, A3
Less reuse of code varies with more reuse of drawings and hardware	A2, A3
A decrease in labor savings varies with less reduction in test time	A, C1
Greater use of a product line based approach varies with less use COTS/GOTS based approach	B, B1
Less use of a component based approach varies with less use of model based approach	B2, B3
Less use of a COTS/GOTS based approach varies independently	B2, B3
Greater reduction in test time varies with greater reduction in items to be tested	C, C2, C3
Greater defect reduction varies with more risk reduction	C1, C2

3.2.5 Lessons Learned and Conclusions

The tests in Table 3.11 are designed to discover relationships between system type, development approach, artifacts selected for reuse, and outcomes. These tests map directly to RQ2-1, RQ2-2, and RQ2-3. We summarize our PCA findings by listing the differences between system type in terms of RQ1 (development approach), RQ2 (artifacts) and RQ3 (outcomes). We then discuss the similarities between system types, found through PCA. We conclude with a summary of the findings between system types. The differences between embedded and nonembedded systems found through PCA are:

- With embedded systems, use of a product line approach goes hand in hand with the heritage/ad hoc approach.
- With nonembedded systems, use of a product line approach avoids use of a heritage/ad hoc approach. This difference relates to RQ1.
- In our PCA, we did not discover differences between embedded and nonembedded systems in terms of artifacts used. This relates to RQ2.
- With embedded systems, risk reduction is not related to any other outcomes.
- For nonembedded systems, risk reduction goes hand in hand with defect reduction. This difference relates to RQ3.

The similarities between embedded and nonembedded systems discovered using PCA are:

- Use of component based, model based and COTS/GOTS development approaches do not seem to be associated with either system type. This relates to RQ1.
- The selection of artifacts does not seem to be associated with system type. This relates to RQ2.
- Labor savings, reduction in test time and reduction in the number of items to be tested is evidently not associated with system type. This relates to RQ3.

The selection of PCA in the analysis of the survey response data is motivated by an interest in relationships between survey responses which are not easily discovered through conventional analytical methods. The results from PCA suggested that the selection of embedded vs non-embedded system type does not influence outcomes such as an increase or reduction in risk, or an increase or reduction in test time. These conclusions cautiously, recognizing opportunities for future research in which a greater volume of survey data is sought for a replicated study.

PCA is used in Pilot studies 1 and 2 to demonstrate success with finding recommendations for help desk improvements. PCA is a process which accounts for the statistical variance between data. The data used in these pilot studies, help desk incidents, also includes a time component. For each incident submitted, the date and time of submission are recorded. We can use this to our advantage by applying techniques which require a time component, namely SRGMs. In pilot study 3 we turn to SRGMs to increase the scope of research in help desk operations.

3.3 Pilot 3: Software Reliability Growth Models

3.3.1 Summary

In Pilot study 3, reliability models are incorporated into the research strategy for help desk process improvements. Incident prediction is addressed as a step toward help desk operational improvements. Reliability models are derived from historical incident data and are used to predict incidents. In the case study conducted for Pilot study 3, software reliability growth models (SRGMs) and their applicability to improvement processes are investigated. A model selection process is proposed and evaluated. Its success is demonstrated using real help desk incident data from eighteen desktop software applications. The results of the pilot study show direct applicability to meeting cost challenges in IT help desk operations.

3.3.2 Purpose

Pilot study 3 compliments research related to help desk operational improvements by fitting incident data to reliability models in order to predict future incidents. The purpose of investigating reliability models and their applicability to help desk data is lies in the business value of predicting help desk workload. Throughout this research we operation on an assumption that incidents drive help desk costs through labor necessary to resolve them. This assumption, along with the business value of incident prediction, is validated through discussions with help desk managers and through empirical evaluation of actual incident data. The application of software reliability models has been investigated for the purpose of defect prediction. Research conducted by Stringfellow and Andrews [105], and replicated by Andersson [7] demonstrates success with a model selection process in software development environments. Another purpose of Pilot study 3 is to evaluate the model selection process for its applicability to IT help desk incidents.

3.3.3 Scope

This pilot study is scoped to an analysis of 18 desktop software products used in the same organization studied in Pilot 1. Specifically, we selected the products from a much larger set based on the number of computers on which the products are installed (i.e., their distribution in the organization), and on the number of incidents associated with failure of the products. The 18 products have similar distributions and incident volume and represent typical products used by most employees in the organization.

3.3.4 Approach

We present a case study of IT help desk operations in a large multi-national aerospace company. In our approach we investigate how software reliability growth models (SRGMs) can be applied to cumulative help desk incidents for software product failures to predict incident volume. Due to the diverse line of products and services offered by the company in our case study, over 800 desktop products are installed on computers across a population of approximately 100,000 employees. Almost all employees have a dedicated computer for their use. There are over 1,000,000 unique software installations on the employee computers. Some common products are installed company-wide and others are installed on a limited number of machines used for a specific purpose such as high-end graphics. A single enterprise help desk is accessible to all employees by several ways (web, telephone, chat) to report all types of failures and service requests related to IT. Because desktop product reliability is of interest in this research, we limit the scope of our investigation to incident types related to desktop software failures. We exclude incidents submitted for software requests, password resets, resolution of hardware failures and general IT assistance. The data used in this pilot study is described in Appendix B.

On a typical workday, hundreds of incidents are opened at the help desk in response to reports of desktop software failures. Enterprise help desk management software is used to control and document incident generation including mapping of incidents to products, the assignment of date and time stamps when an incident is generated, updates when text is added to record the status of troubleshooting efforts, and problem resolution. Each incident contains a fixed number of attributes. A text or numerical field is associated with each attribute. All incidents are maintained as historical records for post-resolution analysis, root cause identification, trending and other aspects of data analytics. A query-based tool is provided to administrative users to obtain lists of incidents which match selection criteria. In our research we use this tool to obtain incidents written against a set of software products of interest. The tool returns the list of incidents, each with the product against which the incident was written and the date and time at which the incident was opened. In this case study we count the accumulation of incidents on the basis of the date and time at which the incident is submitted to the help desk, grouped by one week intervals. From this data we are able to determine cumulative incident volume by product over a time period of interest.

From the initial set of 156 products selected according to the process described in Appendix B, we chose incident datasets for failures associated with 18 desktop software products shown in Table Table 3.14. Our motivation for product selection is based three attributes explained below: 1) product distribution, 2) product incident volume, and 3) the product's distribution and incident volume relative to that of the most typical product in the help desk database. Product distribution is measured in terms of how many active computers have the product installed. For example, a distribution of 6492 for product ID A39 as

Table 3.14: Desktop software products

Product ID	Distribution	Tot Inc	Duration (weeks)
A39	6492	1060	177
A56	6893	305	174
A123	8355	822	137
A165	6394	70	98
A206	5208	35	118
A219	4704	114	178
A228	7570	65	138
A255	5547	351	155
A310	5721	61	164
A373	7767	98	137
A417	4813	49	178
A450	5234	134	171
A459	8284	11	68
A466	6339	27	172
A488	6712	194	180
A495	4732	87	155
A501	5609	236	100
A514	4863	60	175

shown in Table 3.14 means A39 is installed on 6492 active computers in the company, and has remained installed over a specified time period. The range of product distribution in the help desk database is between 1 and 135,000. Incident volume is measured in terms of the number of incidents submitted against the product over the specified time period. Product A39 in Table 3.14 has 1060 incidents.

An incident is uniquely different from another through an alphanumeric identifier automatically assigned when the incident is generated. The range of incident volume in the help desk database is between 1 and 19,500. In order to select a subset of products for our case study, we looked for products with similar distributions and incident volumes, but which are also close to those of the more typical products. A scatter plot of products based on their distribution and incident volume is included in Figure B.1a in Appendix B. The scatter plot shows most products are clustered in the range of less than 20,000 installations and less than 2000 incidents. We define the product whose behavior is the most typical in terms of distribution and number of incidents as the product p_0. In our analysis we select p_0 and the 17 closest products for a total of 18. We label the set of products against which SRGMs are applied A as in Equation B.0.1 in Appendix B.

Managers of help desk operations are challenged with meeting customer affordability constraints while delivering optimal IT service. Predicting costs must include not only estimations of labor, but projections of the cost of building in product reliability [85]. Software reliability growth models (SRGMs) can be used to predict product quality. Practical applications of SRGMs are investigated by Stringfellow and Andrews [105] and by Andersson [7] where a process for the selection of SRGMs is presented in the context of release decisions. In this case study we investigate the generalizability of the Stringfellow et al. approach through application to an IT help desk domain.

Equations and characteristics of the four SRGMs selected for investigation in this case study are shown in Table 3.15. We select these four SRGMs based on our goal to evaluate the generalizability of the research conducted by Stringfellow and Andrews [105] [7]. The equation $u(t)$ for each model expresses the expected number of incidents at time t. All

Table 3.15: SRGMs investigated

SRGM	Equation $u(t)$
G-O [38]	$a(1-e^{-bt}), a \geq 0, b > 0$
Delayed S-shaped [133]	$a(1-(1+bt)e^{-bt}), a \geq 0, b > 0$
Gompertz [65]	$a\left(b^{c^t}\right), a \geq 0, 0 \leq b \leq 1, c > 0$
Yamada [134]	$a\left(1-e^{-bc\left(1-e^{(-dt)}\right)}\right), a \geq 0, bc, d > 0$

SRGMs applied in this case study have asymptotic behavior. Mathematically, this behavior is expressed as a boundary condition $u(\infty) = a$, where a is the theoretical maximum number of incidents that would eventually be reached. In practice, the quantity of incidents

Table 3.16: Prediction error at ten week prediction time frame for converging models not rejected at $R = 0.95$

Prod ID	Model	Calc rem	Actual	Error	Rel Error
A39	D-S	924	895	29	0.032
A56	G-O	262	259	3	0.012
A123	D-S	575	569	6	0.011
A165	G-O	58	58	0	0.000
A206	G-O	25	27	-2	-0.074
A219	D-S	87	87	0	0.000
A255	Gom	339	343	-4	-0.012
A373	D-S	76	75	1	0.013
A417	G-O	35	38	-3	-0.079
A450	G-O	80	81	-1	-0.012
A466	G-O	21	21	0	0.000
A488	D-S	122	121	1	0.008
	GOM	122	121	1	0.008

would approach this number given sufficient time over the life of the product. This behavior is attributed to a decrease in the rate of incident submission as reported problems are fixed. Note that it is possible that incident rates increase, due to product updates, infrastructure changes, etc. What this means is that we may need to collect more incident data before we can estimate (i.e. the model may not converge at this point).

We selected a commercially available curve fitting tool which uses non-linear regression to determine convergence (or non-convergence) of a set of discrete data points to a function. The tool calculates a Goodness of Fit (GOF) measurement to express the degree to which the data points fit the function. It produces numerical values for all parameters in the models. The tool allows us to capture the GOF (R-value), a prediction of the number of incidents for the next time period, and the value of the a parameter for each model, for data points at which there is convergence. Recall the a parameter is an estimate of the total number of incidents that would eventually be submitted. Tables 3.17, 3.18 and 3.19 show the results of curve fitting for cumulative incident data in the final weeks for product IDs A488, A39 and A459 respectively. For each of these products, preferred models can be

Table 3.17: Predicted total number of incidents for product A488 (R=0.95)

Week	Incidents	G-0		Delayed S		Gompertz (*)		Yamada	
		Estimate	R-value	Estimate	R-value	Estimate	R-value	Estimate	R-value
172	113	6656	0.9830	207	0.9953	175	0.9977	–	–
173	114	6621	0.9832	207	0.9953	174	0.9978	–	–
174	115	6591	0.9834	206	0.9954	173	0.9978	–	–
175	115	6552	0.9837	206	0.9955	173	0.9978	–	–
176	116	6518	0.9839	205	0.9955	172	0.9979	–	–
177	117	6489	0.9841	205	0.9956	172	0.9979	–	–
178	117	6451	0.9844	204	0.9957	171	0.9979	–	–
179	117	6406	0.9846	204	0.9957	171	0.9979	–	–
180	119	6378	0.9848	203	0.9958	170	0.9980	–	–

identified on the basis of GOF (R-value) or by using additional restrictive thresholds (best GOF, for example). These models are designated as *Selected* in Tables 3.17, 3.18. For product ID A459 in Table 3.19, no model is selected since the R-values are mostly under the 0.95 threshold. Table 3.16 shows the calculated number of cumulative defects (*Calc rem*) ten weeks into the future. This number is obtained through the function $f_{w0}(t)$ derived by applying the models to the dataset consisting of the first 60% of cumulative incident data collected from the 18 products listed in Table 3.14. We follow the approach taken by Stringfellow and Andrews [105] in using the 60% cumulative data threshold. The (*Actual*) column shows the actual number of cumulative defects known from our dataset at ten weeks past $w0$. We calculate the prediction error and relative error by using (*Calc rem*) as the estimated number of incidents, and (*Actual*) as the actual number. We include in Table 3.16 the model which returns the lowest magnitude of relative error for each product ID. In the case of A488, the relative error determined by two models was the same.

We discussed a selection method to be used in determining which SRGM is the best predictor for incidents in the products we investigate. We propose the approach in Figure 3.5 which applies an analysis of three rejection criteria: 1) does the model converge, 2) is the fit good enough, and 3) is the $estimate$ greater than the $actual$. Our business decision

Table 3.18: Predicted total number of incidents for product A39 (R=0.95)

Week	Incidents	G-0		Delayed S		Gompertz *(*)*		Yamada	
		Estimate	R-value	Estimate	R-value	Estimate	R-value	Estimate	R-value
169	883	3911	0.9739	1056	0.9907	874	0.9968	–	–
170	886	3729	0.9740	1055	0.9908	875	0.9968	–	–
171	886	3563	0.9740	1053	0.9908	877	0.9968	–	–
172	887	3413	0.9741	1052	0.9909	878	0.9968	–	–
173	889	3278	0.9742	1051	0.9910	880	0.9968	–	–
174	892	3158	0.9743	1049	0.9910	881	0.9968	–	–
175	892	3047	0.9743	1048	0.9911	882	0.9968	–	–
176	892	2943	0.9744	1046	0.9912	884	0.9968	–	–
177	893	2848	0.9744	1045	0.9912	885	0.9968	–	–

Table 3.19: Predicted total number of incidents for product A459 (R=0.95)

Week	Incidents	G-0		Delayed S		Gompertz		Yamada	
		Estimate	R-value	Estimate	R-value	Estimate	R-value	Estimate	R-value
60	9	10	0.9239	7	0.8886	16	0.9250	15	0.9254
61	9	11	0.9257	7	0.8882	22	0.9290	17	0.9270
62	9	12	0.9280	7	0.8889	26	0.9329	18	0.9292
63	10	14	0.9286	7	0.9112	159	0.9361	21	0.9293
64	10	15	0.9299	8	0.8880	339	0.9402	25	0.9307
65	10	17	0.9322	8	0.8885	428	0.9440	28	0.9327
66	10	19	0.9348	9	0.8908	392	0.9473	32	0.9352
67	10	21	0.9375	9	0.8937	282	0.9502	36	0.9378
68	10	23	0.9401	9	0.8968	112	0.9527	39	0.9404

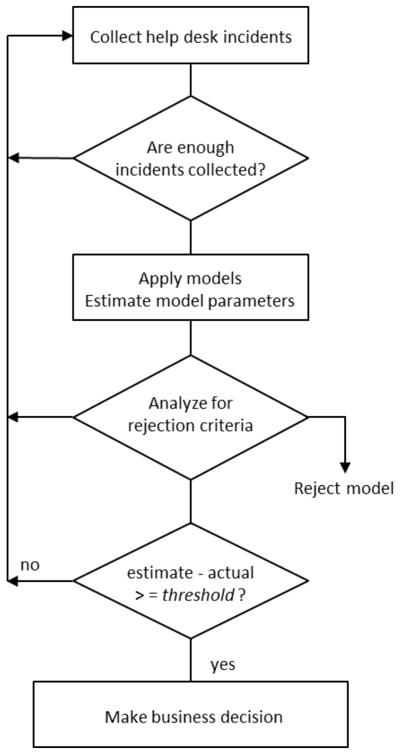

Collect help desk incidents
Are enough incidents collected?
Apply models Estimate model parameters
Analyze for rejection criteria
Reject model
no
estimate - actual > = threshold ?
yes
Make business decision

Figure 3.5: Flowchart for approach

is made on the basis of relative error between predicted number of incidents and the actual number. In our case, the threshold is defined by the model for which the lowest magnitude of relative error is produced from the set of models which are not rejected through the rejection criteria. We model our approach after Stringfellow and Andrews [105]. The results of applying the model selection criteria in Figure 3.5 are shown in Tables 3.20 and 3.21. For $R \geq 0.95$, the less restrictive GOF threshold, Table 3.20 shows the SRGMs most appropriate for the applications listed, based on our approach and model rejection criteria. We note the Gompertz model outperforms the Delayed S-shaped and G-O models in terms of the number of applications for which we determine model suitability. Table 3.21 shows the Gompertz model is the only favorable SRGM. The fewer number of applications in Table 3.21 is explained through the more restrictive GOF. A threshold of 0.99 rejects more curve fits than a threshold of 0.95.

Table 3.20: Results of applying model selection approach using $R = 0.95$

Model	Product ID
G-O	A39, A206, A488
Delayed S-shaped	A39, A56, A65, A219, A488
Gompertz	A39, A123, A255, A373, A450, A466, A488
Yamada	none

Table 3.21: Results of applying model selection approach using $R = 0.99$

Model	Product ID
G-O	none
Delayed S-shaped	A39, A488
Gompertz	A39, A123, A255, A373, A450, A488
Yamada	none

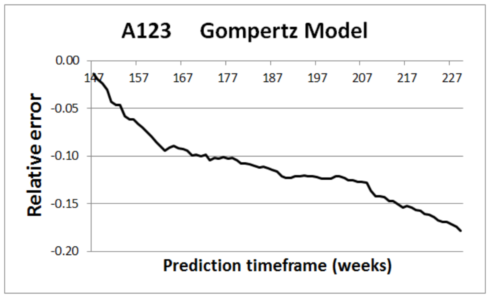


Figure 3.6: SRGM prediction behavior as a function of prediction time frame for product A123 using the Gompertz model

In Table 3.16 we based our analysis on a ten week prediction time frame for all products and all models for which there is convergence in the curve fit at a threshold of $R = 0.95$. We selected the prediction time frame based on business practices when projecting staffing levels at the help desk. Selecting a time frame too soon into the future would require projections to be made more often than necessary. Projecting too far out adds risk in the projections. While a ten week prediction time frame fits our current business practices, we are interested in prediction performance of the models at time frames well beyond ten weeks. We select an approach which investigates relative error of a model's prediction as a function of prediction time frame. In other words, how does prediction error behave across a much larger time frame, instead of only the single ten week time frame influenced by current business practices. Relative error shows how large the error is in relation to the correct value, and can be positive, negative, or zero. A relative error with magnitude close to zero indicates an incident volume prediction close to the actual number of incidents. A view of the Gompertz model in Figure 3.6 shows an the best prediction time frame at ten weeks or more is at week 147, where the magnitude of the relative error is the smallest. The results in this process example suggest predictions from a ten week look-ahead for staffing decisions cannot be improved through selection of a time frame greater than ten weeks. In a business context, the confidence of staffing predictions based on model prediction performance diminishes after the ten week interval. Space limitations preclude additional plots for the other products we investigated.

We conclude our analysis by revisiting the research question *Can an SRGM selection approach be used to estimate IT help desk incidents?* The results of our case study suggest confidence in incident prediction performance with the Gompertz model. The G-O and Delayed S-shaped models are promising, but to a slightly lesser degree than with the Gompertz model using both GOF criteria. The Yamada model must be rejected for incident prediction based on this case study. The G-O, Delayed S-shaped and Gompertz models show strength in incident prediction on the basis of their ability to converge to our case study data sets, but respective of our tolerance for goodness of fit. As with an analysis of prediction ability, the Yamada model did not show us sufficient performance with convergence to build confidence in it as a tool for predicting desktop software quality. Our model selection approach is shown to achieve our goal in selecting a suitable SRGM, reflective of the design of the model and its criteria. We establish relative error as a decision threshold, but demonstrate prediction time frames influence relative error in our selection process. In summary, our approach and results lead us to methodologies which have potential for assisting with IT help desk operations.

3.3.5 Lessons Learned and Conclusions

Four software reliability growth models are investigated in this Pilot study 3 for their applicability to help desk incidents. In the literature we see applications of SRGMs to defect data collected during software testing, often in the interest of evaluating post-release defect prediction accuracy from historical defect data. Our case study is a departure from this approach in that we focus on help desk incidents reported for the resolution of problems with deployed software products. Our assumptions of using incident submission time as a means by which cumulative incident data is collected is consistent with the use of calendar time (vs. execution time) in related studies of SRGMs. Our results demonstrate three out of four reliability models show usefulness in terms of their goodness of curve fit to cumulative help desk data for desktop product software failures. Additionally, we demonstrate success with a model selection framework which assists us with determining a suitable model on the basis of incident prediction error. These results and conclusions are portable to IT help desk operations. Investigation into additional products and product families will likely broaden our outlook on the applicability of these results to IT operations.

3.4 Pilot 4: PCA and SRGMs with Extended Product Set

3.4.1 Summary

In this last of four pilot studies we investigate software reliability models and their applicability to process improvement at an IT help desk, using an extended product set. In Pilot studies 1 and 3 a small number of products was investigated for the purpose of finding process improvements with incident data (Pilot 1) and predicting incidents (Pilot 3). In Pilot study 4, an extended SRGM selection framework built on the selection process described in Pilot 3 is evaluated using real help desk incident data from a portfolio of 156 desktop software applications. Through the use of a much larger set of products, we show success with the scalability of techniques used in Pilots 1 and 3. Help desk incidents are predicted for the portfolio at five prediction intervals and measured against actual numbers of submitted incidents. Incident prediction accuracy is analyzed, and the trend in accuracy based how far into the future incidents are predicted is evaluated. The results demonstrate a model selection framework can assist with predicting the number of incidents that will be submitted to a help desk for a large portfolio of products. The level of accuracy reported in this industry-based case study establishes the proposed estimation technique and reliability model selection framework as novel research in software engineering. Additionally, its practical uses are applicable to help desk process improvement efforts.

3.4.2 Purpose

Pilot study 4 compliments the research strategy of this thesis by demonstrating incident prediction accuracy with a large product portfolio. Successful demonstration of this approach establishes the first of three incident prediction techniques, two of which are presented later in the Large Scale Approach section of this thesis. Discussions with help desk managers at the company in which Pilot studies 1, 3 and 4 are conducted indicate operations are typically managed and executed at a system level rather than at a product level. The help desk provides incident resolution services for a portfolio of products. This establishes motivation to validate incident prediction techniques to a larger set of products than what was selected for the previous pilot studies.

3.4.3 Scope

Pilot study 4 investigates help desk incident predictions in the same organization used with Pilots 1 and 3. Pilot 4 applies the same techniques, but uses incident data from a much larger number of desktop software applications. Rather than selecting a subset of products that are typically used by most employees as was done in Pilot 3, we identify products with a minimum of 50 installations on computers in the company, and a minimum of 50 incidents generated. By applying these thresholds, we identify 156 products from which we gather incident data for analysis.

3.4.4 Approach

Cumulative incident data for each of the 156 products selected according to the process described in Appendix B was obtained from the help desk database. The incident data for each product begins with incidents from April 2008 and goes through a specified month between January 2012 and October 2013. For example, the first dataset for product $A1$ consists of cumulative incident data from April 2008 through January 2012. The second dataset for $A1$ goes from April 2008 through February 2012, and so on. This approach results in 22 datasets for each of the 156 products.

In addition to the actual cumulative monthly incident data for April 2008 through January 2012, we collected incident data for time intervals after January 2012 to use in our validations. We collected the total number of incidents produced by all 156 products in the months of February 2012 through March 2014 for this purpose. Knowing actual incidents for these months allows us to measure prediction accuracy from incident estimates in prior months as part of the validation step.

The process used in this case study predicts incidents by curve fitting help desk incident data to SRGMs. As stated earlier, incident volume drives help desk costs, so a prediction of incidents can help managers make operational decisions. Managers of help desk operations are challenged with meeting customer affordability constraints while delivering optimal IT service. Similar to pilot study 3, practical applications of SRGMs are investigated in this pilot study where a process for the selection of SRGMs is modeled after research conducted by Stringfellow and Andrews [105] and by Andersson [7]. In this pilot study we investigate the generalizability of the Stringfellow et al. approach through application to an IT help desk domain.

Equations and characteristics of the four SRGMs selected for investigation in this pilot study are the same as in pilot study 3, and are shown in Table 3.15. Figure 3.7 shows the incident prediction process used in this case study. We model this process after Stringfellow and Andrews [105]. The steps begin with gathering cumulative incident data for each product through the end of a specified time period, for example through the end of January 2012. The incident data is curve fit to each SRGM, one product at a time. Model parameters are recorded for each SRGM that converges for the product incident data. Models which do not converge are rejected for analysis for the remaining time period. A goodness of curve fit R^2 is reported for each model that converges. The model with an R^2 value closest to 1.0 is selected as the best model for the incident data. Using the model parameters for the best SRGM, incidents are predicted for the next time period t according to the applicable equation in Table 3.15. Additionally, the remaining number of incidents is calculated from the difference between the total number of incidents and the number of incidents produced at time t. Recall, the total number of incidents is the value of the a parameter in the SRGM equations in Table 3.15. After predicting the remaining incidents for the product, the incident prediction process shown in Figure 3.7 is repeated using data for the same time period

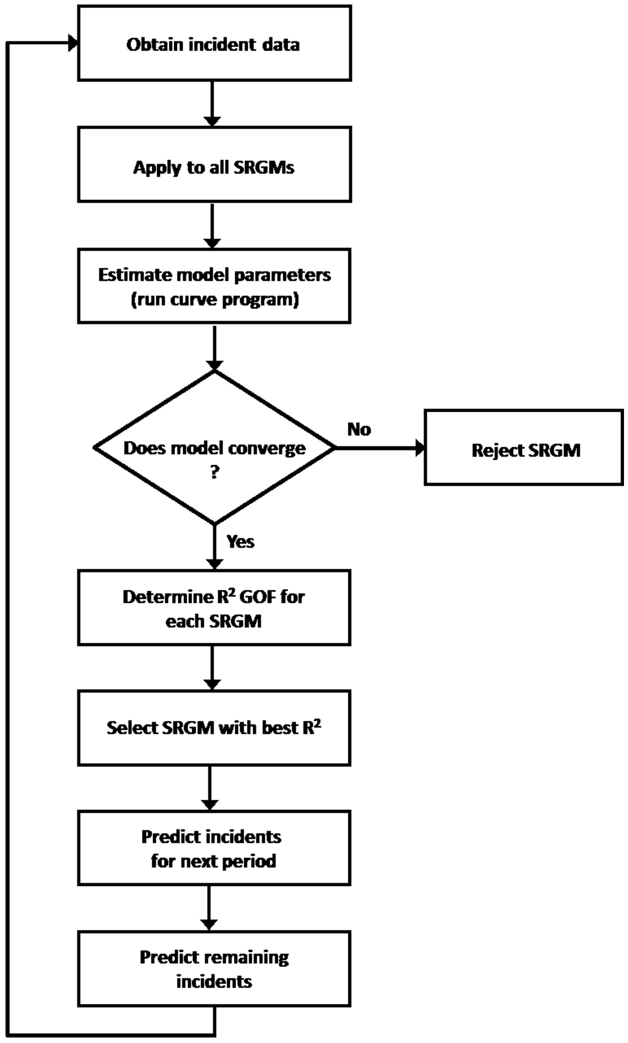

Obtain incident data
Apply to all SRGMs
Estimate model parameters (run curve program)
Does model converge ?
No
Reject SRGM
Yes
Determine R² GOF for each SRGM
Select SRGM with best R²
Predict incidents for next period
Predict remaining incidents

Figure 3.7: Incident prediction process flowchart

(through January 2012 in this example) but for the next product. The steps in the flowchart are followed for each product. At each iteration, all SRGMs in Table 3.15 are evaluated. When the process steps are complete for all products for the current time period, incident data is collected for all products through the next time period (February 2012). The steps are then repeated for all 156 products for the remaining time periods through October 2013.

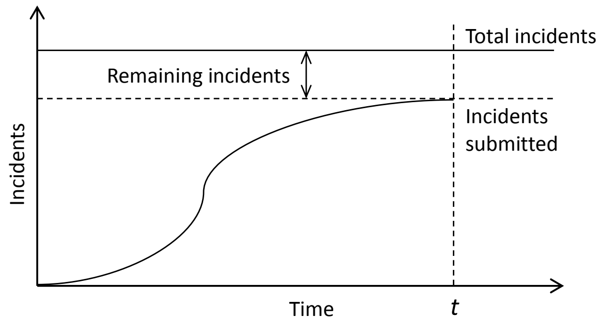


Figure 3.8: Determination of remaining incidents from model plot

Overall, 20,592 curve fits are attempted. For each curve fit, the model with the best SRGM is selected to predict incidents for that product. We designate that model as $u'(t)$ and derive equations for incidents estimations and relative error.

Let $u'_i(t)$ be the curve-fitted model for application A_i $(i = 1 : 156)$ in product set $\mathbf{A}$.

Then $N_{A_i}(t)$, the estimated number of incidents that will be generated by A_i through time period $t \geq 0$, is given by

$$N_{A_i}(t) = u'_i(t) \tag{3.4.1}$$

The estimated total number of incidents $N_A(t)$ to be generated by all 156 applications through t is given by

$$N_A(t) = \sum_{i=1}^{156} N_{A_i}(t) \tag{3.4.2}$$

Let $N_{act}(t)$ be the known number of incidents that are actually produced by all 156 applications in product set $\mathbf{A}$, through t.

Then $RE_A(t)$, the incident prediction relative error at t for all products is given by

$$RE_A(t) = (N_A(t) - N_{act}(t))/N_{act}(t) \tag{3.4.3}$$

Using Equation 3.4.3, incident prediction error is calculated from predictions at 1 month, 2 months, 3 months, 4 months and 5 months beyond the last month in each time period.

Table 3.22: Incident prediction relative error at five prediction intervals, $R^2 \geq 0.95$

	Act. inc	**Predicted incidents (cumulative)**					**Relative error $RE_A(t)$**				
	(cumulative)	**+1 mo**	**+2 mo**	**+3 mo**	**+4 mo**	**+5 mo**	**+1 mo**	**+2 mo**	**+3 mo**	**+4 mo**	**+5 mo**
1/31/2012	**62572**	67595	68754	69881	70982	72059	0.0539	0.0513	0.0483	0.0469	0.0460
2/28/2012	**64135**	68876	70047	71186	72298	73387	0.0532	0.0508	0.0499	0.0495	0.0478
3/31/2012	**65397**	70417	72785	75769	78404	80424	0.0563	0.0735	0.0999	0.1194	0.1297
4/30/2012	**66661**	71661	72903	74154	75438	76787	0.0569	0.0583	0.0587	0.0596	0.0646
5/31/2012	**67804**	72960	74224	75497	76793	78127	0.0591	0.0597	0.0605	0.0647	0.0672
6/30/2012	**68888**	74114	75319	76502	77663	78804	0.0581	0.0580	0.0607	0.0609	0.0628
7/31/2012	**70042**	75335	76492	77626	78739	79832	0.0582	0.0605	0.0604	0.0619	0.0661
8/31/2012	**71192**	76435	77585	78712	79820	80910	0.0597	0.0598	0.0615	0.0660	0.0611
9/30/2012	**72127**	77491	78637	79779	80932	82121	0.0585	0.0605	0.0654	0.0614	0.0529
10/31/2012	**73207**	78615	79727	80815	81883	82934	0.0602	0.0647	0.0599	0.0498	0.0412
11/30/2012	**74149**	79562	80660	81739	82802	83850	0.0625	0.0578	0.0480	0.0396	0.0313
12/31/2012	**74880**	80490	81633	82779	83935	85109	0.0556	0.0466	0.0393	0.0324	0.0259
1/31/2013	**76249**	82016	83195	84476	85926	87628	0.0516	0.0445	0.0390	0.0358	0.0406
2/28/2013	**77995**	83290	84906	86747	88849	91251	0.0457	0.0443	0.0457	0.0551	0.0663
3/31/2013	**79650**	84619	86204	87946	89878	92035	0.0408	0.0392	0.0444	0.0502	0.0617
4/30/2013	**81304**	84890	86118	87400	88752	90189	0.0233	0.0227	0.0213	0.0238	0.0288
5/31/2013	**82956**	89206	90979	92852	94847	96982	0.0593	0.0631	0.0711	0.0819	0.0924
6/30/2013	**84210**	90923	92662	94459	96321	98253	0.0624	0.0689	0.0775	0.0849	0.0928
7/31/2013	**85579**	92894	94756	96681	98674	100737	0.0716	0.0808	0.0890	0.0975	0.1051
8/31/2013	**86685**	93460	95004	96543	98077	99605	0.0661	0.0701	0.0738	0.0759	0.0779
9/30/2013	**87668**	93473	94834	96186	97531	98868	0.0528	0.0548	0.0552	0.0555	0.0557
10/31/2013	**88781**	96081	97493	98880	100240	101573	0.0687	0.0695	0.0701	0.0703	0.0703
11/30/2013	**89908**	-	-	-	-	-	-	-	-	-	-
12/31/2013	**91157**	-	-	-	-	-	-	-	-	-	-
1/31/2014	**92405**	-	-	-	-	-	-	-	-	-	-
2/28/2014	**93654**	-	-	-	-	-	-	-	-	-	-
3/31/2014	**94902**	-	-	-	-	-	-	-	-	-	-

3.4.5 Lessons Learned and Conclusions

We selected a commercially available curve fitting tool which uses non-linear regression to determine convergence (or non-convergence) of a set of discrete data points to a function. In our case the data points are integer values which represent the cumulative number of incidents at each time interval, for a specific product. The tool produces numerical values for all parameters in the models, for models that converge. The tool determines the GOF (R^2 value), and the value of the a parameter for each model as shown in Table 3.15. Recall the a parameter is an estimate of the total number of incidents that would eventually be submitted. The tool also allows us to evaluate the model at future time periods.

We apply the model selection method shown in Figure 3.7 to estimate the values for model parameters. We determine which SRGM has the best curve fit for the current incident data set. This model selection process results in a function $u'(t)$ (Equation 3.4.1) for the selected model, including numerical values for the model parameters of $u'(t)$.

Tables 3.22 and 3.23 show the predicted *cumulative* number of incidents one, two, three, four and five months (labeled as “+1 mo”, “+2 mo”, etc.) from the date in the leftmost

Table 3.23: Incident prediction relative error at five prediction intervals, $R^2 \geq 0.99$

	Act. inc	**Predicted incidents (cumulative)**					**Relative error $RE_A(t)$**				
	(cumulative)	**+1 mo**	**+2 mo**	**+3 mo**	**+4 mo**	**+5 mo**	**+1 mo**	**+2 mo**	**+3 mo**	**+4 mo**	**+5 mo**
1/31/2012	**62572**	65650	66770	67861	68925	69966	0.0236	0.0210	0.0180	0.0165	0.0157
2/28/2012	**64135**	66783	67895	68975	70027	71054	0.0212	0.0185	0.0173	0.0165	0.0144
3/31/2012	**65397**	68231	69366	70477	71567	72639	0.0236	0.0230	0.0231	0.0218	0.0203
4/30/2012	**66661**	68964	70084	71179	72253	73305	0.0171	0.0174	0.0162	0.0149	0.0163
5/31/2012	**67804**	70012	71198	72391	73604	74853	0.0163	0.0165	0.0168	0.0205	0.0225
6/30/2012	**68888**	70976	72087	73172	74232	75268	0.0133	0.0126	0.0145	0.0140	0.0151
7/31/2012	**70042**	73039	74127	75188	76225	77239	0.0259	0.0277	0.0271	0.0280	0.0315
8/31/2012	**71192**	74192	75290	76365	77418	78451	0.0286	0.0285	0.0299	0.0339	0.0289
9/30/2012	**72127**	75155	76229	77280	78311	79323	0.0266	0.0280	0.0321	0.0270	0.0170
10/31/2012	**73207**	75883	76909	77907	78880	79831	0.0234	0.0271	0.0217	0.0113	0.0023
11/30/2012	**74149**	76971	77991	78989	79969	80932	0.0279	0.0228	0.0127	0.0040	-0.0046
12/31/2012	**74880**	77653	78643	79612	80562	81493	0.0184	0.0083	-0.0005	-0.0091	-0.0176
1/31/2013	**76249**	78886	79784	80663	81525	82369	0.0114	0.0017	-0.0079	-0.0173	-0.0219
2/28/2013	**77995**	79009	79876	80735	81586	82429	-0.0080	-0.0176	-0.0268	-0.0312	-0.0368
3/31/2013	**79650**	81017	82313	83715	85251	86947	-0.0035	-0.0078	-0.0059	-0.0038	0.0030
4/30/2013	**81304**	81900	82796	83675	84541	85398	-0.0127	-0.0168	-0.0223	-0.0247	-0.0259
5/31/2013	**82956**	84866	86138	87445	88803	90226	0.0078	0.0065	0.0088	0.0129	0.0163
6/30/2013	**84210**	85684	86810	87927	89040	90152	0.0012	0.0014	0.0030	0.0029	0.0027
7/31/2013	**85579**	87473	88768	90082	91422	92793	0.0091	0.0125	0.0147	0.0168	0.0179
8/31/2013	**86685**	88295	89471	90639	91802	92960	0.0071	0.0078	0.0081	0.0071	0.0060
9/30/2013	**87668**	88344	89369	90381	91383	92377	-0.0049	-0.0060	-0.0085	-0.0111	-0.0136
10/31/2013	**88781**	90951	92092	93209	94305	95379	0.0116	0.0103	0.0087	0.0070	0.0050
11/30/2013	**89908**	-	-	-	-	-	-	-	-	-	-
12/31/2013	**91157**	-	-	-	-	-	-	-	-	-	-
1/31/2014	**92405**	-	-	-	-	-	-	-	-	-	-
2/28/2014	**93654**	-	-	-	-	-	-	-	-	-	-
3/31/2014	**94902**	-	-	-	-	-	-	-	-	-	-

Table 3.24: Incident prediction relative error summary for five prediction intervals

	+1 month			**+2 month**			**+3 month**		
	1st Q	**med**	**3rd Q**	**1st Q**	**med**	**3rd Q**	**1st Q**	**med**	**3rd Q**
R2 >= 0.95	0.0534	0.0582	0.0601	0.0509	0.0590	0.0643	0.0481	0.0602	0.0689
R2 >= 0.99	0.0073	0.0148	0.0236	0.0029	0.0126	0.0224	0.0004	0.0136	0.0178

	+4 month			**+5 month**		
	1st Q	**med**	**3rd Q**	**1st Q**	**med**	**3rd Q**
R2 >= 0.95	0.0496	0.0603	0.0692	0.0465	0.0623	0.0695
R2 >= 0.99	-0.0021	0.0121	0.0167	-0.0029	0.0102	0.0168

column of the tables. Equation 3.4.2 was used to determine the predictions. The predictions are calculated from curve fits with $R^2 \geq 0.95$ in Table 3.22 and from curve fits with $R^2 \geq 0.99$ in Table 3.23. The predictions in both tables are calculated using SRGMs with the best curve fit, by following the process shown in Figure 3.7. Additionally, Tables 3.22 and 3.23 include the actual known (cumulative) number of incidents produced by all 156 products each month. Note that no incident predictions are made from curve fits later than 10/2013. The last five rows in the tables are included only to show the actual incidents for the five months past the last month for which we have cumulative incident data (10/2013). The actual number of incidents and the predicted incidents are used in Equation 3.4.3 to

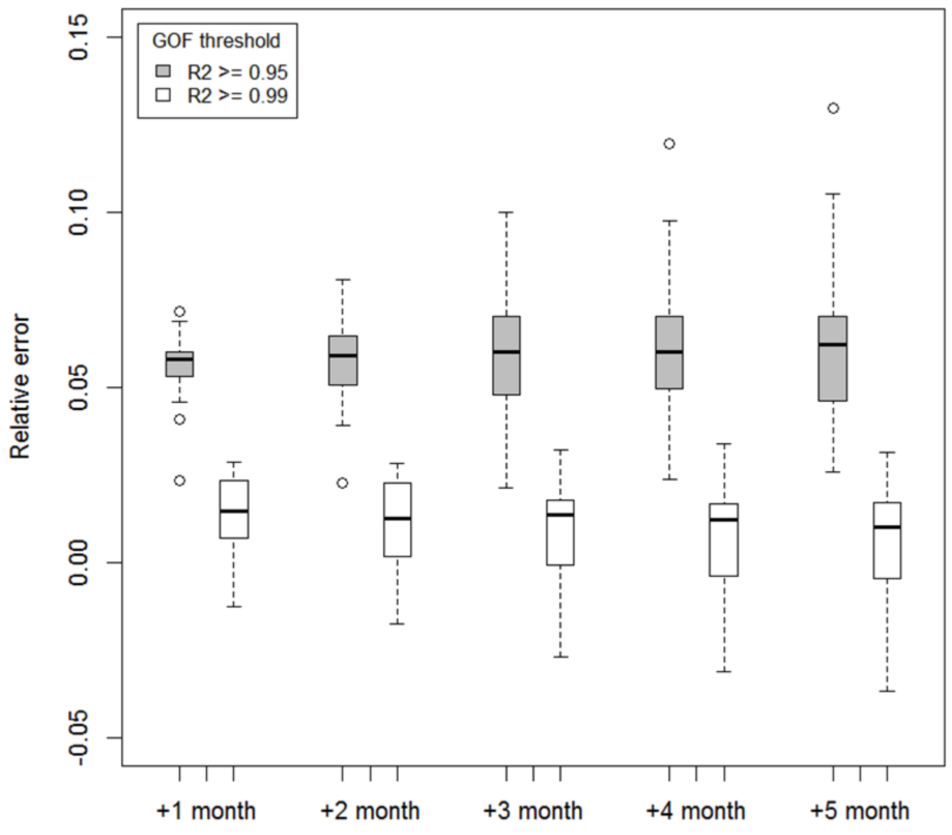


Figure 3.9: Incident prediction relative error at five prediction intervals

calculate incident prediction relative error. The five columns on the right side of Tables 3.22 and 3.23 show the relative error for the one, two, three, four and five month predictions. Box plots showing a visual summary of the relative prediction error in Tables 3.22 and 3.23 are included in Figure 3.9. A numerical summary of incident prediction relative error associated with Figure 3.9 is included in Table 3.24.

We turn our attention to a discussion of the results of this case study through an interpretation of incident prediction accuracy, with a focus on differences in prediction accuracy between the two R^2 thresholds. We discuss 1) median prediction relative error, 2) interquartile range (IQR) of prediction error, 3) trends in median prediction relative as prediction periods increase, and 4) trends in IQR as prediction periods increase. The observed differences for R^2 thresholds of 0.95 and 0.99 are summarized in Table 3.25. Our motivation for selecting these measurements is as follows. Median prediction relative error allows us to see the mid-point in the overall range of relative error, as an indicator of the central tendency of incident prediction error. IQR exposes the spread of prediction error, and is an indicator of prediction stability. Trends in median prediction error and IQR tell us how accurate and stable our incident predictions are for short-term (one month) and long-term (five month) predictions. These prediction time frames are typical of what is used in practice by managers interviewed during this case study.

Starting with median prediction accuracy, we see in Table 3.24 that for $R^2 \geq 0.95$, the median incident prediction error closest to zero (i.e., the most accurate estimate) occurs when predicting incidents one month out. The highest (least accurate) occurs when predicting five months out. This means at the one month prediction interval for $R^2 \geq 0.95$, 50% of the predictions overestimated incidents by at least 5.82%. The other 50% overestimated incidents by at most 5.82%. At the five month prediction interval for $R^2 \geq 0.95$, 50% of the predictions overestimated incidents by at least 6.32% while the other 50% overestimated incidents by at most 6.32%. Continuing with observations of median prediction accuracy, we turn to predictions limited to curve fits with the more restrictive GOF threshold of $R^2 \geq 0.99$. Here we observe the median incident prediction error closest to zero (i.e., the most accurate) occurs when predicting incidents five month out and the highest (least accurate) occurs when predicting only one month out. This is opposite of the median prediction accuracy trend at the less restrictive GOF threshold of $R^2 \geq 0.95$. This seems counterintuitive and is discussed below. We note that median prediction accuracy for all five prediction intervals is better at $R^2 \geq 0.99$, compared to $R^2 \geq 0.95$. For $R^2 \geq 0.99$, 50% of the predictions overestimated incidents by at least 1.48% for one month predictions, and the other 50% overestimated incidents by at most 1.48%. At the five month prediction interval for $R^2 \geq 0.99$, 50% of the predictions overestimated incidents by as little as 1.02%

while the other half overestimated by at most 1.02%. The results from this case study indicate the more restrictive GOF threshold results in more accurate incident predictions in terms of median prediction accuracy for all five prediction periods, compared to the less restrictive GOF threshold. IQR is interpreted as an indicator of incident prediction stability.

Table 3.25: Median incident prediction accuracy and interquartile range (IQR)

		$R^2 \geq 0.95$	$R^2 \geq 0.99$
Median prediction accuracy	best	+1 month	+5 month
	worst	+5 months	+1months
IQR	smallest	+1 month	+1 month
	largest	+5 months	+5 months

The smaller the IQR, the more stable the predictions are. Table 3.24 indicates the smallest IQR (most stable range of predictions) occurs at the one month prediction interval and the largest IQR (least stable predictions) occurs at the five month interval, for both $R^2 \geq 0.95$ and $R^2 \geq 0.99$. These findings confirm a general intuition that prediction accuracy varies more widely the farther out the predictions are made. With median prediction accuracy, similar intuition is confirmed through better accuracy at one month predictions and worse at five month predictions at $R^2 \geq 0.95$. However, achieving slightly better accuracy at *longer* prediction periods for $R^2 \geq 0.99$ seems counterintuitive. Although one might be inclined to infer from these results that better prediction is achieved at longer prediction intervals for $R^2 \geq 0.99$, two additional observations are made. First, the difference in median prediction error between one and five months at $R^2 \geq 0.99$ is 0.46%. (For comparison, the difference in median prediction error between one and five months at $R^2 \geq 0.95$ is 0.41%). These differences in prediction error translate to only a few hundred incidents compared to cumulative incident volumes in the range of 63,000 to 95,000 as shown in Tables 3.22 and 3.23. Secondly, the IQR at five months is approximately twice as large as the one month IQR. This indicates less stability in incident prediction accuracy at five months, compared to stability with one month predictions. Through a combination of prediction stability and accuracy, we are inclined to favor shorter incident predictions at both GOF thresholds, and suggest that appropriate caution is taken with assumptions of similar accuracy for longer predictions.

4 Large Scale Approach

4.1 Summary

Pilot studies 1 through 4 established a foundation for several analytical approaches, the development of incident prediction techniques, and validation of a help desk cost estimation model. Having established success with 1) help desk process improvements and incident prediction based on PCA and SRGMs, 2) generalizability to survey data, and 3) a demonstration of the scalability of the approach to a large set of products, an aggregation of the techniques is applied in a large scale approach in the next section. In this approach, incidents predictions are related to help desk costs through the development and validation of a cost model. A combined use of the techniques successfully demonstrated at a smaller scale in the pilot studies is shown to predict incidents and help desk costs accurately. Additionally, process efficiency is demonstrated through applying the techniques to clusters of products rather than to all products in the large scale approach.

Help Desk operations at a large company can be expensive. Software down time causes revenue loss [103]. Costs associated with IT operations present challenges to profit goals. Minimizing software failures in an operational environment is important to customer experience, but more practically to the cost model for offering IT services. Two costs are associated with software failures. First, short-term losses are realized through missed commitments in daily business rhythms when software resources become nonfunctional or inaccessible to employees. Longer term effects are unmet contractual obligations which can lead to financial penalties or contract loss. Secondly, there are costs associated with the effort to resolve problems by help desk technicians and to manage resolution processes by help desk managers. Workloads at the help desk are dynamic. There is a need for efficient resolution to minimize employee wait time for help desk personnel to begin to work on problems. Employing a sufficiently large team of help desk technicians to handle surges in problems addresses the demand but results in idle staff during periods of nominal help desk operations. Excess staff results in unnecessary costs when there is no backlog of problems to resolve. Conversely, employing too few technicians results in delays with problem resolution due to backlogs. This drives the cost of unproductive employees while problems are being resolved. It would thus be helpful to find a way to predict incidents and the cost to resolve them. Help desk managers need a way to plan staffing levels so that labor costs are minimized while problems are resolved efficiently. An incident prediction method would be useful to help desk managers for planning staffing levels. The potential value of a solution to this problem is important to an IT service provider since software failures are inevitable and their timing is difficult to predict. In addition to assisting with labor predictions for existing operations, a cost prediction model would be a valuable tool in the preparation of bids for additional business through demonstration of management efficiency by presenting evidence of historically accurate labor predictions.

This large scale approach presents a cost model that is derived from incident reports and labor data from a large help desk operation in an industrial setting. The approach uses real help desk incident data and actual labor costs collected over a four and a half year period. A cost model derived from a database of resolved incidents submitted against the full scope of problem types encountered by the help desk in presented in this case study. Through the development and validation of these techniques, the following research questions are addressed:

- RQ1 Can desktop software product reliability data obtained from help desk incidents be used to predict future incident volume?

- RQ2 Can incident prediction accuracy be obtained through the selection of product clusters vs. analyzing a full product portfolio in a prediction model?
- RQ3 Can incident resolution labor data be used to develop a help desk cost prediction model?

4.2 Purpose

The goal of the large scale approach is to develop and validate a model to predict incidents and cost (in terms of effort) for help desk operations. An objective in the development of the cost model is to determine clusters of products which produce incidents, and to investigate whether or not a representative product in each cluster can be used to predict future incidents for all cluster members rather than working with a larger set of products. The large scale approach aggregates techniques from the pilot studies into one case study. By showing success with the large scale approach, we are demonstrating the applicability of the overall approach to actual help desk operations.

4.3 Scope

Like the pilot studies, the large scale approach is a case study conducted in an industrial setting. The large scale approach uses data from the same help desk in pilots 1, 3 and 4, and uses the same 156 products included in pilot 4. The scope of research in the large scale approach is increased to include cost predictions for help desk operations. Additionally, two additional incident prediction methods are introduced in order to compare results from the prediction method used in pilot 4. An overview of help desk operations is included. The data used in the case study is described. The large scale approach is included along with results and conclusions.

4.4 Case Study

4.4.1 Help Desk Operations

Help desk operations in the context of this pilot study are explained in Appendix A.

4.4.2 Data

In addition to the incident data described in Appendix B, we also obtained actual monthly labor figures for effort in resolving help desk incidents. The labor figures have been validated by help desk managers to ensure their accurate interpretation. Effort is measured as the number of hours recorded by help desk technicians responsible for the resolution of incidents submitted for desktop software products, hardware issues, operating system errors and products outside the scope of this research. We are concerned with the amount of labor associated with the resolution of incidents for the products we use in our cluster analysis. Because monthly labor figures account for resolution effort for all types of incidents, we need a way to extract the portion of labor that represents effort only for the incidents submitted for the 156 clustered products. An attempt to obtain reasonable estimates of the applicable portion through interviewing help desk managers was not successful.

The data used for this case study fall into two categories defined by Runeson, et. al. [93]. Knowledge of help desk operations in the industrial setting of this case study is obtained by direct methods through semi-structured interviews with two help desk managers. Data obtained by this method compliments the experienced-based knowledge of help desk data analysis by one of the authors of this paper. A second category of data, one in which the majority of the quantitative data in this case study falls, is obtained through the inspection of archival data in the help desk database of incident records. The quantitative data is collected and maintained by the help desk for the analysis of performance with respect to established service metrics and for trend analyses from which business decisions are made.

We use incident data in two ways. First, we develop an incident volume prediction model using actual incidents resolved by the help desk as described later in our approach. Secondly, we validate the incident prediction process by measuring the accuracy of incident volume prediction compared to the actual number of incidents generated by a set of products over a selected time period.

In this case study we also use help desk cost data in two ways. First, historical cost data is used with incident volume to develop a cost prediction model based on predicted incident volume. Our cost model development will be discussed later in our approach. Secondly, we use actual cost data to validate the accuracy of our cost prediction model. The cost data we collected is in units of hours per calendar month. The data are the collective number of hours for all help desk technicians and managers who spent effort on resolving all incidents for the given month. The actual number of incidents is known as well as the product for which each incident is submitted. Although the numbers represent efforts toward incident resolution, we are careful to note that they include effort for all product-based help desk incidents, inclusive of the 156 products in our case study. The ideal scenario of accounting for labor by product would have greatly assisted our research, but the labor was not accounted for to that level of granularity. The 156 products in this case study is a subset of the overall portfolio of products managed by the help desk. Similarly, the aggregate labor hours each month represents effort spent resolving incidents from products in the overall portfolio. The two entities are related through the identification of a specific product in each incident. This allows us to determine the number of incidents each month that are submitted against the 156 products in our case study and for the balance of the portfolio. Actual labor costs for the effort described above were obtained from help desk managers. We collected monthly labor hours for each month over the time interval of interest in this case study. The labor data is shown in Table 4.1.

Table 4.1: Monthly labor hours for products selected in case study

Month	Hours	Month	Hours	Month	Hours
-	-	01/2010	768	01/2012	719
-	-	02/2010	797	02/2012	924
-	-	03/2010	820	03/2012	782
04/2008	925	04/2010	780	04/2012	742
05/2008	736	05/2010	777	05/2012	686
06/2008	771	06/2010	808	06/2012	615
07/2008	865	07/2010	957	07/2012	724
08/2008	719	08/2010	884	08/2012	690
09/2008	829	09/2010	839	09/2012	562
10/2008	876	10/2010	880	10/2012	646
11/2008	728	11/2010	785	11/2012	595
12/2008	616	12/2010	618	12/2012	455
01/2009	822	01/2011	1008	01/2013	810
02/2009	803	02/2011	796	02/2013	965
03/2009	1083	03/2011	849	03/2013	1015
04/2009	901	04/2011	786	04/2013	854
05/2009	821	05/2011	723	05/2013	854
06/2009	903	06/2011	832	06/2013	755
07/2009	854	07/2011	766	07/2013	788
08/2009	933	08/2011	874	08/2013	660
09/2009	782	09/2011	726	09/2013	629
10/2009	868	10/2011	681	10/2013	713
11/2009	859	11/2011	724	-	-
12/2009	669	12/2011	563	-	-

4.5 Approach

By integrating actual cost data with incident history for a selected set of desktop products, we present an approach by which incident resolution cost is predicted. Figure 4.1 describes the main steps in our approach.

Our approach uses two major phases:

- incident estimation
- cost estimation

Since incident estimation for hundreds of products is expensive, we use cluster analysis to group similarly behaving products in clusters, for which we then estimate incidents based on the representative product in the cluster. We determine this representative product via PCA. We use these representative products to estimate the total number of incidents for all selected products using software reliability growth models. In Phase 2 we use incident estimates and cost data to estimate expected labor cost for these expected incidents. Finally, we validate our approach on the data available through our case study.

For problem resolution of software products in their operational life cycle, incident records related to malfunctioning software reflect product reliability. A view of incident records over time exposes a reliability trend. Knowing how many remaining incidents to expect assists in quantifying information related to product quality, loss of employee productivity associated with product failures, and projected levels of help desk staffing to prepare for problem resolution.

4.5.1 Cluster Analysis

A goal of this step is to avoid having to estimate incidents for 156 products selected in section 4.4. We use cluster analysis to group products based on incident behavior (attributes

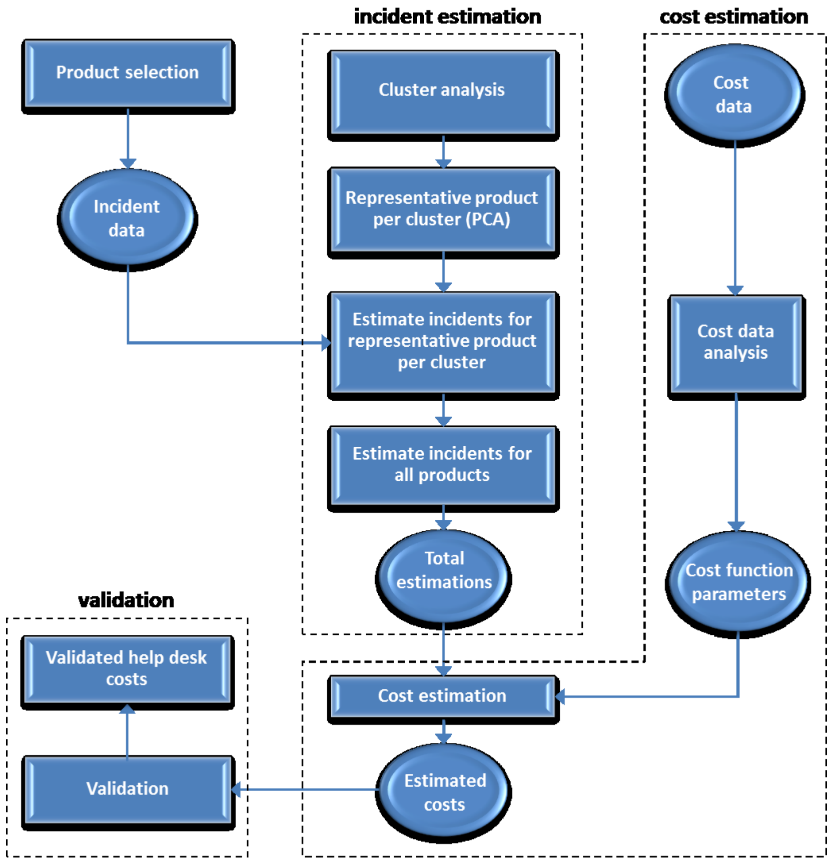


Figure 4.1: Process Flow: product selection through process validation

in Table A.1). This is essentially a data reduction or variable reduction technique using pattern recognition [63] [57].

Cluster analysis is a method use to separate a data set into the most similar data in the same cluster and most dissimilar data in different clusters. We apply an agglomerative hierarchical cluster analysis method [124] to group products together. Briefly, agglomerative cluster analysis assigns each data point to a unique cluster, and then merges data points into clusters such that the distance between the points in each cluster is minimized and the distance between the clusters is maximized. The technique returns a tree structure (dendrogram) in which clusters are related hierarchically. This allows us to select cluster sizes small enough to remain confident of behavior similarity among cluster members (and dissimilarity between clusters) while meeting the goal of data reduction.

A vector of seventeen attributes of Table A.1 for each of the 156 products is input to the cluster analysis. We used the *R* statistical tool *hclust* package. It includes a utility to plot the hierarchical relationship between clusters as a dendrogram. Clusters with relatively small distances between heights in the dendrogram are more similar to one another than those whose height differences are greater. Since we clustered products on the basis of their centroids, products in clusters with smaller differences in height behave more similarly in terms of their incident attributes. The dendrogram produced from the cluster analysis is shown in Figure 4.2. Rather than deciding on clusters by specifying a height in Figure 4.2, we were constrained by the limitations in our linear regression tool to no more than 16 clusters. We also wanted to have a manageable number of products per cluster. An analysis of the dendrogram with these constraints in mind resulted in 11 clusters. While this may be less than ideal from a theoretical perspective, it nevertheless resulted in high quality incident estimation, as shown later. All 156 product are accounted for in the eleven clusters. Table 4.2 shows the eleven clusters. Column 1 shows the cluster ID, column 2 shows the products in each cluster.[1]

[1]In Table 4.2 the numbers in parenthesis may be ignored at this time as they will be discussed later.

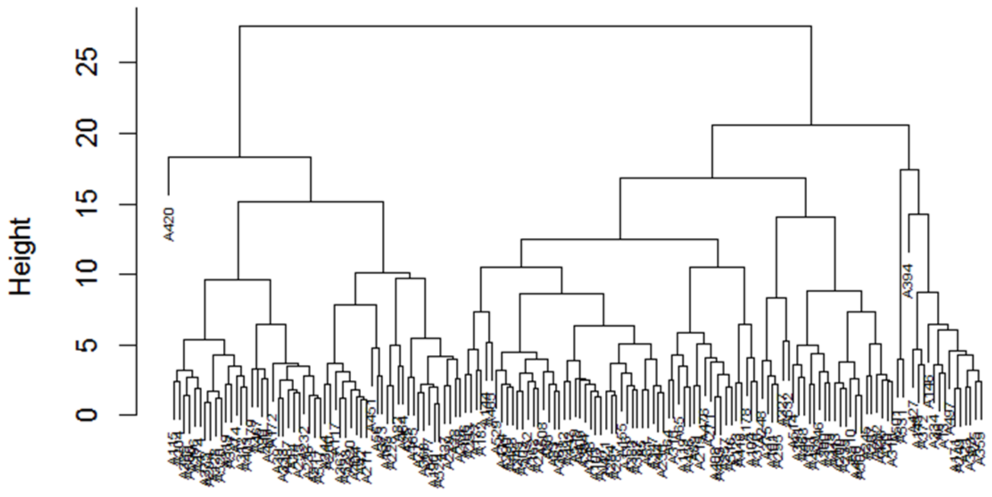


Figure 4.2: Dendrogram from cluster analysis showing all products

4.5.1.1 Representative Product per Cluster

One of the motivations for clustering was to try to determine a representative product in each cluster, so we need only predict future incidents for one product in each cluster rather than for all. The representative product must behave similarly to the other products in its cluster in terms of incident behavior. The behavior of an incident is observed through its attributes shown in Table A.1. We use PCA to determine product similarity. PCA shows us the amount of variance between attributes from a set of incidents for each product in a cluster. We can see how the attributes relate to one another within a product through inspections of the resulting principal components. For example, if attributes *A1* (incident urgency) and *A2* (incident impact) show strong covariance in the principal components from three products, we conclude those three products have similar behavior on the basis of the impact and urgency. By discovering the attribute relationships in each of the products within a cluster, we can rank the products according to their similarity based on the number of attribute relationships that are common between the cluster members. Products with higher numbers of relationships common with other cluster members get ranked higher.

Table 4.2: Cluster product scores based on number of shared and unique principal components within cluster

Cluster ID	Cluster members (raw similarity score)	Selected product
BV	A421 (19), A284 (26), A287 (23), A155 (13), A116 (8), A383 (28), A387 (23), A5 (12), A81 (23), A347 (16), A44 (29), A235 (20)	A44
EW	A501 (29), A531 (25), A394 (22), A427 (25), A145 (47), A372 (66), A146 (24), A334 (61), A6 (41), A13 (32), A497 (53), A171 (64), A249 (71), A21 (63), A393 (23), A75 (48), A359 (28)	A249
EG	A179 (27), A467 (34), A90 (11), A499 (8), A172 (16), A39 (18), A333 (47), A487 (40), A14 (32), A255 (28), A232 (13), A236 (46), A511 (52), A517 (45)	A511
ER	A248 (84), A375 (30), A101 (109), A295 (53), A327 (53), A532 (53), A514 (27), A391 (64), A468 (123), A151 (45), A439 (107), A246 (85), A300 (90), A329 (108), A413 (115), A200 (26), A239 (55), A510 (119), A450 (89), A502 (111), A245 (77), A267 (107), A282 (118), A436 (46), A20 (75), A310 (96)	A468
EI	A185 (98), A228 (106), A82 (36), A464 (40), A25 (62), A165 (53), A77 (94), A368 (68), A277 (34), A190 (44), A524 (39), A27 (46), A229 (39), A28 (40), A278 (47)	A228
EH	A244 (36), A260 (25), A117 (36), A367 (31), A523 (51), A320 (16), A404 (48), A42 (32), A211 (41), A451 (18), A56 (28), A373 (17)	A523
EK	A64 (27), A315 (47), A65 (24), A119 (26), A456 (24), A89 (26), A219 (54), A476 (42), A271 (47), A408 (35), A495 (48), A357 (31), A516 (27), A123 (38), A416 (33), A178 (13), A104 (24), A376 (37)	A219
EF	A115 (56), A204 (76), A385 (72), A500 (77), A396 (69), A474 (71), A275 (49), A294 (86), A128 (79), A520 (71), A319 (62), A397 (46), A74 (34), A312 (43), A403 (45)	A294
DA	A529 (39), A114 (21), A290 (35), A488 (47), A345 (44), A503 (31), A152 (50), A162 (25), A222 (14), A508 (37), A86 (30), A83 (22), A361 (55)	A361
BU	A382 (5), A442 (6), A519 (9), A409 (1), A384 (7), A148 (7), A103 (-2), A187 (3)	A519
DV	A166 (11), A293 (7), A177 (2), A182 (-1), A144 (3), A465 (-7)	A166

To achieve this, we assigned a weighted similarity score to each product in a cluster. We measure similarity based on the number of attribute relationships a candidate product shares with the other products in the same cluster. The score is weighted more heavily by attribute relationships which are shared by larger numbers of products in a cluster and weighted less by relationships shared with fewer products. The similarity score of the candidate product is decreased by the number of attribute relationships which are unique to the product. These are attribute relationships that are not shared with any products in the cluster. For example, consider cluster c containing four products p_1, p_2, p_3, p_4. The goal is to determine the product that best represents the other products in cluster c. We perform a PCA for each product and show the PCA loadings in Tables 4.3 through 4.6. In these tables, the row IDs correspond to the attributes shown earlier in Table A.1. The column IDs are the names of the principal components. The numbers in each column are the loadings for the principal component. The PCA loadings shown in boldface form the principal component relationships for each product. These are identified using techniques in [21] [126] [82] [122] [10]. For example, in Table 4.6 the PCA loadings for PC1 result in a relationship between A3 and A5. This relationship is interpreted as "attribute A3 (incident priority) varies with A5 (incident is resolvable through First Call Resolution) for product p_1".

Table 4.7 shows all of the principal component relationships found in products p_1, p_2, p_3 and p_4 in cluster c. A bullet in Table 4.7 indicates the principal component relationship in the left column is found in the product identified in the column header. Some attributes vary independently, such as A9 in Table 4.7 for PC1. In other words, attribute A9 does not vary with any other attribute.

For each principal component relationship, a relationship score is calculated and shown in the right-most column in Table 4.7, according to the following assignments:

- A relationship found in exactly four products is assigned a score of 3

Table 4.3: PCA loadings for scoring example: product p_1

	PC1	**PC2**	**PC3**	**PC4**	**PC5**
A1	0.2124	0.3130	**-0.3699**	0.1651	-0.2458
A2	-0.0200	-0.2648	**-0.4168**	0.1978	0.1871
A3	**0.8038**	0.0296	0.2538	-0.1603	-0.2805
A4	0.0920	**0.5006**	-0.2795	0.1706	-0.1452
A5	**0.7889**	0.1478	-0.2066	-0.1390	0.0682
A6	0.0680	**-0.6004**	-0.2265	0.3063	0.2982
A7	0.1459	0.2198	-0.0025	**0.5123**	-0.1574
A8	-0.0170	0.3011	0.1069	**-0.8662**	0.0874
A9	-0.2071	0.2267	0.0837	0.0779	**0.4223**
A10	-0.2216	-0.2184	0.2822	0.1746	**0.3298**

Table 4.4: PCA loadings for scoring example: product p_2

	PC1	**PC2**	**PC3**	**PC4**	**PC5**
A1	**-0.3955**	0.3166	0.1674	0.0457	0.3023
A2	**-0.4587**	0.1164	-0.0872	0.1867	-0.1206
A3	-0.0014	**0.4892**	0.0337	0.1202	0.1584
A4	-0.0347	-0.1873	**0.7821**	-0.0747	0.2256
A5	0.1937	**0.6891**	0.0499	0.0998	-0.0116
A6	0.2892	-0.1487	**-0.4520**	-0.0989	-0.3184
A7	-0.0612	0.0632	0.2316	**-0.3972**	-0.2350
A8	-0.2587	-0.2237	0.0771	-0.2857	**-0.6123**
A9	-0.3155	-0.2147	0.1263	0.2232	**0.5563**
A10	-0.0530	0.0961	0.1915	**-0.4235**	-0.1711

Table 4.5: PCA loadings for scoring example: product p_3

	PC1	**PC2**	**PC3**	**PC4**	**PC5**
A1	0.2513	-0.3109	0.0627	-0.0179	0.2369
A2	-0.0269	0.2793	0.1674	0.1557	-0.0635
A3	-0.2033	-0.2574	**0.7756**	0.2081	-0.2958
A4	-0.1123	**0.4561**	0.0039	0.1345	-0.2210
A5	0.2524	0.0489	**0.3287**	-0.0200	0.2015
A6	0.2381	**-0.6154**	0.0947	0.0364	-0.2633
A7	-0.2174	0.1160	-0.1342	**0.6122**	0.0968
A8	0.0526	-0.0044	0.2559	**-0.7708**	0.1678
A9	**-0.8223**	-0.2031	0.0585	-0.1161	-0.0078
A10	-0.3055	-0.0926	0.2675	0.2706	**-0.4800**

Table 4.6: PCA loadings for scoring example: product p_4

	PC1	**PC2**	**PC3**	**PC4**	**PC5**
A1	-0.0378	0.1164	-0.2083	0.2569	0.0725
A2	0.0959	0.2390	**0.5514**	0.1444	-0.2855
A3	**0.5621**	-0.3020	-0.0036	0.1263	-0.1564
A4	0.2028	**0.3298**	0.2717	-0.1862	0.0111
A5	**0.6632**	0.2143	0.0933	0.0893	0.1494
A6	0.2890	**-0.4877**	0.0527	-0.1172	-0.0276
A7	0.1640	0.2642	0.1142	0.1550	**0.3550**
A8	0.0361	0.0793	0.1827	0.2627	-0.0749
A9	-0.3185	0.2864	0.0351	-0.2780	-0.2404
A10	-0.0971	-0.1790	0.0054	**-0.3870**	0.1424

- A relationship found in exactly three products is assigned a score of 2
- A relationship found in exactly two products is assigned a score of 1
- A relationship found in exactly one product is assigned a score of -1

Table 4.7 shows the relationship score summation for each of the four products and the resulting similarity score. The product with the highest similarity score is designated as the cluster representative. In this example, product p_1 has a similarity score of 7, which is higher than the scores for the other three products as shown in the bottom row of Table 4.7. Product p_1 is therefore determined to be the representative of cluster c.

In the similarity scoring technique for n products in c, the attribute relationships for the product to be scored are compared against the $n - 1$ other products. The similarity score is influenced positively by higher numbers of shared relationships and incorporates a penalty for the non-shared (unique) relationships. Table 4.2 shows the eleven clusters and the products which comprise each cluster. The similarity score assigned to each product using the scoring technique described above is shown in parenthesis next to each product ID. The product selected to represent the cluster members is shown in the column labeled "Selected Product". As a means of comparing the cluster representatives, Table 4.8 shows the number of installations and incidents for each cluster representative, and the time period over which the incidents were generated.

4.5.1.2 Incident Estimates Representative Product per Cluster

In pilot study 4, we adapted the SRGM model selection approach from Stringfellow et al., Andrews et al. and Andersson [104] [105] [7] by including a step at which the model with the best GOF value was selected for incident prediction. This model selection process is shown earlier in Figure 3.7. In this large scale approach case study we use the same model selection process. Figure 3.7 shows our estimation process. The process begins with obtaining cumulative incident data for the eleven representative products. The incident data

Table 4.7: Product similarity scoring example for cluster c

	Products in cluster c				
Principal comp. relationship	p_1	p_2	p_3	p_4	**Relationship score**
A1, A2	●	●			**1**
A3, A5	●	●	●	●	**3**
A4, A6	●	●	●	●	**3**
A7, A8	●		●		**1**
A9, A10	●				**-1**
A7, A10		●			**-1**
A8, A9		●			**-1**
A9			●		**-1**
A10			●	●	**1**
A2				●	**-1**
A7				●	**-1**
Rel. score sum	**1 + 3 + 3 + 1 -1**	**1 + 3 + 3 - 1 -1**	**3 + 3 + 1 -1 -1**	**3 + 3 -1 -1 -1**	
Similarity score	**7**	**5**	**5**	**3**	

Table 4.8: Cluster representatives - product characteristics

Cluster ID	Cluster rep ID	Num installations	Num incidents	Dates of incidents
BV	A44	32214	345	4/11/2008 - 10/17/2013
EW	A249	121,259	199	6/24/2008 - 10/23/2013
EG	A511	452	481	4/1/2008 - 10/22/2013
ER	A468	1602	51	8/15/2008 - 1/7/2013
EI	A228	7570	65	4/11/2008 - 8/31/2012
EH	A523	119,341	3358	5/22/2008 - 9/13/2013
EK	A219	4704	111	4/9/2008 - 10/30/2013
EF	A294	3531	1088	4/2/2008 - 10/24/2013
DA	A361	240	174	4/1/2008 - 10/25/2013
BU	A519	1357	269	4/10/2008 - 10/10/2013
DV	A166	11,016	656	4/1/2008 - 5/23/2013

is applied to all four SRGMs by running a curve fit to estimate model parameters for each SRGM. Models which do not converge are rejected. In the next step we determine the R^2 GOF value for each model that converges. The model with the best R^2 value is selected for prediction of incidents. The model is used to predict incidents for the next period (for example, one month), and to predict the remaining number of incidents. In these last two steps the model may be used to predict incidents for more than one period (one month and two month predictions, for example). After the predictions are made, the first process is repeated, starting with the collection of incidents for the next period. Tables 4.9 through 4.12 show the results of applying this approach to the representative products in all eleven clusters for an R^2 threshold of 0.95 (Tables 4.9 and 4.10) and 0.99 (Tables 4.11 and 4.12). Tables 4.9 and 4.11 show the estimated number of incidents for R^2 thresholds of 0.95 and 0.99, respectively. Tables 4.10 and 4.12 show the estimated remaining number of incidents for R^2 thresholds of 0.95 and 0.99, respectively. Incident data from April 2008 through August 2012 was used to obtain the results. To the right of the cluster ID in each table is the ID of the representative product. The SRGM column lists the model that resulted in the best R^2 value. The number of incidents predicted for one month through five months out is shown along with the remaining number of incidents for the one month through five month prediction periods. The remaining number of incidents is obtained by subtracting the predicted number of incidents submitted from the total number of incidents predicted by the SRGM, as shown in Figure 3.8. For $R^2 \geq 0.95$ (Table 4.9), all curve fits resulted in at least one model that converged with R^2 greater than 0.95. However for $R^2 \geq 0.99$ (Table 4.11), cluster EI resulted in R^2 less than 0.99 for all models. The best R^2 value for EI was 0.9541 for the GO Musa model. We note in Table 4.8 that product A228, the representative for cluster EI, produced the second smallest number of incidents over the time period in this case study. In fact, this product has only 65. An inspection of the cumulative incident data shows the number of incidents for this product reaches 100% of its total incidents after 48 months. The remaining 4 months have no more incidents. In contrast, the representative for cluster ER (product A468) which has even fewer total incidents than A228 (51), achieved an R^2 value of 0.9951. An inspection of the incident data for A468 indicates the number of incidents for this product reaches 100% of its total during the last month of analysis. None of the other representative products have stopped producing incidents at the end of the data collection period. These results suggest products at or beyond their incident-producing periods may not be suitable candidates for representing other products in a cluster.

It is interesting to see that this method was able to come up with an individual estimate in all but the EI cluster representative at $R^2 \geq 0.99$ (at least one of the models converged). Having estimated future incidents for the representative product in each cluster, we turn to

Table 4.9: Incident estimations for cluster representatives, $R^2 \geq 0.95$

Cluster ID	**Cluster rep**	**SRGM**	**Incidents for next period**				
			1 mo	**2 mo**	**3 mo**	**4 mo**	**5 mo**
BU	A519	Mod Gompertz	246	249	252	255	259
BV	A44	Mod Gompertz	315	322	328	335	342
DA	A361	Mod Gompertz	167	169	172	174	177
DV	A166	Mod Gompertz	626	629	631	633	636
EF	A294	Mod Gompertz	1064	1069	1073	1077	1081
EG	A511	Mod Gompertz	452	455	458	461	464
EH	A523	Gompertz	3170	3197	3222	3246	3269
EI	A228	GO Musa	59	59	59	59	59
EK	A219	Gompertz	99	100	100	101	102
ER	A468	Gompertz	51	51	52	52	52
EW	A249	Mod Gompertz	130	132	134	137	139

Table 4.10: Remaining incidents for cluster representatives, $R^2 \geq 0.95$

Cluster ID	**Cluster rep**	**SRGM**	**Remaining incidents**				
			1 mo	**2 mo**	**3 mo**	**4 mo**	**5 mo**
BU	A519	Mod Gompertz	242	239	236	233	229
BV	A44	Mod Gompertz	704	697	691	684	677
DA	A361	Mod Gompertz	117	115	112	110	107
DV	A166	Mod Gompertz	43	40	38	36	33
EF	A294	Mod Gompertz	121	116	112	108	104
EG	A511	Mod Gompertz	2792	2789	2786	2783	2780
EH	A523	Gompertz	546	519	494	470	447
EI	A228	GO Musa	0	0	0	0	0
EK	A219	Gompertz	10	9	9	8	7
ER	A468	Gompertz	2	2	1	1	1
EW	A249	Mod Gompertz	292	290	288	285	283

Table 4.11: Incident estimations for cluster representatives, $R^2 \geq 0.99$

Cluster ID	**Cluster rep**	**SRGM**	**Incidents for next period**				
			1 mo	**2 mo**	**3 mo**	**4 mo**	**5 mo**
BU	A519	Mod Gompertz	246	249	252	255	259
BV	A44	Mod Gompertz	315	322	328	335	342
DA	A361	Mod Gompertz	167	169	172	174	177
DV	A166	Mod Gompertz	626	629	631	633	636
EF	A294	Mod Gompertz	1064	1069	1073	1077	1081
EG	A511	Mod Gompertz	452	455	458	461	464
EH	A523	Gompertz	3170	3197	3222	3246	3269
EI	A228	none	-	-	-	-	-
EK	A219	Gompertz	99	100	100	101	102
ER	A468	Gompertz	51	51	52	52	52
EW	A249	Mod Gompertz	130	132	134	137	139

Table 4.12: Remaining incidents for cluster representatives, $R^2 \geq 0.99$

Cluster ID	Cluster rep	SRGM	Remaining incidents				
			1 mo	2 mo	3 mo	4 mo	5 mo
BU	A519	Mod Gompertz	242	239	236	233	229
BV	A44	Mod Gompertz	704	697	691	684	677
DA	A361	Mod Gompertz	117	115	112	110	107
DV	A166	Mod Gompertz	43	40	38	36	33
EF	A294	Mod Gompertz	121	116	112	108	104
EG	A511	Mod Gompertz	2792	2789	2786	2783	2780
EH	A523	Gompertz	546	519	494	470	447
EI	A228	none	-	-	-	-	-
EK	A219	Gompertz	10	9	9	8	7
ER	A468	Gompertz	2	2	1	1	1
EW	A249	Mod Gompertz	292	290	288	285	283

Table 4.13: Actual incidents for predictions by cluster representatives, $R^2 \geq 0.95$

Cluster ID	Actual incidents for next period				
	1 mo	2 mo	3 mo	4 mo	5 mo
BU	239	243	251	258	260
BV	321	329	331	334	338
DA	165	168	170	171	173
DV	627	633	639	641	641
EF	1057	1061	1064	1069	1074
EG	462	463	470	470	470
EH	3227	3251	3265	3274	3276
EI	65	65	65	65	65
EK	101	101	103	105	107
ER	52	52	52	52	52
EW	108	119	125	141	146

estimating incidents for all products in a cluster. Note that this method allows for flexible prediction time frames. We have shown 1-5 month predictions as an example.

4.5.1.3 Incident Estimation Using Cluster Representatives

We present two approaches for estimating the number of incidents that will be generated by each cluster.

- Scalar approach: The estimate of incidents produced by all cluster members in a cluster is determined by multiplying the number of incidents predicted by the cluster

Table 4.14: Relative error for predictions by cluster representatives, $R^2 \geq 0.95$

Cluster ID	**Relative error**				
	1 mo	**2 mo**	**3 mo**	**4 mo**	**5 mo**
BU	0.0293	0.0247	0.0040	-0.0116	-0.0038
BV	-0.0187	-0.0213	-0.0091	0.0030	0.0118
DA	0.0121	0.0060	0.0118	0.0175	0.0231
DV	-0.0016	-0.0063	-0.0125	-0.0125	-0.0078
EF	0.0066	0.0075	0.0085	0.0075	0.0065
EG	-0.0216	-0.0173	-0.0255	-0.0191	-0.0128
EH	-0.0177	-0.0166	-0.0132	-0.0086	-0.0021
EI	-0.0923	-0.0923	-0.0923	-0.0923	-0.0923
EK	-0.0198	-0.0099	-0.0291	-0.0381	-0.0467
ER	-0.0192	-0.0192	0.0000	0.0000	0.0000
EW	0.2037	0.1092	0.0720	-0.0284	-0.0479

representative by the number of products in the cluster, as in Equation 4.5.1.

$$Incidents_S = Incidents_{rep} * |cluster| \tag{4.5.1}$$

- Linear function approach: A function derived through linear regression based on historical incident data is used to estimate incidents produced by all cluster members for a given cluster. The linear function relates actual incidents numbers from the cluster representative to the number of incidents produced by all cluster members. With the linear function approach, the estimated number of incidents for a cluster is a linear transformation of the number of incidents predicted by the cluster representative, as shown in Equation 4.5.2, where a and b are the coefficient and intercept, respectively, resulting from the linear regression.

$$Incidents_L = a * Incidents_{rep} + b \tag{4.5.2}$$

For both methods, the estimated number of incidents for all 156 products is obtained by summing the per-cluster estimates for all eleven clusters.

4.5.1.4 Scalar Method

Let N_{cr_i} be the number of incidents predicted for the representative product of cluster c_i ($i = 1 \ldots n$, where n is the number of clusters).

Let m_i be the number of products in cluster i.

Then N_{c_i}, the number of incidents predicted for cluster c_i is given by

$$N_{c_i} = m_i * N_{cr_i} \quad (i = 1, \ldots, n) \tag{4.5.3}$$

The total number N_S of incidents is then given by

$$N_S = \sum_{i=1}^{n} N_{c_i} = \sum_{i=1}^{n} m_i * N_{cr_i} \tag{4.5.4}$$

Table 4.15: Scaled incident prediction relative error for one month prediction using Scalar Method (September 2012)

Cluster ID	**Actual num inc cluster rep**	**Predicted num inc cluster** N_{cr_i}	**Rel. err. per cluster rep**	**Actual num inc cluster**	**Est. num inc per cluster** N_{c_i}	**Rel. err. per cluster**
BU	239	246	0.0293	4773	1968	-0.5877
BV	321	315	-0.0187	6089	3780	-0.3792
DA	165	167	0.0121	4340	2171	-0.4998
DV	627	626	-0.0016	3409	3756	0.1018
EF	1057	1064	0.0066	11525	15960	0.3848
EG	462	452	-0.0216	12692	6328	-0.5014
EH	3227	3170	-0.0177	12028	38040	2.1626
EI	65	59	-0.0923	1210	885	-0.2686
EK	101	99	-0.0198	7911	1782	-0.7747
ER	52	51	-0.0192	2032	1326	-0.3474
EW	108	130	0.2037	6118	2210	-0.6388
Total	**6424**	**6379**	**-0.0070 (overall rel. err.)**	**72127**	**78206** N_S	**0.0843 (overall rel. err.)**

Table 4.15 shows the results of applying the scalar incident prediction method to estimate the cumulative number of incidents that will be generated by all 156 products in one

month. Incident data from April 2008 through August 2012 was used in the predictions, so the one month predictions in Table 4.15 are for September 2012. Starting in the left two columns of Table 4.15, each cluster ID is shown with the actual number of incidents produced by its cluster representative. In the third column, the predicted number of incidents for each cluster representative is shown. These are the values of N_{cr_i} in Equation 4.5.3. The relative prediction error per cluster representative is included in the table to show the accuracy with which each cluster representative product predicts its own incidents. The overall relative error based on the cluster representatives is shown at the bottom of the column titled “Relative err. per cluster rep”. This number is calculated from the total of the predicted number of incidents for each cluster representative, and the total of the actual number of incidents for each cluster representative, using the formula (predicted - actual)/actual. The column titled “Actual num inc. per cluster” contains the known number of incidents produced by all members of each cluster through the month of September 2012. As with the numbers in the column “Actual num incidents cluster rep”, the actual number of incidents produced by all cluster members is used to calculate relative prediction error by cluster. The numbers in the column titled “Est. num incidents per cluster” are the scaled values for N_{c_i} obtained through Equation 4.5.3. Their sum is used along with the sum of the actual number of incidents per cluster to determine the relative incident prediction error for all 156 products combined.

Using the scalar incident prediction method, the relative error per cluster representative shown in the fourth column from the left in Table 4.15 varies from a minimum (worst under-prediction) of -0.0923 (cluster representative for EI) to a maximum (worst over-prediction) of 0.2037 (cluster representative for EW). The two most accurate cluster representative incident predictions (relative error closest to zero) are -0.0016 and 0.0066. These are from the clusters DV and EF respectively. The mean and median incident prediction relative error for all cluster representatives are 0.0055 and -0.0177 respectively. The overall one month incident prediction error of the cluster representatives is -0.0070. This absolute error

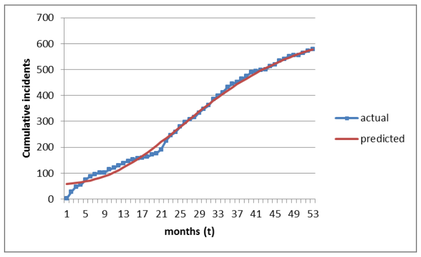


Figure 4.3: Cluster DV representative

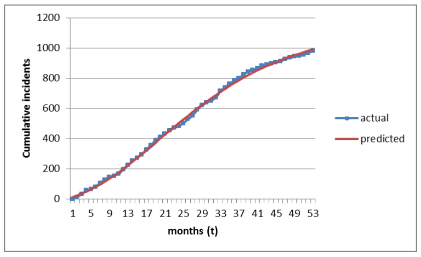


Figure 4.4: Cluster EF representative

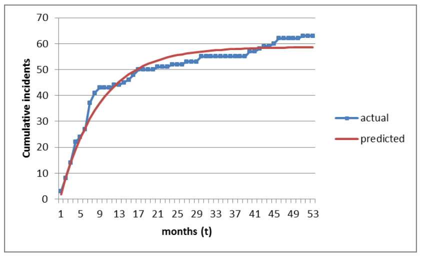


Figure 4.5: Cluster EI representative

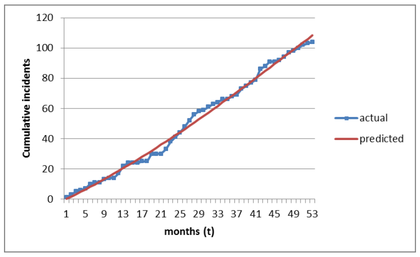


Figure 4.6: Cluster EW representative

of 45 incidents out of an actual number of 6424 results in an underprediction of 0.7%. Although this is a reasonably small overall prediction error, we wanted to investigate why some incident predictions from the representatives were more accurate than others.

The predicted and actual cumulative incidents as a function of time are shown for the two most accurate one month predictions (Figures 4.3 and 4.4), and the two least accurate one month predictions (Figures 4.5 and 4.6). We can see how data points for the actual incidents fit the curve for the predicted incidents somewhat better for DV and EF in Figures 4.3 and 4.4, compared to EI and EW in Figures 4.5 and 4.6. The difference is more evident in months 45 though 53 for all four cluster representatives compared to earlier months. Since one month predictions beyond month 53 are measured against actual incidents (see Table 4.15, column "Relative error per cluster rep"), the error at month 53 detected visually helps explain why predictions one month out have the same approximate magnitude of error as one month earlier. For clusters DV and EF, whose representatives resulted in the two most accurate predictions, the curve fit error shown graphically in Figures 4.3 and 4.4 at month

53 is very small. Another distinction between the four results in Figures 4.3 through 4.6 is that number of incidents at month 53 for the two least accurate predictors is only about 10% of the number of incidents for the two most accurate predictors. The appearance of the predicted curve for EI in Figure 4.5 for the later months is close to horizontal. This denotes the product selected to represent EI is approaching the end of its incident producing period. Each SRGM in this case study has a parameter which holds the value for the cumulative number of incidents that would eventually be reached if sufficient time were permitted. By appearance of Figure 4.5 for cluster representative EI, that number is slightly under 60 incidents. An inspection of the detailed results for the curve fit for EI confirms the number is 59. This validates our suspicion of the life cycle phase of the desktop product selected to represent cluster EI. In contrast, the appearance of Figure 4.6 for cluster representative EW indicates incidents are still being produced at month 53 since there is no evidence of the curve flattening out. Given the resulting prediction error, this may suggest the cluster representative for EW is not sufficiently mature to accurately represent the other cluster members, although for the other seven cluster representatives whose curve fits graphs are not shown for brevity, three indicate a similar level of maturity as seen with EF. These findings present an opportunity for an extension of this research with a focus on product life cycle, product incident volume, and candidacy for cluster representation.

The two right-most columns in Table 4.15 show the results of scaling the number of incidents estimated for each cluster representative, using Equation 4.5.3, and the per-cluster relative error. Table 4.15 also includes the actual number of incidents per cluster. This number is the sum of the actual incidents produced by each member of the cluster identified in the "Cluster ID" column. The per-cluster actual number of incidents is used with the per-cluster estimated number of incidents to calculate the relative error per cluster shown in Table 4.15. Additionally, totals for the actual and estimated number of incidents and overall relative error are shown at the bottom of the table. These values are associated with all 156 products.

In addition to incident predictions based on cluster representatives, Table 4.15 shows the per-cluster incident prediction and relative prediction error. Similar to our analysis of relative error for each cluster representative, we are interested in per-cluster relative error and why some cluster predictions are more accurate than others. In the “Rel. error per cluster” column in Table 4.15 we see a range of -0.7747 for cluster EK (worst underprediction) to 2.1626 for cluster EH (worst overprediction). The mean and median relative error are -0.1226 and -0.3792 respectively. The overall relative error is 0.0843. Our first observation is that overall relative error of 0.0843 (bottom of right-most column of Table 4.15) is worse compared to the overall relative error of the cluster representatives (-0.0070). Secondly, the magnitude of the per-cluster relative error is much greater than that of each cluster representative for all eleven clusters. An inspection of the clusters which resulted in the two most accurate predictions (DV and EI) and two least accurate predictions (EH and EK) indicates weak correlation with the two best and two worst predictions for the cluster representatives. Cluster DV resulted in the most accurate incident prediction for both the cluster representative and for all cluster members. Interestingly, EI is the next most accurate cluster predictor while it was one of the worst as a cluster representative. This tells us that incident prediction accuracy by the cluster representatives (using the scalar incident prediction method) is not a strong indicator of cluster-wide accuracy based on the one month prediction results. This is not surprising since the scalar method uses only one variable, m_i, to transform the prediction of the cluster representative to a prediction of incidents for all members of the cluster it represents. More than one variable in the transformation, as will be discussed with the linear function method, is likely to achieve better prediction accuracy through an improved function fit. As will be seen, however, the trade-off for improved prediction accuracy is the additional effort required to perform the linear regression associated with the linear function method.

4.5.1.5 Linear Function Method

In the linear function approach to incident prediction, we develop a set of linear functions to relate the number of incidents predicted by the cluster representative to the number of incidents for the aggregate of all members of the cluster.

Let $A_{cr_i}(t)$ be the actual incident counts for time frame t for the cluster representative cr_i for cluster c_i $(i = 1, \ldots, n)$, where n is the number of clusters.

Let $A_i(t)$ be the actual incident counts for time frame t for cluster c_i $(i = 1, \ldots, n)$. Assume that $1 \leq t \leq T$, where T is the time period for which measurements are available. We perform linear regressions of the form

$$A_i = x_i A_{cr_i} + y_i \quad (i = 1, \ldots, n) \tag{4.5.5}$$

to determine constants x_i and y_i for each cluster C_i.

Then N_{c_i}, the number of incidents predicted for cluster c_i is given by

$$N_{c_i} = x_i N_{cr_i} + y_i \tag{4.5.6}$$

We then can use these linear functions to predict total number of incidents N_L based on the predicted incidents for each cluster representative:

$$N_L = \sum_{i=1}^{n} N_{c_i} \tag{4.5.7}$$

These functions allow us to predict the number of incidents for all members of a cluster based on the cluster representative alone. Table 4.16 shows for each cluster actual cumulative incidents vs. estimated incidents and relative error. As with the scalar method described earlier, incident data from April 2008 through August 2012 was used in the predictions, so

Table 4.16: Incident prediction relative error for one month prediction using Linear Function Method (September 2012)

Cluster ID	**Actual num incidents per cluster**	**Estimated num incidents per cluster** $x_i N_{cr_i} + y_i$	**Relative error per cluster**
BU	4773	5373	0.1257
BV	6089	8603	0.4129
DA	4340	3236	-0.2544
DV	3409	3849	0.1291
EF	11525	10726	-0.0693
EG	12692	13844	0.0908
EH	12028	12181	0.0127
EI	1210	417	-0.6554
EK	7911	6496	-0.1789
ER	2032	1617	-0.2042
EW	6118	5970	-0.0242
Total	**72127**	**72312** (N_L)	**0.0026 (overall rel. error)**

the one month predictions in Table 4.16 are for September 2012. The overall relative error is shown at the bottom of the column titled “Relative error per cluster”. This is calculated from the estimated number of incidents for each cluster, and the total of the actual number of incidents for each cluster representative, using the formula (predicted - actual)/actual.

Similar to our analysis of relative error for each cluster representative using the scalar incident prediction method, we are interested in per-cluster relative error with the linear function method as well as overall relative prediction error. Specifically, we investigate why some cluster predictions are more accurate than others. The relative error per cluster shown in Table 4.16 varies from a minimum (worst under-prediction) of -0.6554 (cluster representative for EI) to a maximum (worst over-prediction) of 0.4129 (cluster representative for BV). The two most accurate incident predictions (relative error closest to zero) are 0.0127 and 0.0242. These are from the cluster representatives for EH and EW respectively. The mean and median relative error calculated from the set of eleven estimations are -0.0559 and -0.0242 respectively. The overall relative error is 0.0026.

Our first observation with the results of the linear function method is that the overall incident prediction is rather accurate for this particular case of one month incident predictions. An overall relative error of 0.0026 means the one month prediction overestimates incidents for all 156 products combined by 0.26%. The absolute error of 185 incidents is quite small compared to the total incident volume of approximately 72,000 incidents. The other significant finding with the linear function method is that in spite of the relatively inaccurate per-cluster predictions (for example EI and BV in Table 4.16), the overall relative error is small.

4.5.1.6 Estimate of Incidents for All Products

This requires estimating incidents for all 156 products. It is by far the most expensive. This method estimates 77,491 incidents compared to 72,127 actual incidents with a relative error of 6%. So not only is it very expensive, it is also rather inaccurate. A comparison of the estimation methods in terms of effort is presented in the next section.

Table 4.17: Incident prediction relative error for one month prediction: Scalar and Linear function methods (September 2012)

Cluster ID	**Scalar**			**Linear function**	
	Actual incidents per cluster	**Estimated incidents per cluster**	**Relative error per cluster**	**Estimated incidents per cluster**	**Relative error per cluster**
BU	4773	1968	-0.5877	5373	0.1257
BV	6089	3780	-0.3792	8603	0.4129
DA	4340	2171	-0.4998	3236	-0.2544
DV	3409	3756	0.1018	3849	0.1291
EF	11525	15960	0.3848	10726	-0.0693
EG	12692	6328	-0.5014	13844	0.0908
EH	12028	38040	2.1626	12181	0.0127
EI	1210	885	-0.2686	417	-0.6554
EK	7911	1782	-0.7747	6496	-0.1789
ER	2032	1326	-0.3474	1617	-0.2042
EW	6118	2210	-0.6388	5970	-0.0242
Total	**72127**	**78206**	**0.0843 (overall RE)**	**72312**	**0.0026 (overall RE)**

Table 4.18: Incident prediction relative error for one month prediction: All products method (September 2012)

All products		
Actual number of incidents	**Estimated incidents**	**Relative error**
72172	**77491**	**0.0585**

4.5.1.7 Comparison of approaches to estimate future incidents

We compare all three approaches to incident prediction for a one month interval. Table 4.17 shows the comparison in the order in which the estimation methods is presented. The results are obtained using incident data from April 2008 through August 2012, to predict incidents for September 2012. Starting with the left-most column in Table 4.17 we show the cluster IDs. The actual number of incidents produced by each cluster representative is shown to the right of the column of cluster IDs. Column two shows the actual incidents per cluster. Column three shows the number of incidents estimated for each cluster using the Scalar method. In column four we show the incident prediction relative error for the cluster representatives. Columns five and six show the estimated incidents for each cluster and the relative prediction error respectively, for the Linear function method. Table 4.18 shows the relative prediction error of 0.0585 using the All products method, calculated from the estimated number of incidents for all 156 products using this method.

- For the Scalar method, the predicted number of incidents per cluster representative, scaled by the number of products in each cluster, is shown along with the relative prediction error per cluster. The relative prediction error of 0.0843 is determined by the sum of the estimated number of incidents per cluster (78206) and the actual number of incidents per cluster (72127).

- For the Linear function method, the predicted number of incidents per cluster representative, transformed by the linear function described earlier, is shown with the relative prediction error per cluster. The error of 0.0026 is based on estimated total incidents (72312) versus the actual number of incidents per cluster (72127).

- For the All products method, the estimated number of incidents (77491) and the sum of the actual number of incidents (72127) are used to calculate a relative error of 0.0585.

- Overall in Tables 4.17 and 4.18, we observe the Linear function estimation method results in the most accurate prediction of incidents. The All products method is the second best, and the Scalar method is the least accurate of the three estimation methods. We will investigate the quality of prediction beyond one month in the validation section.

We conclude our analysis with a comparison of the incident estimation methods, and discuss the trade-off of effort that is necessary with the more accurate incident prediction method. In terms of overall prediction accuracy, the Linear function method is more accurate than both the Scalar method and the All products method when predicting incidents one month out for the month of September 2012. Additionally, per-cluster incident prediction accuracy with the Linear function method is generally better than with the Scalar method. A more extensive investigation of the All products incident estimation method is presented in our previous work [11], where help desk incidents are estimated for 156 products using 22 cumulative incident data sets. For each data set, incidents are predicted at five prediction intervals (one month, two months, through five months). The median of relative prediction error found at the one month predictions was 0.0582 using the All products method. In this case study the All products method resulted in a relative prediction error of 0.0585 which is very close to the median error from our previous work. As discussed earlier, the Scalar method uses only one variable to estimate the number of incidents for all members of a cluster based only on incidents predicted by the cluster representative, so we expect less accurate predictions than what is produced by the Linear function method which uses two variables.

4.5.1.8 Cost Data Analysis

The cost model validated in this case study is developed using monthly incident data from product clusters and help desk technician labor costs. Using linear regression, a function is produced which relates per-cluster incident predictions to the cost to resolve the incidents. The role of the cost prediction model is shown in Figure 4.7. Per-cluster incident predictions from either the scalar model or the linear function model are inputs to the cost prediction model. Cost is predicted using linear regression.

The actual monthly labor costs obtained from the help desk managers in this case study is a set of monthly labor figures to resolve all incidents from all products used in the organization. As explained earlier, we applied a set of criteria to select the 156 products. Our cost model is based on estimated incidents from the 156 products, and predicts costs to resolve incidents from only the 156 products. In order to validate the cost model we need a way to identify the amount of monthly labor attributable to them. We investigate a proportion method in which we assume the amount of labor to resolve incidents from the 156 products is based on the ratio of the monthly number of incidents produced by the 156 products, to the total number of incidents produced by all products in the month.

Let I_c be the number of incidents generated by the set of clustered products at time t.

Let I be the number of incidents generated by all products at time t.

Then k_c, the proportion of incidents from the clustered products at t is given as

$$k_c = I_c/I \tag{4.5.8}$$

Let R be the cost to resolve I at time t.

Then R_c, the cost to resolve I_c at t is given as

$$R_c = k_c R \tag{4.5.9}$$

For the cost model, we develop a set of linear functions to relate the number of incidents predicted by all members of a cluster to the cost associated with resolving incidents produced by the members of the cluster. The incident predictions can be determined through either the Scalar or Linear Function methods, as shown in Figure 4.7.

Let $A_{c_i}(t)$ be the actual incident counts for time frame t for cluster c_i $(i = 1, \ldots, n)$, where n is the number of clusters.

Then $\bar{A}_c$, the vector of actual incident counts for time frame t is given as

$$\bar{A}_c = \begin{bmatrix} A_{c_1} \\ A_{c_2} \\ \vdots \\ A_{c_n} \end{bmatrix} \tag{4.5.10}$$

Let $R_c(t)$ be the actual cost of resolving incidents in $\bar{A}_c$ for time frame t. We perform linear regressions of the form

$$R_c = k_c R = \bar{A}_c \bar{P} + q \tag{4.5.11}$$

to determine coefficients $p_i \in \bar{P}$ $(i = 1, \ldots, n)$, where n is the number of clusters, and q is the intercept.

Let R_{c_i} be the cost to resolve the estimated number of incidents in cluster c_i.

$$R_{c_i} = p_i N_{c_i} \tag{4.5.12}$$

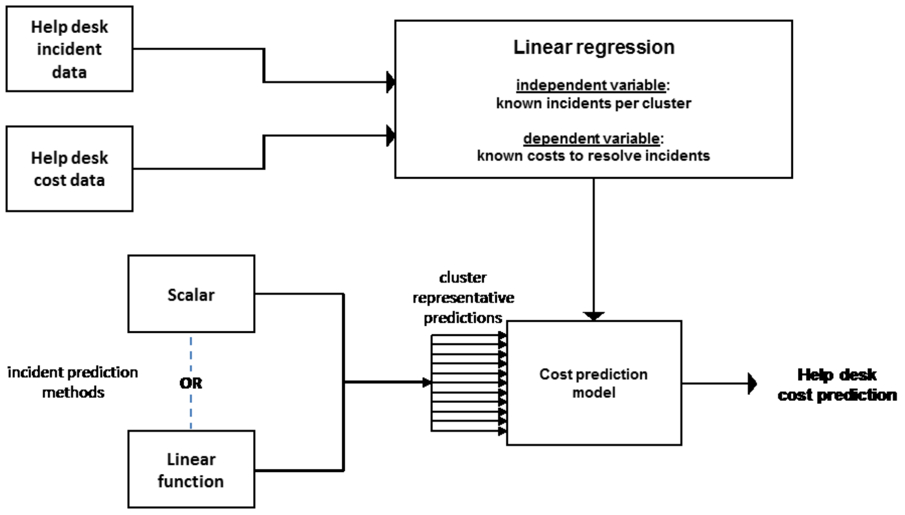

Help desk incident data
Help desk cost data
Linear regression
independent variable: known incidents per cluster
dependent variable: known costs to resolve incidents
Scalar
OR
Linear function
incident prediction methods
cluster representative predictions
Cost prediction model
Help desk cost prediction

Figure 4.7: Help desk cost prediction

Then R, the cost to resolve incidents in all clusters is given as

$$R = q + \sum_{i=1}^{n} R_{c_i} \tag{4.5.13}$$

We can use Equation 4.5.13 to predict the total cost of resolving incidents based on the predicted incidents for each cluster. Table 4.19 shows one month cost estimations for September 2012. Results from incident predictions using the Scalar and Linear function methods are show side by side for comparison. Table 4.19 shows the cluster IDs on the left. In column 2 we show the coefficients and intercept obtained from the linear regression described above. The coefficients are the components of $\bar{P}$ in Equation 4.5.11. The intercept is q. Column s shows the actual total cost of resolving incidents produced by the 156 incidents. The total cost is a cumulative number (total cost through August 2012 to resolve the cumulative incidents produced by the 156 products through August 2012). Columns 3 and 4 show the cost predictions when incidents are estimated using the Scalar method. Columns 5 and 6 show the cost predictions when incidents are estimated using the Linear function method. For both methods, the estimated incidents per cluster are the same as in Table 4.17 since we are predicting incidents and cost for the same month (September 2012). For both incident prediction methods in Table 4.19 we show the estimated cost (labor hours) for each cluster. These numbers are calculated using Equation 4.5.12. The total estimated cost is calculated using Equation 4.5.13. We now proceed to a discussion of the results of our cost predictions.

Table 4.19 shows for each cluster actual labor cost, estimated labor cost, and relative error for both the scalar and linear method of incident estimation. The sum of the per-cluster cost components and the intercept produce the overall cost estimate to resolve the predicted number of incidents. The cost prediction relative error of 0.0165 using the Linear incident estimation method is much better than the cost prediction error of -0.2703 obtained using the Scalar incident prediction method. These results are not surprising since cost predictions are based on the estimates from the incident estimation method (Scalar versus Linear function). Since the Linear function method produced more accurate incident estimations, we expect more accurate cost predictions compared to the scalar method.

Table 4.19: Cost prediction relative error for one month prediction (September 2012)

		Scalar method		**Linear function method**	
Cluster ID	**Actual cost (labor hours)**	**Estimated cost (labor hours)**	**Relative error**	**Estimated cost (labor hours)**	**Relative error**
BU	40822	237	-0.2703	646	0.0165
BV		3769		8577	
DA		2136		3138	
DV		5420		5554	
EF		4627		3109	
EG		3721		8140	
EH		5331		1707	
EI		1031		486	
EK		1091		3977	
ER		79		96	
EW		2158		5828	
intercept		192		192	
Total		**29792**		**41496**	

4.6 Validation

The method presented so far use one month predictions to estimate the number of incidents at the help desk and the labor effort required to deal with them. The results were shown to be generally quite good, but only cover a single month. We now turn to evaluate their performance when used over a longer period of time, i.e.

- RQ How stable are one month incident predictions when used over a long period of time?

4.6.1 Incident Estimation

To answer this research question, we produced a set of one month incident predictions over the 22 month time period of January 2012 through October 2013. To address prediction stability over subsequent months, we apply the three estimation methods to one month incident predictions for all 22 months and observe the range of relative error for each estimation method. Additionally, we compare the difference in prediction stability when the predictions are limited to R^2 GOF thresholds of 0.95 and 0.99. For each approach, we look for median relative error closest to zero as an indicator of prediction accuracy. We also look for the approach with the smallest interquartile range (IQR) of relative error as an indicator of prediction stability. Finally, we look for the approach with the smallest range of outliers as an additional indicator of prediction stability.

Figures 4.8 and 4.9 show relative incident estimation error for the three estimation approaches, at GOF thresholds of 0.95 and 0.99 for the SRGM that results in the best GOF at each of the 22 one month periods. A numerical summary of the box plots in Figures 4.8 and 4.9 is shown in Table 4.20. Figure 4.8 shows the range of relative incident prediction error from the three estimation methods when the predictions are limited to a GOF greater than or equal to 0.95. The All products approach results in a median relative prediction error of -0.0025 with first and third quartiles at -0.0292 and 0.0217 respectively. Although this method shows stability with median relative error centered close to zero, it results in outliers as low as -0.3636 and as high as 0.6829. This wide range of error suggests instability in prediction accuracy over the 22 months of predictions. The Scalar approach results in a median prediction error of -0.5114, with first and third quartiles at -0.6172 and 0.0846 respectively, and outliers at -0.8184 and 2.0770. The Linear function approach shows a median prediction error at -0.0648, first and third quartiles at -0.1645 and 0.0421 respectively, and outliers at -0.7642 and 0.2071. Of the three incident prediction approaches at $R^2 \geq 0.95$ in Figure 4.8 and summarized in Table 4.20, the All products approach has the best median prediction accuracy (median relative error closest to zero) followed by the linear function approach (method 3) as second-best and the scalar approach with the least prediction accuracy. The All products approach also results in the smallest interquartile range, again followed by Linear function then scalar. In terms of the overall range of outliers at $R^2 \geq 0.95$, Linear function results in the smallest range followed by All products then Scalar.

Turning our attention to $R^2 \geq 0.99$ in Figure 4.9, we investigate incident prediction stability for the more restrictive case in which predictions with GOF no less than 0.99 are included. As with $R^2 \geq 0.95$, the more accurate prediction comes from the all products approach, followed by Linear function method, then the Scalar method. An investigation of interquartile range at $R^2 \geq 0.99$ shows us the same order as with $R^2 \geq 0.95$. The All products method results in the smallest interquartile range, followed by the Linear function method and then the Scalar method. In terms of the overall range of outliers at $R^2 \geq 0.99$, the Linear function method results in the smallest range followed by the All products method and then the Scalar method. Although this same rank order of median accuracy, IQR and outlier range was observed with $R^2 \geq 0.95$, the range of outliers is smaller with all three approaches for $R^2 \geq 0.99$ as compared with $R^2 \geq 0.95$. From this we observe the more restrictive GOF threshold also excludes a greater number of the less accurate predictions.

For $R^2 \geq 0.99$, an interpretation of prediction error tells us the Linear function approach results in the most stable predictions over the 22 month period of one month predictions since the spread of prediction error is the smallest of all three approaches. Additionally, at $R^2 \geq 0.99$, incident prediction stability over the 22 month period is greater than at $R^2 \geq 0.95$ through the exclusion of a higher number of less accurate predictions.

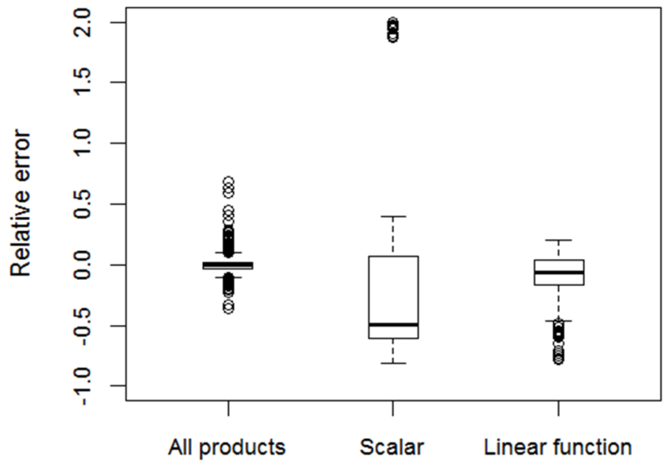


Figure 4.8: One month incident estimation relative error: comparison between three estimation methods: $R^2 \geq 0.95$

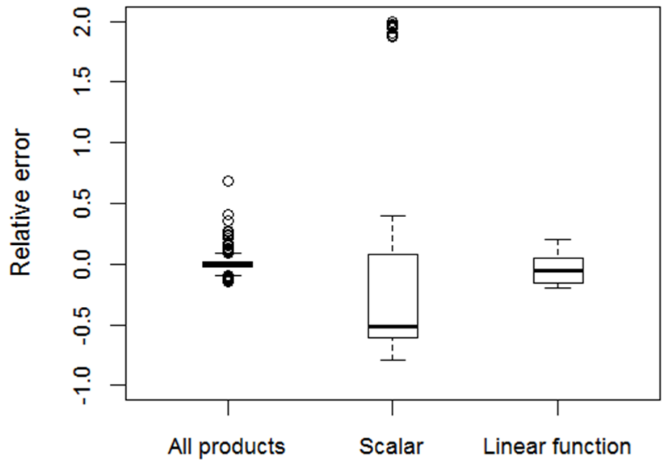


Figure 4.9: One month incident estimation relative error: comparison between three estimation methods: $R^2 \geq 0.99$

Table 4.20: Summary of one month incident estimation relative error at $R^2 \geq 0.95$

Estimation method	R^2 **>= 0.95**				
	Min.	**1st Qu.**	**Median**	**3rd Qu.**	**Max**
1 All products	-0.3636	-0.0292	-0.0025	0.0217	0.6829
2 Scalar	-0.8184	-0.6172	-0.5114	0.0846	2.0770
3 Linear function	-0.7642	-0.1645	-0.0648	0.0421	0.2071

Table 4.21: Summary of one month incident estimation relative error at $R^2 \geq 0.99$

Estimation method	R^2 **>= 0.99**				
	Min.	**1st Qu.**	**Median**	**3rd Qu.**	**Max**
1 All products	-0.1479	-0.0228	-0.0018	0.0223	0.6829
2 Scalar	-0.7889	-0.6128	-0.5117	0.0935	2.0770
3 Linear function	-0.1986	-0.1506	-0.0511	0.0459	0.2070

Table 4.22 ranks the three estimation methods in terms of median prediction accuracy, IQR, and outlier range, using the data in Tables 4.20 and 4.21. The rankings are the same for both GOF thresholds. In Table 4.22, we observe the all products estimation method predicts incidents more accurately than the Linear function approach. The Scalar method is the least accurate. We see the same rank order of estimation methods for IQR, where smaller IQR indicates less spread and therefore more stability. The outlier range is smallest for the Linear function method and largest for the Scalar method. The Scalar method results in the least prediction stability, as was observed with prediction accuracy and IQR.

In the introduction we listed a research question (RQ2) about achieving prediction accuracy through the selection of product clusters rather than analyzing all 156 products. Essentially we are asking if we can eliminate the All products incident estimation method and still predict incidents accurately. To help answer this question, we add up the number of operations required to produce an estimate for each of the estimation methods, and compare the results. If a method requires less work to produce an accurate estimate, that

Table 4.22: Incident estimation method comparison by one month median prediction accuracy, IQR and outlier range

Median prediction accuracy		
best		**worst**
All products	Linear function	Scalar

IQR		
smallest		**largest**
All products	Linear function	Scalar

Outlier range		
smallest		**largest**
Linear function	All products	scalar

method would be preferred by help desk practitioners. In Table 4.23 we present the number of operations required for each method to achieve the goal of incident estimation for all 156 products. We analyze the methods in terms of the number of predictions and number of processing steps that are unique to the method. The All products incident estimation method requires the most number of operations because it requires running predictions for all products. The Scalar method requires the least number of operations and results in the greatest number of operations saved compared to the All products method. The Linear function method requires about twice the number of operations as the Scalar method but still saves 71% compared to the All products method. In terms of processing effort alone, we can reject the All products method and concentrate on the other two methods.

Table 4.23: Comparison of estimation efficiency

	Method 1	**Method 2**	**Method 3**
	All products	**Scalar**	**Linear function**
Operations	156 predictions	11 predictions	11 predictions
	1 summation	11 scalar products	11 incident lookups
		1 summation	11 linear regressions
			11 linear functions
			1 summation
total # operations	**157**	**23**	**45**
Relative estimation efficiency *	**100%**	**85%**	**71%**

* compared to All products method

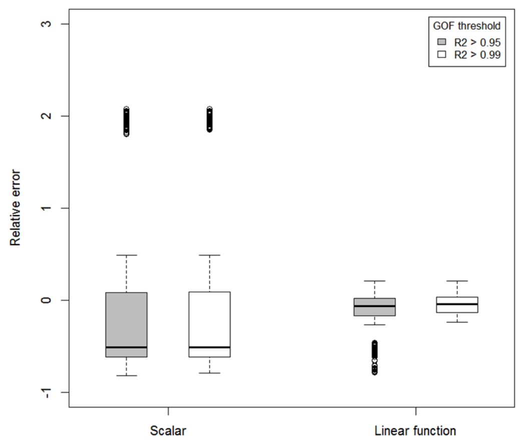


Figure 4.10: Help desk one month incident estimation relative error by incident estimation method

In Figure 4.10, incident estimation error from one month predictions is compared for the scalar and linear function incident estimation methods. We observe the scalar estimation method results in a median error less than zero. Overall, the linear function method shown in Figure 4.10 indicates improved prediction accuracy compared to the scalar method, as previously observed. As with the scalar method, there is little difference in terms of prediction accuracy between the two GOF thresholds.

4.6.2 Effort estimation

In the final part of our validation we analyze the stability of one month cost predictions over a 22 month period. In the approach section we made cost predictions for a single month. Because help desk managers routinely reestimate costs on a monthly basis, we need to ensure our technique produces stable results when applied over time. Therefore, we make a set of 22 one month predictions and analyze the results. As with our analysis of incident prediction stability we look for cost prediction median relative error closest to zero as an indicator of prediction accuracy. We also look for the approach with the smallest interquartile range of relative error as an indicator of cost prediction stability.

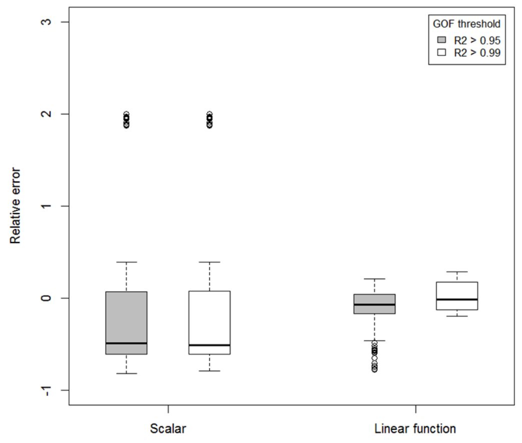


Figure 4.11: Help desk one month labor cost estimation relative error by incident estimation method

Table 4.24: Summary of one month labor cost estimation relative error by incident prediction method

Pred method	R^2 >= 0.95					
	Min	1st Q	Med	Avg	3rd Q	Max
Scalar	-0.8160	-0.6105	-0.4912	-0.1831	0.0735	1.9990
Lin fun	-0.7773	-0.1668	-0.0668	-0.0972	0.0421	0.2071

Pred method	R^2 >= 0.99					
	Min	1st Q	Med	Avg	3rd Q	Max
Scalar	-0.7889	-0.6075	-0.5105	-0.1595	0.0758	1.9990
Lin fun	-0.1930	-0.1250	-0.0110	0.0224	0.1745	0.2861

The set of incident estimates from one month predictions over a span of 22 months results in a range of relative error, as shown in the box plots in Figure 4.11. Since incident estimations are the basis of cost predictions, we can expect to find cost prediction error. Similar to the validation of one month incident prediction accuracy and stability, we investigate cost prediction accuracy and stability over a longer period of time.

The box plots in Figure 4.11 show the range of cost prediction error when the Scalar and Linear function incident estimation methods are used to predict incidents. The results of each method are shown when SRGM curve fit results are limited to the two R^2 thresholds of 0.95 and 0.99. Table 4.24 shows the descriptive statistics for cost prediction results.

Table 4.25: One month cumulative cost estimations: Scalar method $R^2 \geq 0.99$

Mo	**Est. inc**	**Actual inc**	**Inc pred RE**	**Est effort (hrs)**	**Act effort (hrs)**	**Effort pred RE**	**Est cost ($K)**	**Act cost ($K)**	**% var**
1/12	39447	64135	-0.613	89364	34072	1.615	$8,936	$3,418	161%
2/12	39478	65397	-0.519	92257	35293	1.614	$9,226	$3,529	161%
3/12	39507	66661	1.897	17174	36411	-0.528	$1,717	$3,641	-53%
4/12	39538	67804	-0.748	97416	37255	1.615	$9,742	$3,725	162%
5/12	39568	68888	-0.635	17983	38197	-0.529	$1,798	$3,820	-53%
6/12	39599	70042	-0.783	102100	39146	1.615	$10,210	$3,905	161%
7/12	39629	71192	0.094	19599	40039	-0.511	$1,960	$4,004	-51%
8/12	39660	72127	0.084	19291	40822	-0.527	$1,929	$4,082	-53%
9/12	39691	73207	-0.529	19600	41662	-0.530	$1,960	$4,166	-53%
10/12	39721	74149	-0.785	16682	42503	-0.608	$1,668	$4,250	-61%
11/12	39752	74880	1.999	128612	42885	1.999	$12,861	$4,289	200%
12/12	39782	76249	-0.094	41983	43774	-0.041	$4,198	$4,377	-4%
1/13	39813	77995	-0.789	21208	44879	-0.527	$2,121	$4,488	-53%
2/13	39844	79650	-0.593	16037	45677	-0.649	$1,604	$4,568	-65%
3/13	39872	81304	2.007	44539	46459	-0.041	$4,454	$4,646	-4%
4/13	39903	82956	-0.715	16615	47134	-0.648	$1,662	$4,713	-65%
5/13	39933	84210	-0.038	16821	47828	-0.648	$1,682	$4,783	-65%
6/13	39964	85579	-0.215	46638	48618	-0.041	$4,664	$4,862	-4%
7/13	39994	86685	-0.512	53191	49443	0.076	$5,319	$4,944	8%
8/13	40025	87668	-0.545	17612	50101	-0.649	$1,761	$5,010	-65%
9/13	40056	88781	0.858	9454	50883	-0.814	$945	$5,088	-81%
10/13	40086	89922	0.475	49376	51492	-0.041	$4,938	$5,148	-4%

As with incident predictions over 22 one month periods, the Scalar method results in a wider range of outliers compared to the Linear function method. For the Scalar method, the range is 2.8150 for $R^2 \geq 0.95$ and 2.7879 for $R^2 \geq 0.99$. The Linear function method results in a range of 0.9844 for $R^2 \geq 0.95$ and 0.4971 for $R^2 \geq 0.99$. The differences in the outlier ranges as well as the interquartile ranges across the two incident prediction methods can be seen in Figure 4.11. These results indicate cost predictions over the 22 month period are more stable with the Linear function method. The results also indicate little difference in cost prediction stability between the two R^2 thresholds with the Scalar method, but greater stability with the Linear function method at $R^2 \geq 0.99$ versus 0.95. In terms of the median cost prediction relative error, there is not much difference between the two R^2 values using the Scalar method (-0.4912 with $R^2 \geq 0.95$ and -0.5105 with $R^2 \geq 0.99$). There is a more noticeable difference in median cost prediction error with the

Table 4.26: One month cumulative cost estimations: Linear function method $R^2 \geq 0.99$

Mo	**Est. inc**	**Actual inc**	**Inc pred RE**	**Est effort (hrs)**	**Act effort (hrs)**	**Effort pred RE**	**Est cost ($K)**	**Act cost ($K)**	**% var**
1/12	69343	64135	0.081	40926	34072	0.198	$4,093	$3,418	20%
2/12	52833	65397	-0.192	44608	35293	0.264	$4,461	$3,529	26%
3/12	56203	66661	-0.157	42622	36411	0.171	$4,262	$3,641	17%
4/12	59862	67804	-0.117	30065	37255	-0.193	$3,006	$3,725	-19%
5/12	76648	68888	0.113	32903	38197	-0.139	$3,290	$3,820	-14%
6/12	82338	70042	0.176	39921	39146	0.022	$3,992	$3,905	2%
7/12	65149	71192	-0.085	38209	40039	-0.046	$3,821	$4,004	-5%
8/12	72315	72127	0.003	41495	40822	0.017	$4,150	$4,082	2%
9/12	62182	73207	-0.151	34822	41662	-0.164	$3,482	$4,166	-16%
10/12	59423	74149	-0.199	49596	42503	0.167	$4,960	$4,250	17%
11/12	69766	74880	-0.068	50407	42885	0.175	$5,041	$4,289	18%
12/12	79749	76249	0.046	53390	43774	0.220	$5,339	$4,377	22%
1/13	89234	77995	0.144	52156	44879	0.162	$5,216	$4,488	16%
2/13	78226	79650	-0.018	45070	45677	-0.013	$4,507	$4,568	-1%
3/13	98134	81304	0.207	38560	46459	-0.170	$3,856	$4,646	-17%
4/13	78666	82956	-0.052	46948	47134	-0.004	$4,695	$4,713	0%
5/13	71960	84210	-0.146	59394	47828	0.242	$5,939	$4,783	24%
6/13	81206	85579	-0.051	46330	48618	-0.047	$4,633	$4,862	-5%
7/13	89605	86685	0.034	43263	49443	-0.125	$4,326	$4,944	-13%
8/13	77905	87668	-0.111	64435	50101	0.286	$6,443	$5,010	29%
9/13	74001	88781	-0.167	43113	50883	-0.153	$4,311	$5,088	-15%
10/13	92972	89922	0.034	49914	51492	-0.030	$4,991	$5,148	-3%

linear function method, compared to the Scalar method. The linear function method results in a median cost prediction error of -0.0668 for $R^2 \geq 0.95$ and -0.0110 for $R^2 \geq 0.99$. These relative error values indicate greater cost prediction accuracy compared to using the Scalar method.

We turn to an analysis of the cost estimation results from the perspective of help desk managers. Tables 4.25 and 4.26 show cost predictions for each month using the Scalar and Linear function incident estimation approaches. For brevity we limit the results to cost predictions obtained at the $R^2 \geq 0.99$ threshold. Starting with the left column in Tables 4.25 and 4.26 we show the calendar month and year for which one month cost predictions are made. We include the estimated and actual number of incidents, as well as incident prediction relative error, in columns 2, 3 and 4 for each month. The numbers for estimated and actual incidents are cumulative. We include values for incident estimations since they are the basis of cost estimations. The values for effort estimation and actual monthly help desk effort are shown in columns 5 and 6. The relative error of the effort predictions is shown in column 7. The estimated costs in column 8 are calculated from an assumed hourly rate of $100. The values in column 5 are multiplied by this rate to arrive at the estimated cost numbers in column 8. Similarly, the values in column 6 are multiplied by the same hourly rate to arrive at the actual cost numbers in column 9. Column 10 shows the percent variance between the estimated cost in column 8 and the actual cost in column 9. Cost variance is used routinely by the help desk managers in this case study to assess how far off they are each month in their projection of costs.

Table 4.25 shows cost estimations and percent variance when the Scalar method is used for one month cost predictions. The range of cost variance is -81% to 200%. The mean and median cost variance is 5.47% and -51.90%, respectively. From the perspective of the help desk manager, these results mean costs are under-predicted by as much as 81% and over-predicted by nearly 200%. On average, the manager can rely on cost predictions to vary by a little over 5%. While the average monthly cost prediction variance may seam reasonable, predictions that vary at the extreme ends of the range are not useful for predicting help desk costs. We observe the months for which some of the higher cost variances result are the end of the year and the beginning of next year. The organization in this case study has a planned shutdown in the last week of December. Additionally, many employees take vacation late in the year. This results in a reduced number of incidents submitted to the help desk. These conditions may explain the resulting overestimation of costs. We recommend taking unusual months into account when estimates are made and to adjust effort estimates accordingly.

Table 4.26 shows cost estimations and percent variance when the Linear function method is used for one month cost predictions. The range of cost variance is -19% to 29%. The mean and median cost variance is 3.82% and 0.63%, respectively. Compared to the Scalar method, the Linear function method results in more stable one month cost predictions. The mean and median predictions are also an improvement over the Scalar method. As with the Scalar method, we see the high and low ends of the cost prediction range tend to occur around the months at the end of the year and the beginning of the next year.

Consultation with the help desk managers in this case study indicates an average cost prediction variance of less than 5% would be a improvement over what actually occurs in their environment. While historical data for monthly cost estimation was not available for comparison with our results, we confirmed help desk monthly cost prediction variance is usually greater than 10%. Based on this information, cost predictions using either the Scalar or Linear function method would be an improvement over current cost prediction variance, but the extreme under-estimations and over-estimations would likely be cause for rejection of the Scalar method in favor of the Linear function method.

4.7 Lessons Learned and Conclusions

4.7.1 Threats to validity

We analyze the validity of our case study according to the approach taken by Runeson et. al. in their presentation of case study research [92]. *Construct validity* addresses the extent to which operational measures of a case study are accurately represented in the research questions. To ensure our research questions captured the operational aspect of cost estimation, we engaged the help desk managers through discussions focused on daily operations specific to the incident resolution. *Internal validity* concerns an awareness of all factors which influence causal relationships. In this cases study, incidents drive help desk costs. In our discussion of the three categories of help desk incidents, we were clear about our knowledge of the separation of labor between the categories. However, our assumption about relationship between labor associated with the set of software products we investigated and labor for incidents from full product portfolio is not actually used in the industrial setting of this case study since labor is not accounted for by product. *External* *validity* reflects the generalizability of our findings, and the extent to which our conclusions are useful to others. We limited our study to products which run on the Windows operating system since the vast majority of products managed at the help desk run on Windows. Conclusions drawn from the SRGM we used to make incident predictions may not hold with non-Windows products. A replication of this case study using non-Windows products would be of value to further address this aspect of validity. This case study was conducted in a large multinational company which uses hundreds of different software products. Our findings may not be directly applicable to small companies with fewer software products. *Reliability* addresses how repeatable the results of a case study might be if conducted independently. We were careful to describe the incident data and scope of product distribution to the extent the confidentiality of this case study allowed, but the specific product types were not disclosed. Since product type, functionality, and the environment in which they are used influences operational behavior, a repetition of this case study may produce results that are different from ours. However, we could show the level of reliability for a rather large number of months.

4.7.2 Conclusions

This paper describes an approach to estimate incident volume and related labor cost for a product portfolio of an IT help desk. A case study in a major international corporation shows that the approach is quite successful in doing so, in spite of a number of assumptions that had to be made to deal with the limitations of available data. We discuss lessons learned at each process step then conclude with opportunities for future work. *Product selection* criteria established for minimum incident quantity and number of installations per product established for this step of the process resulted in 20% of the products generated 80% of the incidents. Our process accommodates adjustment of the product selection criteria in the event the product portfolio size grows or shrinks. We recommend careful attention to the minimum number of incidents per product, as this influences curve fit results. For

Incident data, the date and time data associated with incidents was critical to this case study. Construction of a parsing tool to produce cumulative incident datasets from raw help desk incident records facilitated data preparation for curve fitting. Since our incident estimation process accommodates 1) the addition of new products, and 2) the addition of more recent cumulative data, use of the parsing tool is recommended. With our *Cost data*, the assumption of cost apportionment to a subset of products could have been avoided had the help desk tracked labor by product, although the cost estimation techniques worked out well in spite of the assumption. Recurring use of the estimation techniques requires monthly updates to actual cost data. Arrangements for standard reporting of actual cost data would have avoided the manual aggregation of cost information obtained through contact with help desk managers. For *Cluster analysis* we were constrained by the limitations of the linear regression tool to no more than 16 clusters motivated a non-standard approach to dendrogram height selection to define clusters. Selection of a different regression tool that has less stringent limitations would facilitate a more standard approach in cluster selection. In spite of this constraint, we were successful in demonstrating incident prediction accuracy through the selection of product clusters vs. analyzing a full product portfolio in a prediction model. For *Incident estimates for representative product per cluster* we demonstrated how incident prediction accuracy can be obtained through the selection of cluster representatives (vs. analyzing all products) to answer RQ2 "Can incident prediction accuracy be obtained through the selection of product clusters vs. analyzing a full product portfolio in a prediction model?". We used PCA to determine the product which best represented the other products in a cluster on the basis of the number of shared principal components. For *Incident estimates for all products*, to answer RQ1 regarding the use of desktop software product reliability data from the help desk to predict future incident volume, we successfully demonstrated three incident prediction methods. The All products, Scalar and Linear function prediction methods were compared on the basis of prediction accuracy and effort to arrive a predictions. The Linear function method was shown to predict incidents more accurately that the Scalar method at one month prediction intervals. Although the All products method shows promise with incident prediction accuracy, it was rejected due to requiring an analysis of all 156 products. The Linear function and Scalar methods required less effort to predict incidents. For *Cost estimation* we successfully demonstrated that help desk labor data can be used to estimate costs of future incidents in response to RQ3. Linear regression was used to predict help desk labor from incident volume. The regression coefficients were used as parameters for the cost estimation model.

5 Future Work

5.1 Error Analysis

The approach taken with the large scale application of incident prediction techniques involved two metrics for assessing how incident data can be used to make cost predictions: 1) How well incident data was curve fit to SRGMs, measured using the R^2 GOF value, and 2) Prediction accuracy. In the SRGM selection process, the model with the highest R^2 was selected as shown in Figure 3.7. Incident predictions obtained from the selected model were later used as the basis for cost predictions. Although the selection was made independent of the resulting incident prediction accuracy, the overall cost prediction process illustrated in Figure 4.1 resulted in accurate predictions. These findings inspire interest in an analysis of incident prediction error vs. SRGM curve fit. The abundance of help desk incident data and the large portfolio of products afford an opportunity for sufficient data to measure the correlation between SRGM curve fit and prediction accuracy. The motivation for this future work is a potential refinement of the process shown in Figures 3.7 and 4.1 to select the best metric for model selection.

5.2 Product Portfolio Upgrades and Removals

The realities of a dynamic IT environment include frequent changes in the desktop software product portfolio. As business conditions change, new products are introduced and older ones are retired. Commercial software products frequently make new features available through upgrades. When an organization undergoes a change in its product portfolio, employees must adapt to new features and functionality. Minimally tested commercial products, or those tested under conditions that differ from the operational profile in which they are later deployed, result in latent failures. The combination of the learning curve employees experience and post-release failures drives help desk calls. Managers would benefit from a cost prediction process that models changes in the product portfolio. Similar to the benefits of cost estimations based on future incidents, estimating impacts to costs due to product upgrades and removals would be useful to help desk managers.

5.3 Cost Estimation Model with Refined Effort Data

The results of cost estimation using techniques validated in this thesis demonstrate a help desk cost model is feasible, in spite of several assumptions made regarding the cost data. Specifically, costs attributed to the subset of products chosen for the extended pilot study (pilot 4) and the large scale approach were estimated through linear regression. The accuracy with which costs were predicted under these assumptions motivates a refinement in the approach to actual help desk cost measurement and data collection. Should effort data be collected by product, fewer assumptions would need to be made regarding allocation of overall help desk labor to the set of products selected for investigation. A replication of the case study described in the large scale approach, using refined cost data, would likely produce a more accurate cost model.

5.4 Generalization to Other Domains

The techniques presented in this thesis were validated in an organization which predominately uses Windows products. Additionally, the organization is a large company with a wide variety of product types. IT services are internal to the company versus outsourced. The organization deals mainly with defense contracts. Although the large amount of help desk data facilitated validation of techniques in this research, organizations which produce fewer help desk incidents, possibly generated from products running on non-Windows devices such as Linux, could also benefit from a cost prediction model. A replication of the cases study presented in the large scale approach could be of interest to demonstrate scalability and any differences in prediction accuracy in other domains.

5.5 Prediction of Incident Severity

The organization in this research characterizes help desk incident severity through a combination of the Urgency, Impact and Priority attributes shown in Table A.1. An increase in any one of these three attributes indicates an increase in the effort necessary to resolve the incident, and thus an increase in cost. The ability to predict incident severity based on incident attributes, and possibly on historical high severity incidents with similar attributes, would supplement the cost model developed in this thesis.

Bibliography

[1] A. B. Costello and J. W. Osborne . Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis. *Practical Assessment, Research and Evaluation*, 10(7):1–9, 2005.

[2] R. Adnan, B. Graaf, A. van Deursen, J. Zonneveld, and J. Zonneveld. Using cluster analysis to improve the design of component interfaces. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, ASE '08, pages 383–386, Washington, DC, USA, 2008. IEEE Computer Society.

[3] R. Akkiraju, T. Lehman, N. Boyette, Haijing Fang, M. Lichtsinn, and Ben Shaw. On the role of analytics in estimating the cost of delivering complex information technology (it) outsourcing services projects. In *SRII Global Conference (SRII), 2012 Annual*, pages 705–714, July 2012.

[4] S.H. Aljahdali and M.E. El-Telbany. Software reliability prediction using multi-objective genetic algorithm. In *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, pages 293–300, May 2009.

[5] V. Almering, M. van Genuchten, G. Cloudt, and P.J.M. Sonnemans. Using software reliability growth models in practice. *Software, IEEE*, 24(6):82–88, Nov 2007.

[6] Beongku An and Symeon Papavassiliou. A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks. *Int. J. Netw. Manag.*, 11(6):387–395, November 2001.

[7] Carina Andersson. A replicated empirical study of a selection method for software reliability growth models. *Empirical Softw. Engr.*, 12(2):161–182, 2007.

[8] A. Andrews and J. Lucente. Predicting incident reports for it help desk. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, pages 678–683, June 2014.

[9] Anneliese Andrews, Philip Beaver, and Joseph Lucente. Towards better help desk planning: Predicting incidents and required effort. *Empirical Software Engineering*, page 4, submitted.

[10] Anneliese Andrews and Joseph Lucente. From incident reports to improvement recommendations: Analyzing it help desk data. In *Australasian Softwre Engineering Conference*, April 2014 (accepted).

[11] Anneliese Andrews and Joseph Lucente. On the viability of using srgms for it help desk incident predictions. In *Procs. International Conference on Software Quality, Reliability and Security*, page (submitted), August 2015.

[12] D. Anthony, E. Hines, D. Taylor, and J. Barham. A study of data compression using neural networks and principal component analysis [of pulmonary scintigrams]. In *Biomedical Applications of Digital Signal Processing, IEE Colloquium on*, pages 2/1–2/5, Nov 1989.

[13] George A. Barnett. Recent developments in the global telecommunication network. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 4435–4444, Jan 2012.

[14] Pankaj Bhawnani, B.H. Far, and G. Ruhe. Explorative study to provide decision support for software release decisions. In *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, pages 617–620, Sept 2005.

[15] Barry Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.

[16] A. Bose, A. Heching, and S. Sahu. A framework for model-based continuous improvement of global it service delivery operations. In *Services Computing, 2008. SCC '08. IEEE International Conference on*, volume 2, pages 197–204, July 2008.

[17] Michael Franklin Bosu and Stephen G. MacDonell. Data quality in empirical software engineering: A targeted review. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, EASE '13, pages 171–176, New York, NY, USA, 2013. ACM.

[18] Rodrigo A. Botafogo. Cluster analysis for hypertext systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, pages 116–125, New York, NY, USA, 1993. ACM.

[19] L.C. Briand, T. Langley, and I. Wieczorek. A replicated assessment and comparison of common software cost modeling techniques. In *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, pages 377–386, 2000.

[20] Ludwig M. Busse, Peter Orbanz, and Joachim M. Buhmann. Cluster analysis of heterogeneous rank data. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 113–120, New York, NY, USA, 2007. ACM.

[21] C. Wohlin, and A. Amschler Andrews. Prioritizing and assessing software project success factors and project characteristics using subjective data. *Empirical Software Engineering: An International Journal*, 8(3):285–308, 2003.

[22] David Cannon, David Wheeldon, Shirley Lacy, and Ashley Hanna. *ITIL Service Strategy*. TSO The Stationary Office, 2011.

[23] Eric Chen and Marvin V. Zelkowitz. Use of cluster analysis to evaluate software engineering methodologies. In *Proceedings of the 5th International Conference on Software Engineering*, ICSE ’81, pages 117–123, Piscataway, NJ, USA, 1981. IEEE Press.

[24] E. Condon, Michel Cukier, and Tao He. Applying software reliability models on security incidents. In *Software Reliability, 2007. ISSRE ’07. The 18th IEEE International Symposium on*, pages 159–168, Nov 2007.

[25] E. Condon, A. He, and Michel Cukier. Analysis of computer security incident data using time series models. In *Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on*, pages 77–86, Nov 2008.

[26] V.A. Danciu. Formalisms for it management process representation. In *Business-Driven IT Management, 2006. BDIM ’06. The First IEEE/IFIP International Workshop on*, pages 45–54, April 2006.

[27] Fernando De la Torre and Takeo Kanade. Discriminative cluster analysis. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML ’06, pages 241–248, New York, NY, USA, 2006. ACM.

[28] Thang Le Dinh and M. Leonard. A conceptual framework for modelling service value creation networks. In *Network-Based Information Systems, 2009. NBIS ’09. International Conference on*, pages 463–468, Aug 2009.

[29] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel. An empirical study of the effect of time constraints on the cost-benefits of regression testing. In *Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering*, pages 71–82. ACM, November 2008.

[30] H. Do and G. Rothermel. An empirical study of regression testing techniques incorporating context and lifetime factors and improved cost-benefit models. In *Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering*, pages 141–151. ACM, November 2006.

[31] H. Do and G. Rothermel. Using sensitivity analysis to create simplified economic models for regression testing. In *Proceedings of the International Symposium on Software Testing and Analysis*, pages 51–61, July 2006.

[32] J. Donovan and E. Murphy. Improvements in reliability-growth modeling. In *Reliability and Maintainability Symposium, 2001. Proceedings. Annual*, pages 296–301, 2001.

[33] J. Duchene and S. Leclercq. An optimal transformation for discriminant and principal component analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(6):978–983, Nov 1988.

[34] S.G. Eick, C.R. Loader, M.D. Long, L.G. Votta, and S.V. Wiel. Estimating software fault content before coding. In *Software Engineering, 1992. International Conference on*, pages 59–65, 1992.

[35] William Everett, Jr. Keene, S., and A. Nikora. Applying software reliability engineering in the 1990s. *Reliability, IEEE Transactions on*, 47(3):SP372–SP378, Sep 1998.

[36] M. Fanaeepour, L. Naghavian, and M.A. Azgomi. Modeling and evaluation of call centers with GSPN models. In *Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference on*, pages 619–622, May 2007.

[37] A.L. Goel. Software reliability models: Assumptions, limitations, and applicability. *Software Engineering, IEEE Transactions on*, SE-11(12):1411–1423, Dec 1985.

[38] A.L. Goel and K. Okumoto. Time-dependent error-detection rate model for software reliability and other performance measures. *Reliability, IEEE Transactions on*, R-28(3):206–211, 1979.

[39] S.S. Gokhale and K.S. Trivedi. Log-logistic software reliability growth model. In *High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International*, pages 34–41, Nov 1998.

[40] K. Gopalan and S. Mahil. Speaker identification and verification via singular value decomposition of speech parameters. In *Circuits and Systems, 1990., Proceedings of the 33rd Midwest Symposium on*, pages 725–728 vol.2, Aug 1990.

[41] K. Goseva-Popstojanova and T. Trivedi. Failure correlation in software reliability models. In *Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on*, pages 232–241, 1999.

[42] O. Grygorash, Yan Zhou, and Z. Jorgensen. Minimum spanning tree based clustering algorithms. In *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, pages 73–81, Nov 2006.

[43] S. Gulati and S.A. Malcolm. Call center scheduling technology evaluation using simulation. In *Simulation Conference, 2001. Proceedings of the Winter*, volume 2, pages 1438–1442 vol.2, 2001.

[44] R.C. Hampshire and W.A. Massey. Variational optimization for call center staffing. In *Diversity in Computing Conference, 2005 Richard Tapia Celebration of*, pages 4–6, Oct 2005.

[45] M.J. Harrold, D. Rosenblum, G. Rothermel, and E. Weyuker. Empirical studies of a prediction model for regression test selection. *Software Engineering, IEEE Transactions on*, 27(3):248–263, 2001.

[46] K. Henningsson, T. Birath, and P. Molin. A fault-driven lightweight process improvement approach. In *Euromicro Conference, 2003. Proceedings. 29th*, pages 343–350, Sept 2003.

[47] C.-J. Hsu and C.-Y. Huang. Optimal weighted combinational models for software reliability estimation and analysis. *Reliability, IEEE Transactions on*, PP(99):1–1, 2014.

[48] Chin-Yu Huang, Sy-Yen Kuo, and M.R. Lyu. Effort-index-based software reliability growth models and performance assessment. In *Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International*, pages 454–459, 2000.

[49] Chin-Yu Huang, Sy-Yen Kuo, and M.R. Lyu. An assessment of testing-effort dependent software reliability growth models. *Reliability, IEEE Transactions on*, 56(2):198–211, June 2007.

[50] Chin-Yu Huang, Chu-Ti Lin, Sy-Yen Kuo, M.R. Lyu, and Chuan-Ching Sue. Software reliability growth models incorporating fault dependency with various debugging time lags. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 186–191 vol.1, Sept 2004.

[51] Chin-Yu Huang, Chu-Ti Lin, and Chuan-Ching Sue. Software reliability prediction and analysis during operational use. In *Information Technology: Research and Education, 2005. ITRE 2005. 3rd International Conference on*, pages 317–321, June 2005.

[52] S. D. Hyman, T.P. Vogl, K.T. Blackwell, G. S. Barbour, J. M. Irvine, and D.L. Alkon. Classification of Japanese Kanji using principal component analysis as a preproces-

sor to an artificial neural network. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume i, pages 233–238 vol.1, Jul 1991.

[53] A. Iannino, J. D. Musa, K. Okumoto, and B. Littlewood. Criteria for software reliability model comparisons. *SIGSOFT Softw. Eng. Notes*, 8(3):12–16, July 1983.

[54] R. Ibrahim, P. L'Ecuyer, N. Regnard, and Haipeng Shen. On the modeling and forecasting of call center arrivals. In *Simulation Conference (WSC), Proceedings of the 2012 Winter*, pages 1–12, 2012.

[55] S. Inoue and S. Yamada. Change-point modeling for software reliability assessment depending on two-types of reliability growth factors. In *Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on*, pages 616–620, Dec 2010.

[56] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[57] A.K. Jain, R.P.W. Duin, and Jianchang Mao. Statistical pattern recognition: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, Jan 2000.

[58] Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley professional computing. Wiley, 1991.

[59] P. Jalote and B. Murphy. Reliability growth in software products. In *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*, pages 47–53, Nov 2004.

[60] Z. Joao, M. Mzyece, and Anish Kurien. Matrix decomposition methods for the improvement of data mining in telecommunications. In *Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th*, pages 1–5, Sept 2009.

[61] I.T. Jolliffe. *Principal Components Analysis*. Springer, 2002.

[62] Stephen H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002.

[63] Leonard Kaufman and Peter J. Rousseeuw. *Frontmatter*, pages i–xiv. John Wiley and Sons, Inc., 2008.

[64] V. Kaulgud and V.S. Sharma. Cdi: Cost of development index. In *Emerging Trends in Software Metrics (WETSoM), 2013 4th International Workshop on*, pages 80–83, May 2013.

[65] D. Kececioglu. *Reliability Engineering Handbook*, volume 2. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[66] P.A. Keiller and T.A. Mazzuchi. Enhancing the predictive performance of the goel-okumoto software reliability growth model. In *Reliability and Maintainability Symposium, 2000. Proceedings. Annual*, pages 106–112, 2000.

[67] Dong Seong Kim, Ha-Nam Nguyen, T. Thein, and Jong Sou Park. An optimized intrusion detection system using pca and bnn. In *Information and Telecommunication Technologies, 2005. APSITT 2005 Proceedings. 6th Asia-Pacific Symposium on*, pages 356–359, Nov 2005.

[68] B.A. Kitchenham. Systematic reviews. In *Software Metrics, 2004. Proceedings. 10th International Symposium on*, pages xii–xii, Sept 2004.

[69] Sy-Yen Kuo, Chin-Yu Huang, and M.R. Lyu. Framework for modeling software reliability, using various testing-efforts and fault-detection rates. *Reliability, IEEE Transactions on*, 50(3):310–320, Sep 2001.

[70] R. C T Lee, Y. H. Chin, and S.C. Chang. Application of principal component analysis to multikey searching. *Software Engineering, IEEE Transactions on*, SE-2(3):185–193, Sept 1976.

[71] Ang Li, Qing Gu, Guang-Cheng Feng, and Dao-Xu Chen. Snn - a neural network based combination of software reliability growth models. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pages 5109–5112, Dec 2009.

[72] P.L. Li, R. Nakagawa, and R. Montroy. Estimating the quality of widely used software products using software reliability growth modeling: Case study of an IBM federated database project. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 452–454, Sept 2007.

[73] Tao Liu and Lieli Liu. Research on forecasting call center traffic through pca and bp artificial neural network. In *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, volume 1, pages 444–447, Oct 2012.

[74] A.G. Malishevsky, G. Rothermel, and S. Elbaum. Modeling the cost-benefits tradeoffs for regression testing techniques. In *Software Maintenance, 2002. Proceedings. International Conference on*, pages 204–213, 2002.

[75] D. McCarthy. Automation of help desks using case-based reasoning. In *Case Based Reasoning: Prospects for Applications, IEE Colloquium on*, pages 9/1–9/3, 1994.

[76] S. Meskini, A.B. Nassif, and L.F. Capretz. Reliability models applied to mobile applications. In *Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on*, pages 155–162, June 2013.

[77] R.E. Mullen. The lognormal distribution of software failure rates: application to software reliability growth modeling. In *Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on*, pages 134–142, Nov 1998.

[78] J.D. Musa and A.F. Ackerman. Quantifying software validation: when to stop testing. *Software, IEEE*, 6(3):19–27, May 1989.

[79] John D. Musa, Anthony Iannino, and Kazuhira Okumoto. *Software Reliability: Measurement, Prediction, Application*. McGraw-Hill, Inc., New York, NY, USA, 1987.

[80] Vu Nguyen. Improved size and effort estimation models for software maintenance. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–2, Sept 2010.

[81] H. Niitsuma and Takashi Okada. Covariance and PCA for categorical variables. In *Advances in Knowledge Discovery and Data Mining*, volume 3518, pages 523–528. Springer Berlin Heidelberg, 2005.

[82] Magnus C. Ohlsson, Anneliese Amschler Andrews, and Claes Wohlin. Modelling fault-proneness statistically over a sequence of releases: a case study. *Journal of Software Maintenance and Evolution: Research and Practice*, 13(3):167–199, 2001.

[83] Magnus C. Ohlsson, Anneliese Von Mayrhauser, Brian McGuire, and Claes Wohlin. Code decay analysis of legacy software through successive releases. In *Proceedings of the IEEE Aerospace Conference*, pages 69–81, 1999.

[84] M.C. Ohlsson and C. Wohlin. Identification of green, yellow, and red legacy components. In *Procs. International Conference on Software Maintenance*, pages 6–15, 1998.

[85] B.J. Osteen, S. Ramanan, and K. Jeganathan. Optimizing the cost of software quality - a path to delivery excellence. In *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*, pages 754–756, April 2013.

[86] H. Pant and D.R. Jeske. Software reliability predictions for distributed software. In *Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on*, pages 11–21, Nov 1998.

[87] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[88] R. Peng, Q.P. Hu, and S.H. Ng. Incorporating fault dependency and debugging delay in software reliability analysis. In *Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on*, pages 641–645, Sept 2008.

[89] Andy Podgurski, Wassim Masri, Yolanda McCleese, Francis G. Wolff, and Charles Yang. Estimation of software reliability by stratified sampling. *ACM Trans. Softw. Eng. Methodol.*, 8(3):263–283, July 1999.

[90] Kimmo E. E. Raatikainen. Cluster analysis and workload classification. *SIGMETRICS Perform. Eval. Rev.*, 20(4):24–30, May 1993.

[91] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Torner. Comparing between maximum likelihood estimator and non-linear regression estimation procedures for NHPP software reliability growth modelling. In *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*, pages 213–218, Oct 2013.

[92] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164, apr 2009.

[93] Per Runeson, Martin Høst, Austen Ranier, and Bjørn Regnell. *Case Study Research in Software Engineering*. John Wiley & Sons, inc., 1st edition, 2012.

[94] Farhad Shahrokhi. An efficient flow routing algorithm to solve the maximum concurrent flow problem with applications to the packed switched telecommunication networks and cluster analysis (abstract). In *Proceedings of the 1986 ACM Fourteenth Annual Conference on Computer Science*, CSC '86, pages 494–, New York, NY, USA, 1986. ACM.

[95] Yanjun Shu, Zhibo Wu, Hongwei Liu, and Xiaozong Yang. Considering fault correction lag in software reliability modeling. In *Dependable Computing, 2008. PRDC '08. 14th IEEE Pacific Rim International Symposium on*, pages 355–356, Dec 2008.

[96] Yanjun Shu, Zhibo Wu, Hongwei Liu, and Xiaozong Yang. Considering the dependency of fault detection and correction in software reliability modeling. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 2, pages 672–675, Dec 2008.

[97] He Shu-guang, Li Li, and Qi Er-shi. Study on the continuous quality improvement of telecommunication call centers based on data mining. In *Service Systems and Service Management, 2007 International Conference on*, pages 1–5, 2007.

[98] J. Singh and L.S. Maurya. Reliability assessment and prediction of open source software systems. In *Image Information Processing (ICIIP), 2013 IEEE Second International Conference on*, pages 6–11, Dec 2013.

[99] Vinay Singh, Vandana Bhattacherjee, and Sandeep Bhattacharjee. An analysis of dependency of coupling on software defects. *SIGSOFT Softw. Eng. Notes*, 37(1):1–6, January 2012.

[100] Dag I. K. Sjoberg, Tore Dyba, and Magne Jorgensen. The future of empirical methods in software engineering research. In *2007 Future of Software Engineering*, FOSE '07, pages 358–378, Washington, DC, USA, 2007. IEEE Computer Society.

[101] T. Sorsa, H.N. Koivo, and H. Koivisto. Neural networks in process fault diagnosis. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(4):815–825, Jul 1991.

[102] S.G. Steckley, S.G. Henderson, and V. Mehrotra. Performance measures for service systems with a random arrival rate. In *Simulation Conference, 2005 Proceedings of the Winter*, pages 566–575, 2005.

[103] E. Stoker and J. Bechta Dugan. When does it pay to make software more reliable? In *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*, pages 321–331, Nov 2003.

[104] C. Stringfellow and A. Andrews. Deriving a fault architecture to guide testing. *Software Quality Journal*, 10:299–330, 2002.

[105] C. Stringfellow and A. Andrews. An empirical method for selecting software reliability growth models. *Empirical Software Engineering*, 7:319–413, 2002.

[106] G. C. Sun and D.L. Chenoweth. Principal components applied to multi-layer perceptron learning. In *Artificial Neural Networks, 1991., Second International Conference on*, pages 100–102, Nov 1991.

[107] Jun Sun, Zhulong Wang, Hao Yu, Fumihito Nishino, Yukata Katsuyama, and Satoshi Naoi. Effective text extraction and recognition for www images. In *Proceedings of the 2003 ACM Symposium on Document Engineering*, DocEng '03, pages 115–117, New York, NY, USA, 2003. ACM.

[108] Mikael Svahnberg and Claes Wohlin. An investigation of a method for identifying a software architecture candidate with respect to quality attributes. *Empirical Software Engineering*, 10(2):149–181, 2005.

[109] S.M. Syed-Mohamad and T. McBride. A comparison of the reliability growth of open source and in-house software. In *Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific*, pages 229–236, Dec 2008.

[110] S.M. Syed-Mohamad and T. McBride. Reliability growth of open source software using defect analysis. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 2, pages 662–667, Dec 2008.

[111] T. M. Khoshgoftaar and D. L. Lanning . Are the principal components of software complexity stable across software products? In *Procs. International Symposium on Software Metrics*, pages 61–72, October 1994.

[112] T. M. Khoshgoftaar and E.B. Allen and N. Goel and A. Nandi and J. McMullan . Detection of software modules with high code debug churn in a very large legacy system. In *Procs. International Symposium on Software Reliability Engineering*, pages 364–371, October 1996.

[113] Y. Tamura and S. Yamada. Comparison of software reliability assessment methods for open source software. In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, volume 2, pages 488–492, July 2005.

[114] Y. Tamura and S. Yamada. Software reliability growth model based on stochastic differential equations for open source software. In *Mechatronics, ICM2007 4th IEEE International Conference on*, pages 1–6, May 2007.

[115] Min Tang, Bryan Pellom, and K. Hacioglu. Call-type classification and unsupervised training for the call center domain. In *Automatic Speech Recognition and Understanding, 2003. ASRU '03. 2003 IEEE Workshop on*, pages 204–208, 2003.

[116] B.K. Tannahill, C.E. Maute, Y. Yetis, M.N. Ezell, A. Jaimes, R. Rosas, A. Motaghi, H. Kaplan, and M. Jamshidi. Modeling of system of systems via data analytics: Case for big data in sos. In *System of Systems Engineering (SoSE), 2013 8th International Conference on*, pages 177–183, June 2013.

[117] Telephone call centers quarterly update, May 2014.

[118] J. Tian. Better reliability assessment and prediction through data clustering. *Software Engineering, IEEE Transactions on*, 28(10):997–1007, Oct 2002.

[119] Martin Trachtenberg. A general theory of software-reliability modeling. *Reliability, IEEE Transactions on*, 39(1):92–96, Apr 1990.

[120] N. Ullah and M. Morisio. An empirical analysis of open source software defects data through software reliability growth models. In *EUROCON, 2013 IEEE*, pages 460–466, July 2013.

[121] Julia Varnell-Sarjeant. *An Empirical Comparison of Reuse in Embedded and Nonembedded Systems*. PhD thesis, University of Denver, October 2013.

[122] Julia Varnell-Sarjeant, Anneliese Amschler Andrews, Joe Lucente, and Andreas Stefik. Comparing development approaches and reuse strategies: An empirical evaluation of developer views from the aerospace industry. *Journal of Information and Software Technology*, 61(0):71 – 92, 2015.

[123] Guan-Wei Wang, Chun-Xia Zhang, Jian Zhuang, and De-Hong Yu. Clustering based on sequential representation of minimum spanning tree. In *Wavelet Analysis and Pattern Recognition (ICWAPR), 2011 International Conference on*, pages 132–137, July 2011.

[124] U. Wattanachon and C. Lursinsap. Agglomerative hierarchical clustering for nonlinear data analysis. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 1420–1425 vol.2, Oct 2004.

[125] E.J. Weyuker. Empirical software engineering research - the good, the bad, the ugly. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 1–9, Sept 2011.

[126] Claes Wohlin and Anneliese Amschler Andrews. Analysing primary and lower order project success drivers. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, SEKE '02, pages 393–400, New York, NY, USA, 2002. ACM.

[127] Claes Wohlin and Anneliese Amschler Andrews. Evaluation of three methods to predict project success: a case study. In *Proceedings of the 6th international conference*

on Product Focused Software Process Improvement, PROFES'05, pages 385–398, Berlin, Heidelberg, 2005. Springer-Verlag.

[128] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering: An introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[129] A. Wood. Predicting software reliability. *Computer*, 29(11):69–77, Nov 1996.

[130] A. Wood. Software reliability growth models: assumptions vs. reality. In *Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on*, pages 136–141, Nov 1997.

[131] Victor Wu, R. Manmatha, and Edward M. Riseman. Finding text in images. In *Proceedings of the Second ACM International Conference on Digital Libraries*, DL '97, pages 3–12, New York, NY, USA, 1997. ACM.

[132] Huiqi Xu, Zhen Li, Shumin Guo, and Keke Chen. Cloudvista: Interactive and economical visual cluster analysis for big data in the cloud. *Proc. VLDB Endow.*, 5(12):1886–1889, August 2012.

[133] Shigeru Yamada, Mitsuru Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *Reliability, IEEE Transactions on*, R-32(5):475–484, 1983.

[134] Shigeru Yamada, Hiroshi Ohtera, and H. Narihisa. Software reliability growth models with testing-effort. *Reliability, IEEE Transactions on*, 35(1):19–23, April 1986.

[135] Shigeru Yamada and S. Osaki. Software reliability growth modeling: Models and applications. *Software Engineering, IEEE Transactions on*, SE-11(12):1431–1437, Dec 1985.

[136] M.C.K. Yang and Anne Chao. Reliability-estimation and stopping-rules for software testing, based on repeated appearances of bugs. *Reliability, IEEE Transactions on*, 44(2):315–321, Jun 1995.

[137] T.-J. Yu, V.Y. Shen, and H.E. Dunsmore. An analysis of several software defect models. *Software Engineering, IEEE Transactions on*, 14(9):1261–1270, Sep 1988.

[138] Carmen Zannier, Grigori Melnik, and Frank Maurer. On the success of empirical studies in the international conference on software engineering. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 341–350, New York, NY, USA, 2006. ACM.

[139] Xiaoni Zhang, Kellie B. Keeling, and Robert J. Pavur. Information quality of commericial web site home pages: An explorative analysis. In *Proceedings of the Twenty First International Conference on Information Systems*, ICIS '00, pages 164–175, Atlanta, GA, USA, 2000. Association for Information Systems.

[140] Jing Zhao, Hong wei Liu, Gang Cui, and Xiao-Zong Yang. A software reliability growth model from testing to operation. In *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, pages 691–694, Sept 2005.

[141] Yuxin Zhao, Chengcheng Wan, Feng Gao, and Shuai Chang. Change points estimation and simulation for software reliability model. In *Measurement, Information and Control (ICMIC), 2013 International Conference on*, volume 01, pages 434–438, Aug 2013.

[142] Kunlin Zhou and Rongsheng Guo. Research on process monitoring method based on SPC and PCA technology. In *Control and Decision Conference (CCDC), 2011 Chinese*, pages 691–694, May 2011.

Appendix A Help Desk Operations

We present an overview of help desk operations and the data used in this case study to establish motivation and context for our selection of related work. The primary goal of the help desk is to provide cost effective IT services to an organization. Amongst a wide variety of services ranging from infrastructure planning to the provisioning of IT assets, the help desk resolves problems that occur when using products in the desktop software portfolio. Each help desk request is recorded on an incident record. Help desk labor is managed through resource planning, tracking planned labor costs to actual costs, and adjusting staffing levels to balance service delivery commitments with labor budget targets. The basis of estimate for staffing levels is driven primarily by service delivery metrics rather than through help desk workload predictions. Help desk managers are expected to find ways to improve problem resolution efficiency to reduce costs.

There are three categories of incidents, as shown in Figure A.1. The first category consists of incidents submitted to report software product specific issues. For example, an employee contacts the help desk to report an error message produced by Adobe Reader. Help desk technician labor is spent resolving this category of incidents through a body of knowledge comprised of technician experience and scripted solutions. A second major category of incidents relates to defective hardware such as broken keyboards, crashed hard drives and problems that are not specific to any particular installed software product. This category also includes incidents associated with general operating system errors and computer performance problems. These types of issues are usually not resolved by addressing the functionality of any product other than the operating system itself or an un-

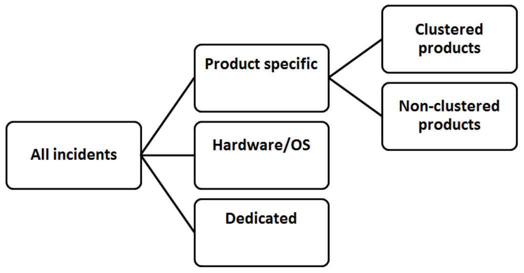


Figure A.1: Help desk incident resolution labor categories

derlying component such as the .Net framework. A third major category of incidents is designed to address issues for well-defined situations, through dedicated resolution mechanisms. Incidents in this category are typically routed to dedicated help desk team members through interactive voice prompts that ensure a trained help desk technician is assigned to the incident. Dedicated effort can be directed to products with high call volume through standard scripts or known resolution methods to achieve resolution efficiency. Business trends influence which products need dedicated support. For example, software products designed to assist employees in the selection of health benefits during a fixed time period of benefit selection may require an interactive telephone prompt system to offer a selection only for health benefit software problems. The quantity of incidents resolved through dedicated, call routed mechanisms is usually high compared to product-specific incidents. This is attributed to problems which have the potential to affect large numbers of employees. Software upgrades to highly distributed products such as email and time card software also influence the level of dedicated help desk effort.

Help desk labor is reported by technicians on a daily basis. Weekly and monthly reports are created. The weekly reports are generally reserved for internal help desk management reviews and are not distributed outside of the labor accounting department. Monthly roll ups are made available to permitted employees.

Incidents are submitted to the help desk by employees who experience problems with products and need help fixing problems or repairing product installations. Each incident record includes a unique identifier and 43 fields for incident attributes. Textual attributes are used to record problem descriptions, reported source (phone, web, chat), the geographic location and business unit of the employee who submitted the incident, product categories, a summary of the resolution, resolution categories, and the name of the organization in which the product is managed. Five attributes are included to record the date and time for incident submission, assignment, last updated time, and resolution events. Date time stamps on incident records for help desk events are recorded with precision to the minute. Table A.1 describes seventeen incident attributes that quantify the behavior of the incidents in terms of resolution urgency, relative priority, transfer and escalation activity and processing time. Table A.1 lists all seventeen attributes with their unique identifier, description, numerical range of values and their role as an input or output during incident processing. All seventeen attributes are mandatory fields in the help desk database. Attributes A1 and A2 are categorical attributes whose range of values consists of four discrete levels for incident urgency, impact and priority. Urgency is assigned by the help desk technician based on standard guidance which describes the amount of time by which resolution is expected based on general problem types. Impact is assigned through discussion with the employee who reports the problem based on the number of employees or devices affected by the unresolved issue. Incident priority is automatically determined through a mapping of the combination of Urgency and Impact to one of fourteen levels as indicated for attribute A3. The lowest priority incidents are assigned a value of zero. Attributes A4 and A5 are binary categorical levels assigned by the incident management system. A4 is assigned a value of 1 if the incident is resolved on the call or chat session initiated to report the problem. A4 is a service metric used as an indicator of the level of service provided by the help desk technician. In general, the help desk service model is designed around a maximization of the level of service and a minimization of cost and effort. A small number of incidents are excluded from First Call Resolution service metrics. A5 is used for this indicator. These are incidents for which specific business rules apply such as trouble alerts automatically generated through application health monitoring mechanisms. A set of six attributes describe the extent to which new incidents are similar to existing incidents, documented solutions, known problems and those related to specific artifacts. These are attributes A6 through A11. Attribute A6 is set to true if an incident matches any type of incident record in the system. Values for A7 through A11 are the number of existing help desk records with matching resolution methods. An increase in resolution efficiency achieved with effort and cost reduction is targeted through quick reference to other help desk records which assist in problem resolution, although no quantitative studies have been conducted to substantiate the benefits of incident matching. Metrics derived from attributes A6 through A11 may also be used to identify problems which are potentially resolvable through cost-saving self-service methods.

Attributes A12 and A13 measure impacts to the cost of resolving incidents. Some incidents are not resolvable by the technician providing assistance during the initial engagement with the help desk. These incidents must be transferred to subject matter experts. While first call resolution is a cost-based goal, resolution quality is often achieved through reassignment of responsibility to more knowledgeable individuals, at the expense of increasing resolution effort. The number of times an incident is transferred for reassignment is recorded in attribute A12. Managerial assistance is sometimes required to ensure incidents are resolved. This may become necessary for three reasons. 1) Incident volume may exceed the capacity of available technicians, 2) experience and knowledge of the technician may impact the speed and quality with which incidents are resolved, and 3) high priority incidents sometimes need managerial focus to restore productivity to the affected individuals or systems, especially with wide-scope service outages. Similarly, the priority of some incidents may demand the coordinated efforts of managers to bring in subject matter experts. Attribute A13 indicates whether or not an incident was escalated in that way for resolution. A14 and A15 are indicators of cost and effort associated with incident resolution. A16 is a per-incident measure of service restoration within time spans defined by incident priority. Not all incidents must meet a time span, as deemed by the help desk managers. A17 identifies whether they do or not.

Table A.1: Incident Attributes

Attr ID	Description	Range	Role
A1	Urgency	1=low, 2=med, 3=high, 4=crit	input
A2	Impact		
A3	Priority	(0, 3, 5, 9, 10, 13, 15, 18, 19, 20, 23, 24, 25, 29)	input
A4	Resolved through First Call Resolution (FCR)	0 = False, 1 = True	output
A5	Resolvable through First Call Resolution (FCR)	0 = False, 1 = True	input
A6	Incident Matches a help desk historical record		
A7	Number of Matches to any other incident	(0, 1, 2, ... , n)	input
A8	Number of Matches to any known solution		
A9	Number of Matches to any known error		
A10	Number of Matches to any known problem		
A11	Number of Matches to any configuration item		
A12	Total Transfers of Incident	(0, 1, 2, ... , n)	output
A13	Incident Escalated	0 = False, 1 = True	output
A14	Incident Response Time	(0, 1, 2, ... , n)	output
A15	Incident Resolution Time		
A16	Return to Productivity (RTP) Compliance	0 = False, 1 = True	output
A17	Return to Productivity (RTP) Eligibility	0 = False, 1 = True	input

The content, accuracy and context of the incidents are monitored for quality through involvement by help desk managers. Employees impacted by loss of productivity usually contribute to incident accuracy in the database through feedback to the help desk during the resolution process.

Appendix B   Case Study Data

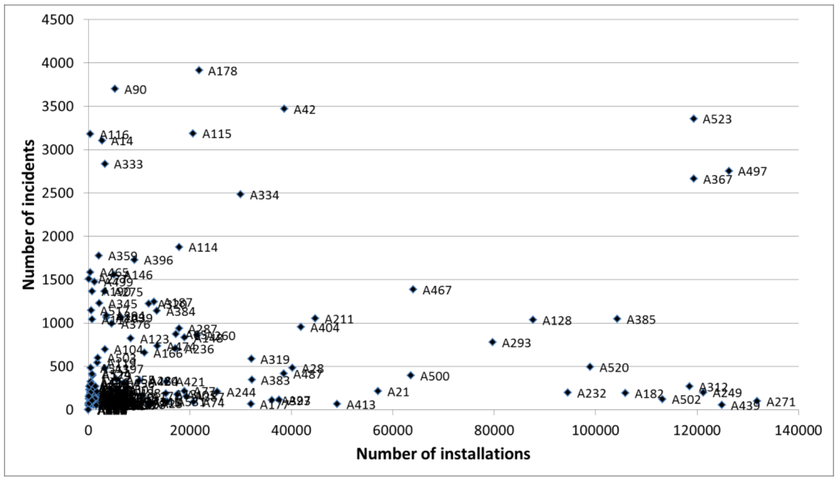
This case study describes a large scale IT operation in a large, multinational company. The IT operation services 800 products installed on over 120,000 desktop machines.

Twenty percent of the products in use by the organization in this case study are widely distributed to the majority of employee computers. These are mainly office productivity applications (email and word processing), Internet browsers, domain-specific business tools and security-related applications. We based our initial selection on the following product characteristics.

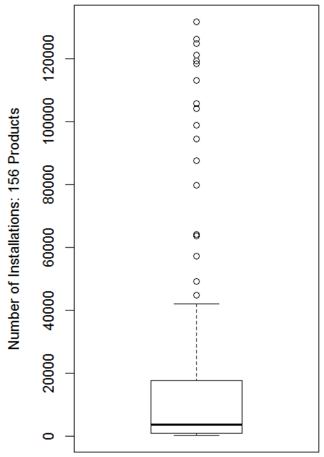
1. Each product is installed on a Windows operating system.
2. There are 50 or more installations of the product.
3. There are at least 50 incidents per product.
4. Product incident records were generated between 4/1/2008 and 10/31/2013.

Searching the organization's software distribution system resulted in 361 products which were installed on at least 50 computers. Incident counts for each of the initial set of 361 products were obtained from the help desk database. Incidents marked as unresolved or canceled were not included in our analysis. We removed 205 products for which the 50 incident minimum criterion was not met. The resulting 156 products were used for analysis in the remainder of the case study. We label the set of products A as in Equation B.0.1.

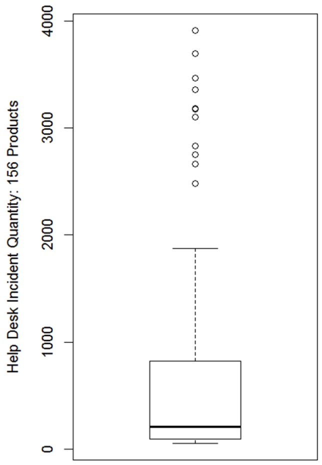
$$\mathbf{A} = \{A1, A2, A3, ..., An\} \tag{B.0.1}$$



(a) Product installations and incident quantities



(b) Number of product installations



(c) Product incident quantities

Figure B.1: Product installations and incident quantities: 156 Products

Figure B.1a shows these 156 products in a scatter plot based on number of installations and number of incidents. Most products are clustered in the range of less than 20000 installations and less than 500 incidents. Figure B.1b shows a box plot of the number of installations of the products. Half of the products are installed on fewer than 3544 computers. The interquartile range shows that 50% of the products are installed on between 815 and 17,990 computers. The average number of installations is 17,470. There are approximately 20 outlier products with installations on 4000 to 120,000 computers. Figure B.1c shows a

Table B.1: Installation and incident data for products selected in case study

Prod ID	num installs	num inc	Prod ID	num installs	num inc	Prod ID	num installs	num inc
A5	193	232	A190	756	1367	A382	201	269
A6	1806	122	A200	1874	81	A383	32208	346
A13	15286	184	A204	3238	153	A384	13502	1138
A14	2692	3105	A211	44758	1054	A385	104296	1045
A20	1058	55	A219	4704	111	A387	19278	151
A21	57133	215	A222	2625	153	A391	93	52
A25	12452	75	A228	7570	65	A393	36197	107
A27	37615	115	A229	4483	94	A394	102	79
A28	40267	483	A232	94514	198	A396	9085	1729
A39	6492	1063	A235	238	140	A397	3312	475
A42	38652	3470	A236	17169	705	A403	17557	174
A44	32214	345	A239	578	59	A404	41940	954
A56	6893	305	A244	25440	207	A408	2204	69
A64	249	89	A245	107	67	A409	1691	207
A65	17271	872	A246	1483	52	A413	49067	65
A74	20700	87	A248	442	218	A416	490	322
A75	890	113	A249	121259	199	A421	15398	326
A77	18895	216	A255	5547	351	A427	862	87
A81	4587	213	A260	21434	861	A436	3072	66
A82	176	76	A267	799	71	A439	124882	54
A83	699	201	A271	131874	100	A442	418	197
A86	1037	116	A275	3201	1363	A450	5234	134
A89	1449	107	A277	95	1510	A451	1763	67
A90	5257	3701	A278	70	59	A456	3166	138
A101	255	78	A282	905	84	A464	3356	50
A103	3556	1064	A284	10108	340	A465	399	1587
A104	3231	698	A287	17876	935	A467	64073	1386
A114	17885	1877	A290	393	204	A468	1602	51
A115	20638	3188	A293	79696	777	A474	13567	736
A116	402	3179	A294	3531	1088	A476	3208	91
A117	782	1042	A295	889	178	A487	38512	416
A119	1764	542	A300	4134	66	A488	6712	195
A123	8355	823	A310	5721	61	A495	4732	87
A128	87716	1034	A312	118566	268	A497	126289	2755
A144	480	290	A315	614	81	A499	1194	1477
A145	10611	90	A319	32115	588	A500	63555	392
A146	5094	1559	A320	11861	1224	A501	5609	235
A148	18972	831	A327	345	226	A502	113203	123
A151	3704	52	A329	3133	53	A503	1861	600
A152	1392	162	A333	3246	2837	A508	613	298
A155	522	203	A334	29993	2484	A510	10786	94
A162	82	125	A345	2115	1227	A511	452	481
A165	6394	70	A347	1428	131	A514	4863	60
A166	11016	656	A357	500	85	A516	213	70
A171	10646	161	A359	2074	1778	A517	513	1144
A172	428	175	A361	240	174	A519	1357	269
A177	32022	64	A367	119384	2662	A520	98952	493
A178	21815	3916	A368	763	237	A523	119341	3358
A179	874	395	A372	1754	86	A524	757	413
A182	105885	191	A373	7767	98	A529	760	103
A185	359	120	A375	1304	126	A531	15570	95
A187	12911	1244	A376	4630	990	A532	95	156

Table B.2: Comparison of selected product descriptive statistics to full help desk product portfolio

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
Incident volume (all products)	1	3	17	321	408	18840
Incident volume (selected products)	50	94	204	592	812	3916
Number of installations (all products)	1	27	320	7450	2589	131900
Number of installations (selected products)	70	815	3544	17470	17990	131900

box plot of the number of incidents generated by the products. Half of the products produce fewer than 204 incidents as indicated by the median value. The interquartile range shows that 50% of the products produced between 94 and 812 incidents, with an average of 592 incidents. Twelve products shown as outliers produced between 2500 and 4000 incidents. Descriptive statistics are included in Table B.2 for a comparison of the selected products and the full product portfolio in terms of installations and incidents. The descriptive statistics in Table B.2 indicate one half of the 800 products are installed on fewer than 320 machines. Similarly, one half of the products produce 17 or fewer incidents. This skewness in both product installations and number of incidents generated is evident in Figure B.1a. Of the roughly 123000 incidents used in this case study, 93000 (76%) were produced by the 156 products selected for investigation. This confirms an instance of the 80/20 Pareto rule.

Figure B.2 shows incident volume from April 2008 to October 2013. The average monthly incident volume for the software products and hardware/OS categories is roughly 2000 and 1000 respectively, whereas the average volume of incidents for the dedicated effort category is near 10,000. The help desk incident database separates incidents by product name and by time interval, but it does not automatically separate incidents into the three categories. For example, the database can be queried for all incidents reported against Adobe Reader in the May 2013 time frame. Similarly, incidents against the Windows 7 operating system and those against peripheral devices (keyboards, mouse, etc.) are easily obtained, but their separation from named software products is not made. The data used for this case study fall into two categories defined by Runeson, et. al. [93]. Knowledge of help desk

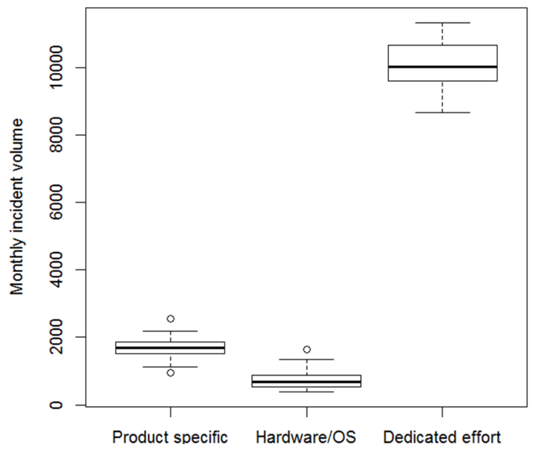


Figure B.2: Monthly incident volume distribution by category

operations in the industrial setting of this case study is obtained by direct methods through semi-structured interviews with two help desk managers. Data obtained by this method compliments the experienced-based knowledge of help desk data analysis by one of the authors of this paper. A second category of data, one in which the majority of the quantitative data in this case study falls, is obtained through the inspection of archival data in the help desk database of incident records. The quantitative data is collected and maintained in the help desk database for the analysis of performance with respect to established service metrics and for trend analyses from which business decisions are made.

For each product in our analysis we obtain the following data.

1. Number of incidents submitted within the time interval for our analysis
2. Date and time at which each incident is submitted
3. Values for seventeen incident attributes as shown in Table A.1 for all incidents

Appendix C The Survey[1]

This is an anonymous survey for all company systems and software engineers to determine various experiences we have had with different types of reuse. This information can be used to help analyze and apply best practices for reuse. It can also be used as data to include in proposals, for research papers such as doctoral dissertations and masters theses.

For purposes of this survey:

- A strategy is the choice of approach or combination of approaches the program makes to employ reuse
- An approach is the one of the development methods that allows for reuse (such as those four listed below):
  - Model-based reuse is reuse that is based on reusing models created on other programs or components.
  - Component-based is reuse based on already developed components or designed for reuse on a component basis.
  - Product line reuse is reuse based on a standardized but tailorable product line.
  - Ad Hoc is reuse that the engineer is familiar with, that happens to meet a requirement but was not designed for reuse.

If you have been involved in more than one project that employed reuse and had different experiences you would like to share, please respond once per program. In these cases please respond from the standpoint of that program and your experiences there. Summary results (and the resulting papers) will be posted here. We hope the results provide guidance in our future practices and help identify the best strategies and approaches for different types of programs We hope this survey can be used as a first step to assess what works and what does not work in reuse. All surveys used for academic research need to make the following disclaimer: Participation is voluntary. You can delete your response at any time. You can respond without identifying yourself.

[1]This survey is included with reference to contributions by Dr. Julia Varnell-Sarjeant in her PhD thesis on the subject of software reuse in embedded vs. non-embedded systems [121]. This survey is also used by Dr. Varnell-Sarjeant et al. [122]. Dr. Varnell-Sarjeant developed the survey and collected the data. This survey is included to assist in understanding Pilot Study 3.2.

Respondent Information

RQ-1 The purpose of these questions is to correlate reuse experience with the type of engineer (ie hardware, software, systems), the company (which corresponds to the types of programs and the culture), and the experience level of the engineer.

- 1. What type of engineer are you?
  - a. Systems
  - b. Software
  - c. Software Systems
  - d. Specify your own answer
- 2. What company and location do you work for?
- 3. How many years of experience do you have with system or software development?
  - a. 0-5 years
  - b. 6-10 years
  - c. More than 10 years
- 4. How many years of experience do you have with incorporating reuse into programs
  - a. 0-5 years
  - b. 6-10 years
  - c. More than 10 years

Program/Application Information

RQ-2, RQ-3 The purpose of these questions is to correlate the size of the program, the nature of the system/program, the software type. This should offer insight into whether embedded and non-embedded use the same strategies and whether successful strategies are similar.

- 5. Is the system you are working on or reporting on embedded or non-embedded?
  - a. Embedded
  - b. Non-embedded
- 6. It is possible for a system to be embedded and the software to be non embedded (for example, database software may be part of a flight system but not itself embedded). Conversely, it is possible to work on embedded software for a non-embedded system (for example, embedded flight software components for a desktop simulator). Is the software you are working with or on or reporting on embedded or non-embedded?
  - a. Embedded
  - b. Non-embedded
  - c. Both
- 7. What type program are you working on (i.e. what is the final product)?
  - a. Satellite
  - b. Ground Station
  - c. Missile/Rocket
  - d. Helicopter
  - e. Submarine
  - f. Deep Space (probe or lander)
  - g. Logistics
  - h. Data Collection
  - i. Other (describe)
- 8. How large is the software effort on your program, or the program you are reporting on (approximately) (KSLOC)?
- 9. What type of application is the focus of the work product you are producing? i.e. graphics, GNC, algorithm, web-based, business, data mining, hardware components, architecture etc?

Reuse Information
RQ-4, This set of questions helps identify the type of reuse strategy employed, whether success is improved with being part of the decision, and whether the program is far enough along to measure factors that occur late in the program. We are able to compare development strategies and artifacts used on embedded systems vs. nonembedded systems.

- 10. Is your program employing product reuse? (artifacts, models, etc)
  - a. Yes (Branch to 11)
  - b. No
- 11. What approach to reuse did your program take? Check all that apply
  - a. Component based
  - b. Model based
  - c. Product Line
  - d. COTS/GOTS
  - e. Heritage/legacy
  - f. Ad Hoc (using already developed code that you happen to have around)
  - g. Other
- 12. Did you have input in the reuse decisions?
- 13. What phase has your program reached?
  - a. Capture
  - b. Requirements
  - c. Architecture
  - d. Design
  - e. Implementation
  - f. Integration and Test
  - g. Deployment
  - h. Maintenance/Operations and Maintenance
- 14. What product(s) is/are being reused? (i.e. requirements, architecture, models (what type?), use cases, code, drawings, hardware, test products, already tested clusters)
  - a. Requirements
  - b. Code
  - c. Architecture
  - d. Models
  - e. Drawings
  - f. Hardware
  - g. Use Cases
  - h. Test Products
  - i. Already tested clusters
  - j. Other (fill in)

Reuse Effectiveness Information
RQ-4 This set of questions will help correlate the effectiveness of the strategy against the strategy by identifying and scoring the change in outcomes attributed to reuse.

- 15. Did the reuse save labor hours?
  - a. No
  - b. Yes, from 10 to 20 per cent
  - c. Yes, from 20-30 per cent
  - d. Yes, more than 30 per cent
  - e. It cost us time (10-20 per cent)
  - f. It cost us time (more than 20 per cent)
  - i. * How much of the time savings to you attribute to reuse? Please explain how it saved time
  - i. * Please explain how the reuse cost you labor hours
- 16. Did you notice fewer defects than when reuse was not employed?
  - a. No
  - b. Yes, 0-10 per cent fewer

  - c. Yes, 10-30 per cent fewer
  - d. Yes, more than 30 per cent fewer
  - e. No, we observed more defects

- 17. Did it reduce testing labor hours?

  - a. No
  - b. Yes, 0-10 per cent reduction
  - c. Yes, 10-30 per cent reduction
  - d. Yes, more than 30 per cent reduction
  - e. No, we had to test more than 10 per cent more

- 18. Did it reduce items that needed to be tested?

  - a. No
  - b. Yes, 0-10 per cent reduction
  - c. Yes, 10-30 per cent reduction
  - d. Yes, more than 30 per cent reduction
  - e. No, we had to test more than 10 per cent more

- 19. Did you feel risk (cost, schedule, technical) was reduced? yes/no
- 20. * If you felt risk was reduced, please explain how. If not, please explain.

Reuse Experience
RQ-5 This set of questions will help analyze the user the experience of the reuse approach.

- 21. Could you please comment on your reuse experience, both bad and good?

- 22. How important do you consider software reuse to achieve the following benefits? Answer with 5 being greatest benefit, 1 being nearly useless:

  - a. * Lower Development Costs
    - ∗ i. 5 Great benefit
    - ∗ ii. 4 Some benefit
    - ∗ iii. 3 Little benefit
    - ∗ iv. 2 Neither helpful nor unhelpful
    - ∗ v. 1 Nearly useless
  - b. * Shorter "time to market" (including all phases of development through delivery)
    - ∗ i. 5 Great benefit
    - ∗ ii. 4 Some benefit
    - ∗ iii. 3 Little benefit
    - ∗ iv. 2 Neither helpful nor unhelpful
    - ∗ v. 1 Nearly useless
  - c. * More confidence in quality of delivered product
    - ∗ i. 5 Great benefit
    - ∗ ii. 4 Some benefit
    - ∗ iii. 3 Little benefit
    - ∗ iv. 2 Neither helpful nor unhelpful
    - ∗ v. 1 Nearly useless
  - d. * Known standardized product
    - ∗ i. 5 Great benefit
    - ∗ ii. 4 Some benefit
    - ∗ iii. 3 Little benefit
    - ∗ iv. 2 Neither helpful nor unhelpful
    - ∗ v. 1 Nearly useless

Appendix D Published Papers

The following co-authored papers related to this thesis have been submitted or published as noted. Cross references to applicable sections in this thesis are included as footnotes.

1. Anneliese Andrews and Joseph Lucente. From incident reports to improvement recommendations: Analyzing it help desk data. In *Australasian Software Engineering Conference*, April 2014 (published). [10][1]

2. Julia Varnell-Sarjeant, Anneliese Amschler Andrews, Joseph Lucente, and Andreas Stefik. Comparing development approaches and reuse strategies: An empirical evaluation of developer views from the aerospace industry. *Journal of Information and Software Technology*, 2014 (published). [122][2]

3. Joseph Lucente and Anneliese Andrews. Predicting incident reports for IT help desk. In *The First International Workshop on Dependability and Security of System Operation*, June 2014 (published). [8][3]

4. Anneliese Andrews and Joseph Lucente. On the Viability of Using SRGMs for IT Help Desk Incident Predictions. In *International Conference on Software Quality, Reliability and Security*, August 2015 (submitted). [11][4]

5. Anneliese Andrews, Philip Beaver and Joseph Lucente. Towards Better Help Desk Planning: Predicting Incidents and Required Effort. In *Empirical Software Engineering* (submitted). [9][5]

[1]Pilot study 3.1 Application of Principal Components Analysis
[2]Pilot study 3.2 Extension of PCA to Survey Data
[3]Pilot study 3.3 Software Reliability Growth Models
[4]Pilot study 3.4 PCA and SRGMs with Extended Product Set
[5]Chapter 4 Large Scale Approach