

ENHANCING SOFTWARE PROJECT OUTCOMES:  
USING MACHINE LEARNING AND OPEN SOURCE  
DATA TO EMPLOY SOFTWARE PROJECT  
PERFORMANCE DETERMINANTS

Lalit N. Jagtiani

Under the Supervision of Dr. Christian Bach

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN TECHNOLOGY MANAGEMENT

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

November 27th, 2017


# ENHANCING SOFTWARE PROJECT OUTCOMES: USING MACHINE LEARNING AND OPEN SOURCE DATA TO EMPLOY SOFTWARE PROJECT PERFORMANCE DETERMINANTS

## Approvals

### Committee Members

Name	Signature	Date
Dr. Christian Bach, University of Bridgeport		11/27/17
Dr. Lesley Frame, University of Bridgeport		11/27/17
Dr. Christopher Huntley, Fairfield University		11/27/17
Dr. Michael Lohle, University of Bridgeport		11/27/17
Dr. Gad Selig, University of Bridgeport		11/27/17
Dr. Robert Todd, University of Bridgeport		11/27/17

### Ph.D. in Technology Management Program Director

Dr. Elif Kongar, University of Bridgeport		11/27/2017
---	--	------------

### Dean, School of Engineering

Dr. Tarek M. Sobh, University of Bridgeport		11/27/2017
---	--	------------

# ENHANCING SOFTWARE PROJECT OUTCOMES: USING MACHINE LEARNING AND OPEN SOURCE DATA TO EMPLOY SOFTWARE PROJECT PERFORMANCE DETERMINANTS

© Copyright by Lalit N. Jagtiani 2017

All Rights Reserved

# ENHANCING SOFTWARE PROJECT OUTCOMES: USING MACHINE LEARNING AND OPEN SOURCE DATA TO EMPLOY SOFTWARE PROJECT PERFORMANCE DETERMINANTS

## **Abstract**

Many factors can influence the ongoing management and execution of technology projects. Some of these elements are known a priori during the project planning phase. Others require real-time data gathering and analysis throughout the lifetime of a project. These real-time project data elements are often neglected, misclassified, or otherwise misinterpreted during the project execution phase resulting in increased risk of delays, quality issues, and missed business opportunities.

The overarching motivation for this research endeavor is to offer reliable improvements in software technology management and delivery. The primary purpose is to discover and analyze the impact, role, and level of influence of various project related data on the ongoing management of technology projects. The study leverages open source data regarding software performance attributes. The goal is to temper the subjectivity currently used by project managers (PMs) with quantifiable measures when assessing project execution progress.

Modern-day PMs who manage software development projects are charged with an arduous task. Often, they obtain their inputs from technical leads who tend to be significantly more technical. When assessing software projects, PMs perform their role subject to the limitations of their capabilities and competencies. PMs are required to

contend with the stresses of the business environment, the policies, and procedures dictated by their organizations, and resource constraints.

The second purpose of this research study is to propose methods by which conventional project assessment processes can be enhanced using quantitative methods that utilize real-time project execution data. Transferability of academic research to industry application is specifically addressed vis-à-vis a delivery framework to provide meaningful data to industry practitioners.

# TABLE OF CONTENTS

Title Page .....	i
Approvals .....	ii
Copyright Statement .....	iii
Abstract .....	iv
<b>TABLE OF CONTENTS .....</b>	<b>vi</b>
<b>LIST OF TABLES .....</b>	<b>x</b>
<b>LIST OF FIGURES .....</b>	<b>xi</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
<b>CHAPTER 2: RESEARCH SCOPE .....</b>	<b>3</b>
2.1. Motivation and Research Questions .....	3
2.2. Software Life Cycle and Performance Measures.....	4
2.3. Research Goals and Potential Contribution .....	9
2.4. Limitations and Constraints .....	12
<b>CHAPTER 3: LITERATURE REVIEW .....</b>	<b>14</b>
3.1. Macro Analysis .....	16
3.2. Software Quality Attributes .....	25
3.3. Research Data Validity .....	31
3.4. Software Management and Predictive Modeling.....	41
<b>CHAPTER 4: RESEARCH PROCESS.....</b>	<b>49</b>

<b>CHAPTER 5: RESEARCH DATA .....</b>	<b>51</b>
5.1. Data Plan and Expected Outcomes .....	51
5.2. Data Management .....	52
5.3. Data Extraction .....	54
5.4. Data for Analysis .....	55
5.5. Research Variables.....	58
<b>CHAPTER 6: RESULTS AND IMPLICATIONS .....</b>	<b>61</b>
6.1. Software Quality Attributes .....	62
6.2. Descriptive Statistics – Research Variables.....	67
6.3. Multiple Regression Model – Research Variables.....	68
6.3. Towards Building and Validating a Predictive Model .....	70
6.4. Research Transferability .....	72
6.5. Developing an Applied Research Framework (ARF).....	76
<b>CHAPTER 7: FUTURE RESEARCH OPPORTUNITIES.....</b>	<b>83</b>
<b>CHAPTER 8: CONCLUDING REMARKS .....</b>	<b>86</b>
<b>CHAPTER 9: GLOSSARY OF TERMS.....</b>	<b>88</b>
<b>CHAPTER 10: BIBLIOGRAPHY .....</b>	<b>94</b>
<b>APPENDIX A: THE SRDA .....</b>	<b>109</b>
A.1. Data Entity Relationship Diagram—SourceForge.net.....	109
A.2. Data Entity Relationship Model – SRDA – Artifact.....	110
A.3. Data Entity Relationship Model – SRDA – Documents.....	111

A.4. Data Entity Relationship Model – SRDA – Forums.....	112
A.5. Data Entity Relationship Model – SRDA – FRS .....	113
A.6. Data Entity Relationship Model – SRDA – Job .....	114
A.7. Data Entity Relationship Model – SRDA – Tasks.....	115
A.8. List of tables analyzed from OSS Research.....	115
A.9. Select SRDA Data Stored in Local MS Access Database .....	117
<b>APPENDIX B: EXISTING RESEARCH – THE SRDA.....</b>	<b>120</b>
B.1. Reference List by Research Focus .....	120
B.2. Yearly Cumulative Publications – SRDA.....	122
B.3. Publications by Publisher .....	123
<b>APPENDIX C: DATA RESULTS .....</b>	<b>124</b>
C.3. Weka 3.8.1 Machine Learning Algorithm – Results.....	124
C.3.1. Random Forest .....	124
C.3.2. Meta Bagging .....	125
C.3.3. J48 Decision Tree.....	126
C.3.4. Decision Table.....	127
C.3.5. K-Nearest Neighbor .....	128
C.3.6. Multi-Layer Perceptron .....	129
C.3.7. Iterative Classifier .....	130
C.3.8. Naïve-Bayes .....	131



C.3.9. AdaBoost M1 .....	132
--------------------------	-----

## LIST OF TABLES

Table 1. Project Performance Statistics [9].....	17
Table 2. Comparison of Management and Technical Performance Levels .....	18
Table 3. Probability of Selected Outcomes [17] .....	20
Table 4. Average Software Schedules (in Calendar Months).....	21
Table 5. Summary of Measures of Success, Indicators, and Potential Issues .....	26
Table 6. SPI Success Dimensions .....	29
Table 7. OSS by the Numbers [53] .....	33
Table 8. Gap Assessment – Learning Outcomes for Course Curricula [68] .....	42
Table 9. Gap Assessment – Software Quality Prediction Models in Research Studies ...	46
Table 10: Research Variables and Relevance .....	59
Table 11. Descriptive Statistics – Research Variables.....	67
Table 12. Research Variables – Data Quartiles .....	68
Table 13. Research Variables – Multiple Regression Results (95% C.I.) .....	69
Table 14. Relative Importance of Key Attributes in Predictive Modeling .....	70
Table 15. Classification Algorithm Performance Summary .....	71

## LIST OF FIGURES

Figure 1. Software Project Delivery Life Cycle Framework .....	5
Figure 2. Objectivity of Metrics and Research Scope .....	6
Figure 3. Sample Dashboards - Release Predictability and Defects .....	11
Figure 4. Sample Dashboards - Stakeholder Engagement and Work Effort .....	12
Figure 5. Areas of Literature Review .....	15
Figure 6. Automation in Key Software Project Management Activities .....	19
Figure 7. Planned Versus Actual Software Schedules [17] .....	21
Figure 8. DeLone and McLean’s Success Model [33] .....	26
Figure 9. Mining Software Research Spanning 5 Years –.....	31
Figure 10. Select Data Entities from SRDA .....	37
Figure 11. SRDA – Existing Research – 10 Year Period, 2007 to 2016 .....	38
Figure 12. Review of 266 Research Papers Using OSS vs. PSS Artifacts .....	39
Figure 13. Influence of Top Software Management Practices on Software Quality .....	44
Figure 14. Process for Research Study Execution .....	49
Figure 15. Data plan - Key Activities and Preliminary Outcomes .....	51
Figure 16. Data analysis - Key Activities and Preliminary Outcomes .....	52
Figure 17. Research Data Management Plan .....	53
Figure 18. SRDA SourceForge Query Form .....	54
Figure 19. Key Data Relationships for Select Data from the SRDA .....	55
Figure 20. Key Steps to Derive Key Data Relationships from the SRDA .....	57
Figure 21. Research Variables and Classifications .....	59

Figure 22. Sample 3D Data Plot –Sample Size of 18,019 Software Packages.....	63
Figure 23. Mean Time Between Software Release Dates and Defects Logged .....	64
Figure 24. Density Map – Group Rank and Number of Downloads .....	65
Figure 25. Relationship between Group Rank and Number of Downloads .....	66
Figure 26. MLP – Neural Network Diagram to Predict Software Rank.....	72
Figure 27. Root Cause Analysis - Select Review of Research Papers.....	73
Figure 28. Improving Software Management through Research Transferability .....	74
Figure 29. Pasteur’s Quadrant Model of Scientific Research.....	75
Figure 30: Applied Research Framework (ARF) for Research Transferability .....	78
Figure 31: ARF – An Illustrative Example of Using the SRDA .....	81

# CHAPTER 1: INTRODUCTION

While the origins of computational methods and programming date back a few centuries or more, the discipline of software engineering originated in 1968 [1]. The ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB) defines software engineering as “the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.”<sup>1</sup> Software engineering is therefore far more than computer programming; it is both the science and art of applying mathematical and computational logic to create a defined technological capability within resource and time constraints.

Numerous industrial and academic oriented studies have examined software development performance and prediction models for more than fifty years [2]. Theoretical and practice-oriented experts recognize several reasons which contribute to sub-optimal performance. While much of the failure can be attributed to industry, business climate, and other external forces, further research is required to determine methods that can improve aggregate results over time. Experience and research show that current software project management practices use far more subjective methods than objective data analysis to assess project progress [3].

The goal of this research study is to demonstrate concrete ways to increase objectivity in the management of software engineering. The scope of research has been aligned to address technology management issues specific to the software development process. Research objectives are established at the onset of the study; they guide the study to focus

---

<sup>1</sup>[1] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*: IEEE Computer Society Press, 2014.

on examining software quality predictors, establishing objective metrics, and recommending revised methods for improved software technology management. A thorough literature review is conducted to examine macro factors impacting software management, identify software quality attributes, establish research data validity, and discover improved software management and predictive modeling opportunities. Research study variables are established, classified, and analyzed using data extracted from an open source data archive exclusively developed for software research purposes at the University of Notre Dame. Quality prediction modeling opportunities are examined using the research variables with select machine learning enabled models trained and tested with archive data for optimal predictive performance. An applied research framework is introduced to enable the transfer of research outcomes from academia to industry. Finally, the framework is explored as a mechanism for employing predictive models within the industry to sustainably improve software project outcomes.

## **CHAPTER 2: RESEARCH SCOPE**

### **2.1. Motivation and Research Questions**

Although software engineering as a discipline has been around for nearly half a century, failure rates of large projects have remained high with no significant improvements reported over time. While it is generally accepted that several external factors have an influence on the management and execution of software technology projects, there needs to be a thorough examination of internal software project data that can be leveraged at appropriate times during the project life cycle. Moreover, many organizations use overly subjective (i.e., qualitative) analysis to improve project execution by relying heavily on the personal experiences of project managers. Biases associated with personal experiences often result in continued performance and quality variances and missed targets. Technological advances in software development now allow for real-time data to be utilized for rapid analysis and infusion into project management processes. This research aims to discover viable predictor(s) of software project execution quality and their potential usefulness in improving processes. More specifically, the research questions examined in this study include the following three categories:

#### **1. Predicting the quality of the software project execution process.**

- a. Is software defect data a good predictor of overall project quality during the execution stage of the project?
- b. What attributes of software are better predictors than others? Are there other reliable predictors?

- c. Can organizations predict software release dates based on these predictors? Is it feasible to develop a learning-based, predictive model for practical application by software management professionals?

## **2. Increasing the objectivity of software project management.**

How can objective data from past experiences with software management offer improvements to future projects using quantitative analysis?

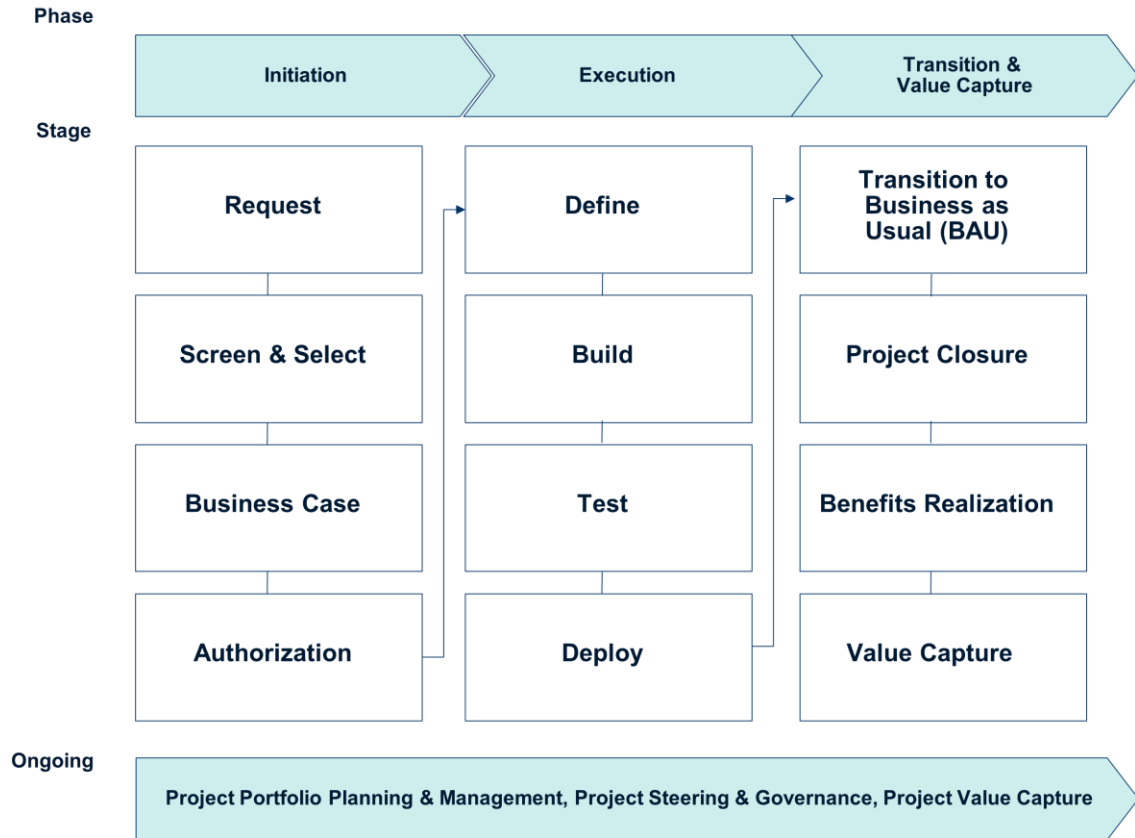
## **3. Improving methods used by software project management processes.**

How can project management tools and techniques be modified to incorporate quantitative methods and predictive models for use software technology management professionals?

## **2.2. Software Life Cycle and Performance Measures**

The study of software project life cycles has interested academia and industry since the advent of the software engineering discipline. Figure 1 depicts a project life cycle framework that was developed by the primary researcher of this research study based on collective industry experiences. The framework is also supported by the report *IEEE Standard for Developing a Software Project Life Cycle Process* published by the IEEE Computer Society in 2006 and follows the guidelines set forth [4].

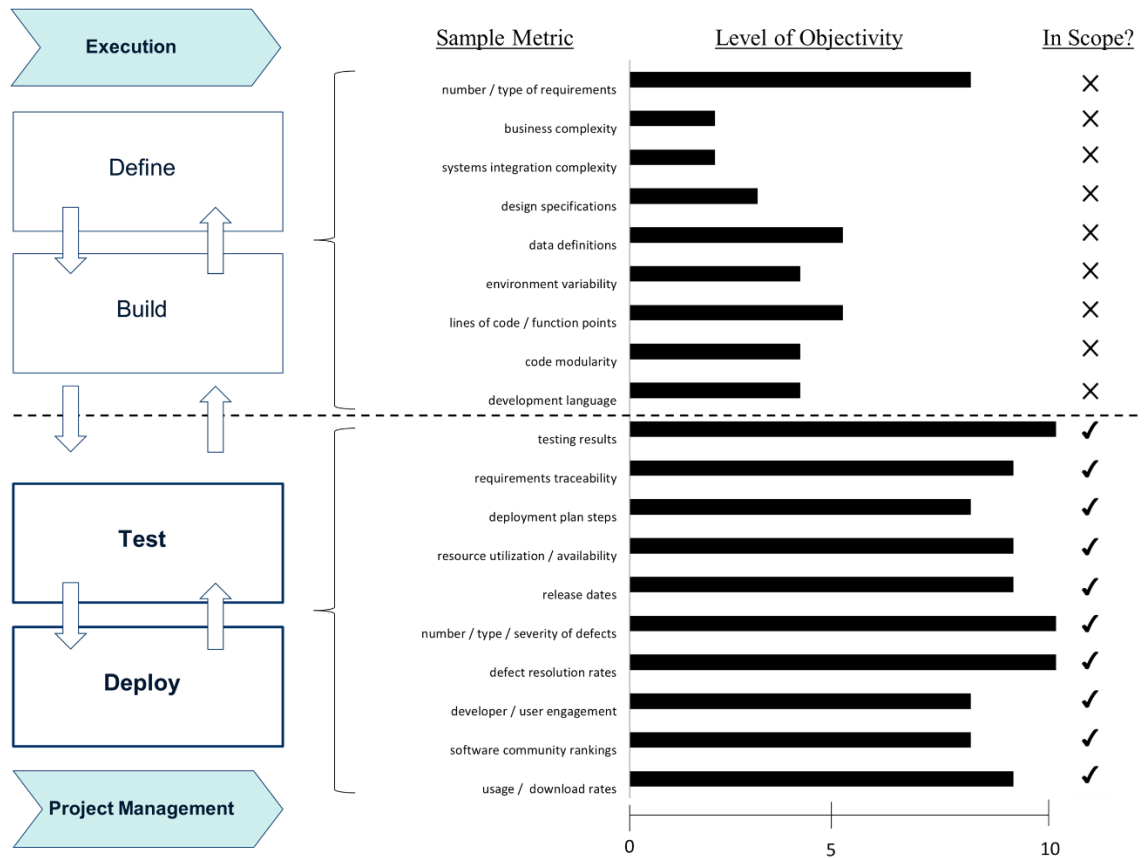




**Figure 1. Software Project Delivery Life Cycle Framework**

While there are numerous metrics that can be measured during each stage of the software project life cycle, the scope of this study is limited to the execution phase of software projects. Several factors influence the management and execution of software technology projects. Some of these elements are known a priori during the project planning phase. Others may require real-time data gathering and analysis during project execution [5]. These real-time project data elements are often neglected, misclassified, or otherwise misinterpreted during the project execution phase. The results are higher risk of delays, quality problems, and lost opportunities. The primary purpose of this research study is to discover and analyze the impact, role, and level of influence of various project related data on the execution and ongoing management of software technology projects. This research study is focused on measuring and predicting project execution variables

which are typically subject to high levels of uncertainty. Measures that are highly subjective or reflect the quality of the product instead of the quality of the process and may be of interest to other researchers are not in the scope of this study. Figure 2 shows the areas of focus within the project execution stage for measurement and predictability related variables in the context of this research study.



**Figure 2. Objectivity of Metrics and Research Scope  
– Software Project Execution Phase**

Sample metrics have been depicted for each stage of project execution. The level of objectivity inherent in each of these metrics has been assessed accordingly. Metrics in the early stages of execution are measured with a higher level of subjectivity, i.e., lower relative objectivity, and are therefore considered less reliable and difficult to measure. Early stage metrics are out of the scope of the study as they are typically useful in

determining the size, complexity, and features of the software product and not the quality of the development process itself. Typically, project managers use these metrics to define the project and baseline estimates against the triple constraint model of time, cost, and product scope. During the early and formative stages of project execution, i.e., design and build, the need for predictability is relatively low, the level of discovery is relatively high, and consequently, project execution risk is relatively transparent and easier to assess in these stages. In contrast, the metrics in the later stages of the project execution phase, i.e., test and deployment, can be more actionable by helping to identify and mitigate risk otherwise not visible to PM's. Also, it is important to note that while, for example, the testing stage, is a widely accepted validation approach in industry, it is often ad-hoc, expensive, and unpredictable [6]. Some earlier studies suggest that the testing phase by itself could constitute 50 percent more of the total development cost [6, 7].

Metrics pertaining to the later stages of project execution demonstrate varying levels of objectivity with respect to their measurement and predictability. Examples of metrics that demonstrate lower levels of objectivity include the number of use cases, validity of test scripts, test case accuracy, requirements traceability, deployment plans, resource availability, and utilization. These metrics are product and organization defining, less reflective of the software development process, and can, therefore, be difficult to quantify. Such metrics are considered out of scope for this research study. However, there are a group of key metrics that is relevant to the later stages of project execution which provide greater objectivity and ease when measuring quality. They are also process-indicative by their very nature. Sample metrics in this group and in the scope for this research study include release dates, number/type/severity of defects, defect resolution

rates, developer and user engagement, software community rankings, and usage and download rates.

This research study fully recognizes the overriding premises of software development and management processes and in no way attempts to minimize the complexities and interdependencies of the subjective and objective aspects of project management that are required to produce quality software under the triple constraint model. The goal is to increase the objectivity within the software project management process where possible, making it more pragmatic and effective. While Figure 2 depicts sample metrics that are in and out of scope and draws attention to the later stages of project execution, the analysis also suggests interdependencies and the iterative nature of processes that transcend all the stages of execution including the earlier stages of software product definition and build.

This research study leverages open source software (OSS) data available from the public domain. Though sometimes difficult and complex, leveraging OSS data can provide a viable platform for research if a fit-for-purpose database environment is created that addresses specific measurement related requirements [3]. It is essential to create an environment that provides a normalized dataset and one that can reduce the layers of abstraction that would otherwise hinder research efforts. In this manner, certain insights that are developed after the analysis of OSS data can be applied to privately and commercially developed software projects.

In order to ensure feedback, there should be quantifiable improvement measures for various system states over time. The goal of this research is to add a novel body of knowledge to the technology management discipline by exploring one or more of the modern-day predictive modeling techniques.

PM's who manage software development projects have an increasingly arduous task. Often managers receive their inputs from technical leads who are more technically oriented. When assessing software projects, PM's perform their duties according to the limitations of their capabilities and competencies. PM's have the need to contend with the limitations of resources made available to them such as human capital and project management tools. Furthermore, PM's tend to follow common and industry-tested practices within their organizations.

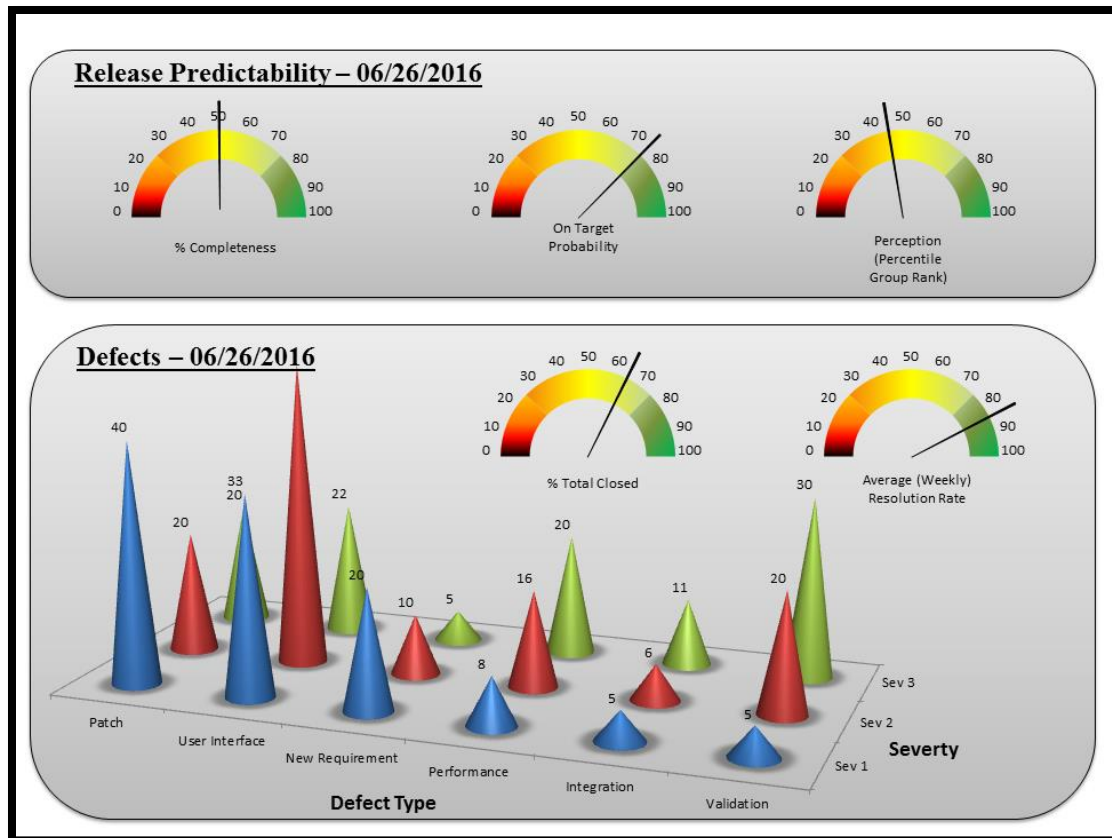
With this motivation in mind, the secondary purpose of this research is to examine how conventional project assessment processes can be enhanced by using quantitative methods utilizing real-time project execution data.

### **2.3. Research Goals and Potential Contribution**

This study aims to make measurable and implementable contributions in the area of software technology management. Specifically, the goals of this research study are to:

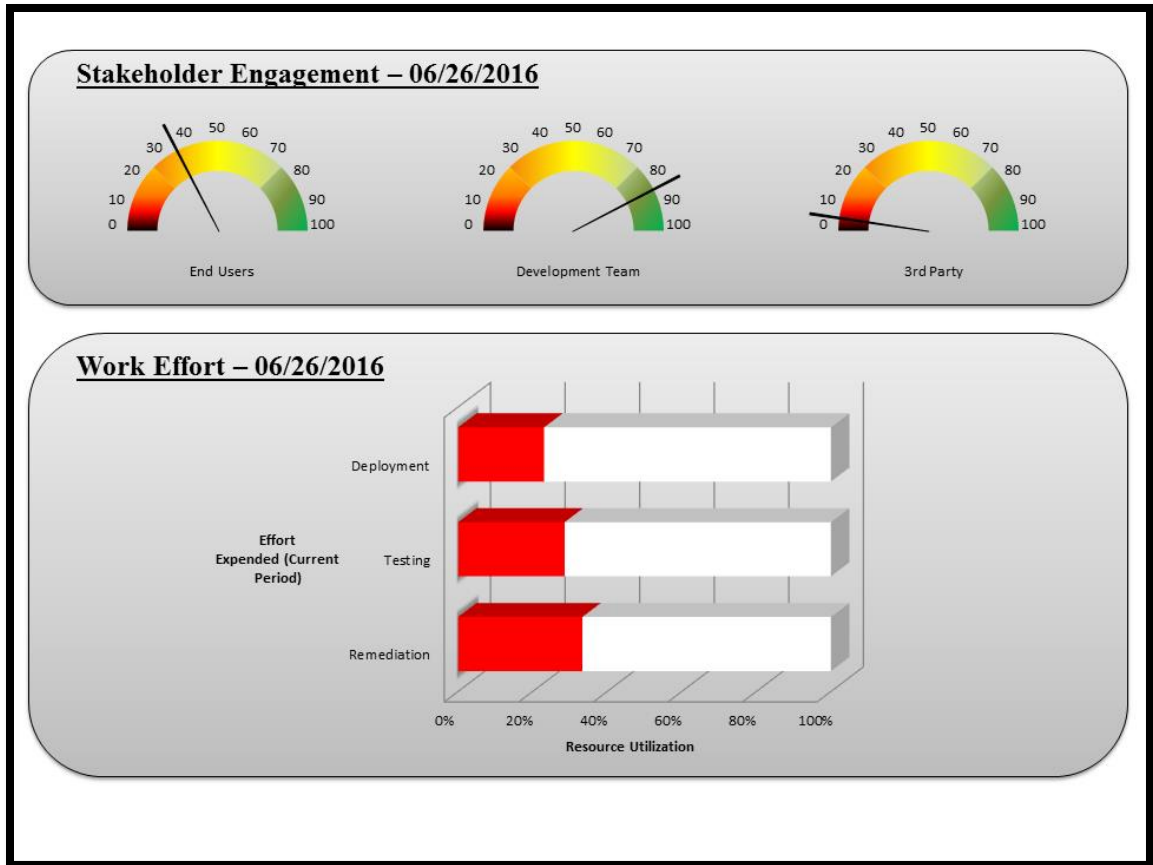
1. Discover and analyze the impact of various project related data on ongoing software technology management projects.
2. Improve upon conventional project assessment processes by using quantitative methods, which utilize real-time project execution data. In particular, the study aims to:
  - a. Enhance the predictors of software project execution.
  - b. Improve the methods used by software project management processes.
  - c. Develop a framework to increase the objectivity of software project management.

Based on previous work conducted using OSS data, this study aims to extend the research so that findings may be leveraged by future software project management professionals under appropriate conditions and parameters. By operationalizing these findings, practitioners can reasonably expect to improve the predictability and reliability of their software projects. For example, software project quality and predictability related dashboards that could be created and used by software PM's to assess more objectively the quality and predictability of outcomes from the later stages of the software project execution phase of active projects. Conceivably, real-time data could be leveraged from projects and instantly benchmarked with historical data from similar projects to determine quality and predictability attributes. Insights provided by these types of dashboards could be valuable for mitigating project risk. PM's who traditionally manage their projects by overly relying on subjective project information, can use such data-based insights as early warning indicators giving more time for corrective actions to be performed before it is too late. Figure 3 and Figure 4 depict representative sets of sample dashboards. In Figure 3, the Release Predictability dashboard shows predictability attributes for software release while Defects dashboard shows defect information that provides indicative information about the quality of the development process currently underway on a project:



**Figure 3. Sample Dashboards - Release Predictability and Defects**

Figure 4 depicts another pair of sample dashboards representing stakeholder engagement and work effort being expended during the current stage of the project. The first dashboard shown in Figure 4 provides objective information regarding the level and type of engagement signaling potential team concerns, interest concerns, and the quality of testing underway. This can be of crucial importance to the unsuspecting PM responsible for managing a project. The second dashboard shows how resource time is utilized during the current stage of the project's execution phase signaling potential risks related to skill set gap, product development, productivity, and resource backlog:



**Figure 4. Sample Dashboards - Stakeholder Engagement and Work Effort**

## **2.4. Limitations and Constraints**

While this research study strives to add an incremental and novel value to software project management, by no means will this research study be terminal or decisive with all possible attributes, parameters, or factors. For instance, as noted earlier, numerous subjective factors influence project performance, such as organizational culture, team dynamic, competitive environment, timing, and resources. These macro factors will not be the subject of this research. In addition, there can be several other quantitative factors that influence project performance in various ways that are out of the scope of this study such as code performance, code complexity, code modularity, and hardware related parameters. While these variables could have an impact on software product quality, this



study focuses solely on software process quality. Therefore, discovery efforts for this study will be limited to process related attributes.

This research study is also bound by a few key constraints that are anticipated. Unless grant funds are awarded, research efforts for this study are not expected to be funded from external sources. When possible, university resources and the author's personal resources are relied upon exclusively to complete this research study. Ample opportunity is expected for further research that extends beyond the scope of this study including the identification of additional quality predictors, the development of industry-specific quality predictive models, and the automation of methods to incorporate research data into project management tools.

## **CHAPTER 3: LITERATURE REVIEW**

While the review of existing research continues towards the advancement of this research, the goals of this study were further explored with a thorough literature review. Specifically, research goals for this study have stemmed from the following two areas of literature review:

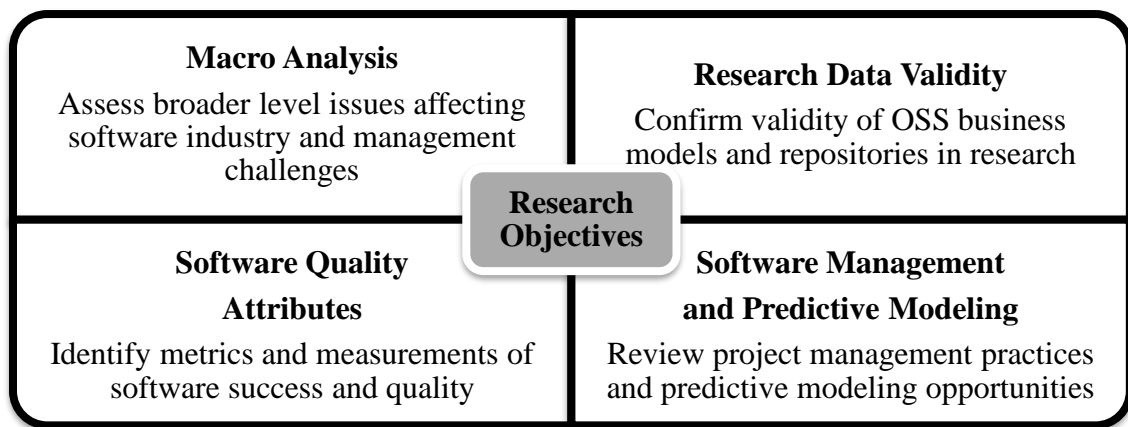
1. Past researchers have specifically acknowledged an opportunity for further research in a specific area based on their own published research.
2. A gap within existing research has been observed (i.e., there was no evidence found of significant progress made in the particular area of concern).

The following benefits from this research project were realized as a result of the literature review:

1. Shape and refine this research study by increasing the added value and ensuring that the research goals are novel and achievable. A thorough literature review provides clarity, conviction, and a pragmatic approach provides continuing guidance and validity to this research study.
2. Provide an interesting opportunity to review past academic research to identify gaps and refine research goals that can maximize intellectual merit while minimizing implementation complexity.
3. Provide substantive background, context, and motivation for this research study.

4. Develop and validate assumptions regarding required data and an appropriate approach to research – the selection of right data is an important component for this research study.
5. Validate data sources and discovery of criteria for data source selection – selected data requirements will require the proper data sources that are easy to access, supported by a credible research community and tested by academic researchers.
6. Provide insights on the selection of statistical and modeling methods to use when analyzing the data to create the best prediction model within the scope of this study.

The research goals of this study are developed and refined based on a comprehensive review of literature in specific areas of software management. Numerous academic publications, authored by experts in their respective fields, are reviewed and categorized in the following areas as shown in Figure 5:



**Figure 5. Areas of Literature Review**

### **3.1. Macro Analysis**

While the global information technology industry has grown to over \$3.5T dollars including a software segment that generates in excess of \$310M revenues annually. In just over half a century, the practices of software technology management have been severely stressed [8]. As is the case with most disruptive technologies that experience high growth rates, the software industry has experienced extreme challenges that have resulted from the hyper the growth rate. Numerous studies, surveys, and assessments are routinely conducted by organizations and independent third parties to better understand and alleviate the challenges. This study starts by examining software project failure rates.

In 2013, the CHAOS Report published by The Standish Group showed an alarming rate of 79% for projects that either failed or where challenged [9]. Failed projects were defined as projects that are canceled at some point during the development cycle. Challenged projects were defined as projects that are completed and operational but over-budget, over the time estimate, and/or that offer fewer features and functions than originally specified. Successful projects were defined as projects that are completed on time and on budget with all features and functions as initially specified. The Standish Group has been publishing the report for more than thirty years.

Analyses show high failure or challenged project rates with the root cause centered squarely on lack of adequate planning, readiness, and assessment methods. Mandal and Pal establish, with their research, that more than 50% of all Information Technology (IT) projects become “runaway” projects [10]. These projects exceed budgets and schedules while failing to deliver the expected outcomes [10-12]. Furthermore, project results based on the triple constraint model of time, cost, and scope become even more concerning.

Table 1 depicts that most of the projects exceed on two of the most important constraints: time and cost:

**Table 1. Project Performance Statistics [9]**

<b>Year</b>	<b>Successful (A)</b>	<b>Challenged (B)</b>	<b>Failed (C)</b>	<b>Unsuccessful (B + C)</b>	<b>Time Overrun</b>	<b>Cost Overrun</b>	<b>Undelivered Scope</b>
<b>1994</b>	16%	53%	31%	84%	N/A	N/A	N/A
<b>1996</b>	27%	33%	40%	73%	N/A	N/A	N/A
<b>1998</b>	26%	46%	28%	74%	N/A	N/A	N/A
<b>2000</b>	28%	49%	23%	72%	N/A	N/A	N/A
<b>2002</b>	34%	51%	15%	66%	N/A	N/A	N/A
<b>2004</b>	29%	53%	18%	71%	84%	56%	36%
<b>2006</b>	35%	46%	19%	65%	72%	47%	32%
<b>2008</b>	32%	44%	24%	68%	79%	54%	33%
<b>2010</b>	37%	42%	21%	63%	71%	46%	26%
<b>2012</b>	39%	43%	18%	61%	74%	59%	31%

For years 2004 to 2012, Table 1 shows no significant annual improvement when examining software project performance for these two constraints (*i.e.*, time and cost) even while scope remains generally under delivered. Project success seems to be arbitrarily achieved by the reduction of project scope rather than improving performance using other drivers [5]. These findings should raise a significant concern for software technology managers. The implications are sub-optimal aggregate productivity, increased risk of missing project expectations, and a greater allocation of project resources than planned. Software quality predictors have been studied by several researchers in the past. In one such study, the analysis of an OSS repository, SourceForge, showed that software quality indicators such as the number of downloads, rank, operating system, language, and days-to-build can, in fact, be examined to predict outcomes [13].

Lee, Kim, and Gupta also point out abysmal statistics on OSS projects [14]. They note that most of the success with large OSS projects can be attributed to backend servers and internet-related software. The number of failed or dormant OSS projects is notable.

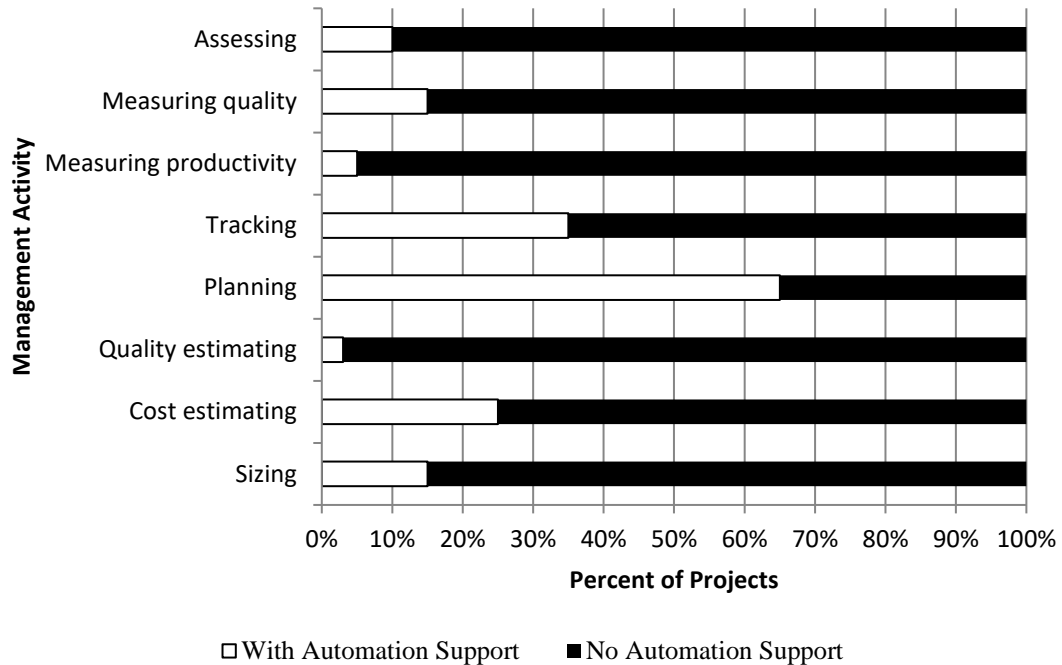
They base their findings on data extracted from SourceForge and confirm that most OSS projects have ended in failure. An alarming 58% of the projects do not advance beyond the alpha developmental stage, 22% remain in the planning phase, whereas the remaining 17% remain in the pre-alpha phase, and some become inactive. The authors point out similar results which have been reported by the World Bank which cites an excess of 50% failure rates for OSS projects [15]. With such results and claims, additional research on software process improvement and related investments appears justified.

A study conducted over one hundred assessments as part of a research project to better understand software productivity [16]. The findings on the performance levels of managers and technical staff are summarized in Table 2:

**Table 2. Comparison of Management and Technical Performance Levels**

<b>Management Activities</b>	Sizing	Fair	<b>Technical Activities</b>	Analysis	Fair
	Estimating	Poor		Design	Fair
	Planning	Fair		Coding	Good
	Tracking	Poor		Reviews	Poor
	Measuring	Poor		Testing	Good
	Overall	Poor		Overall	Fair to Good

A comparison of projects using automation with those that do not use any automation in their assessment processes is shown in Figure 6 below.



**Figure 6. Automation in Key Software Project Management Activities**

A correlation between the presence or absence of project management automation and practical results on software projects has been observed. 40% of large software projects having 2,000 function points or more miss their anticipated delivery dates by more than six months and about 15% miss by more than 12 months. In addition, some large projects are canceled and not delivered at all [16]. When either automated planning or automated estimating methods are used, approximately 12% of software projects miss their scheduled dates in excess of 6 months and about 5% were delayed by more than 1 year. When both methods are used, less than 5% of software projects miss their delivery dates by more than 6 months, and only 1% were delayed by more than 1 year. A secondary benefit is also observed. Since automated estimating and planning tools provide a much stronger grounding, the use of these tools prevents arbitrary efforts to compress schedules without rationale.

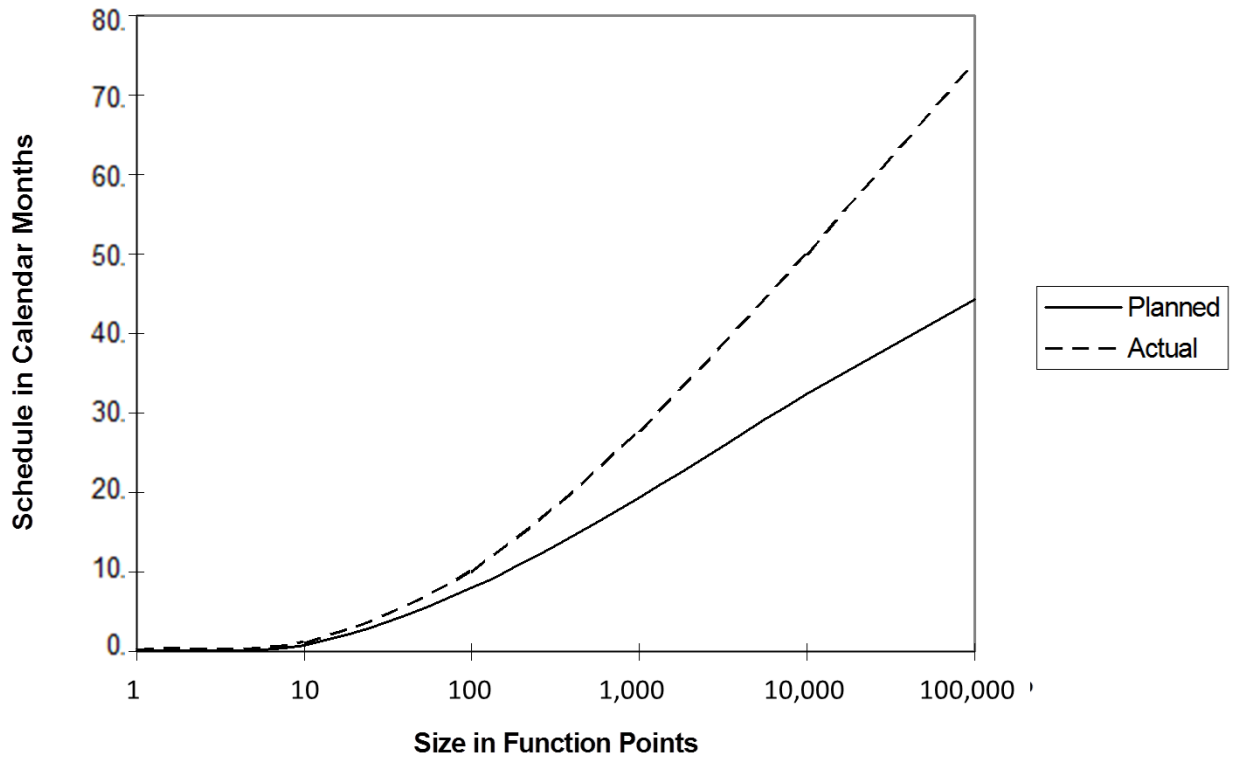
Table 3 shows that larger software projects have a higher risk of cancelations or major delays when compared to smaller projects:

**Table 3. Probability of Selected Outcomes [17]**

	<b>Early</b>	<b>On Time</b>	<b>Delayed</b>	<b>Canceled</b>
<b>1 FP</b>	14.68%	83.16%	1.92%	0.25%
<b>10 FP</b>	11.08%	81.25%	5.67%	2.00%
<b>100 FP</b>	6.06%	74.77%	11.83%	7.33%
<b>1000 FP</b>	1.24%	60.76%	17.67%	20.33%
<b>10000 FP</b>	0.14%	28.03%	23.83%	48.00%
<b>100000 FP</b>	0.00%	13.67%	21.33%	65.00%
<b>Average</b>	5.53%	56.94%	13.71%	23.82%

In his paper, Jones further highlights the risk of client conflict, lost credibility, and risk of litigation resulting in immense financial stress to organizations because of the delays [18]. The author confirms that the most frequent complaint about software projects from executives in the private and public sectors is that the larger the software system, the greater the delays experienced with delivery schedules. Figure 7 depicts this problem in terms of function points (FP) and delivery schedule delays:





**Figure 7. Planned Versus Actual Software Schedules [17]**

A study conducted in 2001 of 1,000 U.S. software projects further substantiates the notion that on average, larger projects experience greater delays. Software development schedules depicted in Table 4 are in calendar months (in decimal units) for 6 size ranges and 6 categories of software projects: end-user development; management information systems (MIS), outsource projects (OutS), commercial software (Comm), system software, and military software [19]:

**Table 4. Average Software Schedules (in Calendar Months)**

	End-User	MIS	OutS	Comm	System	Military	Average
1FP	0.05	0.10	0.10	0.20	0.20	0.30	0.16
10FP	0.50	0.75	0.90	1.00	1.25	2.00	1.07
100FP	3.50	9.00	9.50	11.00	12.00	15.00	10.00
1000FP	0.00	24.00	22.00	24.00	28.00	40.00	27.60
10000FP	0.00	48.00	44.00	46.00	47.00	64.00	49.80
<b>100000FP</b>	<b>0.00</b>	<b>72.00</b>	<b>68.00</b>	<b>66.00</b>	<b>78.00</b>	<b>85.00</b>	<b>73.80</b>

Table 4 highlights some potential nuances among software variants in various yet representative categories. As supported by the data, larger and more complex software is deployed by military organizations, core systems, and those having their own intrinsic commercial value in the market place. This is also highlighted by the longer delivery schedules to allow for longer testing cycles for these categories of software. For example, if these categories of software are contrasted with software that serves the end-user community, the data show that end-user software is generally of lower complexity and consequently shorter delivery timelines. It is also reasonable to assume that this is due to simpler requirements and perhaps lower risk perceived by sponsoring organizations catering to individual end-user needs than those catering to larger constituents with expansive software utility and more at stake.

In a study that examined 250 large software projects over a 9 year period, software management practices were examined to determine patterns inherent in project failures and successes. The study confirmed a few patterns which are of particular relevance to this research study and provides added impetus [20]:

1. The majority of schedule and cost overrun related failures occur in the testing stage caused by poor project quality control, management of defect resolution, and planning for the remaining stages of the project. These factors are not as easily predictable in advance by the project management team. This naturally substantiates premise of this research study which is to focus metrics that most closely correspond to the later stages of the project execution phase – testing and deployment.

2. Most project failures trace back to poor project planning while successful project planning methods tend to be highly automated. This declaration further supports the need to improve automation tools for the software PM in particular and is, therefore, a focus area of this research study.
3. Successful projects have a higher defect resolution rate when compared to unsuccessful projects. The study found that successful projects experience 4.0 defects per function point and remove about 95% before deployment. Unsuccessful projects experience 7.0 defects per function point and remove only about 80% before deployment.
4. Projects use one or more project management tools with varying degree of proficiency and sophistication. However, most of these tools were built for other applications aside from software engineering and do not offer any estimating capabilities, quality control features, or measuring of efficiency issues (e.g. defect removal efficiency).

It must be recognized that organizations work towards managing projects to maximize business benefit while minimizing the risk of related financial losses. Since delayed schedules can often result in increased cost, larger projects are of particular concern to sponsoring organizations since they tend to have longer delivery schedules and therefore an increased risk of experiencing substantial delays. By some measures and reports, software project failure costs and its associated implications are staggering:

- A report by Roger Sessions in 2009 stipulated that cost of IT project failure as a percent of GDP to be as follows: \$6,180B (world), \$1,225B (USA), \$3.9B (New Zealand), \$200B (UK), and \$110B [21]. The report further shares that IT project

failure costs have surmounted to over \$500B per month and the problem is getting worst. While these numbers represent IT aggregate numbers, the percent attributed to software related failures is lower yet still staggering [21].

- 80% of technology projects actually cost more than they return [22].
- Up to 80% of budgets are consumed fixing self-inflicted problems [23].

As discussed earlier, there continues to be a large need and associated benefit of using greater data-based automation to manage software engineering projects. In fact, the need and benefits of doing so are universal in engineering and have also been confirmed for other types of projects showing that less-performing projects present significantly lower system utilization levels than the other projects [24]. Another study by Raymond established that the use of robust project management information systems is advantageous to PM's. The study confirmed improvements in effectiveness and efficiency in conducting managerial tasks related to project planning, scheduling, monitoring, and control. Improvements in productivity were also observed in terms of timelier decision-making [25]. This study also acknowledged that benefits often extend beyond the PM as an individual to the performance of the overall project. Such benefits included improved budget control, meeting of deadlines, and addressing technical complexity with greater ease than without having such systems in place [25].

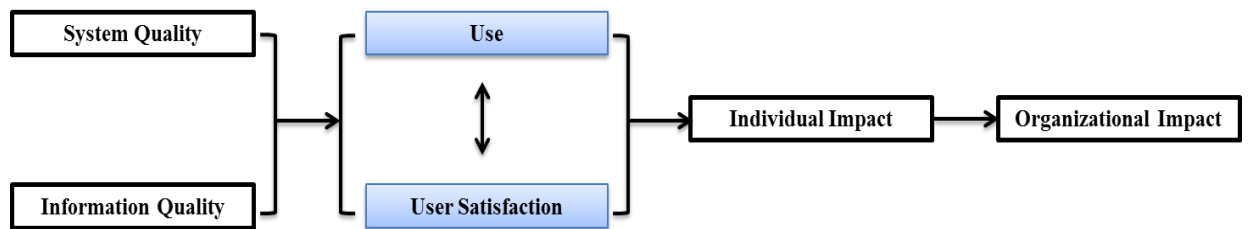
Previous research substantiates the need for improved predictive modeling tools and techniques to alleviate challenges that result in project delays such as quality, number of defects, and complexity of software. This call for action is a key motivator for this research study.

The study of the software industry and the associated management challenges would not be complete without some important notes on factors that make projects in this industry different from other industries such as construction, automotive, scientific exploration, and medicine. Many of the techniques of general project management can be applied but software projects have certain characteristics that make them different. Most software is inherently invisible, generally complex, virtually changeable, and easily conformable [26-31]. Software project management is a process of making visible that which is invisible. Unlike a bridge being constructed, software progress is not immediately visible. Software products are more complex when measured per unit of currency, than other engineered artifacts. The ease with which software can be changed is usually seen as one of its strengths. However, this means that where the software system interfaces external systems, it is expected that software can be changed to accommodate when necessary. Software systems are likely to be subject to a high degree of change. This results in higher pressures on software project management practitioners resulting in greater variability of outcomes. Software project managers need to trade-off characteristics, preferences, and quantities while balancing requirements, expectations, perceptions, opportunities, and risks [32]. Real-time decision-making frameworks and techniques are crucially important as they can help alleviate these challenges.

### **3.2. Software Quality Attributes**

As stated previously, many researchers have utilized a vast amount of OSS data which is publicly accessible through various software repositories. These OSS repositories can be used to gain insights into the software development process, its management, and its effectiveness. In this research study, OSS data can be used to

understand and develop improved measures of success for software projects. Measuring project success is useful for the effective and reliable assessment of ongoing projects. Measuring success is also extremely useful when software is monetized in both the OSS and PSS communities. Software project sponsors can only evaluate the return on investment if success criteria can be identified and subsequently measured. Over the years, software success has been measured in numerous ways and with varying levels of sophistication. A commonly cited model for Information Systems (IS) success was developed by DeLone and McLean [33] and is shown in Figure 8. This model suggests six interrelated measures of success: system quality, information quality, use, user satisfaction, individual impact, and organizational impact:



**Figure 8. DeLone and McLean's Success Model [33]**

While the model above is considered reasonable and complete by many researchers, the literature review also suggests several challenges when trying to measure results for some of the variables referenced in the model. Crowston et al. describe each measure of success and identify key indicators [34]. They also admit to potential issues for each based on their research especially as it relates to OSS, the primary data source of interest for this study. Table 5 below summarizes these findings and claims:

**Table 5. Summary of Measures of Success, Indicators, and Potential Issues**

Measure of Success	Indicators	Potential issues
<b>System and</b>	Code quality (e.g. understandability,	• Code quality is

Measure of Success	Indicators	Potential issues
<b>Information Quality</b>	completeness, conciseness, portability, consistency, maintainability, testability, usability, reliability, structure, efficiency), Documentation quality.	<p>generally good in OSS so the measure may prove to be of minimal value – in software engineering code quality does not imply software project execution quality.</p> <ul style="list-style-type: none"> <li>• Data related to code quality in OSS repositories may not be adequate.</li> <li>• Many of the quality indicators are highly subjective in nature making it difficult for researchers to code accurately.</li> </ul>
<b>User Satisfaction</b>	User ratings, Opinions on mailing lists, User surveys.	<ul style="list-style-type: none"> <li>• Surveys are the only reasonable way of ascertaining this measure; surveys are often subjective and based on a non-random sample (i.e., users who take the time to volunteer a rating within the OSS community).</li> </ul>
<b>Use</b>	Use (e.g. Debian Popularity Contest), Number of users, Number of downloads, Inclusion in package distributions, Popularity or views of information page, Package dependencies, Reuse of code.	<ul style="list-style-type: none"> <li>• The best measure of the four identified by DeLone and McLean's Success Model.</li> <li>• Used by many studies as an indicator of success.</li> <li>• Especially relevant for OSS.</li> <li>• In research, must adjust for the phenomena that highly successful (and stable) software may not be downloaded too often</li> </ul>

Measure of Success	Indicators	Potential issues
<b>Individual and Organization Impacts</b>	Economic and other implications.	<p>as there would be no need by users to do so.</p> <ul style="list-style-type: none"> <li>• Impact measures are difficult to define for OSS and PSS projects; they are even harder for OSS due to the difficulty in defining the intended user base and expected outcomes.</li> </ul>

Usage and user satisfaction related measures are easiest to measure with the least amount of issues that can be anticipated as shown in Table 5 and therefore highlighted in Figure 8. Based on these findings, many researchers have settled on relating software success to the level of use of the software over periods of time [14, 34]. It has been acknowledged that software measurement is required for practical software process improvement (SPI) to ensure improvements are actually addressing the correct issues [35]. SPI has been in the spotlight in industry and academia in recent years. Additional bodies of research and publications have focused on practical SPI. Despite the increased focus placed on SPI by researchers, change management professionals, quality assurance managers, process owners, and researchers continue to be challenged in defining success achieved in SPI [35-37]. After conducting an independent literature review, Abrahamsson appropriately lays out SPI success dimensions and differentiates between “hard” and “soft” measures with a relative estimate of difficulty in attainment in Table 6 below [38]:



**Table 6. SPI Success Dimensions**

<b>Success Dimension</b>	<b>Types of Measurements</b>	<b>Relative Estimate Difficulty</b>
<b>Project Efficiency</b>	Hard measures (e.g. work effort)	Low
<b>Impact on the Process User</b>	Soft measures (e.g. satisfaction, ease of use; work morale; level of excitement, teaming, collaborative practices used)	High
<b>Business Success</b>	Hard measures (e.g. productivity)	Moderate
<b>Direct Operational Success</b>	Hard measures (e.g. defect ratio)	Moderate
<b>Process Improvement</b>	Both (experience database)	High

As with software metrics, one can reasonably assume that SPI related soft measures such as satisfaction levels, morale, and level of excitement are more difficult to measure with sufficient objectivity than hard measures such as level of productivity and defect ratios. Soft measures present the greatest challenges when measuring their direct impact on SPI. Such measures have increased subjectivity and may not be easily attributable to a specific SPI related variable (e.g. individual morale can be influenced by many variables).

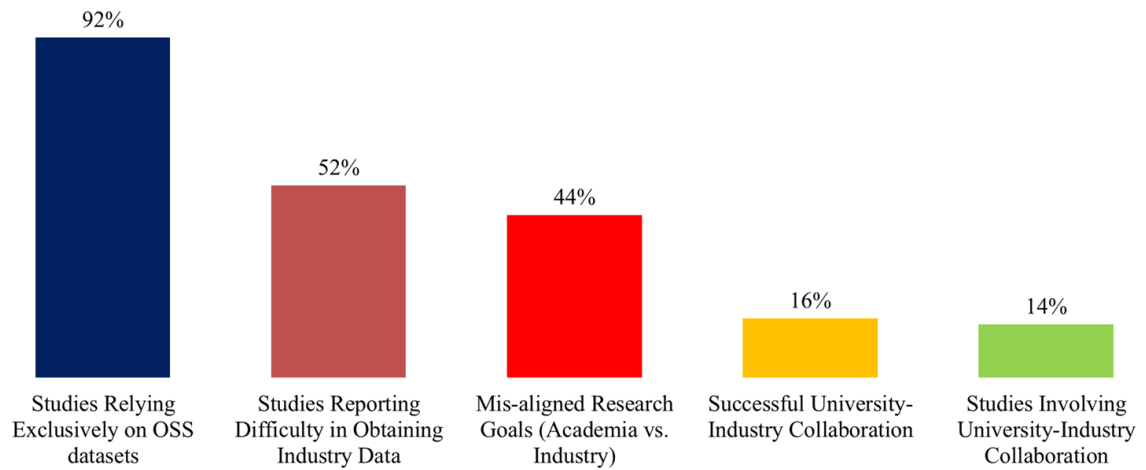
Equally important to using proven predictors is the notion of establishing metrics that can be tied to processes being addressed programmatically as part of the software engineering process. Catal suggests that metrics based models are so important that they must be frequently revised (i.e., real-time) while the project is underway – perhaps even rebuilt from scratch each time the process or the organization experiences a change [39]. Other researchers have also suggested that evaluation and prediction are two separate learning schemes using historical defect data to predict defects for new data [40]. Song, Shepperd, and Liu note considerable variations in the performances of predictors across

data sets suggesting that a simple search for the “best” predictor may prove to be pointless unless the research is targeted towards answering a very specific question [40].

At this time, there is an important and final point to be made regarding the dangers of measurement and software metrics in general. While DeMarco classically reminds us that in order to be able to control a process or a product, measurements are a definite prerequisite, he equally reminds us that it can be very expensive to collect “good” metrics and even more expensive if “bad” ones are inadvertently collected and used [41]. However, the author goes on to support his claim that software metrics are worth it in the end as they can help the organization improve processes and focus management attention on the real drivers to course correct when necessary. Dekkers and McQuaid submit that measurements can enhance or misguide software projects [42]. The authors explicitly state that it behooves software project management spend the required time on the people and cultural issues that ultimately can provide lasting remedies. These findings support the motivation and premise for this research study and guide the efforts accordingly. This research will focus on hard measurements (i.e., easier to measure, more objective, and more quantifiable measures) in favor of subsequent management action required to course-correct projects by adjusting the levers in softer aspects (i.e., culture, environment, and change management aspects) that impact projects. The social science aspect of timely management decisions in these softer areas of software project management has been amply studied and conjectured by academia and industry practitioners. However, a gap appears to be in the use of the hard measures in real-time while project activities are underway to drive timely management intervention to achieve course-correction *before* failure occurs. This represents a real opportunity for this research study.

### 3.3. Research Data Validity

One of the greatest challenges experienced when researching the quality of software management is the reluctance of organizations to share unbiased data related to their own software management practices. In their research study, Tripathi et al. reviewed 187 research studies that span over a period of 5 years (2010 to 2014) and found that 91.9% of them rely solely on OSS datasets and only 14.4% involve university-industry collaboration [43]:



**Figure 9. Mining Software Research Spanning 5 Years – Study of 187 research papers, 2010 – 2014**

Subsequently, Sureka et al. reported that more than 50% of the researchers which they surveyed had reported difficulties in obtaining industry data on software and indicated that this was a major impediment to greater university-industry [44]. OSS communities have helped to address this research data gap to some extent [13, 44-48]. Many researchers and practitioners have acknowledged that there is an abundance of OSS data available and some of these are proven and tested as reliable inputs for conducting unbiased research in related areas. Large OSS communities of actively engaged contributors have documented the features of numerous software products and various

other key process related attributes that help researchers understand the quality of software development and management practices. Proper due diligence to appropriately consume OSS data can be facilitated to determine track record, performance, and maintenance aspects of software [49].

The validity of software engineering research and supporting sources of data continues to be investigated by researchers. By recognizing the benefits and biases of various data sources, researchers can better improve research quality and address the issues of validity given the differences between proprietary and open source software development. Mathieu et al. [50] establish a key connection between OSS creators and entrepreneurs by examining OSS creator motivations across 3 entrepreneurial dimensions: the opportunity, the organization, and the business models. They find similarities as both constituents exclusively aim for value creation. With regards to OSS-based business models specifically, the authors find that they fall into one of 5 categories: donations or gifts from users, enhancements of preexisting products, software sold for commercial interest, services-based offerings, and services-based partnerships. This finding supports modern day market place realities of OSS and explains how globally based open entrepreneurs have self-organized themselves initially and virtually to ultimately create enhanced, mega-sized commercial software adaptations leveraging the open source movement. Open source software (OSS) and related communities provide product offerings that compete head-to-head with proprietary source software (PSS) across most emerging software categories, including cloud-based operating systems (e.g. Linux) web server technology (e.g. Apache HTTP Server), database engines (e.g. MySQL Database), Web 2.0+ development environments (e.g. PHP), and widely adopted

internet browsers (e.g. Firefox) [50-52]. In fact, OSS provides ample opportunity for generating revenue and reducing certain costs and is therefore heavily leveraged by most enterprises on a global basis. Table 7 shows relevant statistics and summarized results of a recent OSS survey (2016) by BlackDuck Software Enterprises, an independent authority on OSS and has been surveying the market annually for the past 10 years:

**Table 7. OSS by the Numbers [53]**

	Statistic
<b>Percent of enterprises globally run on OSS</b>	78%
<b>Percent of enterprises that do not rely on OSS</b>	3%
<b>Estimated number of active OSS projects</b>	180,000+
<b>Percent of enterprises contributing to OSS</b>	65%
<b>Percent increased use of OSS within the enterprise</b>	65%
<b>Top reasons cited for using OSS</b>	Increased efficiency, improved interoperability, and greater innovation
<b>Emerging technologies leveraging OSS</b>	Cloud computing components, big data, content management, databases, operating systems, development tools, and mobile technologies

Notable examples of such OSS-based collaborative efforts include LINUX, Apache, MySQL, and the Java programming language.

In the category of large distributed projects, proprietary software systems (PSS) and OSS is very similar in nature. However, PSS teams generally operate with a greater degree of privacy, resulting in weaker datasets that are a barrier for research. This is especially true of software fault prediction and related data that has been experienced by researchers in the past [54]. OSS data, on the other hand, can provide a richer data set full of insights that are transferable when trying to understand PSS management processes. Many organizations leverage the possibilities of increased globalization to widely

distributed teams in an effort to maximize PSS team productivity. Such software development practices mirror those of the open source [55].

A separate body of research has theorized extensively on organizational learning in the context of software development. Organizational learning is a prerequisite for long-term, continued adoption of software quality predictive models. In order for predictive models to be effective and sustaining, sponsoring organizations must be willing to learn and they must have a robust knowledge management process already in place. Classic research studies such as Raymond's Cathedral and Bazaar [56] explain that OSS and PSS communities have vastly different cultures in the way in which they release software and their fear threshold for rejection by their peers. The author claims that, by nature, OSS is served by iterative communities who strive for small incremental wins whereas, in contrast, commercial vendors (considered as part of the PSS community) are expected to strive for increased perfection than the frequency of release, defect resolution, and predefined software quality goals. The implication is that it is the commercial aspect which drives PSS delivery, not the individuality and free spirit which drives OSS community members. More generally, the authors seem to imply that intrinsic motivators behind the actions taken by key actors are perceived as being different for the two communities being compared. Israeli and Feitelson [57] highlight the dominating indicators of software success as being market share in the case of PSS and the number of downloads in the case of OSS. However, the findings of this body of existing research need to be tempered appropriately by recognizing that the main purpose being strived for is singular in either case; it is to develop good quality software products which command high rates of adoption by the target set(s) of users. A second argument can be made that

organizational learning happens within both OSS and PSS communities as long as the appropriate tools are made available to capture the right data, organize the information, and engage community members adequately. Third, from a practical perspective, OSS and PSS communities increasingly co-mingle and work in cooperation with each other since many of their members also benefit from having joint membership of both communities.

Huntley, while elaborating on the same topic, credits Raymond's contrast of the two communities [58]. The author provides a strong argument and a perspective – while the individual OSS developer operates under minimal supervision and is not confined by lack of rigor and process, collectively these individuals are highly effective and are able to perform and create quality products over periods of time. The OSS communities achieve this by employing rigorous learning processes using specialized tools at each stage of the process (e.g. use of formalized bug tracking and version management processes). Some of these tools can be used to learn more about software quality processes which can have implications that are transferrable to both OSS and PSS projects.

Most primary research projects are bounded by the quality, availability, and accessibility of relevant data. This study is no different. SourceForge provides a large and adequate set of accessible data on OSS development projects. This has made the world's largest repository a highly valuable data source for research [59]. Even still there have been concerns expressed about the accuracy and validity of data available in SourceForge. One of the key concerns that have been raised is the quality of the data that is generally available as some of it is self-reported by project owners and administrators [59]. Lerner and Tirole [60] provide a reasonable explanation that alleviates this concern.

The authors explain that project leaders use the data to recruit new developers, attract new users, and solicit donations for their projects. As such, any deliberate entry of inaccurate data regarding projects will be naturally avoided by those who are involved with OSS development on SourceForge. Software evolution researchers have found it acceptable to use OSS data repositories for certain types of research [44]. Notably, select Lehman's laws [61] of software evolution specifically pertaining to measuring continuing software change and growth, also established measures of software quality in the PSS domain, were deemed to be applicable to OSS [62].

The SourceForge repository was selected as the primary source of research data for this study. The criteria used for selection include relevancy, accessibility, reliability, sufficiency, validity, and longevity. SourceForge originated in 1999 and was the first provider of free and open source software (FLOSS). Over the years, the independent company was owned by many other larger ventures such as VA Software, Dice, and BizX LLC. SourceForge remains the industry default OSS repository and boasts having industry-leading tools, a collection of 430K+ projects, 3.7M+ registered users, 41.8M+ customers of software, and 4.8M downloads per day [63]. Earlier in the research effort, there were concerns that the repository's vastness may prove to be its limitation. After further research, a solution was discovered. Exclusively for research purposes, the SourceForge Research Data Archive (SRDA), located on the University of Notre Dame data servers was created in 2003 after a group of researchers received several grants from the National Science Foundation (CISE IIS-Digital Society & Technology program under Grant ISS-0222829 and by the CISE Computing Research Infrastructure program under Grant CNS-0751120) [64].



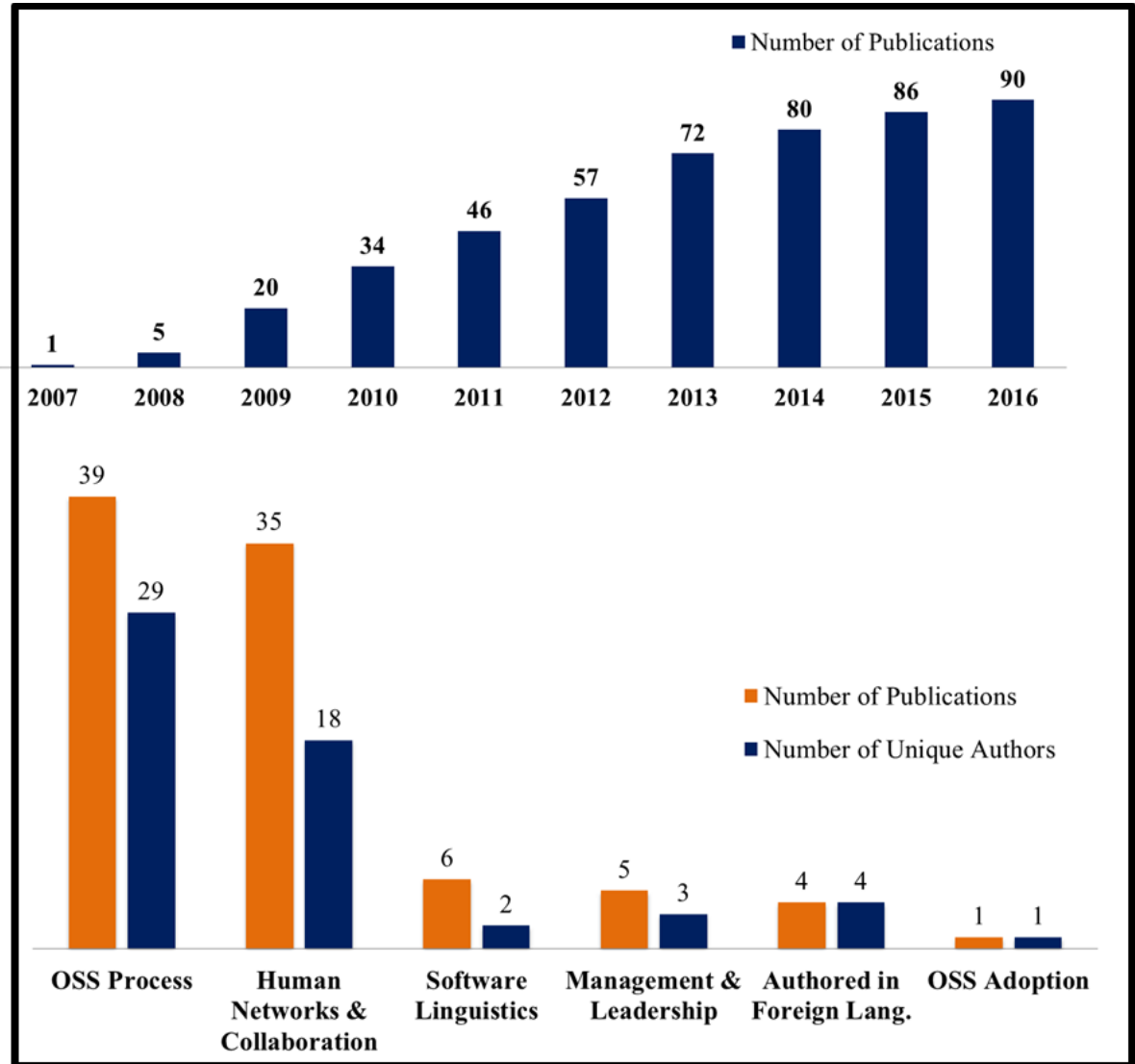
The SRDA assimilates and normalizes a collection of OSS data directly from SourceForge on a monthly basis [46]. This practice has continued until September 2014 after which the data was frozen and has been made available for use for further research. Selected data feeds from 2003 to 2014 are licensed and provided by SourceForge to SRDA developers for research consumption. Over 100 researchers worldwide have used this data archive for research purposes because of its accessibility, ease of use, and reliability [64]. The data is made available in a relational database format which can be queried using standard SQL procedures. It is anticipated that the following data entities from SRDA will be used in support of this research study. Detailed data relationship diagrams are supplemented in Appendix A section of this report. The key data entities are shown in Figure 10 below:

1. Artifact-Bugs	• Contains bugs (defects logged) for all projects on the SRDA
2. FRS Package	• Contains relationships between groups and packages
3. FRS Release	• Contains activity related to releasing software
4. Groups	• Contains information regarding software groups
5. Stats Project All	• Contains select aggregate data about projects
6. Trove Category	• Contains categories of software
7. Trove Group Link	• Contains relationships between groups and categories

**Figure 10. Select Data Entities from SRDA**

The SRDA has served as an established and reasonably reliable source of data used by software researchers in the past. As part of this research effort, 91 research

publications that were readily accessible have been analyzed to determine the knowledge contributions made by the studies. The analysis has been summarized in Figure 11:

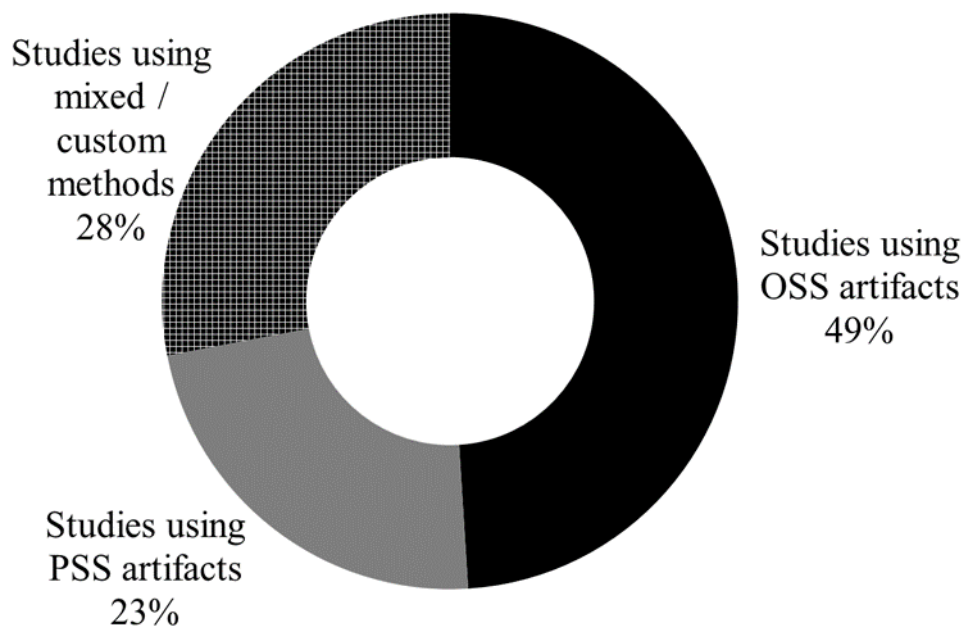


**Figure 11. SRDA – Existing Research – 10 Year Period, 2007 to 2016**

During the past 10 years, most of the focus of existing SRDA based research has (i.e., 82%) has been placed on better understanding OSS processes and the study of human networks and collaboration mechanisms within OSS communities. While such research provides valuable insights and support to this research study, there remains ample

research opportunity to leverage SRDA data for direct benefit to software project management and industry practitioners. It is also evident from our comprehensive review that existing research outcomes, while interesting and quite possibly applicable to the software management industry, remain in academia; there have been no frameworks, blueprints, or methods offered to aid in institutionalizing the research to help firms realize the potential organization benefits of the research. Cumulative year-wise analysis of SRDA based publications, publisher analysis and a detailed cross-reference table supporting the analyses presented in this section has been included in Appendices B.1, B.2 and B.3 respectively.

With further regard to the validity of OSS data in academic research efforts, Wright et al. investigated 266 empirical studies and found that 49% used OSS artifacts exclusively while only 23% used PSS artifacts for research [55]. The remainder used a custom (e.g. derived) or a combination approach as shown in Figure 12:



**Figure 12. Review of 266 Research Papers Using OSS vs. PSS Artifacts**

Another similar claim was provided by Sureka et al. based on a survey conducted in 2015 where the authors reported that 54% of research studies used OSS data exclusively and 9% used PSS related data solely to conduct their research [44]. The remainder of the population surveyed used a mixed approach. The authors also found that over 77% of the research studies surveyed reported either exclusive use of OSS data or mostly OSS data to conduct their research.

OSS networks rely heavily on newcomers that can actively participate in the development of software over longer periods of time. Steinmacher et al. highlight several barriers that threaten newcomer entry [65]. In their literature review, the authors report that the barriers to entry were largely centered on lack of social interaction with project members, receiving timeliness of response, and good project documentation. While still geographically dispersed and individually motivated to participate, OSS community members must still value that they are a small part of a larger team organization that works on a software project over a period of time. Faraj et al. [66] in their research and Sahin [67] in his respective research work uniformly establish that teams are a primary mechanism for accomplishing organizational work, especially on software projects. Team building, team size, and cooperation amongst a team are critical factors in developing a quality software project. Interestingly, these findings which suggest the importance of quality documentation, timeliness of data, and the importance of teaming are highly appealing to researchers contemplating the use of OSS data to study software and process quality measures.

The literature review suggests that careful use of OSS data repositories to understand software engineering processes can be effective, more manageable, and more reliable in

some cases when compared to PSS data. Research shows that there remains a unique opportunity provided by OSS to advance the study of the software development process and its associated quality attributes. This research study intends to utilize this opportunity.

### **3.4. Software Management and Predictive Modeling**

The final area of focus by the literature review has been on reviewing current practices of software project management and predictive modeling opportunities conducive for further research. Since this research study confirms that objective planning, measurement, and benchmarking are largely missing in PM practices, it is important to determine the root cause. Software management practices start at the onset of the education process for most technical developers and are subsequently adopted by experienced PM's. Given this working philosophy, it is important to shift the focus from a review of industry practices into a brief review of current academic course work related to software engineering. Current software engineering curricula at the undergraduate level were primarily examined. The findings were consistent and possibly a consequence of current industry practices – there is insufficient focus being placed on up-front planning, benchmarking, and metrics in the software education curricula. As a result, important changes must be made to these training methods to teach future matriculated students how to utilize experienced-based knowledge to better predict software project outcomes. Jagtiani and Lewis [68] reported that a greater focus is required on planning and ongoing metrics validation at our universities than evident with current practices. Their research was based on the gaps which they identified between the learning outcomes provided by the authoritative IEEE Software Engineering 2004 Curriculum

Guidelines [69] and a review of standard software engineering course content shared by the same report. Their research summarizes the gaps in Table 8:

**Table 8. Gap Assessment – Learning Outcomes for Course Curricula [68]**

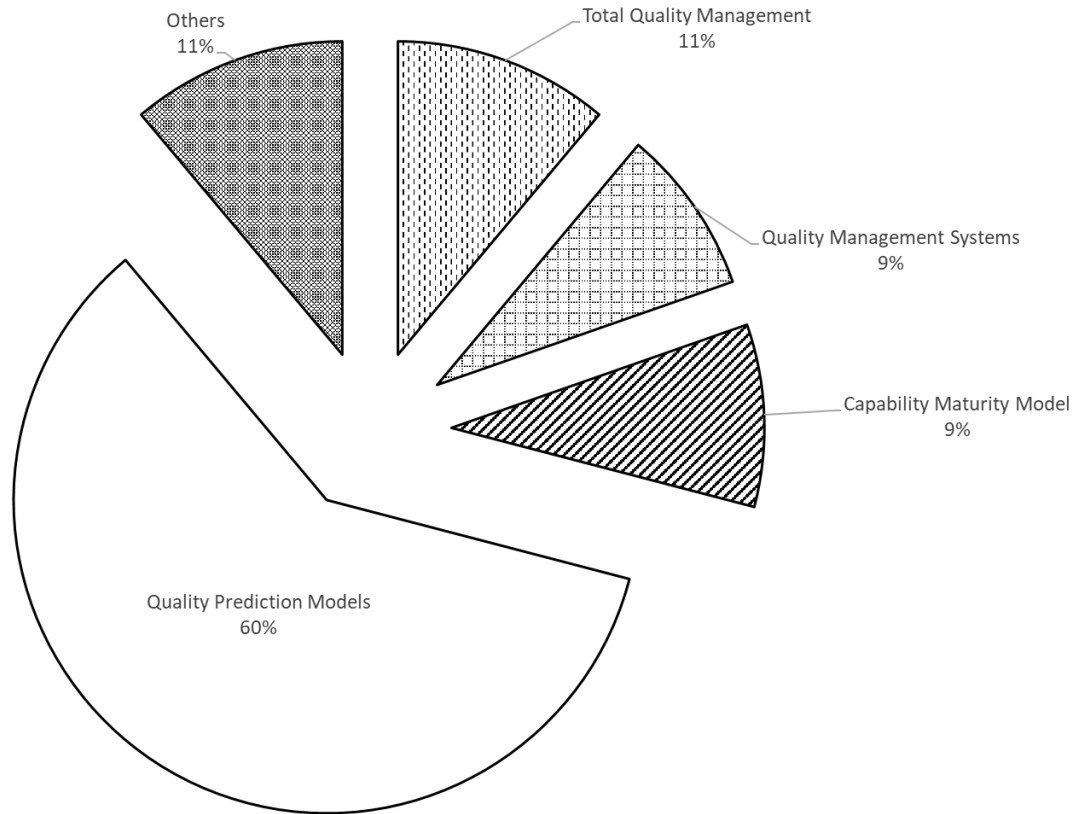
<b>Summary of Intended Learning Outcomes – Software Engineering Undergraduate Curricula</b>	<b>Gap Assessment: Course Topics Coverage</b>
1. Gaining knowledge of software engineering and issues.	Planning, Metrics Required
2. Learning to work individually and in teams.	Topics Adequately Cover
3. Resolving conflicts between cost, time, knowledge, existing systems, and organization.	Planning Required
4. Designing appropriate solutions that satisfy and integrate ethical, social, legal, and economic concerns.	Planning, Metrics Required
5. Learning to apply theories, models, and techniques to identify problems, implement solutions and verify results.	Planning, Metrics Required
6. Understanding the importance of negotiation, effective work habits, leadership, and good communication.	Topics Adequately Cover
7. Learning emerging models, techniques, and technologies as they emerge and the importance of doing so for ongoing efforts.	Unable To Determine

Undergraduate level training on software engineering is generally in need of revision and requires an increase in focus on theory application, models, and techniques to improve planning, metrics, measurements, and overall predictability of software development efforts. While defining solutions and evaluating recommendations potentially useful to software engineering education and related processes is not directly in scope of this research study, it is important to note the gap to a) substantiate the need for PM training and methods in this area which is important to this study and b) identify future opportunity for research and impact.

Since software predictive modeling is a clear objective of this study, accuracy, and cost and schedule estimation aspects of software project management were been studied. Estimation implies risk which is the result of recognizing uncertainties and balancing it with benefits and utilization of organization resources. DeMarco and Lister [70] state it

appropriately in their book, *Waltzing with Bears*, when they convincingly conjecture by stating that the only projects that are worth doing are those that come with risk and that without any risk we may no longer expect returns. The authors remind us that while risks are a reality, poorly developed cost and schedule estimates definitively and adversely affect project success. 41 years ago, in 1975, Frederick Brooks [71] stated that the biggest reason for projects to go off track is due to schedule compared to all other reasons combined. Optimistic estimation continues to be one of the two most common reasons for out-of-control projects [72], and cost and time-related faults are the biggest reason for software failures in day-to-day practice [73]. Boehm also affirms the same by including overly optimistic schedules and budgets in his list of top ten risks faced by software projects [74]. Given the reported and repeat project failure rates reviewed earlier coupled with previous research findings on the challenges related to estimation, it appears that project management practices require more sustainable improvements in this area.

John et al. conducted an extensive review spanning 117 publications of past research focused on software quality management practices [75]. The results of the review were published in June 2016 and show the relative influence of various software management practices over software quality. Figure 13 summarizes the results and provides overwhelming support for the use of predictive models to drive quality:



**Figure 13. Influence of Top Software Management Practices on Software Quality**

In addition, the above research presented several key findings of particular importance to this study:

- Project management practices that employed increased Total Quality Management (TQM) and Capability Maturity Model (CMM) processes and associated implementations also demonstrate increased software quality.
- Regarding the selection of software quality prediction models, multiple models have been developed in the last several years and there is no single approach that is applicable for all software projects.



- Regarding the use of software quality prediction models, most models use static attributes such as code complexity which are not routinely measured or influenced by the PM's.
- Regarding the design of software quality prediction models, while process performance-based models use quantitative management techniques to manage the software process, such models are mostly based on regression analysis and simulation techniques. Different algorithms, in particular, of the AI or machine learning type, are a recognized opportunity for future research. While there are examples of past research which have successfully incorporated machine algorithms to predict project success in other industries (e.g., Wang et al., (2012) use of neural networks in the construction industry [76]), greater opportunities to extend research remain in the software industry.
- Performance characteristics of software projects continue to be largely qualitative.
- A positive correlation exists between software quality, productivity, cycle time, and development effort. Future predictive models should use quantitative methods to manage multiple performance characteristics.
- Because of the above two findings, the authors affirm that models for simultaneously monitoring of quantitative and qualitative performance characteristics are a substantive future research opportunity.

Since the development of a valuable and novel predictive model is of importance to this research study, the research findings shared specifically by John et al. regarding the generally limited use of machine learning in the development of such models were further examined [75]. This research study further examined the results of the authors'

identification of quality prediction model based research depicted as 60% penetration in Figure 13 by contrasting it with a list of widely accepted machine learning algorithms as summarized by Chao [77]. Table 9 shows the gap in research with regards to the use of machine learning models. This research study is particularly interested in examining supervised learning and parametric based learning models (e.g., Naïve Bayesian) given the nature of software quality attributes inherent in the research hypotheses:

**Table 9. Gap Assessment – Software Quality Prediction Models in Research Studies**

Type	Category	Representative Methods	Research Gap	Coverage (No. of Studies)	References
<b>Supervised Learning</b>	Linear model	Perceptron	Yes	2	[78], [79]
		<b>Multi-layer perceptron</b>	Yes		
		Support vector machine	No		
		Support vector regression	No		
		Linear regression	No		
		Rigid regression	No		
		Logistic regression	No		
<b>Supervised Learning</b>	Non-parametric model	<b>K-nearest neighbors</b>	No	6	[80], [81], [82], [83], [84], [85]
		Kernel density estimation	No		
		Kernel regression	No		
		Local regression	No		
<b>Supervised Learning</b>	Non-metric model	Classification, regression tree	No	13	[86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98]
		<b>Decision tree-based systems</b>	No		
<b>Supervised Learning</b>	Parametric model	<b>Naïve Bayes</b>	Yes	23	[99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123]
		Gaussian discriminant analysis	No		
		Probabilistic graphical models	No		
		Bayesian Networks	No		
		Neural Networks	No		
<b>Supervised Learning</b>	Mixed method	<b>Bagging (bootstrap + aggregation)</b>	Yes	3	[124], [125], [126]
		<b>Adaboost</b>	Yes		
		<b>Random forest</b>	Yes		
<b>Unsupervised Learning</b>	Clustering	K-means clustering	No	3	[127], [128], [129]
		Spectral clustering	No		
		Association rule mining	No		

Type	Category	Representative Methods	Research Gap	Coverage (No. of Studies)	References
<b>Unsupervised Learning</b>	Density Estimation	Gaussian mixture model Graphical models	Yes Yes	0	N/A
<b>Unsupervised Learning</b>	Dimensionality reduction	Principal component analysis Factor analysis	No No	1	[130]

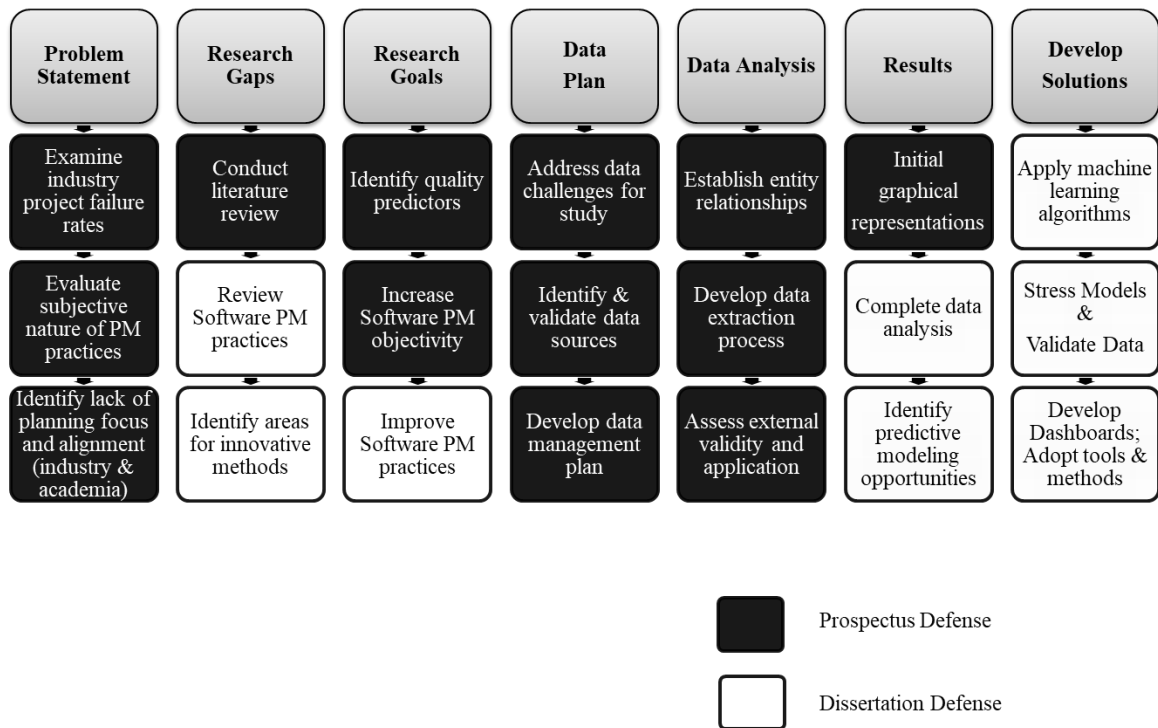
Naïve-Bayes is widely used, simple to set up, and robust with is accompanied with demonstrated examples in various applications such as pattern recognition [131], medical diagnosis [132], and defect prediction [102, 133-135]. Defect prediction models using Naïve Bayesian classifiers deliver the best prediction accuracy on public datasets compared with models with other classifiers [102]. A key reason for the success of the Naïve Bayesian classifier over other methods is that it combines inputs from multiple sources in a given process. The Naïve Bayesian-based learning method is not impacted by minor changes associated with the training data samples. The algorithm recognizes such changes and prevents unnecessary variations in the predictive results since it polls numerous Gaussian approximations to the numeric distributions [102]. Therefore, minor correlations between attributes or samples in the training set within the field of software defect prediction do not confuse Naïve Bayesian classifiers. For these reasons, Misirli et al. successfully used Naïve Bayesian techniques as the sole algorithm to develop a software defect prediction model and calibrated the model based on locally available and public data [133].

More recently, in 2016 a study which evaluated different families of prediction methods for estimating software project outcomes found that four classifiers had relatively high performance – Random Forest, Support Vector Machines, Multilayer Perceptron (a class of neural networks), and Naive Bayes [136]. These findings provide

further support and direction to this research project. However, the scope of the study was limited as it did not provide methods by which the said prediction models can be used by practitioners for project management decision-making.

## CHAPTER 4: RESEARCH PROCESS

This research study has been work-in-progress since January 2014. Figure 14 shows an overview of the process being followed to conduct this research study and to achieve its goals:



**Figure 14. Process for Research Study Execution**

The prospectus defense focused on defining the research problem, identifying research gaps, setting research goals, creating a data plan, conducting preliminary data analysis, and demonstrating initial results. For completing the dissertation thesis and preparing for final defense, feedback from the prospectus defense has been addressed. In addition, a thorough review of software PM practices was conducted and new changes to existing methods and processes have been addressed and supporting research goals were

revised accordingly. Additionally, as part of completing the dissertation thesis, data analysis was completed, results were reviewed, predictive models were developed and validated, and implications were discussed vis-à-vis an application framework as a suggested method for implementation of the research outcome.

## CHAPTER 5: RESEARCH DATA

### 5.1. Data Plan and Expected Outcomes

This study requires careful consideration of the data and analysis to ensure the progress towards achievement of the research goals. Figure 15 shows challenge areas, key activities, and research outcomes for the data planning process which has been followed:

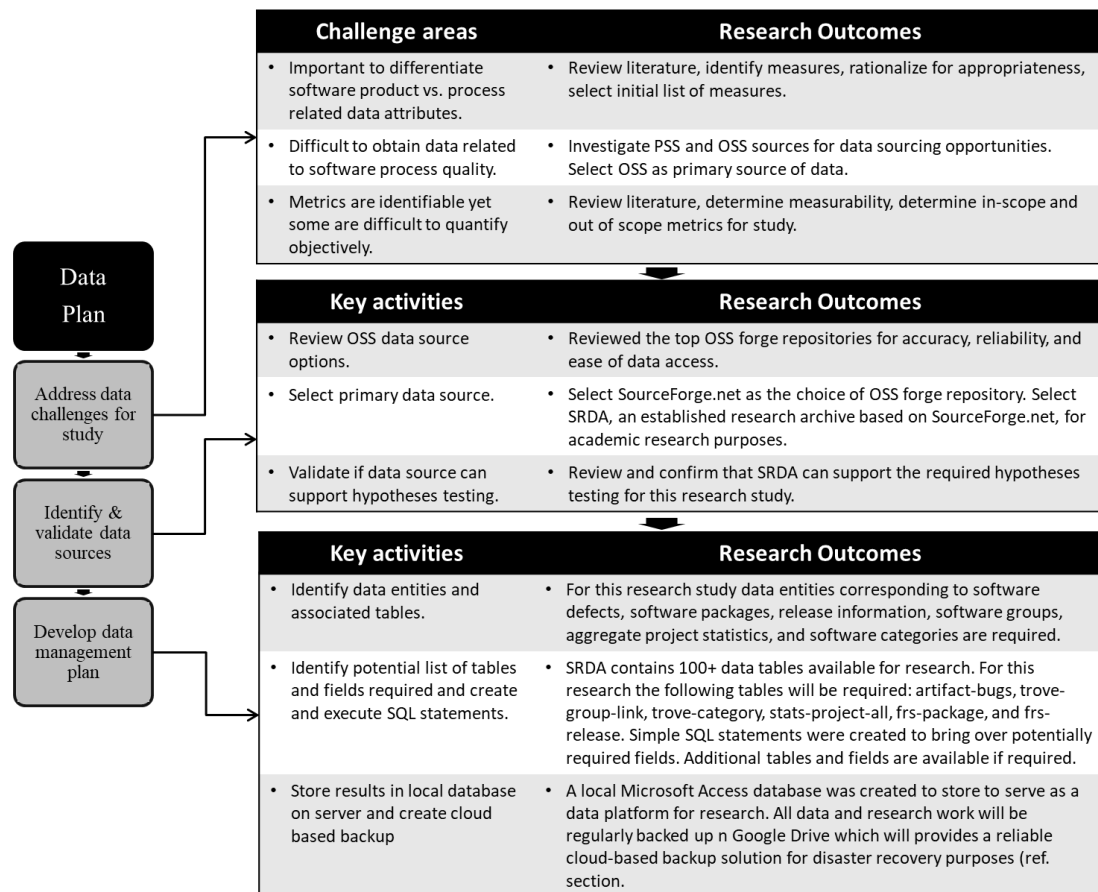
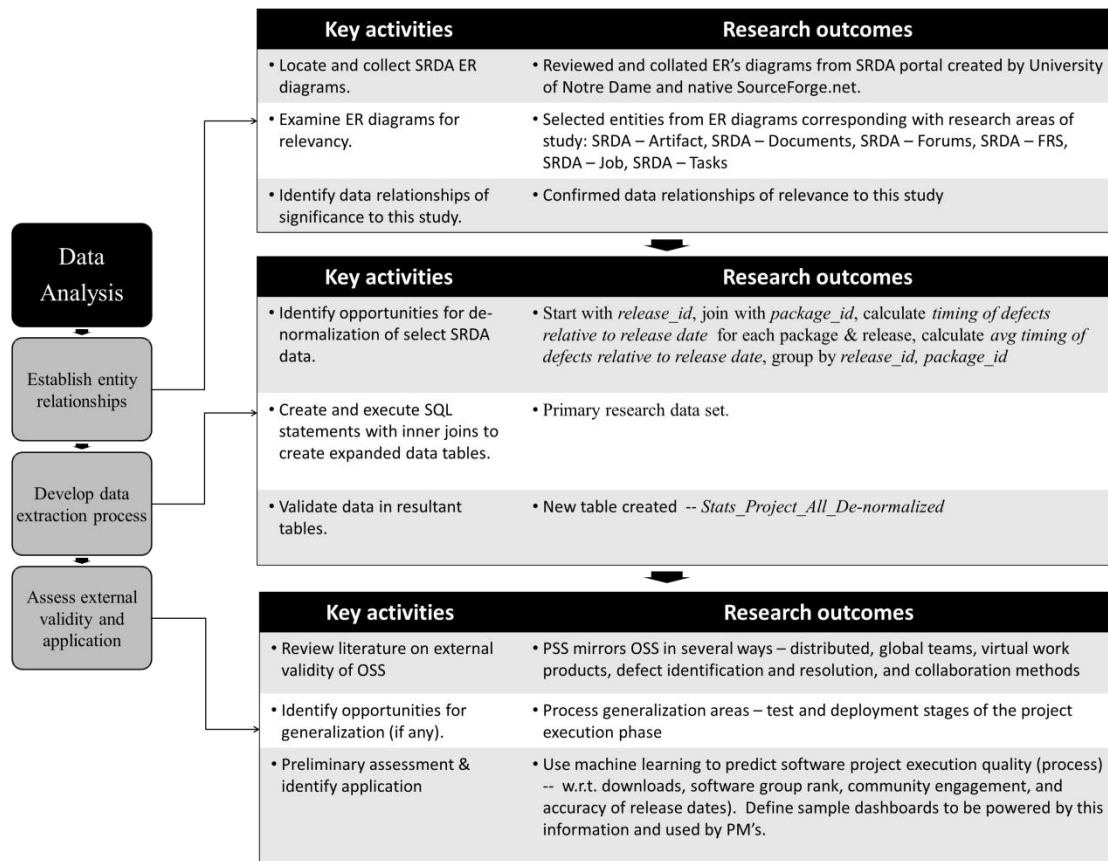


Figure 15. Data plan - Key Activities and Preliminary Outcomes

Similarly, Figure 16 shows key activities and research outcomes for the data analysis process which was followed:



**Figure 16. Data analysis - Key Activities and Preliminary Outcomes**

## 5.2. Data Management

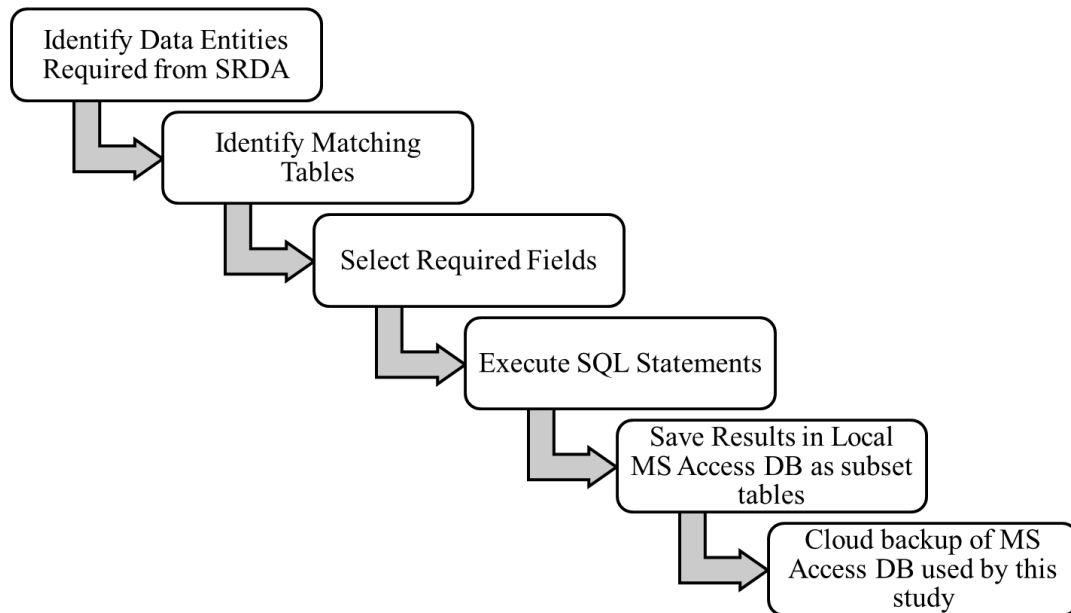
Selection, storage, and management of research data is a crucial prerequisite for this research study. While access to the SRDA servers was granted specifically for this research study, integrity and availability of the data relied upon specifically by this research study must be ensured. Furthermore, it is expected that the data archive, which cumulatively spans 11-year period (i.e., January 2003 through September 2014), will sufficiently serve the objectives set forth by this study as findings are expected to be thematic and aimed at improving software quality and predictability over periods of time.



As the web-service provided by SRDA has the following several limitations, they must be duly addressed to allow for research flexibility and completeness:

1. Long-running queries time out after 60 seconds.
2. The query interface does not allow for complex query definitions. Complex and nested queries involving multiple joins and unions are not possible using the interface.
3. Full downloads of the database are not permissible by the licensing agreement with SourceForge.
4. Long term availability of required SRDA data is unclear.

To address the limitations listed above and to address the requirements of this research study, the following mitigating steps were taken shown in Figure 17:



**Figure 17. Research Data Management Plan**

### 5.3. Data Extraction

As specified by the data management plan for this research study, the SRDA data warehouse was reviewed for potential software and process attributes. SQL queries were coded to extract fields that could be likely candidates for research and analysis as part of the project. A short sample of a SQL query used for selection of data is shown below:

```
SELECT release_id,package_id,status_id, preformatted, release_date, released_by  
FROM sf0914.frs_release r  
WHERE r.release_id > 200000 and r.release_id <= 500000
```

Once developed, such queries are entered into the supplied query form tool as shown in Figure 18 [64]:

**SOURCEFORGE.net**

**SourceForge.net Research Archive Query Form**

**SELECT:**

**FROM:**

**WHERE:**

**Separator**

- ☒ :
- ☐ ;
- ☐ #
- ☐ .
- ☐ XML

**Add SQL query to result file?**

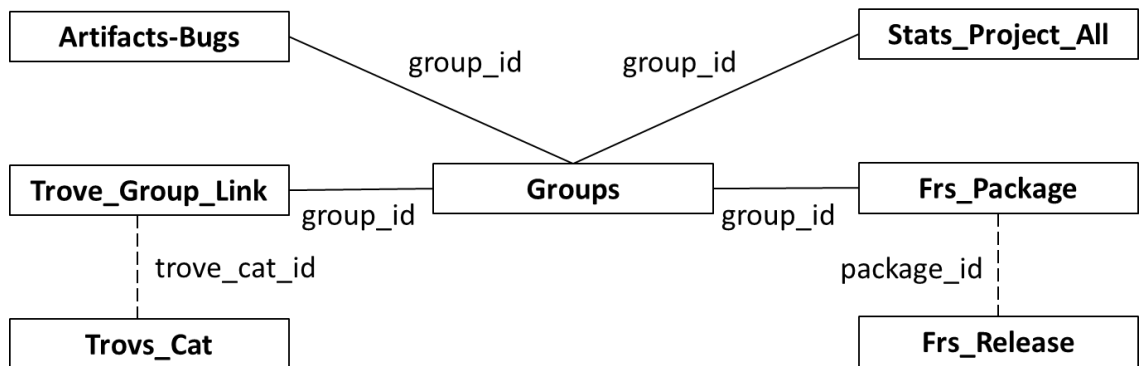
- ☒ yes
- ☐ no

[Query History](#)

1. Mon Sep 26 13:31:56 2016 -- SELECT count(*) FROM sf0914.user_group
2. Mon Sep 26 10:45:05 2016 -- SELECT release_id,package_id,status_id,preformatted,release_date,released_by FROM sf0914.frs_release r

**Figure 18. SRDA SourceForge Query Form**

Multiple queries were run to select the required data in small groupings to compensate for timeouts experienced by the SRDA Web Server. To alleviate query performance issues and to prevent web server timeout issues, a local MS Access database has been created to serve as a holding container for the data. Appendix B lists the tables and fields queried from and stored in the local database for further analysis. Over 9.5M records of data were leveraged for this research study before building out the data relationships that use inner joins that result in substantially increased the record counts in resultant tables. Specific field mappings were selected as they are suitable candidates for the data analyses required to test the research hypotheses. Figure 19 below shows the key relationships between the tables which are leveraged by the research study. These relationships are important and allow for accurate querying of the data and for building secondary relationships:



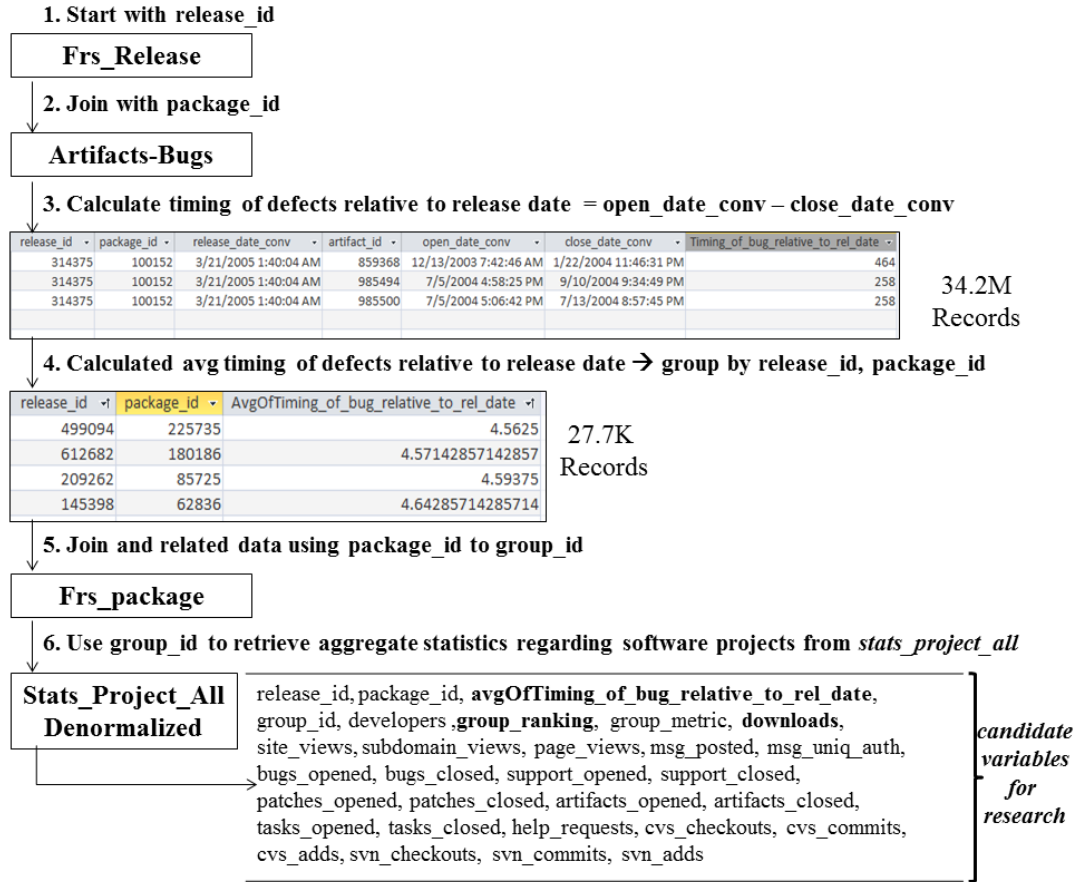
**Figure 19. Key Data Relationships for Select Data from the SRDA**

#### **5.4. Data for Analysis**

After conducting a thorough qualitative and quantitative analysis of the data from SRDA, relevant observations are being shared in this research paper. Specifically, the study set out to discover predictors which can offer enhanced levels of assessment for software quality and project execution. At this stage of the research, the project has

focused on identifying early candidates for software attributes that can be good predictors of success with respect to quality measures based on the data collected. Defect attributes can also be in scope for future research and can be based on the data set used for this research study. There is enough evidence based on past research literature review and analysis of viable OSS data archives to demonstrate that OSS can provide valuable historical information about software projects to benefit software project management and improve project quality.

Key relationships amongst the relevant data entities in the SRDA have been identified and established. Figure 20 shows the steps required to build these relationships are important to calculate important software project attributes and to determine interdependencies such as the average time between software release dates and the number defects logged, group ranking, and the number of downloads:



**Figure 20. Key Steps to Derive Key Data Relationships from the SRDA**

Confirmed by the literature review, it is reasonable to define software success by the two key attributes as researchers have concluded in the past:

1. Software ranking is given by the community and for the community it serves.
2. Software usage which best identified in OSS communities as the total number of downloads.

In addition, we define a third calculated attribute:

3. Average days of defects logged relative to the release date. This derived attribute is being introduced specifically by this research study to address a gap. In order to develop a software quality predictive model usable in real-time, static quality

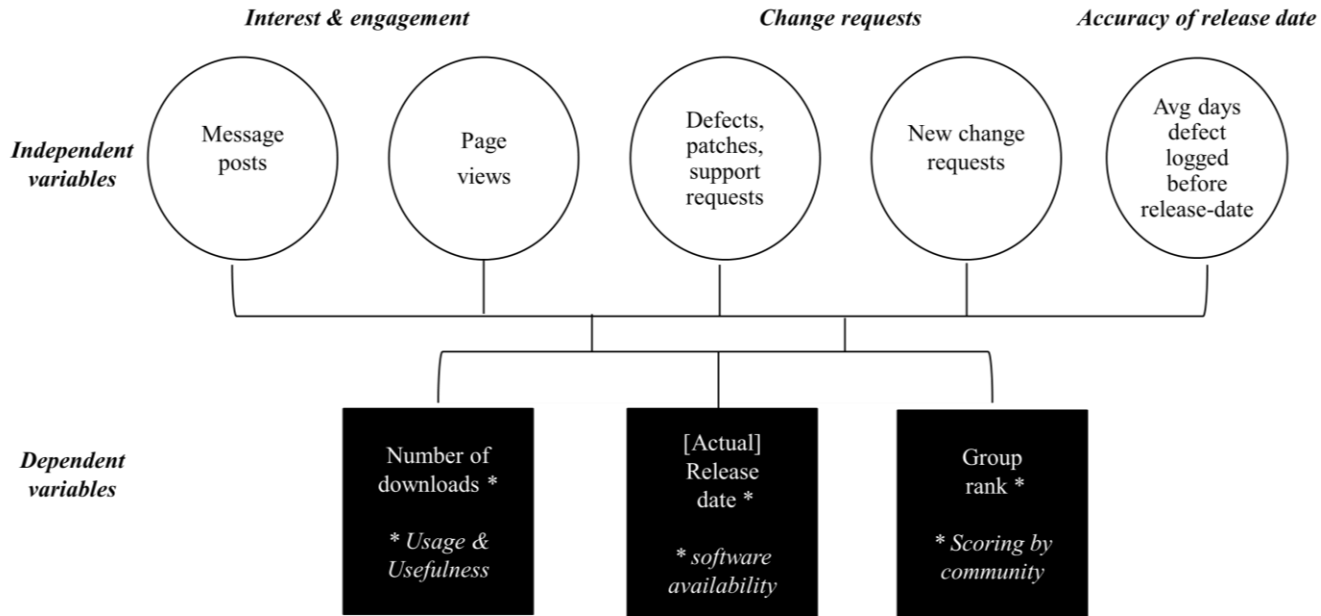
attributes such as the first two attributes above are not enough as they only account for a posteriori result.

By introducing a third aspect focused on measuring release date accuracy, a learning model can be envisioned to better predict the accuracy of an upcoming software release date established by software management organizations by leveraging additional information. Software management can improve the accuracy of estimated release dates by examining the rate of software defects logged just before an upcoming release date or shortly after a premature software release.

### **5.5. Research Variables**

Based on the analysis done thus far, we determine that it is feasible to develop a data store by leveraging OSS data archives to facilitate research towards improving software process quality and to develop predictive models. Key relationships can be established amongst relevant SRDA data to calculate the mean time between software release dates and number defects logged, group rank, and number of downloads. This can be of crucial importance to software managers looking for more empirical data to support decision making and time frame estimation efforts during project execution.

Figure 21 shows the research variables that are directly or indirectly relevant to this study:



**Figure 21. Research Variables and Classifications**

Table 10 below provides a detailed explanation for the research variables and their relevance to this study:

**Table 10: Research Variables and Relevance**

Variable	Classification	Type	Research Relevance
<b>Message Posts</b>	Independent	Whole Number	Represents the total number of messages generated by individuals within the communities. This number indicates message forum penetration by measuring the level of activity evidenced on the community forum(s) related to the software package. This variable shows the level of community engagement regarding the software. A higher number is more favorable and demonstrates greater engagement.
<b>Page views</b>	Independent	Whole Number	Represents the total number of individual hits on the web page that serves the software and its related information to the community. This variable shows the level of interest in the software. Similar to message posts, a higher number of page views is more favorable and shows greater engagement.
<b>Defects, patches, and support requests</b>	Independent	Whole Number	Represents an aggregate sum of three distinct request types: defects, patches, and support requests. Such types of requests indicate issues having occurred during the development process. Although it is preferred that a higher number of issues are identified and resolved before the release of software, a higher number also indicates process quality issues.

Variable	Classification	Type	Research Relevance
<b>New requests</b>	Independent	Whole Number	Represents the total number of changes or revisions to software that have been identified during the test or execution stages and were consequently not intended or specified during the requirements gathering stage. A higher number indicates process quality issues since, ideally, new changes should not be uncovered during the test and execution stages of the project.
<b>Average days defects logged before release date</b>	Independent	Rational Number	Represents a calculated variable and indicates the arithmetic average number of defects reported (i.e., defects logged) relative to the release date. As release date accuracy is of paramount importance to the anticipating user community, release date accuracy has been highly regarded as a quality indicator of the software development process. Release date prediction occurs with a degree of uncertainty and is based on several quantitative and qualitative factors. A viable quantitative factor which can be used to predict the accuracy of release date is the average number of defects logged for the software prior to the estimated release date. Typically, software communities operate under a premise that software will be released as soon as it is estimated to be ready or usable (i.e., with the average number of defects logged being as close to zero as possible). Therefore, from a historical data perspective, we can use the measure of “the average number of days during which defects are logged before the release date” to assess the accuracy of the release date itself. The average number of defects logged can be a positive number indicating that, on average, more defects were logged before the release date of the software. A negative number indicates that, on average, more defects were logged after the software release date. The closer the number is to zero (i.e., minimal skew towards a positive or negative number), the higher the accuracy of the release date.
<b>Number of downloads</b>	Dependent	Whole Number	Represents a variable which has been traditionally considered as an overall indicator of OSS usability and usefulness. A higher number of downloads is always more favorable.
<b>Release date</b>	Dependent	Date Value	Represents the date on which the software is made generally available (GA).
<b>Group rank</b>	Dependent	Whole Number	Represents the rank ordering which the SourceForge library calculates for every software package in its repository. The library uses sophisticated algorithms to calculate the rank which is generally regarded as a measure of software success within the community [137, 138]. A lower rank is more favorable.



## CHAPTER 6: RESULTS AND IMPLICATIONS

Software development is comprised of a series of knowledge-based activities involving discovery, coding, and usage of knowledge that then translates into viable systems solutions. Based on the acceptance of this reasonable premise, a strong connection between knowledge-management processes and software development processes in organizations can be established. As such, Meso et al. examined if software engineering methodologies actually impact the knowledge management processes in organizations and the quality of systems design from a cognitive-theory perspective [139]. The authors confirmed that information systems development is a knowledge-intensive activity and therefore is influenced by the quality of the knowledge management processes employed in support of the activity. This shows that that effective knowledge management processes yield high-quality software solutions and that learning-oriented organizations can indeed benefit from new knowledge which then can lead to better outcomes from future software projects. Therefore, it is rightfully expected that the implications from this research study including any predictive models developed and shared can be of paramount importance to such organizations.

After a predictive model is developed, tested, and finalized, the resulting implications will be shared. Specifically, it is expected that several implications will be highlighted by the final research report:

- The external validity of the predictive model will be examined. Once each of the predicting variables has been identified, each of them will be tested against OSS and PSS drivers for relevance, applicability, and reliability. The predictive model

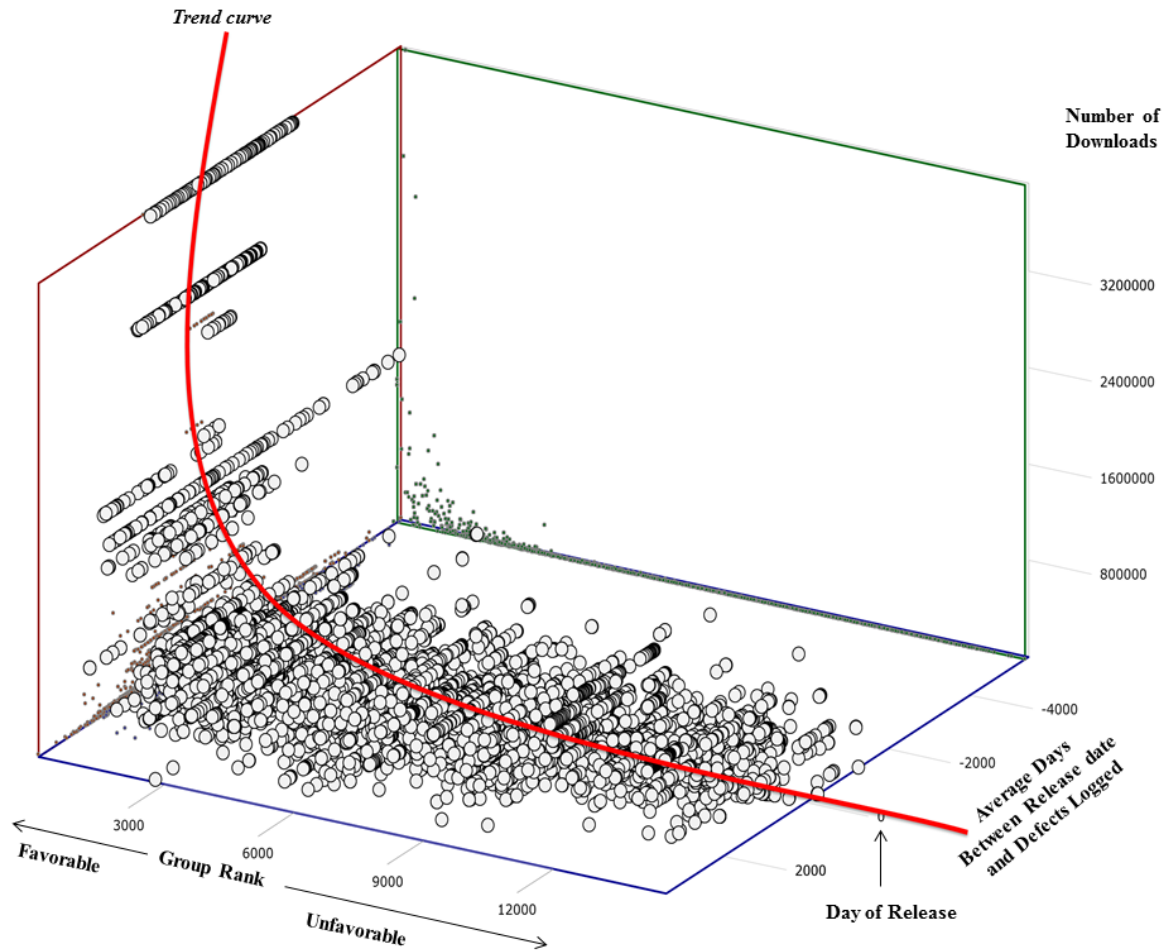
will be tested for performance against 3 distinct OSS software case studies (i.e., specific software examples) that can demonstrate reliable data within the SoureForge based SRDA.

- The context of usefulness and applicability of the model will be examined. The model should be easy to understand and use. The model must incorporate quantifiable methods of assessment.
- The expected benefits of the model must be explained clearly in the report.
- Any constraints and limitations of the model must be acknowledged clearly and completely in the report. Practitioners must be able to easily personalize the constraints and limitations shared by the report in the context of their own or other software projects.

In this section of the report, we discuss the results of the data analysis and research implications.

### **6.1. Software Quality Attributes**

The progress made towards achieving the goals of this research project is encouraging. While further analysis towards testing attribute relationships continues and the development of an improved software quality predictive model is being further developed, preliminary results of the analysis of 3 key attributes are being shared in this paper. Specifically, we examine relationships between group rank, number of downloads, and number of defects relative to release dates (i.e., release date accuracy). A random sample of 18,019 software package releases from the SRDA was selected for the analysis. A 3D plot depicted in Figure 22 provides a visual representation of the data which also supports the findings shared below:

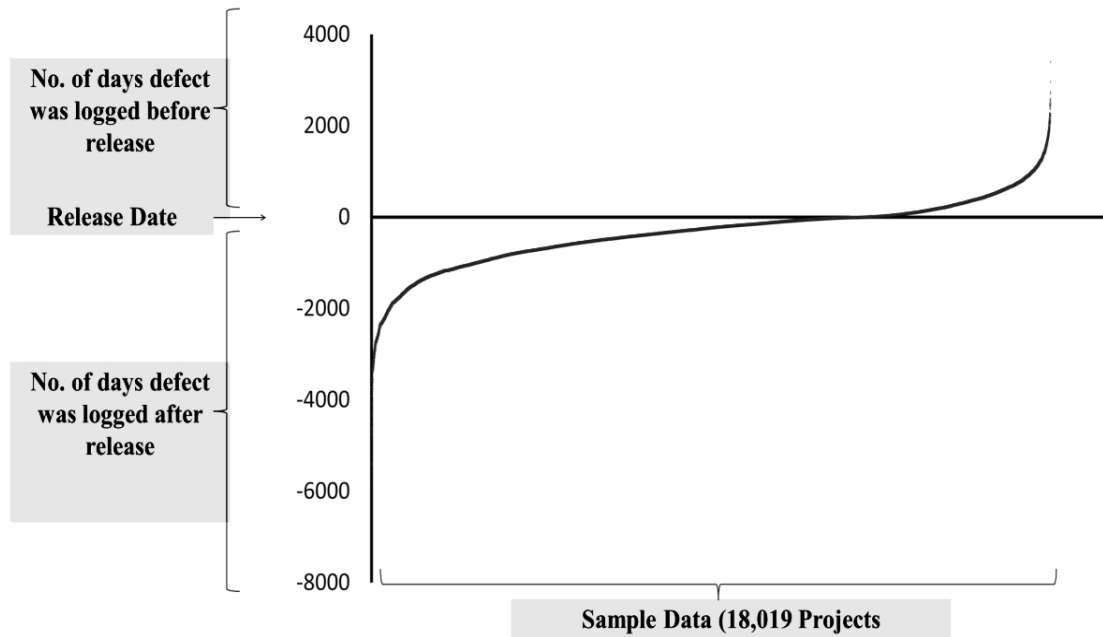


**Figure 22. Sample 3D Data Plot –Sample Size of 18,019 Software Packages**

The 3 axes shown in Figure 22 demonstrate the relationship between 3 variables: “group rank”, “number of downloads”, and “average days defects logged before release date”. The trend line visually depicts the correlation and the clustering of project data points around the axes. A few important observations and associated implications are being shared below:

*1. The accuracy of software release dates is related to how well the software is ranked by the OSS community and is, therefore, an indicator of software development process quality.* We first examine the axis in Figure 22 labeled, “average days between

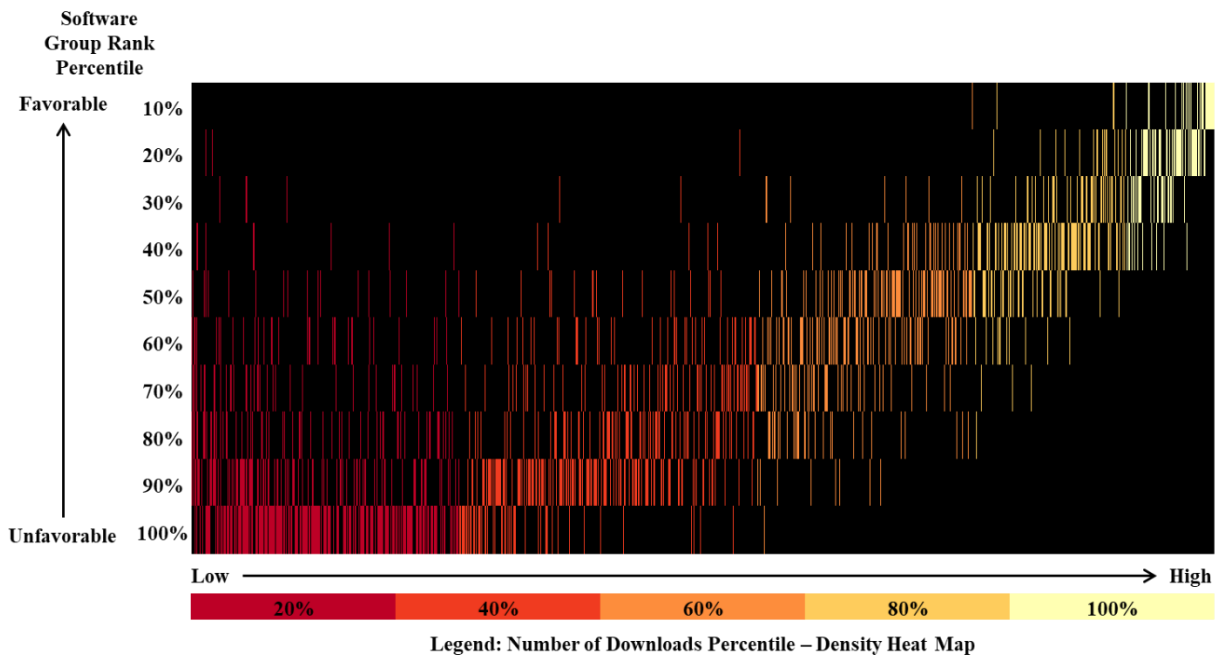
release dates and defects logged”. The relevance of this research variable has been described in Table 10 in the previous section of this paper. As discussed, we expect the value of this variable to be closer to zero for software that exhibits higher accuracy of release dates. A graphical representation of this phenomenon is represented in Figure 23 for the sample dataset. Overall, the results meet expectations:



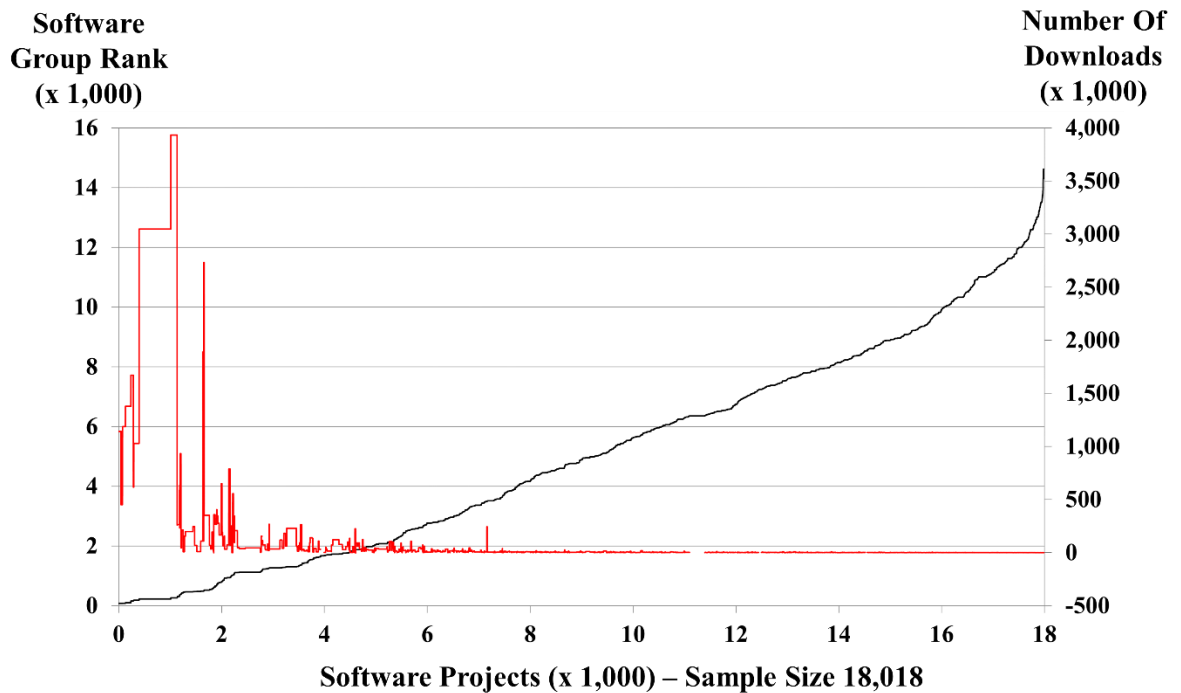
**Figure 23. Mean Time Between Software Release Dates and Defects Logged**

Furthermore, we can presume release date accuracy to be a predictor of software engineering management success. We expect projects with higher accuracy of release dates to be more favorably ranked by the communities which they serve (i.e., they exhibit lower group rank values which are more favorable than higher values). For this regard, the results shown in Figure 22 are consistent with our expectations. Communities react more favorably towards software that demonstrates greater accuracy of release dates. Release date accuracy can be considered as a viable indicator of software project execution quality based on our random data sample.

2. *Community engagement and user perception are important predictors of software interest software adoption.* Figure 22 demonstrates that software group rank and number of downloads are inversely correlated. A density plot and a scatter plot diagram represented by Figure 24 and Figure 25 visually contrasts the data points related to the two axes to further confirm the correlation:



**Figure 24. Density Map – Group Rank and Number of Downloads**



**Figure 25. Relationship between Group Rank and Number of Downloads**

Analysis of the sample data shows well-ranked software experiences high usage download rates. Since ranking is derived based on OSS community engagement and is a measure of the quality of software development, this research finding will be of special interest to software management and practitioners. This finding is highly encouraging, provides further impetus for research, and provides additional motivation for predictive modeling.

**3. Software that is more favorably ranked and is properly timed for release experiences higher usage rates.** Figure 22 shows evidence of a larger cluster of data points (i.e., correlation) with favorable group ranking, accuracy of release dates, and number of downloads. Users seem to rank software more favorably and download it more times presumably for intended use when the accuracy of software release is high. Users perceive software that has been timed for release properly as a measure of software

readiness. Software which is ready for release is also generally free of issues and unanticipated changes.

## 6.2. Descriptive Statistics – Research Variables

Variables shown in Figure 21 were determined to be high contenders for software process quality prediction and therefore have been selected for this research study. Table 11 provides a statistical summary of the selected research variables:

**Table 11. Descriptive Statistics – Research Variables**

Descriptive Statistics	Average Time of Defects (bugs) Relative to Release Date				Total Defects		
	Group Ranking	Total number of downloads	to_rel_date	Number of Page Views	Number of Messages Posted	Patches Support Requests	Total New Requests
Mean	5088.64	186403.84	614.51	699719.33	179.95	116.79	153.25
Standard Error	26.26	4907.06	4.52	18722.62	9.38	3.33	4.12
Median	4869.00	3884.00	434.19	26892.00	5.00	7.00	11.00
Mode	225.00	0.00	1162.08	6084854.00	3.00	0.00	0.00
Standard Deviation	3524.91	658698.16	606.98	2513227.88	1259.30	446.87	552.43
Sample Variance	1.2E+07	4.3E+11	3.7E+05	6.3E+12	1.6E+06	2.0E+05	3.1E+05
Kurtosis	-0.95	16.67	4.61	88.58	442.49	48.12	42.24
Skewness	0.34	4.19	1.77	8.18	19.22	6.55	6.13
Range	14619	3932779	5432	33604019	31905	4346	5313
Minimum	0	0	0	0	0	0	0
Maximum	14619	3932779	5432	33604019	31905	4346	5313
Count	18019	18019	18019	18019	18019	18019	18019

OSS metadata, whether user input or calculated, is global in nature and therefore high variance has been expected and observed. The large data ranges for each variable are in line with expectations as software is generally either highly regarded or otherwise discarded in its own category. For these reasons, each variable has been further normalized by attributing the associated data into quartiles. We use the resulting data quartiles shown in Table 12 as a basis for the predictive modeling aspects of this research study:

**Table 12. Research Variables – Data Quartiles**

Data Quartile		Rank		Downloads		AvgOfTimingOf BugRelDateTime	
1	Very Low	-	1,768	-	673	-	164
2	Low	1,769	4,869	674	3,884	165	434
3	High	4,870	7,852	3,885	31,040	435	893
4	Very High	7,853	14,619	31,041	3,932,779	894	5,432
		Page Views		Msgs Posted			
1	Very Low	-	3,390	-	3		
2	Low	3,391	26,892	4	5		
3	High	26,893	192,791	6	35		
4	Very High	192,792	33,604,019	36	31,905		
		TotalDefectsPatches SupportRequests		Total New Requests			
1	Very Low	-	1	-	2		
2	Low	2	7	3	11		
3	High	8	39	12	55		
4	Very High	40	4,346	56	5,313		

The group ranking variable represents a viable indicator and reflects the perception of the software communities. Since group ranking demonstrates the most favorable Kurtosis and Skewness than the other two candidates identified as dependent variables for this study (i.e., downloads and avg\_of\_timing\_of\_bugs\_relative\_to\_rel\_date). This variable is selected for predictive modeling to serve as an illustrative example.

### 6.3. Multiple Regression Model – Research Variables

As there are several predictor variables, the group ranking variable is used again as the dependent variable to conduct a multiple regression analysis. The results are shared in Table 13:



**Table 13. Research Variables – Multiple Regression Results (95% C.I.)**

<i>Regression Statistics</i>	
Multiple R	0.83
R Square	0.70
Adjusted R Square	0.70
Standard Error	0.62
Observations	18019

<b>ANOVA</b>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	6	15591.62	2598.60	6868.19	0
Residual	18012	6814.90	0.38		
Total	18018	22406.53			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	4.85	0.02	3.11E+02	0.00E+00	4.82E+00	4.88E+00	4.82E+00	4.88E+00
Download-quartile	-0.64	0.01	-9.17E+01	0.00E+00	-6.55E-01	-6.27E-01	-6.55E-01	-6.27E-01
AvgTime-Quartile	-0.05	0.00	-1.29E+01	3.92E-38	-6.20E-02	-4.57E-02	-6.20E-02	-4.57E-02
page_views-Quartile	-0.03	0.01	-4.80E+00	1.62E-06	-4.59E-02	-1.93E-02	-4.59E-02	-1.93E-02
msg_posted-Quartile	-0.08	0.00	-1.76E+01	7.24E-69	-8.87E-02	-7.10E-02	-8.87E-02	-7.10E-02
Total_D_P_S-Quartile	0.04	0.01	3.10E+00	1.95E-03	1.35E-02	6.01E-02	1.35E-02	6.01E-02
Total_Nreq-Quartile	-0.18	0.01	-1.57E+01	3.23E-55	-2.04E-01	-1.59E-01	-2.04E-01	-1.59E-01

A multiple correlation coefficient (R) of 0.83 demonstrates that the six predictor variables combined are highly correlated to the group ranking assigned by the software communities. The coefficient of determination (R<sup>2</sup>) demonstrates that the six predictor variables, when combined, can explain 70% of the variance in group ranking. The F-test and Significance F (p-value) of the overall model shows statistically significant results using a 95% confidence level for ANOVA. From the results of the multiple regression analysis shown in Table 13, the regression equation to demonstrate prediction can be represented as follows:

$$gr = 4.85 - 0.64(dl) - 0.05(avgtime) - 0.03(p) - 0.08(m) + 0.04(t\_dps) - 0.18(t\_nr)$$

The regression equation highlights that group ranking is indeed negatively correlated to the six predictor variables by varying degrees. The total number of software downloads has the maximum impact to how the group ranks the software followed by total new requests for changes prior to release, the level of engagement as evidenced by the number

of message posts and page views for the software. The average time of defects logged relative to release date and total number of defects logged have some impact on how software is ranked albeit less than the other predictor variables considered by this study.

Results of multiple regression analysis and ANOVA analysis at 95% confidence interval shows the relative importance of the various metrics in predictive modeling for group rank and downloads as follows:

**Table 14. Relative Importance of Key Attributes in Predictive Modeling**

Attributes and Objective Metrics from OSS*	Relative Importance in Predictive Modeling*	
	Group Rank (R=0.74)	Downloads (R=.0.80)
Page views	9.6x	6.1x
Message posts	5.9x	1.8x
Defects, support , patches	1.0x	1.4x
<u>New requirements</u>	<u>5.9x</u>	<u>1.0x</u>

### 6.3. Towards Building and Validating a Predictive Model

This study explores the use of Artificial Intelligence (AI) technology involving select machine-learning algorithms to develop a simple, easy to understand, and easy to use predictive model. The *Waikato Environment for Knowledge Analysis (Weka)* software (version 3.8.1) was leveraged for this study. Weka is a suite of machine learning software written in Java and was developed in 1993 with university collaboration. It is a fully supported, research-based, graphical, and widely adopted open source platform. Developers have actively continued to enhance the software with new algorithms and improved user interfaces since its inception [140].

Nine established machine learning algorithms were selected and implemented using the Weka software with the sample data from the SRDA data source. The selection was

guided by the literature review and the analysis summarized vis-a-vis Table 9. Detailed results of the model build and training data validation are provided in Appendix C.3.

Weka 3.8.1 Machine Learning Algorithm – Results. Summary of performance has been provided in Table 15:

**Table 15. Classification Algorithm Performance Summary**

Weka ML Algorithm Results Sample 18,019 Projects 66% Model / Test	J48								
	Random Forest	Meta Bagging	Decision Tree	Decision Table	K-Nearest Neighbor	Multi-Layer Perceptron	Iterative Classifier	Naïve- Bayes	AdaBoost M1
Correctly Classified Instances	78.65%	78.57%	78.42%	77.51%	76.97%	73.08%	67.42%	63.50%	45.82%
Incorrectly Classified Instances	21.35%	21.43%	21.58%	22.49%	23.03%	26.92%	32.58%	36.50%	54.18%
Kappa statistic	0.7154	0.7143	0.7123	0.7001	0.693	0.6412	0.5656	0.5135	0.2757
Receiver operating characteristic	0.95	0.946	0.936	0.94	0.938	0.617	0.881	0.859	0.692
Precision	0.79	0.791	0.788	0.781	0.778	0.063	0.672	0.626	0.292
Mean absolute error	0.1376	0.1484	0.1493	0.1858	0.1909	0.1802	0.2243	0.202	0.2974
Root mean squared error	0.2653	0.2696	0.2764	0.2833	0.2897	0.3074	0.3297	0.3567	0.3856
Relative absolute error	36.69%	39.56%	39.81%	49.54%	50.92%	48.04%	59.82%	53.86%	79.30%
Root relative squared error	61.27%	62.26%	63.84%	65.43%	66.89%	70.99%	76.15%	82.38%	89.05%
Total Number of Instances	6126	6126	6126	6126	6126	6126	6126	6126	6126
Model build time (secs)	2.13	1.52	0.37	0.94	0.01	793.71	6.18	0.07	0.14
Model test time (secs)	1.37	0.38	0.06	0.19	128.18	0.07	0.08	0.38	0.02

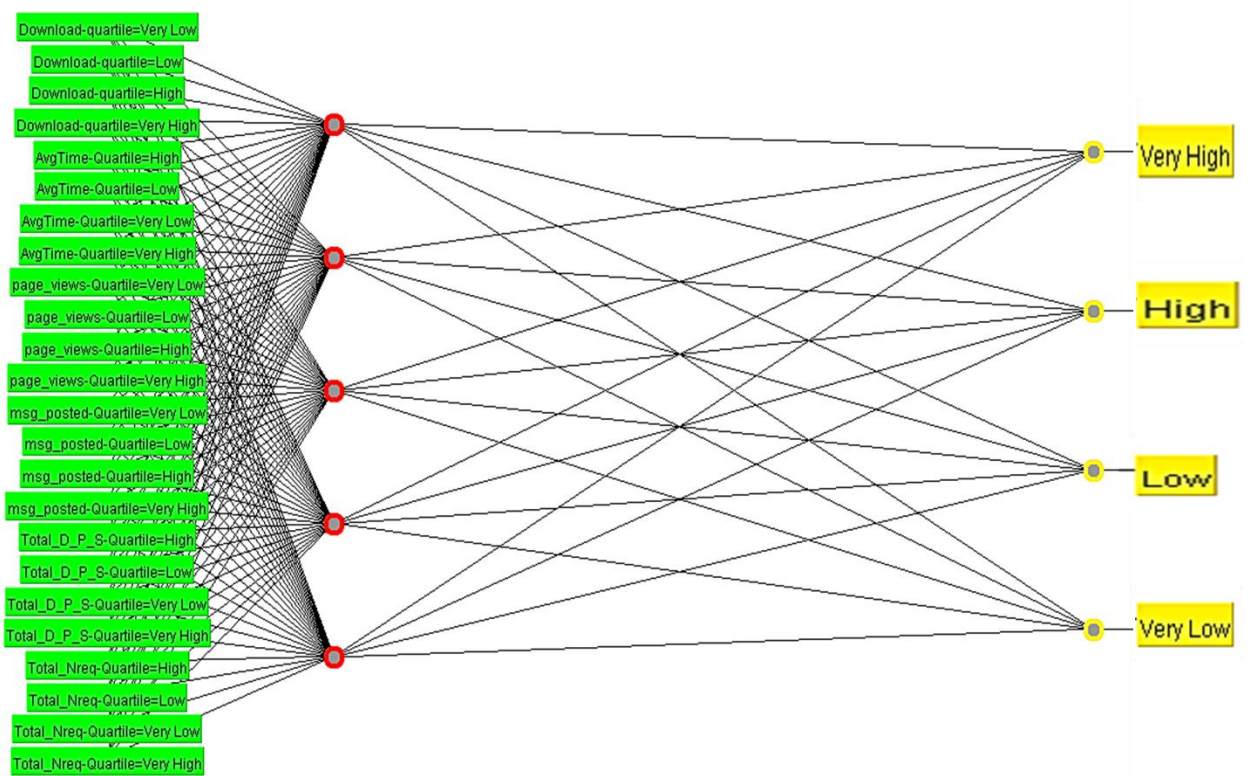
The following noteworthy observations are being shared after examination of the model results:

1. The sample of 18,019 projects was randomly split with 66% or 11,893 projects used for model development and the remainder of the data or 6,126 projects were used for model validation.

2. Actual model fit and reliability was observed to be moderate to good based on the selection criteria used in this study. Random forest, meta-bagging, and J48 decision tree were among the best-performing algorithms. Classification accuracy (i.e., correctly classified instances), a chance-corrected measure of prediction versus actual class (i.e., Kappa statistic), receiver operating characteristic (i.e., ROC), precision, and root mean squared error (i.e., RMSE) were the key factors determining performance. While

important for real-time and industrial application, the time to build and the validation of the model against training data were treated as secondary factors.

3. Simpler algorithms such as Naïve-Bayes as well as the more complex deep learning, neural network based multi-layer perceptron (MLP) experienced suboptimal performance on a relative basis. Figure 26 shows the neuron complexity depicted by the run-time, graphical output from Weka based, on the MLP model build and test parameters executed:

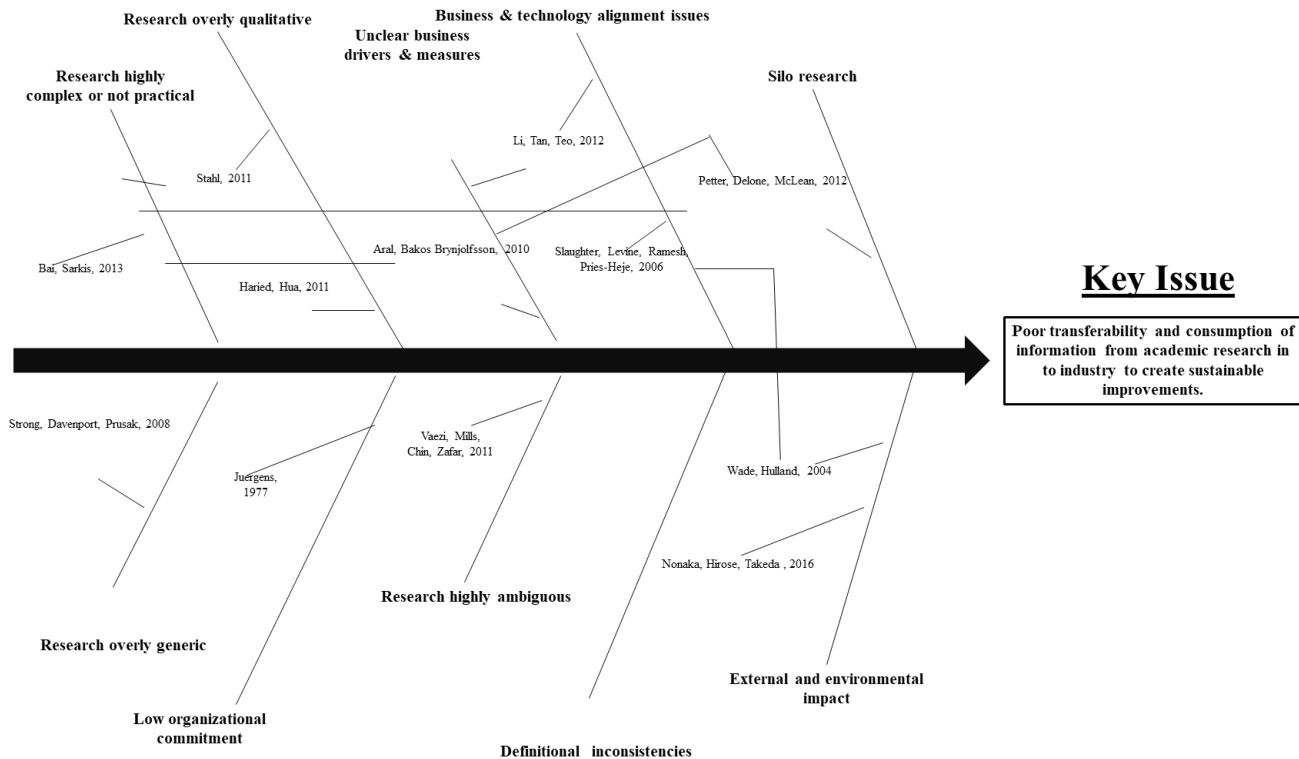


**Figure 26. MLP – Neural Network Diagram to Predict Software Rank**

#### **6.4. Research Transferability**

Current state industry practices highlighted by project failure rates and supporting academic literature demonstrate poor transferability of research outcomes. Furthermore, minimal automation has been leveraged by sponsors of project assessment related

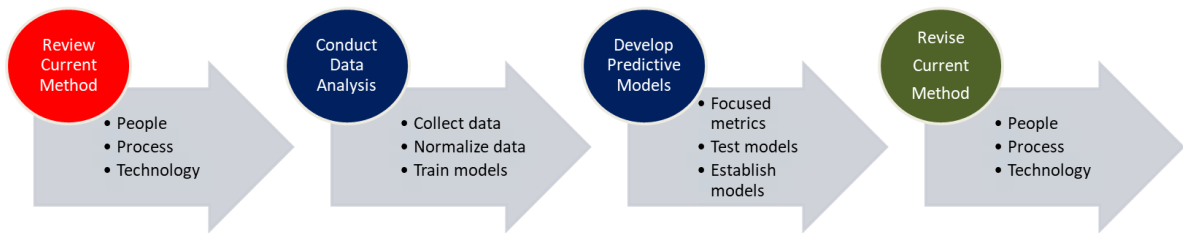
processes within the industry. This is especially true of quality, productivity, and estimation related processes [19, 24, 25, 141]. To better understand the current state, a further review of past research work was conducted. Twelve representative studies from the past were further analyzed. The studies spanned across related disciplines including information systems, knowledge management, and software management. Collectively, these papers utilized varying research methods and considered a total of 1,877 research studies, cases studies, IT organizations, and open source projects [142-153]. A root cause analysis has been conducted and the results have been mapped and shown in Figure 27:



**Figure 27. Root Cause Analysis - Select Review of Research Papers**

Figure 27 shows that while relevant research is conducted in academia, transferability, and consumption of research is limited. Furthermore, tools and methods lack sufficient training and automation, which, if addressed, can enable practitioners to utilize research outcomes.

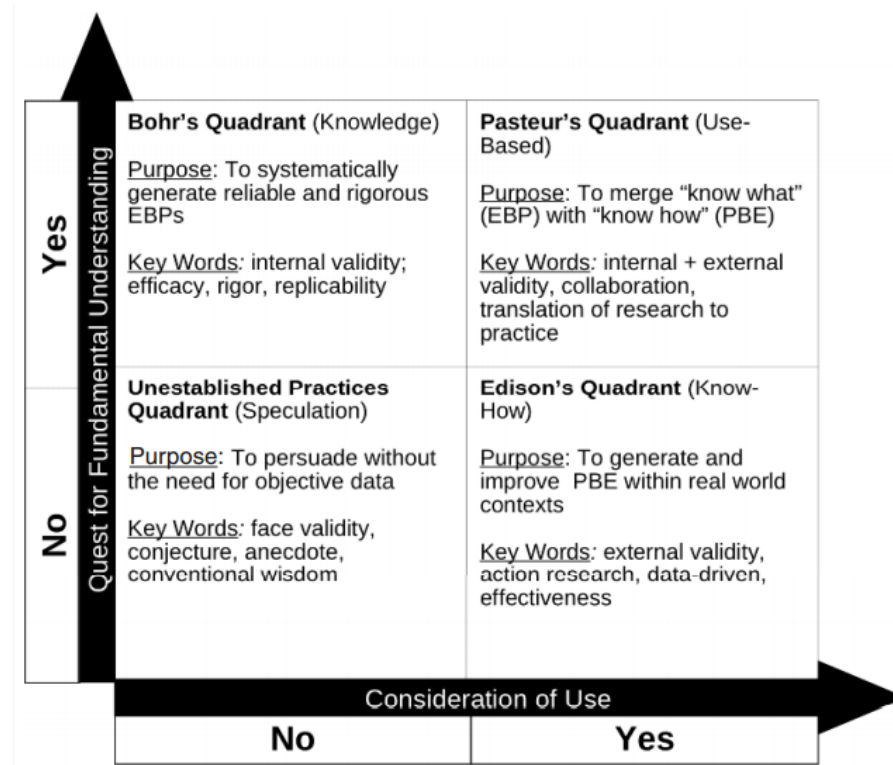
Industry practitioners can only benefit if research outcomes can be translated easily to their own respective environments. As acknowledged earlier in this work, research transferability has been challenged in the software engineering management field. Improved methods and tools must be explicitly developed which utilize proposed models to improve software project management practices. Figure 28 shows a framework that systematically incorporates research outcomes such as those discussed in this paper to real-time, scalable, and practical application:



**Figure 28. Improving Software Management through Research Transferability**

To improve software project execution results, academic research without clear methods to adapt and apply the outcomes in a straightforward and sustainable manner is of minimal value to practitioners. As affirmed earlier, for any study related to technology sustainability, the practical application of methods is required. Due diligence must be done with respect to the managerial and behavioral aspect which helps to confirm research validity and necessitates action for subsequent follow-up [142]. Consequently, every research must translate into application in a straight forward and practical manner. To understand this better, we briefly start by reviewing the classic work of Louis Pasteur which helps bridge the gap between basic research and applied research. Pasteur's quadrant shown in Figure 29 is a classification of scientific research projects that seek

fundamental understanding of scientific problems, while also having an immediate use for society [154]:



EBP = evidence-based practice, PBE = practice-based evidence.

**Figure 29. Pasteur's Quadrant Model of Scientific Research**

We presume that software practitioners remain interested in applying research to achieve improved execution results. Often is the case when academic research deals with the problems without an adequate conduit to the industry it aims to serve. Pasteur's Quadrant analysis suggests that research for knowledge's sake is contrasted with invention – purely towards creating something new whereas engineering research must combine utility and knowledge simultaneously.

Sustainable application of research is only possible if the limitations of past research efforts are adequately addressed and a clear mechanism for implementation is provided to the practitioners. Business leaders do not have time to conduct quality research; they rely

on academic and research professionals who can neutrally develop insights based on using scientific methods coupled with a synthesis of real-world data. The question to ponder is who takes the responsibility to develop a clear mechanism for implementation to build real-world tools and techniques that are adaptable, flexible, and fully exploit the research outcomes? The answer to this question depends on another question which is largely philosophical – which stakeholder needs it the most, the researcher or the practitioner?

### **6.5. Developing an Applied Research Framework (ARF)**

There are many ways to achieve the goal of adapting traditional research methods to ensure greater industry application as it relates to improving software project execution quality. In this section of the paper, we share an illustrative example which utilizes OSS, the SRDA research data archive, and predictive modeling opportunities suggested earlier in the paper.

A closer examination of the proposed rationale behind the reasons that inhibit sustainable application of research reviewed earlier and shown in Figure 27 reveals the following:

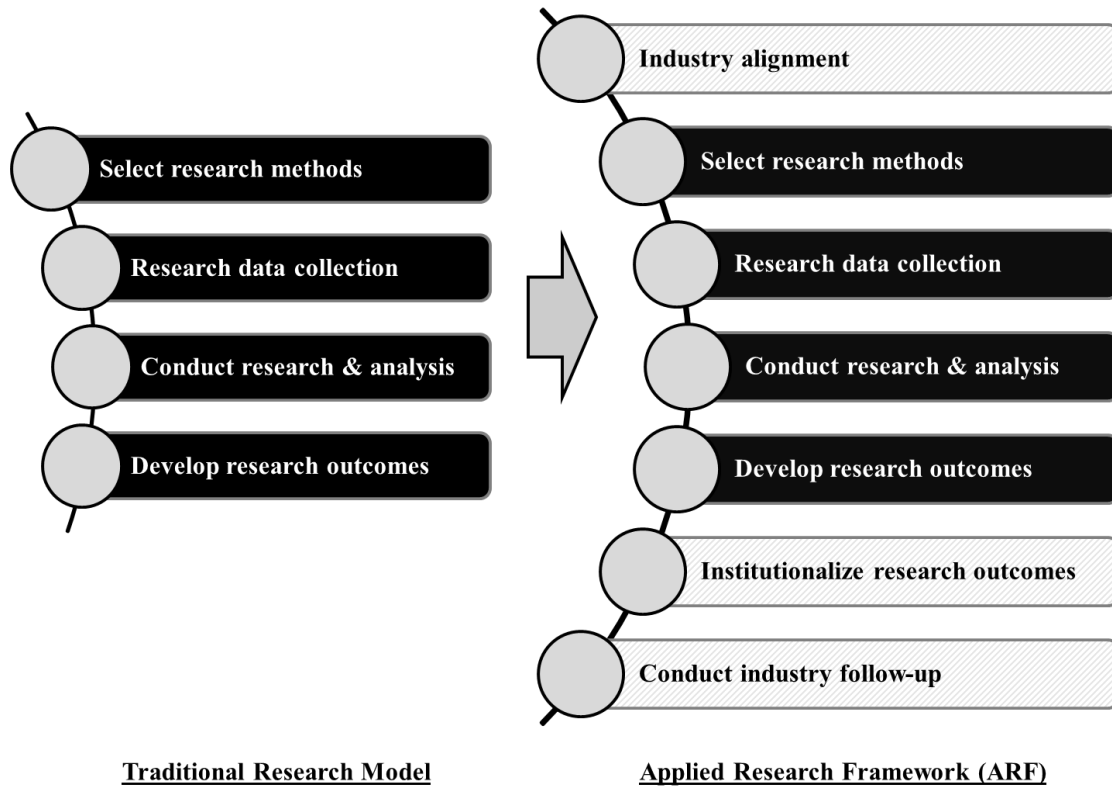
1. Organizational learning is required in order to transfer research knowledge into action.
2. Key metrics must be defined and tracked with the sponsorship and engagement from leadership.
3. Specific tools and techniques to facilitate the use of research outcomes are required to accompany outcomes for firms to realize their benefits.
4. Consistency and scalability must be maintained across the entire process



5. A business process must be adapted, on-site, to leverage the tools and techniques as designed to maximize their effectiveness.

Most research conducted is based on a specific need in industry or academia or a gap in existing research. The research process traditionally includes the selection of a research method, data collection, and related tools, qualitative or quantitative analysis, and the formulation of specific research outcomes. For successful application, the issues highlighted in Figure 27 must be addressed. Notwithstanding one exception, our literature review did not find adequate support towards methods that can aid the transfer of research outcomes to actionable steps which organizations can embrace out-of-the-box for information systems development. The noteworthy exception was a 1987 research study which loosely described how measurement of information systems can be achieved using the financial services sector as a case study. The report concluded that information systems research should follow five sequential steps: *performance assessment and consistent measurement approach, performance and importance ratings using factors, correlation of performance to ratings, action plans with prioritization in line with findings, and finally, adjusting the process based on on-going reviews*. While the research study marked a clear step in the right direction towards improving research transferability, the study does not address specific tools, framework, or examples on how to achieve results and the authors note future opportunities to extend the research further [155].

Based on our findings, we have proposed a revised research model in Figure 30 referred to as the Applied Research Framework (ARF) which focuses on ensuring sustainable application of research in industry:



**Figure 30: Applied Research Framework (ARF) for Research Transferability**

The framework has been conceptualized for adoption within the software management industry although it can adapt for external use. The framework introduces three crucial additions to the traditional research model. The suggested revisions are required for long term sustainable adoption of research by practitioners:

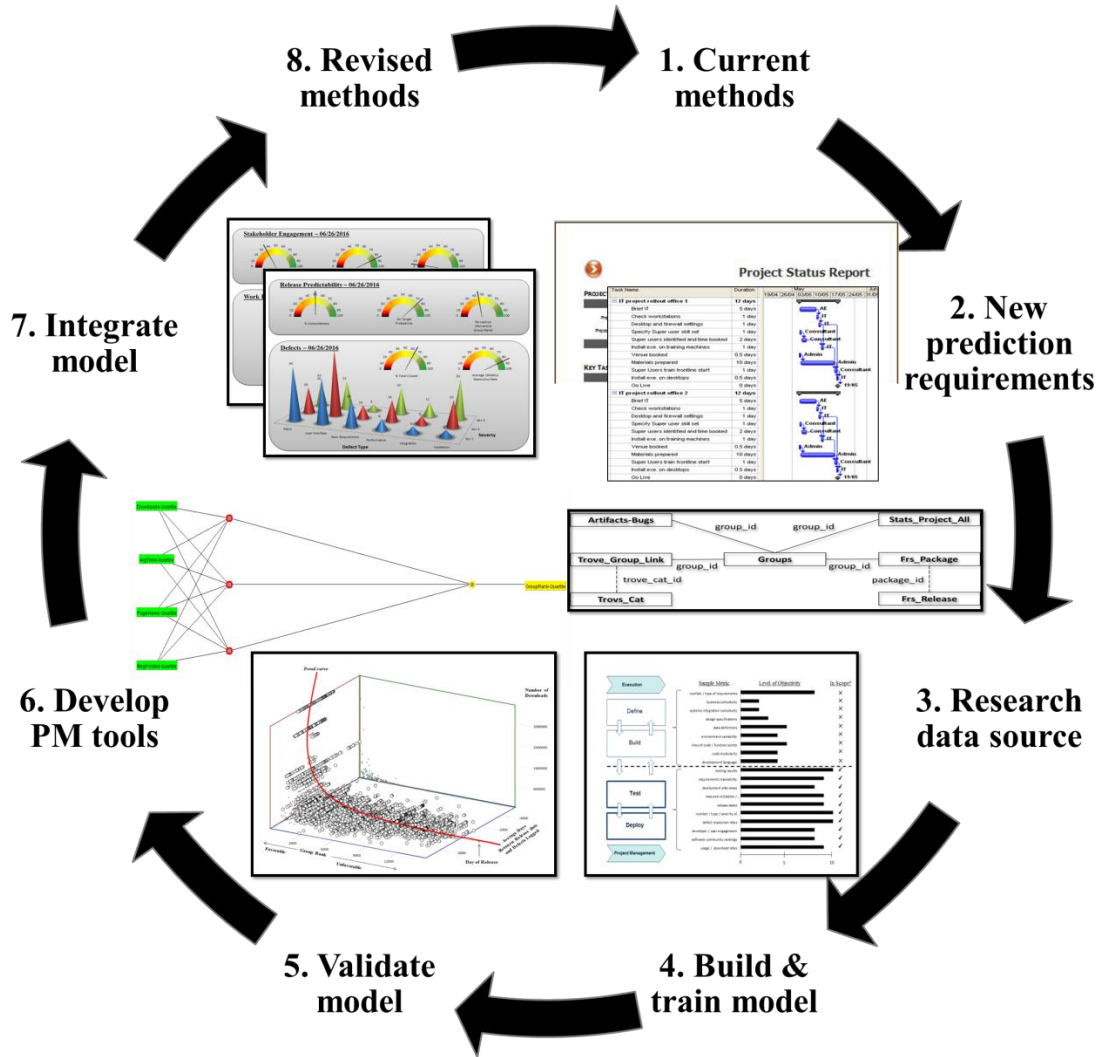
**1. Creating Industry Alignment.** It is imperative that all research aimed towards adding value and remaining transferrable to industry commence with a full understanding of key business needs, performance outcomes, and business drivers. This is particularly important to the software industry where success and outcomes are not always defined consistently as discussed earlier in this paper. Metrics are rarely used in component-based software engineering efforts [156]. This lack of metrics orientation inherent to the

industry can make matters more challenging when introducing the concept of measurable research outcomes. Nevertheless, researchers must align their methods, data, and results to the specific needs of the industry which they aim to serve or at a minimum for a subset of the industry (e.g. one or more organizations in the industry). Explicit alignment is required between the organization conducting the research and organization(s) owning the industrial use case towards which the research is aimed. The best way to achieve this is through the development of key performance measures which are aligned between the two parties. Very often this is recognized but not practiced. Past research has found that many times when firms engage with researching universities more formally, they do so with varying agendas, to gain early access to innovative techniques without a realistic way to quantify the benefits. Hence the motivation for creating a business case with specific metrics is deemed unrealistic and often neglected [157]. These challenges must be overcome at the onset. Researchers must be involved directly at some capacity with external stakeholders to help develop the value proposition for the research efforts. At the onset of any study, metrics are necessary as they can serve as guideposts for all research efforts. In an earlier section of this paper, Figure 2 serves as an illustrative example of how metrics-based research can be aligned with business requirements to support research efforts. From a practical standpoint, the scope and influence of research should be agreed to by business and research stakeholders a priori. This has not been common practice in the area of software engineering management.

**2. Institutionalizing Research Outcomes.** Creating mechanisms for the specific application of research is perhaps the most meaningful of the revision areas proposed by the framework. Research outcomes must come with a set of tools, techniques, or specific

instructions for practitioners to be able to put into use and for organizations to embrace the net contributions of the research. Only then can they be successfully institutionalized. Often this will include a set of tools, techniques, and sometimes business processes (or recommended practices). Also, since learning and knowledge management can be difficult for organizations to embrace consistently given changing business dynamics [149], any automation or learning mechanisms for implementing the research outcomes that can be suggested by researches can only improve the probability of successful transferability of research into the desired industry segment(s).

In an earlier section of this paper, we discussed past research that leveraged the SRDA as a key data source. Specifically, we discussed the value and the limitations related to the application and institutionalization of such research by software management as an example. We now extend the SRDA example by applying the ARF to develop a revised research method that includes several key components that can substantially increase the chances of business adoption. The illustration in Figure 31 shows the modified method which was developed:



**Figure 31: ARF – An Illustrative Example of Using the SRDA**

**3. Conducting Industry Follow-Up.** Once organizations can institutionalize research outcomes, researchers should be encouraged to conduct regular follow-ups to get feedback and provide support by suggesting revised methods to address change management related challenges. Furthermore, industry follow-ups can provide new data that can enable future research opportunities.

In continuation of the software metrics illustrative example shown in Figure 2 and after steps 1 through 3 shown in Figure 31 are completed in the process, each in-scope

metric can be measured based on data collected by the firm. Results can be analyzed and new data can enable additional research until the process is refined and material benefits are realized. Only then can a particular phase of research be deemed complete.

Admittedly, there is a strong caveat to our revised model; researchers must have the required resources including funding in order to conduct the new steps that have been introduced by the model. Without the required resources, many research efforts fall short and are never completed. Sustainable research is iterative, long-term, and incremental by its very nature. As long as research interest remains and access to required resources is possible, all research aimed at improving industry methods must be approached holistically as depicted by the framework.

## CHAPTER 7: FUTURE RESEARCH OPPORTUNITIES

The further proliferation of Big Data in virtually every industry segment is expected to be of natural consequence as data becomes further commoditized. Software engineering management methods have lagged in leveraging data available in the open source environment. This research study explored several tangible methods by which open source data can be leveraged to build, validate, and implement predictive models that are practitioner-friendly which management can leverage for decision-making. Research presented in this paper can be extended in many ways:

**1. Identification of additional software quality predictors.** Five select and independent variables related to software quality prediction were examined by this research study in addition to three dependent variables. Some of these variables can be further refined for future study. For instance, an aggregate number of message posts can be further refined to examine the uniqueness of message subject threads and uniqueness of authorship. Page views can be further delineated by viewer profile and activity. Defects, patches, service requests, and change requests can be categorized by type and severity. An average number of days when defects are logged before the release date can be further segregated by software package, type, and segment. Average number of software downloads can be analyzed by user and demographic related attributes. Software release date and software rank predictability for initial releases versus subsequent releases can be more closely examined. Finally, additional (new) variables can be identified, tested, and included to enhance prediction models.

**2. Development of new OSS metadata resources.** The SRDA has served as an example of a data archive that was built with metadata from the SourceForge OSS repository. The successful data archive project was supported by grant funding and sponsored a time-bound research effort. There remains ample opportunity to create other resources similar to the SRDA that can offer greater extensibility. Additional repositories could be designed, decoded, and mapped to create usable metadata that is timeless.

**3. Development of industry-specific, application specific predictive models for greater accuracy and relevancy.** Predictive models are inherently prone to being generic which can compromise their effectiveness in real-world application and transferability. Predictive models are flexible and can be rendered more specific by the inclusion of additional and specific data filters such as industry segment, types of software package, complexity, team size, dispersion of users, and other criteria. Resultant models can offer greater relevancy to decision-makers.

**4. Selection of new machine learning algorithms.** Regression and classification models offer ample opportunities for additional study in this area. New machine learning algorithms have been an important area of research growth. New and existing algorithms can be further explored and optimized to further improve on the research results shared in this paper.

**5. Automation of project management tools.** Industry practitioners require leading-edge tools and techniques that offer ease of use, flexibility, speed, and accuracy. As confirmed by the literature review, automation has been lacking in the project management space. Seamless integration of predictive models with decision-enabling



dashboards, reporting, activity planning, risk planning, and other project management tools will be of keen interest to software project managers.

Usefulness of academic research has been increasingly important to industry practitioners. For this reason, each of the suggested areas of further research promise to offer greater research transferability to software engineering and management processes.

## **CHAPTER 8: CONCLUDING REMARKS**

All technology projects assume a certain level of execution risk during their lifecycle. The overarching goal for technology management is to maximize risk-adjusted returns from their technology investments. The premise of this research study recognizes the same to be the case for software technology project management. Estimation of software quality is of primary concern and the impetus for this research study. Over the last fifty years, software project performance levels have consistently demonstrated lackluster performance and therefore the need for improvement is warranted. Organizational turnover rate, practitioner skillset, selective memory, alternative motives, and short-term business pressures are representative of factors that contribute to poor project performance.

Research and experience show that project assessment and evaluation techniques used by PM's remain largely subjective. This research presents novel methods that can be infused in traditional software project management practices. This research is the direct outcome of the compelling change required and one that has highlighted by the author and supported by the abysmal status of software project execution performance within the industry. This research also addresses the gaps evident in existing research efforts that have been often discussed explicitly by researchers.

The goal of this research study is to suggest easy implement predictive methods based-on real-time data which is gathered both a priori and a posteriori of project execution. The research insights shed light on how PM's can increase the use of quantitative yet practical methods to assess project execution status and related estimates.

A few such methods based on the validation of select machine learning algorithms were developed and extended vis-à-vis this research study. A framework for research transferability (ARF) was introduced and explored for general use by practitioners.

As a final point and to address the inherent challenges associated with obtaining quality and performance related data from corporate and commercially-based software, the vast realm of OSS has been pragmatically leveraged. The insights drawn from the data is done so with OSS and PSS applicability wherever reasonably possible.

## CHAPTER 9: GLOSSARY OF TERMS

- **Artificial Intelligence (AI)** – This term refers to intelligence seemingly exhibited by computing machines. An "intelligent" machine is a flexible rational agent that perceives its environment and takes actions that maximize its chance of success at some goal. The term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem-solving".
- **Business as Usual (BAU)** – This term often refers to a set of activities that are of normal course of business outside the context of the project. Typically, after a project has been completed, business operations are revised to leverage the outcomes of the project. Planning for BAU activity is generally required during the project to ensure a success full transition after the completion of the project.
- **Capability Maturity Model (CMM)** – The Capability Maturity Model (CMM) is a development model created after study of data collected from organizations that contracted with the U.S. Department of Defense, who sponsored the research. The term "maturity" relates to the degree of formality and optimization of processes, from ad hoc practices, to formally defined steps, to managed result metrics, to active optimization of the processes. The model's aim is to improve existing software development processes, but it can also be applied to other processes. Watts Humphrey began developing his process maturity concepts during the later stages of his 27-year career at IBM. Active development of the model by the US Department of Defense Software Engineering Institute (SEI) began in 1986 when Humphrey joined the Software Engineering Institute located at Carnegie Mellon

University in Pittsburgh, Pennsylvania after retiring from IBM. At the request of the U.S. Air Force, he began formalizing his Process Maturity Framework to aid the U.S. Department of Defense in evaluating the capability of software contractors as part of awarding contracts.

- **Free / Libre of Open Source Software (FLOSS)** – software distributed under an open source license that permits modification and redistribution of the source code. The "L" for "libre" is sometimes included to supplement the word "free" and emphasize that it is referring to freedom of action, not free as in "no cost". In fact, many companies sell open source software, such as Red Hat and Novell. However, the end user is permitted to acquire the source code of their products, modify it, and redistribute it. Products such as CentOS are created this way.
- **Function Point (FP)** – A function point is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points are used to compute a functional size measurement (FSM) of software. Function points were defined in 1979 in Measuring Application Development Productivity by Allan Albrecht at IBM.
- **Hidden Markov Model (HMM)** – A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be presented as the simplest dynamic Bayesian network. In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but the output, dependent on the state, is visible. Each

state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states. The adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly. Hidden Markov models are especially known for their application in temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following,[8] partial discharges and bioinformatics.

- **Machine Learning** – Machine learning, in the context of this research study, refers to a collection of methods that can be used to devise complex models and algorithms that facilitate predictions. This is often referred to as predictive analytics. These analytical methods allow researchers, data scientists, engineers, and analysts to produce reliable, repeatable decisions and results and uncover hidden insights through learning from historical relationships and trends evident in the data.
- **Management Information System (MIS)** – A management information system (MIS) focuses on the management of information systems to provide efficiency and effectiveness of strategic decision making. The concept may include systems termed transaction processing system, decision support system, expert system, or executive information system. The term is often used in the academic study of businesses and has connections with other areas, such as information systems,

information technology, informatics, e-commerce, and computer science; as a result, the term is used interchangeably with some of these areas.

- **Microsoft (MS)** – Microsoft (MS) is an American multinational technology company headquartered in Redmond, Washington and develops, manufactures, licenses, supports and sells computer software, consumer electronics, and personal computers and services. Its best-known software products are the Microsoft Windows line of operating systems, Microsoft Office suite, and Internet Explorer and Edge web browsers. Its flagship hardware products are the Xbox video game consoles and the Microsoft Surface tablet lineup. Microsoft is one of the largest software companies in the world.
- **Naïve Bayesian Classifiers** – In machine learning, Naïve Bayesian classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. These classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time rather than by expensive iterative approximation as used for many other types of classifiers.
- **Open Source Software (OSS)** – Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose.[1] Open-source software may be developed in a

collaborative public manner. According to scientists who studied it, open-source software is a prominent example of open collaboration.

- **Project Manager (PM)** – A project manager (PM) is a professional in the field of project management. Project managers have the responsibility of the planning, procurement, and execution of a project, in any domain of engineering. Project managers are the first point of contact for any issues or discrepancies arising from within the leads of various departments in an organization before the problems escalate to higher authorities. Project management is the responsibility of a project manager. This individual seldom participates directly in the activities that produce the end result, but rather strives to maintain the progress, mutual interaction and tasks of various parties in such a way that reduces the risk of overall failure, maximizes benefits and minimizes costs.
- **Proprietary Software System** – A proprietary software system (PSS) is computer software with its source code which is copyrighted, trademarked, patented, or otherwise unavailable to the general end-user. Software execution rights are required to be purchased or licensed by the owner of the individual or company that owns the rights to the software code.
- **SourceForge** – SourceForge is a web-based service that offers software developers a centralized online location to control and manage free and open-source software projects. It provides a source code repository, bug tracking, mirroring of downloads for load balancing, a wiki for documentation, developer and user mailing lists, user-support forums, user-written reviews and ratings, a news bulletin micro-blog for publishing project updates, and other features.



- **Software Process Improvement (SPI)** – Software process improvement (SPI) often refers to specific methods that can serve as an integrated collection of procedures, tools, and training for the purpose of increasing software product quality or development team productivity, or reducing development time. Software process improvement upgrades an immature organization to a mature organization. An immature organization cannot generate a good quality product. A software process improvement model is an approach or method or both by which process improves and give better result rather than a normal process. By software process improvement a better and high-quality product can be found within budget and time.
- **SourceForge Repository Data Archive (SRDA)** – The SourceForge Research Data Archive (SRDA) is a collection of OSS data and resources developed by researchers at the University of Notre Dame from 2003 to 2014 with the sole purpose of advancing software research. It is based on data from the SourceForge repository and has been utilized by over 100 research studies to date.
- **Total Quality Management (TQM)** – A core definition of total quality management (TQM) describes a management approach to long-term success through customer satisfaction. In a TQM effort, all members of an organization participate in improving processes, products, services, and the culture in which they work.

## CHAPTER 10: BIBLIOGRAPHY

- [1] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*: IEEE Computer Society Press, 2014.
- [2] M. Shepperd, M. Cartwright, and G. Kadoda, "On Building Prediction Systems for Software Engineers," *Empirical Software Engineering*, vol. 5, pp. 175-182, 2000.
- [3] L. Qifeng and L. Bing, "Mining Open Source Software data using regular expressions," in *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, 2011, pp. 550-554.
- [4] *IEEE Standard for Developing a Software Project Life Cycle Process (1074-2006)*. Piscataway, USA: IEEE, 2006.
- [5] L. J. Jagtiani and N. Lewis, "The Impact of Big Data on the Management of Business Software Technology Projects," in *Faculty Research Conference*, Bridgeport, CT, 2015.
- [6] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," ed, 2007, pp. 85-103.
- [7] B. Beizer, *Software Testing Techniques*: Dreamtech Press, 2003.
- [8] (2016). *Gartner Says Worldwide IT Spending is Forecast to Grow 0.6 Percent in 2016*. Available: <http://www.gartner.com/newsroom/id/3186517>
- [9] "The CHAOS Manifesto, 2013: Think Big, Act Small," The Standish Report International 2013.
- [10] A. Mandal and S. Pal, "Identifying the Reasons for Software Project Failure and Some of their Proposed Remedial through BRIDGE Process Models," 2015.
- [11] W. Al-Ahmad, K. Al-Fagih, K. Khanfar, K. Alsamara, S. Abuleil, and H. Abu-Salem, "A taxonomy of an IT project failure: Root Causes," *International Management Review*, vol. 5, p. 93, 2009.
- [12] M. Fabriek, M. van den Brand, S. Brinkkemper, F. Harmsen, and R. Helms, "Reasons for Success and Failure in Offshore Software Development Projects," in *ECIS*, 2008, pp. 446-457.
- [13] G. Tsatsaronis, M. Halkidi, and E. A. Giakoumakis, "Quality Classifiers for Open Source Software Repositories," *arXiv preprint arXiv:0904.4708*, 2009.
- [14] S.-Y. T. Lee, H.-W. Kim, and S. Gupta, "Measuring open source software success," *Omega*, vol. 37, pp. 426-438, 2009.
- [15] P. Dravis, "Open source software: perspectives for development," 2003.
- [16] C. Jones, "Software management," *Computer*, vol. 27, pp. 10-11, 1994.
- [17] C. Jones, *Patterns of software system failure and success*: Itp New Media, 1996.
- [18] C. Jones, "Conflict and litigation between software clients and developers," *IEEE Engineering Management Review*, vol. 26, pp. 46-54, 1998.
- [19] J. Capers, "Applied software measurement," ed: McGraw-Hill, 1996.
- [20] C. Jones, "Software project management practices: Failure versus success," *CrossTalk: The Journal of Defense Software Engineering*, vol. 17, pp. 5-9, 2004.
- [21] R. Sessions, "The IT complexity crisis: Danger and opportunity," *White paper*, November, 2009.

- [22] R. Scheier. (2001, May) Stabilizing Your Risk - Quoted from Firoz Dosani, Mercer Consulting. *Computer Word*. 5-7.
- [23] "IT Projects: Experience Certainty. Independent Market Research Report," in *Dynamic Markets Limited*, ed.
- [24] R. Pellerin, N. Perrier, X. Guillot, and P.-M. Léger, "Project Management Software Utilization and Project Performance," *Procedia Technology*, vol. 9, pp. 857-866, 2013.
- [25] L. Raymond and F. Bergeron, "Project management information systems: An empirical study of their impact on project managers and project success," *International Journal of Project Management*, vol. 26, pp. 213-220, 2008.
- [26] L. Sarigiannidis and P. D. Chatzoglou, "Quality vs risk: An investigation of their relationship in software development projects," *International Journal of Project Management*, vol. 32, pp. 1073-1082, 2014/08/01/ 2014.
- [27] M. Jørgensen, "Software quality measurement," *Advances in Engineering Software*, vol. 30, pp. 907-912, 1999/12/01/ 1999.
- [28] P. Savolainen, J. J. Ahonen, and I. Richardson, "Software development project success and failure from the supplier's perspective: A systematic literature review," *International Journal of Project Management*, vol. 30, pp. 458-469, 2012/05/01/ 2012.
- [29] G. H. Subramanian, J. J. Jiang, and G. Klein, "Software quality and IS project performance improvements from software development process maturity and IS implementation strategies," *Journal of Systems and Software*, vol. 80, pp. 616-627, 2007/04/01/ 2007.
- [30] B. Boehm, "Some future trends and implications for systems and software engineering processes," *Systems Engineering*, vol. 9, pp. 1-19, 2006.
- [31] Brooks, "No Silver Bullet Essence and Accidents of Software Engineering," *Computer*, vol. 20, pp. 10-19, 1987.
- [32] J. A. O. G. Cunha, H. P. Moura, and F. J. S. Vasconcellos, "Decision-making in Software Project Management: A Systematic Literature Review," *Procedia Computer Science*, vol. 100, pp. 947-954, 2016/01/01/ 2016.
- [33] W. H. DeLone and E. R. McLean, "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research*, vol. 3, pp. 60-95, 1992.
- [34] K. Crowston, J. Howison, and H. Annabi, "Information systems success in free and open source software development: Theory and measures," *Software Process: Improvement and Practice*, vol. 11, pp. 123-148, 2006.
- [35] S. Zahran, *Software process improvement*: Addison-wesley, 1998.
- [36] R. B. Grady, *Successful software process improvement*: Prentice-Hall, Inc., 1997.
- [37] W. S. Humphrey, *Managing the software process*: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [38] P. Abrahamsson, "Measuring the Success of Software Process Improvement: The Dimensions," 2013.
- [39] C. Catal, "Software fault prediction: A literature review and current trends," *Expert systems with applications*, vol. 38, pp. 4626-4636, 2011.
- [40] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A General Software Defect-Proneness Prediction Framework," *IEEE Transactions on Software Engineering*, vol. 37, pp. 356-370, May/Jun 2011 2011.

- [41] T. DeMarco, *Why does software cost so much?: and other puzzles of the information age*: Dorset House Publishing Co., Inc., 1995.
- [42] C. A. Dekkers and P. A. McQuaid, "The dangers of using software metrics to (mis)manage," *IT Professional*, vol. 4, pp. 24-30, 2002.
- [43] A. Tripathi, S. Dabral, and A. Sureka, "University-industry collaboration and open source software (oss) dataset in mining software repositories (msr) research," in *Software Analytics (SWAN), 2015 IEEE 1st International Workshop on*, 2015, pp. 39-40.
- [44] A. Sureka, A. Tripathi, and S. Dabral, "Survey Results on Threats To External Validity, Generalizability Concerns, Data Sharing and University-Industry Collaboration in Mining Software Repository (MSR) Research," 2015.
- [45] R. Sen, S. S. Singh, and S. Borle, "Open source software success: Measures and analysis," *Decision Support Systems*, vol. 52, pp. 364-372, 2012.
- [46] Y. Gao, M. V. Antwerp, S. Christley, and G. Madey, "A Research Collaboratory for Open Source Software Research," in *Emerging Trends in FLOSS Research and Development, 2007. FLOSS '07. First International Workshop on*, 2007, pp. 4-4.
- [47] N. Arul Kumar and S. Chandra Kumar Mangalam, "Release process on quality improvement in open source software project management," *Journal of Computer Science*, vol. 8, pp. 1008-1011, 2012.
- [48] B. Rossi, B. Russo, and G. Succi, "Download patterns and releases in open source software projects: A perfect symbiosis?," in *IFIP International Conference on Open Source Systems*, 2010, pp. 252-267.
- [49] P. Dowling and K. McGrath, "Using Free and Open Source Tools to Manage Software Quality," *Queue*, vol. 13, pp. 20-27, 2015.
- [50] R. G. Mathieu, J. L. May, and H. L. Reif, "Investigating open source software creators through the lens of an entrepreneur," *International Journal of Innovation and Learning*, vol. 21, 2017.
- [51] Z. Yetis-Larsson, R. Teigland, and O. Dovbysh, "Networked Entrepreneurs: How Entrepreneurs Leverage Open Source Software Communities," *The American Behavioral Scientist*, vol. 59, p. 475, 2015.
- [52] Z. Iskoujina and J. Roberts, "Knowledge sharing in open source software communities: motivations and management," *Journal of Knowledge Management*, vol. 19, pp. 791-813, 2015.
- [53] 2016, *The Future of Open Source* [Annual Survey Report]. Available: <https://www.blackducksoftware.com/2016-future-of-open-source>
- [54] B. Twala, "Predicting Software Faults in Large Space Systems using Machine Learning Techniques," *Defence Science Journal*, vol. 61, pp. 306-316, 2011.
- [55] H. K. Wright, M. Kim, and D. E. Perry, "Validity concerns in software engineering research," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 411-414.
- [56] E. S. Raymond, "The Cathedral & The Bazaar," ed: O'Reilly. , 1999.
- [57] A. Israeli and D. G. Feitelson, "Success of open source projects: Patterns of downloads and releases with time," in *Software-Science, Technology & Engineering, 2007. SwSTE 2007. IEEE International Conference on*, 2007, pp. 87-94.

- [58] C. L. Huntley, "Organizational learning in open-source software projects: an analysis of debugging data," *IEEE Transactions on Engineering Management*, vol. 50, pp. 485-493, 2003.
- [59] J. Howison and K. Crowston, "The perils and pitfalls of mining SourceForge," in *Proceedings of the International Workshop on Mining Software Repositories (MSR 2004)*, 2004, pp. 7-11.
- [60] J. Lerner and J. Tirole, "The scope of open source licensing," *Journal of Law, Economics, and Organization*, vol. 21, pp. 20-56, 2005.
- [61] L. A. Belady and M. M. Lehman, "A model of large program development," *IBM Systems journal*, vol. 15, pp. 225-252, 1976.
- [62] I. Neamtiu, G. Xie, and J. Chen, "Towards a better understanding of software evolution: an empirical study on open-source software," *Journal of Software: Evolution and Process*, vol. 25, pp. 193-218, 2013.
- [63] (2016, 10/16/2016). *About Page* [Web Page]. Available: <https://sourceforge.net/>
- [64] M. Van Antwerp and G. Madey, "Advances in the sourceforge research data archive," in *Workshop on Public Data about Software Development (WoPDaSD) at The 4th International Conference on Open Source Systems*, Milan, Italy, 2008.
- [65] I. Steinmacher, M. A. G. Silva, M. A. Gerosa, and D. F. Redmiles, "A systematic literature review on the barriers faced by newcomers to open source software projects," *Information and Software Technology*, vol. 59, p. 67, 2015.
- [66] S. Faraj and L. Sproull, "Coordinating Expertise in Software Development Teams," *Management Science*, vol. 46, pp. 1554-1568, 2000.
- [67] Y. G. Sahin, "A team building model for software engineering courses term projects," *Computers & Education*, vol. 56, pp. 916-922, 2011.
- [68] L. Jagtiani and N. Lewis, "Enhancing Software Engineering Curricula By Incorporating Open, Data-Driven Planning Methods," in *2016 ASEE National Conference*, New Orleans, LA, 2016.
- [69] A. f. C. M. The Joint Task Force on Computing Curricula -- IEEE Computer Society, "Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering -- A Volume of the Computing Curricula Series," 2004.
- [70] T. DeMarco and T. Lister, "Waltzing with Bears: Managing Risk Software on Software Projects Dorset House Publishing," *New York, USA*, 2003.
- [71] F. P. Brooks, "Jr. The Mythical Man-Month," *Essays on Software Engineering*. Addison-Wesley Publishing Company, 1975.
- [72] R. L. Glass, "Frequently forgotten fundamental facts about software engineering," *IEEE Software*, vol. 18, pp. 112-111, 2001.
- [73] R. L. Glass, "Error-free software remains extremely elusive," *IEEE Software*, vol. 20, pp. 104-103, 2003.
- [74] B. W. Boehm, "Software risk management: principles and practices," *IEEE Software*, vol. 8, pp. 32-41, 1991.
- [75] B. John, R. Kadadevaramath, and E. A. Immanuel, "Recent Advances in Software Quality Management: A," 2016.
- [76] Y.-R. Wang, C.-Y. Yu, and H.-H. Chan, "Predicting construction cost and schedule success using artificial neural networks ensemble and support vector

- machines classification models," *International Journal of Project Management*, vol. 30, pp. 470-478, 2012/05/01/ 2012.
- [77] W.-L. Chao, "Machine Learning Tutorial," *DISP Lab, Graduate Institute of Communication Engineering, National Taiwan University*, 2011.
  - [78] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *Journal of Systems and Software*, vol. 81, pp. 649-660, 2008.
  - [79] H. Joshi, C. Zhang, S. Ramaswamy, and C. Bayrak, "Local and global recency weighting approach to bug prediction," in *Proceedings of the Fourth International Workshop on Mining Software Repositories*, 2007, p. 33.
  - [80] K. Ganesan, T. M. Khoshgoftaar, and E. B. Allen, "Case-based software quality prediction," *International Journal of Software Engineering and Knowledge Engineering*, vol. 10, pp. 139-152, 2000.
  - [81] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "Comparing case-based reasoning classifiers for predicting high risk software components," *Journal of Systems and Software*, vol. 55, pp. 301-320, 2001.
  - [82] T. M. Khoshgoftaar and N. Seliya, "Analogy-based practical classification rules for software quality estimation," *Empirical Software Engineering*, vol. 8, pp. 325-350, 2003.
  - [83] T. M. Khoshgoftaar, N. Seliya, and N. Sundaresh, "An empirical study of predicting software faults with case-based reasoning," *Software Quality Journal*, vol. 14, pp. 85-111, 2006.
  - [84] V. U. Challagulla, F. B. Bastani, and I.-L. Yen, "A unified framework for defect data analysis using the mbr technique," in *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, 2006, pp. 39-46.
  - [85] D. Zhu and Z. Wu, "The Application of Gray-Prediction Theory in the Software Defects Management," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, 2009, pp. 1-5.
  - [86] A. A. Porter and R. W. Selby, "Evaluating techniques for generating metric-based classification trees," *Journal of Systems and Software*, vol. 12, pp. 209-218, 1990.
  - [87] T. M. Khoshgoftaar and E. B. Allen, "Logistic regression modeling of software quality," *International Journal of Reliability, Quality and Safety Engineering*, vol. 6, pp. 303-317, 1999.
  - [88] A. G. Koru and H. Liu, "Building effective defect-prediction models in practice," *IEEE Software*, vol. 22, pp. 23-29, 2005.
  - [89] T. Menzies, J. S. Di Stefano, and M. Chapman, "Learning early lifecycle IV & V quality indicators," in *Software Metrics Symposium, 2003. Proceedings. Ninth International*, 2003, pp. 88-96.
  - [90] T. M. Khoshgoftaar and N. Seliya, "Software quality classification modeling using the SPRINT decision tree algorithm," *International Journal on Artificial Intelligence Tools*, vol. 12, pp. 207-225, 2003.
  - [91] T. M. Khoshgoftaar and N. Seliya, "Tree-based software quality estimation models for fault prediction," in *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, 2002, pp. 203-214.

- [92] P. Knab, M. Pinzger, and A. Bernstein, "Predicting defect densities in source code files with decision tree learners," in *Proceedings of the 2006 international workshop on Mining software repositories*, 2006, pp. 119-125.
- [93] E. Ceylan, F. O. Kutlubay, and A. B. Bener, "Software defect identification using machine learning techniques," in *32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06)*, 2006, pp. 240-247.
- [94] W. Afzal and R. Torkar, "A comparative evaluation of using genetic programming for predicting fault count data," in *Software Engineering Advances, 2008. ICSEA'08. The Third International Conference on*, 2008, pp. 407-414.
- [95] W. Afzal, R. Torkar, and R. Feldt, "Prediction of fault count data using genetic programming," in *Multitopic Conference, 2008. INMIC 2008. IEEE International*, 2008, pp. 349-356.
- [96] A. B. de Carvalho, A. Pozo, S. Vergilio, and A. Lenz, "Predicting fault proneness of classes through a multiobjective particle swarm optimization algorithm," in *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, 2008, pp. 387-394.
- [97] C. Jin, E.-M. Dong, and L.-N. Qin, "Software fault prediction model based on adaptive dynamical and median particle swarm optimization," in *Multimedia and Information Technology (MMIT), 2010 Second International Conference on*, 2010, pp. 44-47.
- [98] A. B. De Carvalho, A. Pozo, and S. R. Vergilio, "A symbolic fault-prediction model based on multiobjective particle swarm optimization," *Journal of Systems and Software*, vol. 83, pp. 868-882, 2010.
- [99] N. E. Fenton and M. Neil, "A critique of software defect prediction models," *IEEE Transactions on Software Engineering*, vol. 25, pp. 675-689, 1999.
- [100] S. Amasaki, Y. Takagi, O. Mizuno, and T. Kikuno, "A bayesian belief network for assessing the likelihood of fault content," in *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*, 2003, pp. 215-226.
- [101] N. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, and R. Mishra, "Predicting software defects in varying development lifecycles using Bayesian nets," *Information and Software Technology*, vol. 49, pp. 32-43, 2007.
- [102] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *Software Engineering, IEEE Transactions on*, vol. 33, 2007.
- [103] G. J. Pai and J. B. Dugan, "Empirical analysis of software fault content and fault proneness using Bayesian methods," *IEEE Transactions on Software Engineering*, vol. 33, pp. 675-686, 2007.
- [104] B. Turhan and A. Bener, "A multivariate analysis of static code attributes for defect prediction," in *Seventh International Conference on Quality Software (QSIC 2007)*, 2007, pp. 231-237.
- [105] B. John, "Modeling the Defect Density of Embedded System Software Using Bayesian Belief Networks: A Case Study," *Software Quality Professional*, vol. 14, 2012.
- [106] L. Guo, B. Cukic, and H. Singh, "Predicting fault prone modules by the dempster-shafer belief networks," in *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, 2003, pp. 249-252.

- [107] Z. Xu, T. M. Khoshgoftaar, and E. B. Allen, "Prediction of software faults using fuzzy nonlinear regression modeling," in *High Assurance Systems Engineering, 2000, Fifth IEEE International Symposium on. HASE 2000*, 2000, pp. 281-290.
- [108] X. Yuan, T. M. Khoshgoftaar, E. B. Allen, and K. Ganesan, "An application of fuzzy clustering to software quality prediction," in *Application-Specific Systems and Software Engineering Technology, 2000. Proceedings. 3rd IEEE Symposium on*, 2000, pp. 85-90.
- [109] M. Reformat, "A fuzzy-based meta-model for reasoning about the number of software defects," in *International Fuzzy Systems Association World Congress*, 2003, pp. 644-651.
- [110] B. Yang, L. Yao, and H.-Z. Huang, "Early software quality prediction based on a fuzzy neural network model," in *Third International Conference on Natural Computation (ICNC 2007)*, 2007, pp. 760-764.
- [111] L. Hribar and D. Duka, "Software component quality prediction using KNN and Fuzzy logic," in *MIPRO, 2010 Proceedings of the 33rd International Convention*, 2010, pp. 402-408.
- [112] M. Reformat, W. Pedrycz, and N. J. Pizzi, "Software quality analysis with the use of computational intelligence," *Information and Software Technology*, vol. 45, pp. 405-417, 2003.
- [113] A. Mahaweerawat, P. Sophatsathit, and C. Lursinsap, "Adaptive self-organizing map clustering for software fault prediction," in *Fourth International Joint Conference on Computer Science and Software Engineering, KhonKaen, Thailand*, 2007, pp. 35-41.
- [114] T. M. Khoshgoftaar, A. S. Pandya, and D. L. Lanning, "Application of neural networks for predicting program faults," *Annals of Software Engineering*, vol. 1, pp. 141-154, 1995.
- [115] T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl, and S. J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system," *IEEE Transactions on neural networks*, vol. 8, pp. 902-909, 1997.
- [116] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster, "An investigation of machine learning based prediction systems," *Journal of Systems and Software*, vol. 53, pp. 23-29, 2000.
- [117] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object oriented software quality prediction using general regression neural networks," *ACM SIGSOFT Software Engineering Notes*, vol. 29, pp. 1-6, 2004.
- [118] F. Padberg, T. Ragg, and R. Schoknecht, "Using machine learning for estimating the defect content after an inspection," *IEEE Transactions on Software Engineering*, vol. 30, pp. 17-28, 2004.
- [119] M. M. T. Thwin and T.-S. Quah, "Application of neural networks for software quality prediction using object-oriented metrics," *Journal of Systems and Software*, vol. 76, pp. 147-156, 2005.
- [120] M. E. Bezerra, A. L. Oliveira, and S. R. Meira, "A constructive rbf neural network for estimating the probability of defects in software modules," in *2007 International Joint Conference on Neural Networks*, 2007, pp. 2869-2874.



- [121] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," *Information and Software Technology*, vol. 49, pp. 483-492, 2007.
- [122] Y. Singh, A. Kaur, and R. Malhotra, "Predicting software fault proneness model using neural network," in *International Conference on Product Focused Software Process Improvement*, 2008, pp. 204-214.
- [123] T. Wang and W.-h. Li, "Naive bayes software defect prediction model," in *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, 2010, pp. 1-4.
- [124] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*, 2004, pp. 417-428.
- [125] A. Kaur and R. Malhotra, "Application of random forest in predicting fault-prone classes," in *2008 International Conference on Advanced Computer Theory and Engineering*, 2008, pp. 37-43.
- [126] O. Mizuno, S. Ikami, S. Nakaichi, and T. Kikuno, "Spam filter based approach for finding fault-prone software modules," in *Proceedings of the Fourth International Workshop on Mining Software Repositories*, 2007, p. 4.
- [127] Q. Song, M. Shepperd, M. Cartwright, and C. Mair, "Software defect association mining and defect correction effort prediction," *IEEE Transactions on Software Engineering*, vol. 32, pp. 69-82, 2006.
- [128] S. Morasca and G. Ruhe, "A comparative study of two techniques for analyzing software measurement data," in *Proceedings of Annual Meeting, ISERN*, 1996.
- [129] W. Yang and L. Li, "A rough set model for software defect prediction," in *Intelligent Computation Technology and Automation (ICICTA), 2008 International Conference on*, 2008, pp. 747-751.
- [130] N. K. Nagwani and S. Verma, "Predictive data mining model for software bug estimation using average weighted similarity," in *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, 2010, pp. 373-378.
- [131] L. I. Kuncheva, "On the optimality of NaA[macron]ve Bayes with dependent binary features," *Pattern Recognition Letters*, vol. 27, p. 830, 2006.
- [132] A. Uyar, A. Bener, H. N. Ciray, and M. Bahceci, "ROC based evaluation and comparison of classifiers for IVF implantation prediction," in *International Conference on Electronic Healthcare*, 2009, pp. 108-111.
- [133] A. T. Misirli, A. Bener, and R. Kale, "AI-based software defect predictors: applications and benefits in a case study.(IAAI Articles)(Case study)," *AI Magazine*, vol. 32, p. 57, 2011.
- [134] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," ed, 2008, pp. 181-190.
- [135] A. Tosun, A. Bener, B. Turhan, and T. Menzies, "Practical considerations in deploying statistical methods for defect prediction: A case study within the Turkish telecommunications industry," *Information and Software Technology*, vol. 52, pp. 1242-1257, 2010.

- [136] N. Cerpa, M. Bardeen, C. Astudillo, and J. Verner, "Evaluating different families of prediction methods for estimating software project outcomes," *The Journal of Systems and Software*, vol. 112, p. 48, 2016.
- [137] P. Buxmann and T. Hess, *The software industry: economic principles, strategies, perspectives*: Springer Science & Business Media, 2012.
- [138] B. Wang and M. Shi, "Mining and analyzing the characteristic of projects collaborative relationship in open source software," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, 2012, pp. 1277-1280.
- [139] P. Meso, G. Madey, M. D. Troutt, and J. Liegle, "The knowledge management efficacy of matching information systems development methodologies with application characteristics—an experimental study," *Journal of Systems and Software*, vol. 79, pp. 15-28, 2006.
- [140] I. Russell and Z. Markov, "An Introduction to the Weka Data Mining System," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 2017, pp. 742-742.
- [141] C. Jones, *Assessment and control of software risks*: Yourdon Press, 1994.
- [142] C. Bai and J. Sarkis, "Green information technology strategic justification and evaluation," *Information Systems Frontiers*, pp. 1-17, 2013.
- [143] B. C. Stahl, "Teaching ethical reflexivity in information systems: How to equip students to deal with moral and ethical issues of emerging information and communication technologies," *Journal of Information Systems Education*, vol. 22, p. 253, 2011.
- [144] P. Haried and H. Dai, "The Evolution of Information Systems Offshoring Research: A Past, Present and Future Meta Analysis Review," *Journal of International Technology and Information Management*, vol. 20, pp. 83-102, 2011.
- [145] S. Aral, Y. Bakos, and E. Brynjolfsson, "How Trust, Incentives and IT Shape Global Supplier Networks: Theory and Evidence from IT Services Procurement," *Social Science Electronic Publishing, Inc*, 2010.
- [146] Y. Li, C.-H. Tan, and H.-H. Teo, "Leadership characteristics and developers' motivation in open source software development," *Information & Management*, vol. 49, pp. 257-267, 2012.
- [147] S. A. Slaughter, L. Levine, B. Ramesh, J. Pries-Heje, and R. Baskerville, "Aligning Software Processes with Strategy," *MIS Quarterly*, vol. 30, pp. 891-918, 2006.
- [148] S. Petter, W. Delone, and E. R. McLean, "The past, present, and future of "IS success"," *Journal of the Association of Information Systems*, vol. 13, pp. 341-362, 2012.
- [149] B. Strong, T. H. Davenport, and L. Prusak, "Organizational governance of knowledge and learning," *Knowledge and Process Management*, vol. 15, pp. 150-157, 2008.
- [150] H. F. Juergens, "Attributes of Information System Development," *MIS Quarterly*, vol. 1, pp. 31-41, 1977.

- [151] R. Vaezi, A. Mills, W. Chin, and H. Zafar, "User Satisfaction Research in Information Systems: Historical Roots and Approaches," *Communications of the Association for Information Systems*, vol. 38, pp. 501-532, 2016.
- [152] M. Wade and J. Hulland, "Review: The Resource-Based View and Information Systems Research: Review, Extension, and Suggestions for Future Research," *MIS Quarterly*, vol. 28, pp. 107-142, 2004.
- [153] I. Nonaka, A. Hirose, and Y. Takeda, "'Meso'-Foundations of Dynamic Capabilities: Team-Level Synthesis and Distributed Leadership as the Source of Dynamic Creativity," *Global Strategy Journal*, vol. 6, pp. 168-182, 2016.
- [154] D. E. Stokes, *Pasteur's quadrant: Basic science and technological innovation*: Brookings Institution Press, 2011.
- [155] J. Miller and B. A. Doyle, "Measuring the Effectiveness of Computer-Based Information Systems in the Financial Services Sector," *MIS Quarterly*, vol. 11, pp. 107-124, 1987.
- [156] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. d. M. Silveira Neto, Y. C. Cavalcanti, and S. R. d. L. Meira, "Twenty-eight years of component-based software engineering," *Journal of Systems and Software*, vol. 111, pp. 128-148, 2016.
- [157] M. Perkmann and K. Walsh, "University–industry relationships and open innovation: Towards a research agenda," *International Journal of Management Reviews*, vol. 9, pp. 259-280, 2007.
- [158] E. Georg von Krogh, C. R. Lamastra, E. Zurich, and S. Haeffliger, "Phenomenon-based research in management and organization science: Towards a research strategy."
- [159] G. von Krogh, C. Rossi-Lamastra, and S. Haeffliger, "Phenomenon-based Research in Management and Organisation Science: When is it Rigorous and Does it Matter?," *Long Range Planning*, vol. 45, pp. 277-298, 2012.
- [160] S. Tsugawa, H. Ohsaki, and M. Imase, "Inferring success of online development communities: Application of graph entropy for quantifying leaders' involvement," in *Information and Telecommunication Technologies (APSITT), 2010 8th Asia-Pacific Symposium on*, 2010, pp. 1-6.
- [161] K. Crowston, A. Wiggins, and J. Howison, "Analyzing leadership dynamics in distributed group communication," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, 2010, pp. 1-10.
- [162] S. Tsugawa, H. Ohsaki, and M. Imase, "Inferring leadership of online development community using topological structure of its social network," *Journal of the Infosociomics Society*, vol. 7, pp. 17-27, 2012.
- [163] A. Gupta and R. Singla, "Quantitative and Qualitative Evaluation of F/OSS Volunteer Participation in Defect Management," *International Journal of Software Engineering & Applications*, vol. 3, p. 71, 2012.
- [164] J. Howison, A. Wiggins, and K. Crowston, "Validity issues in the use of social network analysis with digital trace data," *Journal of the Association for Information Systems*, vol. 12, p. 767, 2011.
- [165] Z. Leila and J. Davis, "Modeling Service Interaction Networks," 2012.

- [166] R. Sen, M. L. Nelson, and C. Subramaniam, "Application of Survival Model to Understand Open Source Software Release," *Pacific Asia Journal of the Association for Information Systems*, vol. 7, 2015.
- [167] J. Howison, "Alone Together: A socio-technical theory of motivation, coordination and collaboration technologies in organizing for free and open source software development," 2008.
- [168] C. C. M. Campaign, "Daily Archives: April 18, 2014."
- [169] B. Weikel, "FROM CODING TO COMMUNITY," Citeseer, 2009.
- [170] M. Van Antwerp, "Temporal and Topological Analysis of Open Source Software Networks," Citeseer, 2013.
- [171] L. Yu, "The coevolution of mobile os user market and mobile application developer community," *Compusoft*, vol. 2, p. 44, 2013.
- [172] S. Von Engelhardt and A. Freytag, "Geographic allocation of oss contributions: the role of institutions and culture," *Jena economic research papers*2009.
- [173] S. v. Engelhardt and A. Freytag, "Institutions, culture, and open source," *Journal of Economic Behavior & Organization*, vol. 95, pp. 90-110, 2013.
- [174] F. Rullani and S. Haefliger, "The periphery on stage: The intra-organizational dynamics in online communities of creation," *Research Policy*, vol. 42, pp. 941-953, 2013.
- [175] M. Van Antwerp and G. Madey, "Open source software developer and project networks," in *IFIP International Conference on Open Source Systems*, 2010, pp. 407-412.
- [176] L. Zamani and J. G. Davis, "Analyzing Service Interaction Network Effectiveness," in *SRII Global Conference (SRII), 2011 Annual*, 2011, pp. 781-789.
- [177] K.-Y. Huang and N. Choi, "Relating and Clustering Free/Libre Open Source Software Projects and Developers: A Social Network Perspective," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, 2011, pp. 1-10.
- [178] Q. Le and J. H. Panchal, "Building smaller sized surrogate models of complex bipartite networks based on degree distributions," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 42, pp. 1152-1166, 2012.
- [179] S. Daniel, K. Stewart, and D. Darcy, "Patterns of evolution in open source projects: A Categorization Schema and Implications," *Patterns of Evolution in Open Source Projects: A Categorization Schema and Implications*, 2009.
- [180] S. Daniel, R. Agarwal, and K. J. Stewart, "The effects of diversity in global, distributed collectives: A study of open source project success," *Information Systems Research*, vol. 24, pp. 312-333, 2013.
- [181] P. Fly, J. Sims, and H. Kim, "A Study on Characteristics of Open Source Software Development Projects in the Areas of Engineering and Games."
- [182] S. von Engelhardt, A. Freytag, and C. Schulz, "On the geographic allocation of open source software activities," *Jena economic research papers*2010.
- [183] C. Zhang, "A network perspective on open source software development: Team formation and community participation," Purdue University, 2008.

- [184] M. Becker, F. Rullani, and F. Zirpoli, "Coordinating distributed innovation processes: The case of the automotive and open source software industries," in *Papier présenté à la conférence DRUID à Copenhague*, 2009.
- [185] D. Aksoy-yurdagul, F. Rullani, and C. Rossi-lamastra, "Organizing a Firm-Community Collaboration for growth: How to benefit from open source projects without hurting them."
- [186] C. Lampe, "Behavioral Trace Data for Analyzing Online Communities," *The SAGE Handbook of Digital Technology Research*, p. 236, 2013.
- [187] A. Wiggins, J. Howison, and K. Crowston, "Heartbeat: measuring active user base and potential user interest in FLOSS projects," in *IFIP International Conference on Open Source Systems*, 2009, pp. 94-104.
- [188] M. Van Antwerp and G. Madey, "The importance of social network structure in the open source software developer community," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, 2010, pp. 1-10.
- [189] G. Madey, "Open Source Software Developer and Project Networks," ed: Springer, 2010.
- [190] E. Anjos, J. Brasileiro, D. Silva, and M. Zenha-Rela, "Using Classification Methods to Reinforce the Impact of Social Factors on Software Success," in *International Conference on Computational Science and Its Applications*, 2016, pp. 187-200.
- [191] F. Bolici, J. Howison, and K. Crowston, "Coordination without discussion? Socio-technical congruence and Stigmergy in Free and Open Source Software projects," in *Socio-Technical Congruence Workshop in conj Intl Conf on Software Engineering, Vancouver, Canada*, 2009.
- [192] C. D. R. Smit, "The influence of social network structure on the chance of success of Open Source software project communities."
- [193] M. Van Antwerp, "Evolution of Open Source Software Networks."
- [194] S. Lakka, T. Stamati, C. Michalakelis, and D. Anagnostopoulos, "Cross-national analysis of the relation of eGovernment maturity and OSS growth," *Technological Forecasting and Social Change*, vol. 99, pp. 132-147, 2015.
- [195] B. Foushee, J. L. Krein, J. Wu, R. Buck, C. D. Knutson, L. J. Pratt, and A. C. MacLean, "Reflexivity, raymond, and the success of open source software development: A sourceforge empirical study," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, 2013, pp. 246-251.
- [196] C. Brindescu, M. Codoban, S. Shmarkatiuk, and D. Dig, "How do centralized and distributed version control systems impact software changes?," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 322-333.
- [197] W. Wen, C. Forman, and S. J. Graham, "The Impact of Intellectual Property Enforcement on Open Source Software Adoption," in *ICIS*, 2010, p. 187.
- [198] R. E. Vlas and C. Vlas, "A Requirements-Based Analysis of Success in Open-Source Software Development Projects," in *AMCIS*, 2011.
- [199] B. Hosack and G. Sagers, "Participation in oss projects: Does it support release early release often?," 2011.

- [200] D. P. S. Wagle, "SciBrowser: Exploration and Analysis of the Complexity, Structure, and Activity Dynamics of Open Source Science Communities," Auburn University, 2011.
- [201] A. C. MacLean, "Commit Patterns and Threats to Validity in Analysis of Open Source Software Repositories," Brigham Young University, 2012.
- [202] L. J. Pratt, "Cliff Walls: Threats to Validity in Empirical Studies of Open Source Forges," Brigham Young University, 2013.
- [203] B. D. Foushee, "Prevalence of Reflexivity and Its Impact on Success in Open Source Software Development: An Empirical Study," 2013.
- [204] A. Gupta and R. Singla, "Defect Management Practices and Problems in Free/Open Source Software Projects," *International Journal of Software Engineering (IJSE)*, p. 105.
- [205] V. Garousi and J. Leitch, "IssuePlayer: An extensible framework for visual assessment of issue management in software development projects," *Journal of Visual Languages & Computing*, vol. 21, pp. 121-135, 2010.
- [206] A. H. Ghapanchi, "Predicting software future sustainability: A longitudinal perspective," *Information Systems*, vol. 49, pp. 40-51, 2015.
- [207] S. Daniel and K. Stewart, "Open source project success: Resource access, flow, and integration," *The Journal of Strategic Information Systems*, vol. 25, pp. 159-176, 2016.
- [208] P. Kanwal, A. Gupta, and R. K. Singla, "Sampling Process Model for Coordination and Communication in Free/Open Source Software Projects," *International Journal of Computer Applications*, vol. 98, 2014.
- [209] J. M. Gonzalez-Barahona, D. Izquierdo-Cortazar, and M. Squire, "Repositories with public data about software development," *International Journal of Open Source Software and Processes (IJOSSP)*, vol. 2, pp. 1-13, 2010.
- [210] T. Abdou, "Towards Understanding Open Source Software Testing: An Empirical Study," *Social Media and Publicity*, p. 3.
- [211] E. C. Ramos, F. M. Santoro, and F. A. Baião, "A Method for Discovering the Relevance of External Context Variables to Business Processes," in *KMIS*, 2011, pp. 399-408.
- [212] C. Schweik and R. English, "Preliminary steps toward a general theory of internet-based collective-action in digital information commons: Findings from a study of open source software projects," *International Journal of the Commons*, vol. 7, 2013.
- [213] T. Abdou, P. Grogono, and P. Kamthan, "Managing Corrective Actions to Closure in Open Source Software Test Process," in *SEKE*, 2013, pp. 306-311.
- [214] A. Lemnar, "Open source framework usage: an investigation of the user's intention to continue using a framework," Lethbridge, Alta.: University of Lethbridge, Faculty of Management, 2013.
- [215] A. Blekh, "Governance and organizational sponsorship as success factors in free/libre and open source software development: An empirical investigation using structural equation modeling," 2015.
- [216] M. Codoban, "A Comparative Study on How SVN and Git Affect Software Changes," 2015.

- [217] C. Rossi-Lamastra, F. Rullani, and E. Piva, "Firmsâ€™ participation strategies and performances of OSS projects: an empirical analysis."
- [218] A. Vernet, M. Kilduff, and A. Salter, "Information, control, and small worlds: studying returns to individual network positions under different global structures," in *35th DRUID Celebration Conference*, 2013, pp. 17-19.
- [219] L. ROZENBERG and M. KIERUZEL, "Using Markowitz's idea to the valuable estimation of the IT projects risk."
- [220] V. Garousi, "Investigating the success factors of open-source software projects across their lifetime," *Journal of Software Engineering Studies*, vol. 4, pp. 1-15, 2009.
- [221] V. Garousi, "Evidence-based insights about issue management processes: an exploratory study," in *International Conference on Software Process*, 2009, pp. 112-123.
- [222] A. Wiggins and K. Crowston, "Reclassifying success and tragedy in FLOSS projects," in *IFIP International Conference on Open Source Systems*, 2010, pp. 294-307.
- [223] M. Van Antwerp and G. Madey, "Warehousing and Studying Open Source Versioning Metadata," in *IFIP International Conference on Open Source Systems*, 2010, pp. 413-418.
- [224] E. C. Ramos, F. M. Santoro, and F. Baião, "Thinking out of the box: discovering the relevance of external context to business processes," in *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, 2011, pp. 455-470.
- [225] C. Jensen and W. Scacchi, "Computer support for discovering OSS processes," in *Proceedings of the 3rd International Workshop on Public Data about Software Development, Milan, Italy*, 2008.
- [226] K. Crowston, C. Østerlund, J. Howison, and F. Bolici, "Work as coordination and coordination as work: A process perspective on FLOSS development projects," in *Third International Symposium on Process Organization Studies. Corfu, Greece*, 2011.
- [227] C. M. Schweik, R. English, Q. Paienjton, and S. Haire, "Success and abandonment in open source commons: Selected findings from an empirical study of sourceforge. net projects," in *Proceedings of the Sixth International Conference on Open Source Systems (OSS 2010) Workshops*, 2010.
- [228] Y. Gao, "Clustering Dependencies over Relational Tables," 2016.
- [229] S. Syed, "A Z-score for Open Source Projects," 2013.
- [230] F. Schweitzer, V. Nanumyan, C. J. Tessone, and X. Xia, "How do OSS projects change in number and size? A large-scale analysis to test a model of project growth," *Advances in Complex Systems*, vol. 17, p. 1550008, 2014.
- [231] J. L. Krein, "Programming Language Fragmentation and Developer Productivity: An Empirical Study," 2011.
- [232] R. Vlas, "A Requirements-Based Exploration of Open-Source Software Development Projects—Towards a Natural Language Processing Software Analysis Framework," 2012.
- [233] A. C. MacLean, L. J. Pratt, J. L. Krein, and C. D. Knutson, "Threats to validity in analysis of language fragmentation on SourceForge data," in *Proceedings of the*

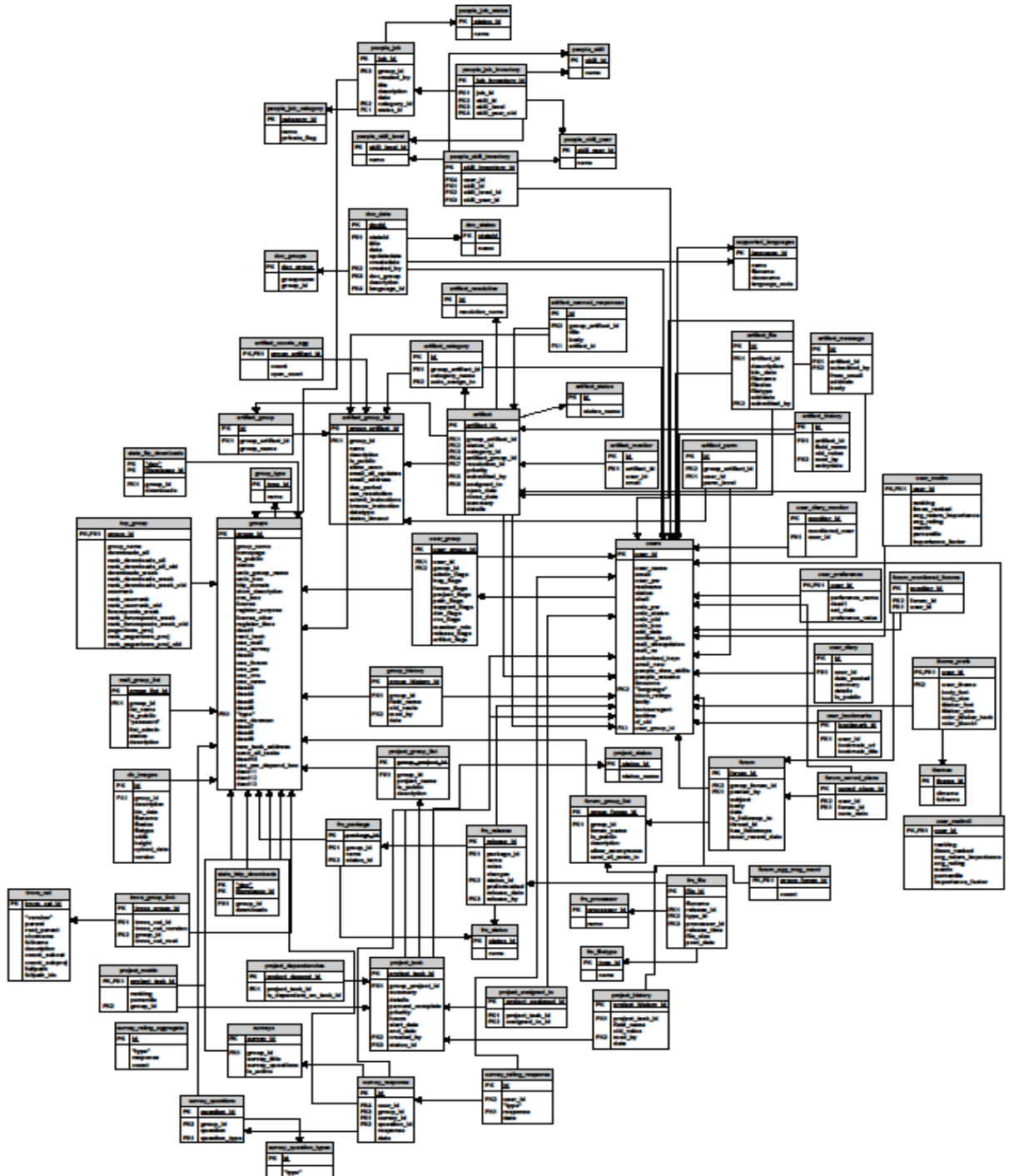
*1st International Workshop on Replication in Empirical Software Engineering Research*, 2010.

- [234] R. Vlas and W. N. Robinson, "A rule-based natural language technique for requirements discovery and classification in open-source software development projects," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, 2011, pp. 1-10.
- [235] R. Vlas and W. N. Robinson, "Applying a Rule-Based Natural Language Classifier to Open Source Requirements: a Demonstration of Theory Exploration," in *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, 2013, pp. 3158-3167.
- [236] R. E. Vlas and W. N. Robinson, "Two rule-based natural language strategies for requirements discovery and classification in open source software development projects," *Journal of Management Information Systems*, vol. 28, pp. 11-38, 2012.

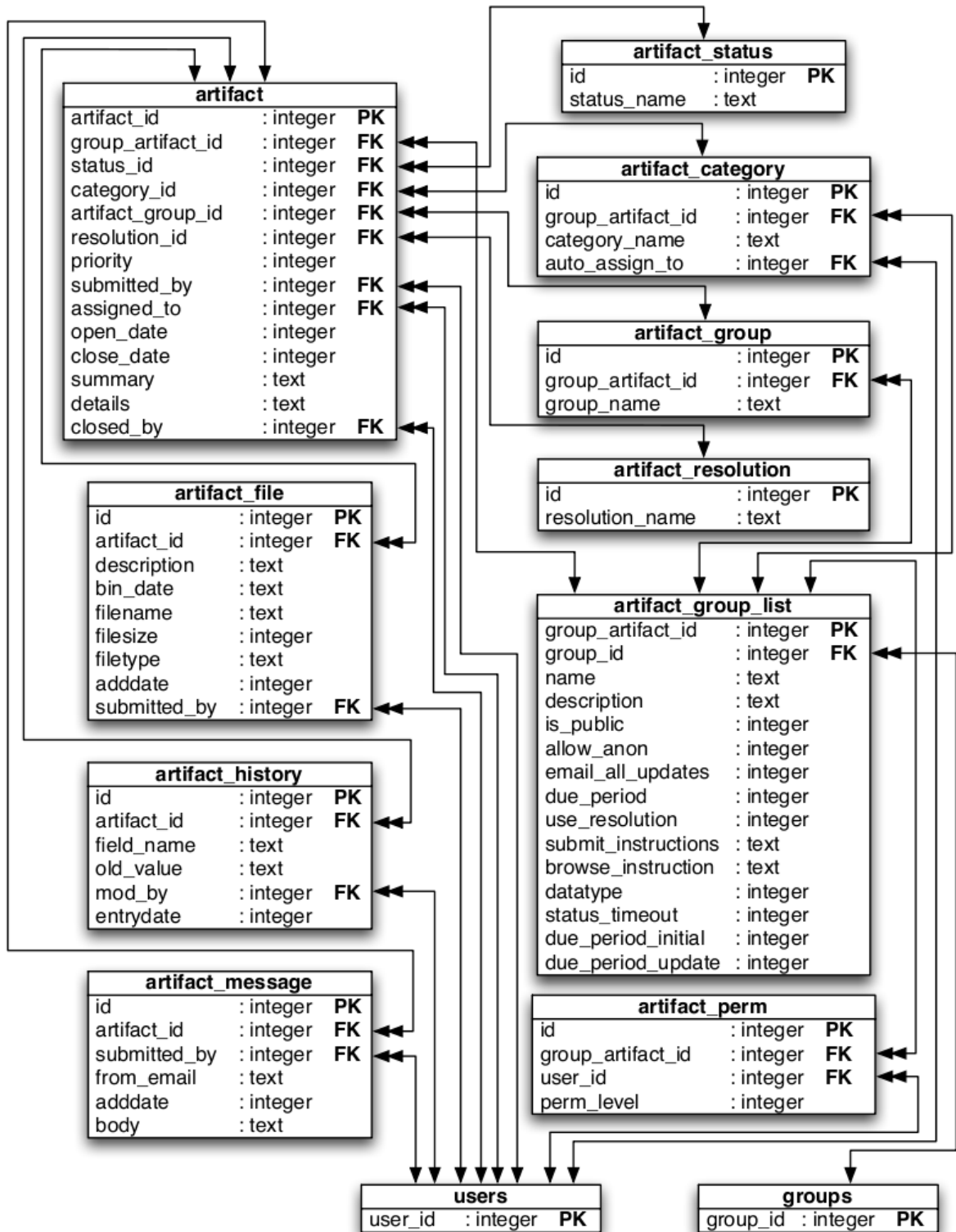


# APPENDIX A: THE SRDA

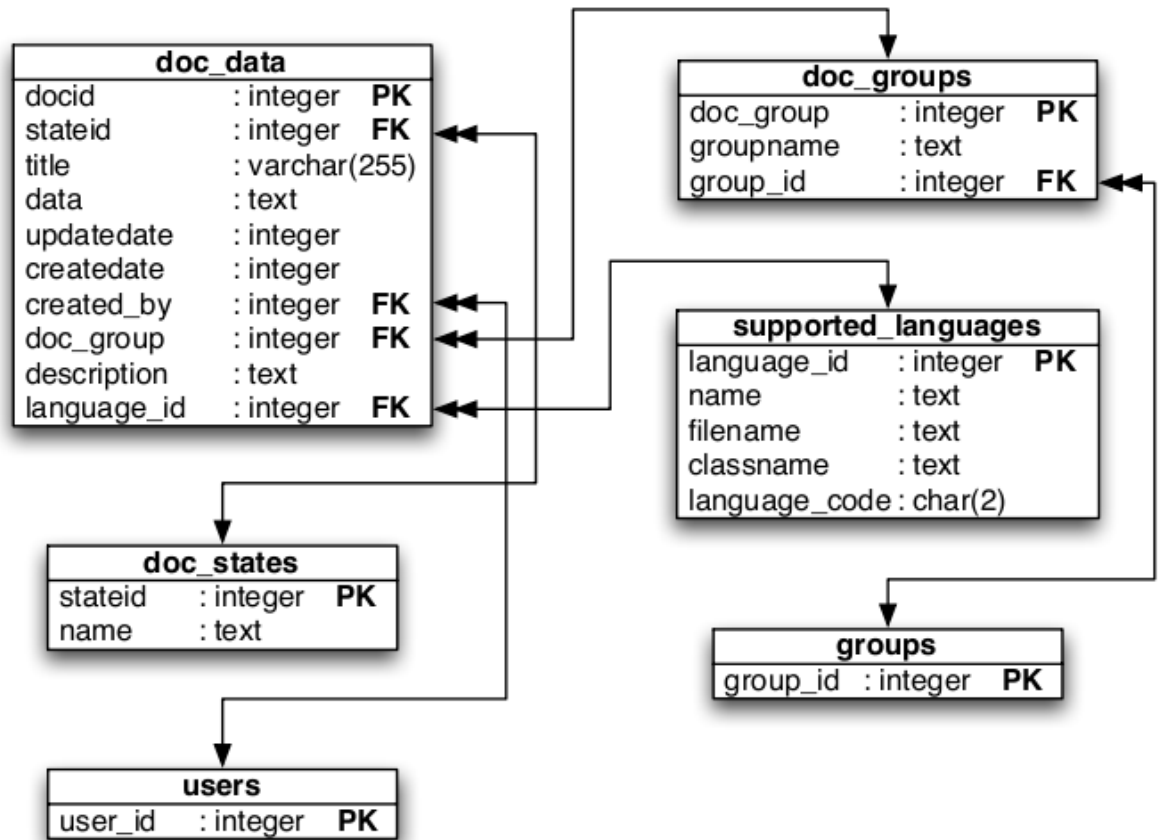
## A.1. Data Entity Relationship Diagram—SourceForge.net



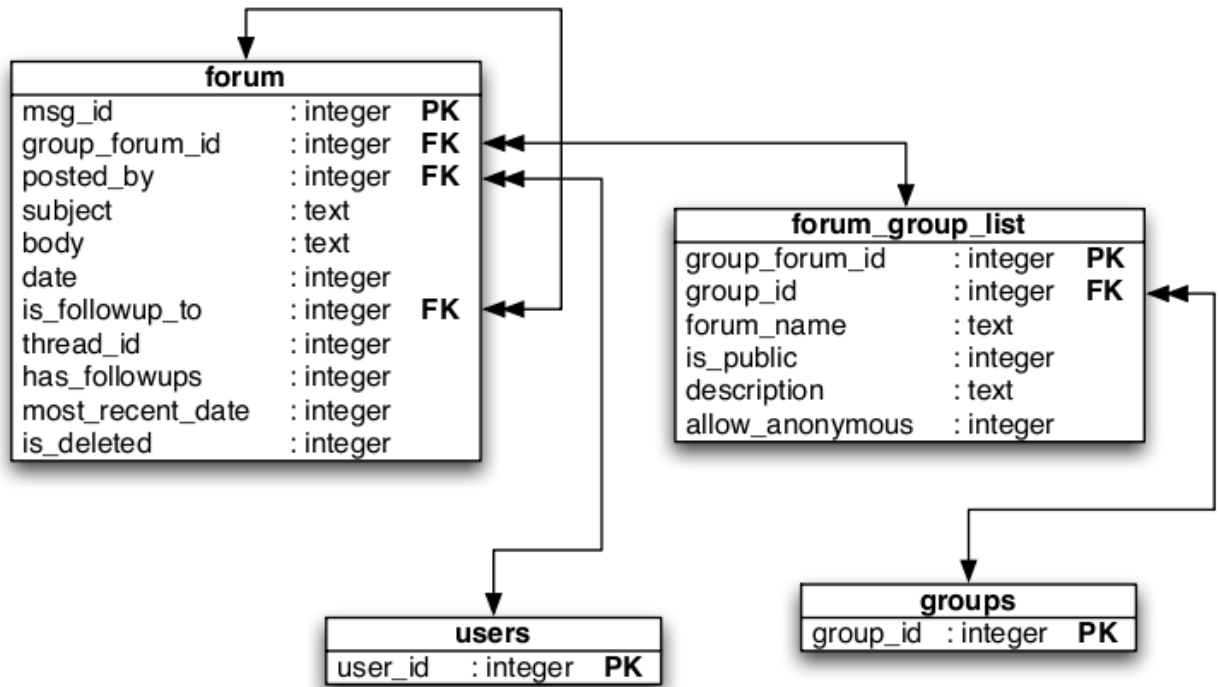
## A.2. Data Entity Relationship Model – SRDA – Artifact



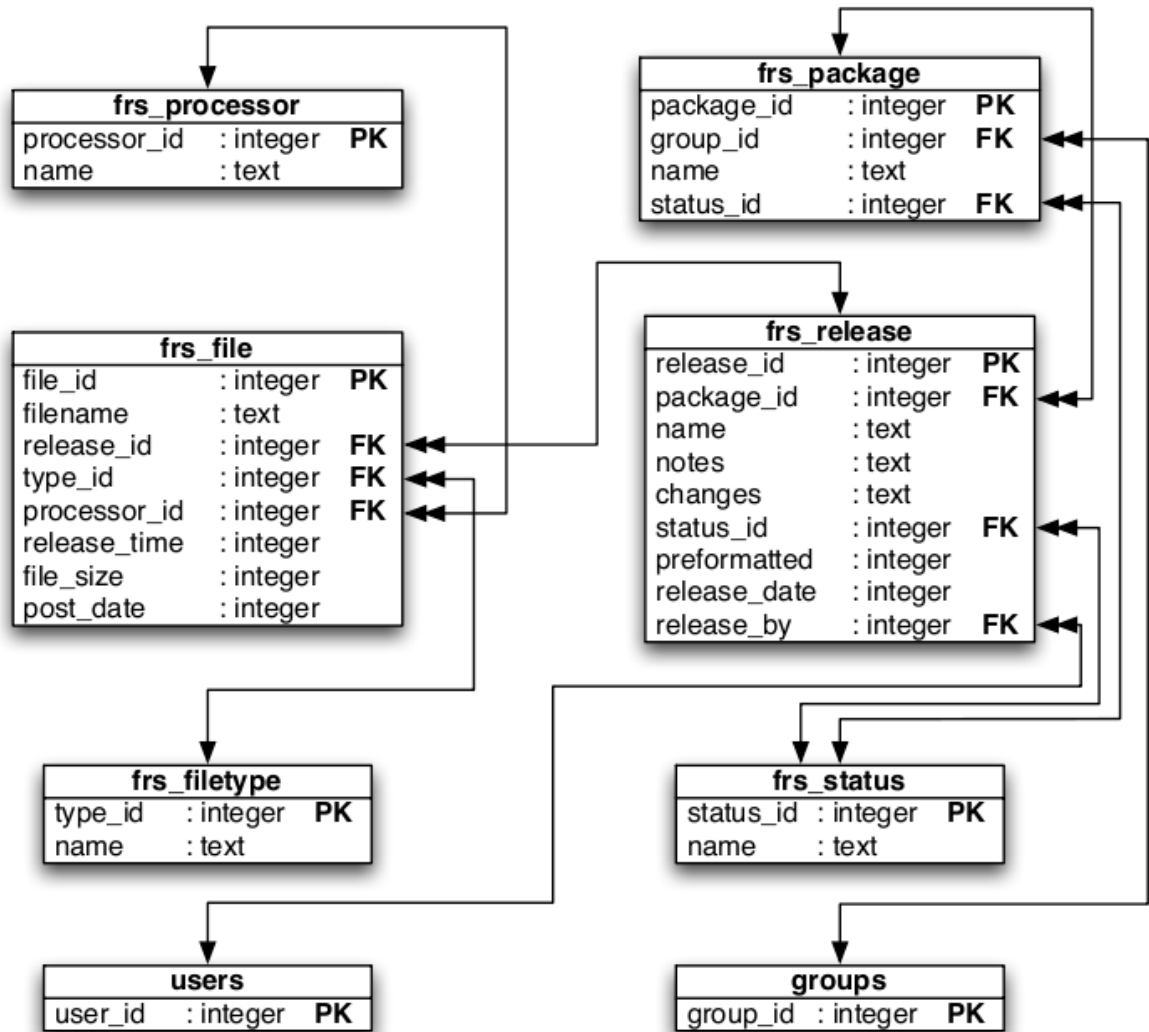
### A.3. Data Entity Relationship Model – SRDA – Documents



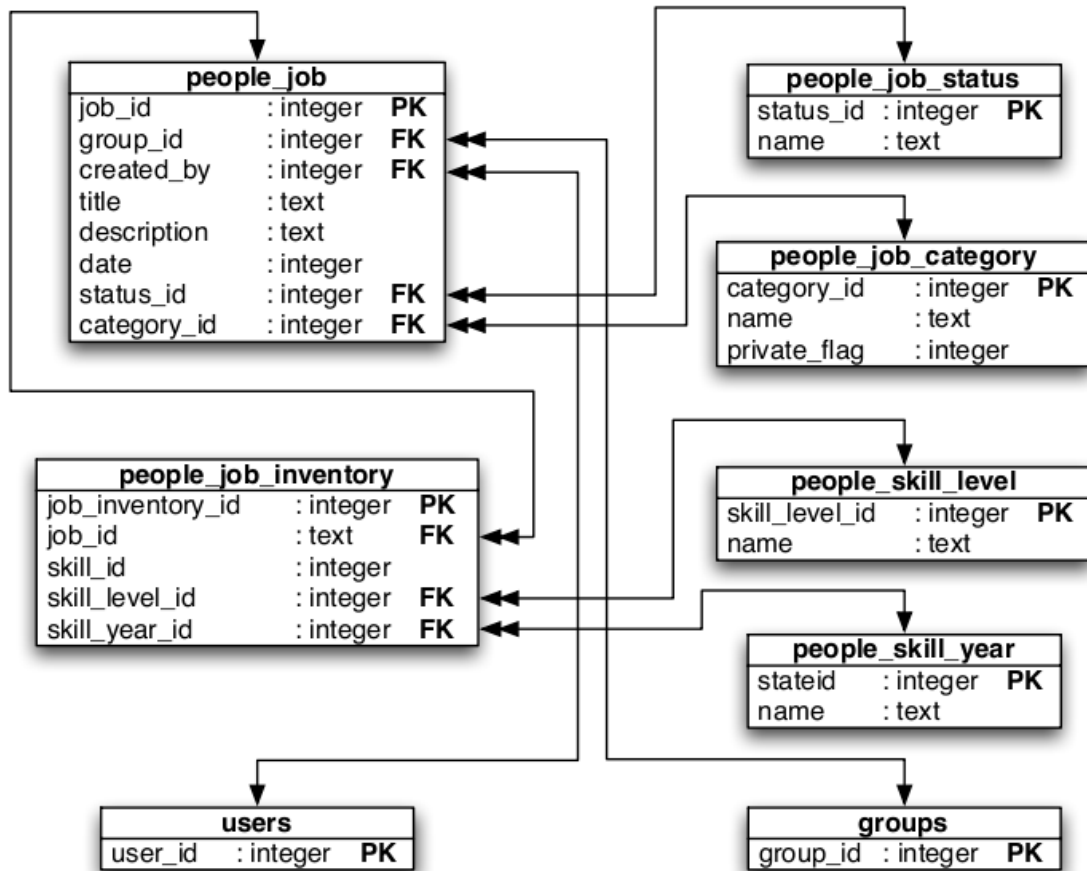
#### A.4. Data Entity Relationship Model – SRDA – Forums



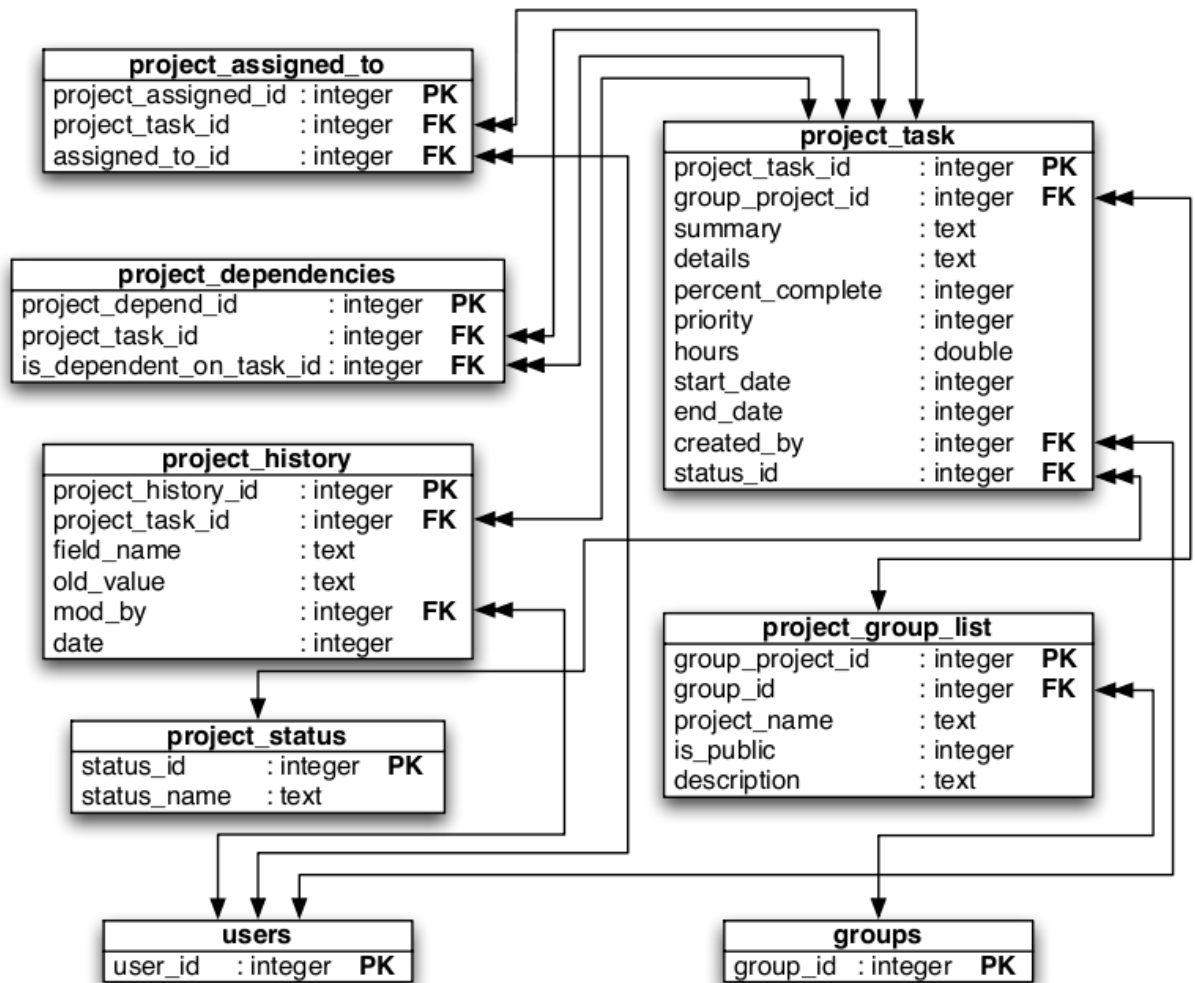
### A.5. Data Entity Relationship Model – SRDA – FRS



## A.6. Data Entity Relationship Model – SRDA – Job



### A.7. Data Entity Relationship Model – SRDA – Tasks



### A.8. List of tables analyzed from OSS Research

The following are the data tables which were analyzed for this research study:

activity_log	moorman_massmail_20060224	moorman_massmail_20060630
activity_log_old	moorman_massmail_20060511	moorman_massmail_20060630b
activity_log_old_old	moorman_massmail_20060523	moorman_massmail_20060908
activity_log_regs	b	moorman_proj2
activity_log_regs_tmp	ostg_contest	moorman_sdmaillist
admin_annotations	payment_option	moorman_sdmaillist2
artifact	people_job	mypstat
artifact_canned_responses	people_skill	mysql_auth
artifact_category	people_skill_inventory	news_bytes

artifact_counts_agg	people_skill_level	specialty
artifact_file	people_skill_year	stats_agg_logo_by_day
artifact_ftispool	pg72_bug	stats_agg_logo_by_group
artifact_group	pg_autovac_skip	stats_agg_pages_by_day
artifact_group_list	pg_stat_database_historical	stats_agg_site_by_group
artifact_history	pg_ts_cfg	stats_cvs_group
artifact_message	pg_ts_cfgmap	stats_cvs_user
artifact_monitor	pg_ts_dict	stats_fileid_alltime_agg
artifact_perm	pg_ts_parser	stats_ftp_downloads
artifact_resolution	prdb_dbs	stats_groupid_alltime_agg
artifact_status	prdb_states	stats_group_rank
audit_trail	prdb_types	stats_group_rank_alltime
audit_trail_data	project_assigned_to	stats_group_rank_byday
audit_trail_group	project_counts_weekly_tmp	stats_group_rank_byday_0528
audit_trail_group_data	project_dependencies	stats_group_rank_byday_backup_0
audit_trail_user	project_group_list	50528
autopurge_exempt	project_history	stats_group_rank_byday_backup_0
autopurge_projects	project_metric	50529
beta_members	project_metric_tmp1	stats_group_rank_bymonth
beta_offerings	project_metric_weekly_tmp1	stats_http_downloads
blocks	project_purge	stats_multi_rank_history_byday
cache_store	project_status	stats_outage_log
canned_responses	project_sums_agg	stats_project
category_management	project_task	stats_project_all
charities	project_weekly_metric	stats_project_developers
cloudscape_answers	prweb_vhost	stats_project_developers_last30
cloudscape_contest	purge_history	stats_project_last_30
cronjob_history	purge_queue	stats_project_metric
db_images	ranking_tmp	stats_project_months
doc_data	rating	stats_rank_oldformula_byday
doc_groups	ref_timezones	stats_sfweb_recent_hit
doc_states	reputation	stats_site
dup_emails	scm_repo_trigger	stats_site_last_30
entity_charity	scm_trigger	stats_site_months
external_tool_links	screenshots	stats_site_pages_by_day
filemodule_monitor	search_data_groups	stats_subd_pages
finance_audit	seller_profile_language	stats_toplist_week
foo	seller_profile_payment_option	stats_trove_topic_activity
forum	service_contract	subscriptions
forum_agg_msg_count	service_listing	supported_languages
forumemail	service_listing_language	survey_questions
forum_ftispool_new	snippet_package	survey_question_types
forum_group_list	snippet_package_item	survey_rating_aggregate
forum_monitored_forums	snippet_package_version	survey_rating_response
forum_saved_place	snippet_version	survey_responses
forum_threadinfo	svn_migration_log	surveys



foundry_data	svn_migration_queue	trove_agg
foundry_news	test1	trove_agg_counts
foundry_preferred_projects	theme_prefs	trove_agg_minix
imported_projects	themes	trove_agg_tmp
intel_agreement	tmp_stats_fileid_alltime_agg_1	trove_cat
invalid_name	118629848	trove_cat_activity
kernel_traffic	tmp_stats_fileid_alltime_agg_1	trove_frontpage
lucene_searchspool	140665539	trove_group_link
lucene_searchspool2	tmp_stats_fileid_alltime_agg_1	trove_monitor
mail_group_list	142736917	trove_monitor_event_queue
massmail_queue	tmp_stats_fileid_alltime_agg_1	trove_ref_translation_to_iso639
mllist_subscriber	145005244	trove_treesums
mllist_subscriber_count	tmp_stats_fileid_alltime_agg_1	tshirt_codes
money_in	147219524	user_auth_keys
monitor_enable	tmp_stats_fileid_alltime_agg_1	user_bookmarks
monitor_project	151460886	user_diary
moorman_defunct_unix_uids	tmp_stats_fileid_alltime_agg_1	user_diary_monitor
service_listing_payment_option	155778896	user_group
service_order	tmp_stats_fileid_alltime_agg_1	user_ip_dl_auth
session	156537400	user_metric
sfce_api_mapping	tmp_stats_fileid_alltime_agg_1	user_metric0
snippet	158781396	user_metric_history
people_job_category	tmp_stats_groupid_alltime_agg_1	user_perms
people_job_inventory	1117227330	user_preferences
people_job_status	tmp_stats_group_rank_byday_1	user_ratings
	151598926	user_role
	top_group	users
	top_group_tmp	users-bak_regs
		users_lookup
		users_registration

#### A.9. Select SRDA Data Stored in Local MS Access Database

**Table: Artifact-Bugs – 4,731,734 Records**

<b><u>Columns</u></b>		
<b>Name</b>	<b>Type</b>	<b>Size</b>
artifact_id	Long Integer	4
group_id	Long Integer	4
open_date	Long Integer	4
close_date	Long Integer	4
category_name	Text	255

---

**Table: frs\_package – 1,883,112 Records**

<u>Columns</u>		
Name	Type	Size
package_id	Text	255
group_id	Text	255
name	Text	255
status_id	Text	255

**Table: frs\_release – 507,951 Records**

<u>Columns</u>		
Name	Type	Size
release_id	Long Integer	4
package_id	Long Integer	4
status_id	Long Integer	4
preformatted	Long Integer	4
release_date	Long Integer	4
released_by	Long Integer	4

**Table: groups – 1,885,588 Records**

<u>Columns</u>		
Name	Type	Size
group_id	Text	255
group_name	Text	255
status	Text	255
short_description	Text	255
license	Text	255
register_time	Text	255

**Table: stats\_project\_all – 63,785 Records**

<u>Columns</u>		
Name	Type	Size
group_id	Text	255
developers	Text	255
group_ranking	Text	255
group_metric	Text	255
logo_showings	Text	255
downloads	Text	255
site_views	Text	255
subdomain_views	Text	255
page_views	Text	255
msg_posted	Text	255
msg_uniq_auth	Text	255
bugs_opened	Text	255
bugs_closed	Text	255
support_opened	Text	255
support_closed	Text	255

---

---

patches_opened	Text	255
patches_closed	Text	255
artifacts_opened	Text	255
artifacts_closed	Text	255
tasks_opened	Text	255
tasks_closed	Text	255
help_requests	Text	255
cvs_checkouts	Text	255
cvs_commits	Text	255
cvs_adds	Text	255
svn_checkouts	Text	255
svn_commits	Text	255
svn_adds	Text	255

**Table: trove\_cat – 870 Records**

**Columns**

<b>Name</b>	<b>Type</b>	<b>Size</b>
trove_cat_id	Text	255
version	Text	255
parent	Text	255
root_parent	Text	255
shortname	Text	255
fullname	Text	255
description	Text	255
fullpath	Text	255
fullpath_ids	Text	255
parent_only	Text	255
people_skill	Text	255

**Table: trove\_group\_link – 2,135,315 Records**

**Columns**

<b>Name</b>	<b>Type</b>	<b>Size</b>
trove_group_id	Long Integer	4
trove_cat_id	Long Integer	4
trove_cat_version	Long Integer	4
group_id	Long Integer	4
trove_cat_root	Long Integer	4
entity_type	Long Integer	4

---

## APPENDIX B: EXISTING RESEARCH – THE SRDA

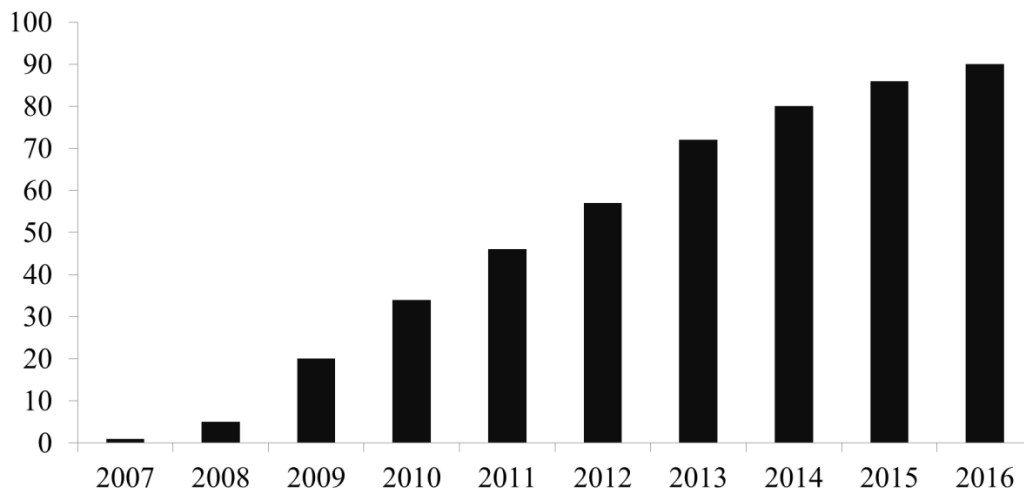
### B.1. Reference List by Research Focus

Research focus	Reference
<b>Leadership</b>	<p>Georg von Krogh, ETH, Lamastra, Cristina Rossi, Zurich, ETH, Haefliger, Stefan, 2009 [158]</p> <p>Georg von Krogh, ETH, Lamastra, Cristina Rossi, Zurich, ETH, Haefliger, Stefan, 2012 [159]</p> <p>Tsugawa, Sho, Ohsaki, Hiroyuki, Imase, Makoto, 2010 [160]</p> <p>Crowston, Kevin, Wiggins, Andrea, Howison, James, 2010 [161]</p> <p>Tsugawa, S, Ohsaki, Hiroyuki, Imase, Makoto, 2012 [162]</p>
	<p>Gupta, Anu, Singla, RK, 2012 [163]</p> <p>Howison, James, Wiggins, Andrea, Crowston, Kevin, 2011 [164]</p> <p>Leila, Zamani, Davis, Joseph, 2012 [165]</p> <p>Sen, Ravi, Nelson, Matthew L, Subramaniam, Chandrasekar, 2015 [166]</p> <p>Howison, James, 2008 [167]</p> <p>Campaign, Cross Channel Marketing, 2014 [168]</p> <p>Weikel, Brad, 2009 [169]</p> <p>Van Antwerp, Matthew, 2013 [170]</p> <p>Yu, Ligu, 2013 [171]</p> <p>Engelhardt, Sebastian von, Freytag, Andreas, 2009 [172]</p> <p>Engelhardt, Sebastian v, Freytag, Andreas, 2013 [173]</p> <p>Rullani, Francesco, Haefliger, Stefan, 2013 [174]</p> <p>Weikel, Bradley N, 2009 [169]</p> <p>Van Antwerp, Matthew, Madey, Greg, 2010 [175]</p> <p>Zamani, Leila, Davis, Joseph G, 2011 [176]</p> <p>Huang, Kuang-Yuan, Choi, Namjoo, 2011 [177]</p> <p>Le, Qize, Panchal, Jitesh H, 2012 [178]</p> <p>Daniel, Sherae, Agarwal, Ritu, Stewart, Katherine, 2009 [179]</p> <p>Daniel, Sherae, Agarwal, Ritu, Stewart, Katherine J, 2013 [180]</p> <p>Fly, Pervis, Sims, James, Kim, Hyunju, 2012 [181]</p> <p>Von Engelhardt, Sebastian, Freytag, Andreas, 2009 [172]</p> <p>von Engelhardt, Sebastian, Freytag, Andreas, Schulz, Christoph, 2010 [182]</p> <p>Zhang, Chen, 2008 [183]</p> <p>Becker, Markus, Rullani, Francesco, Zirpoli, Francesco, 2009 [184]</p> <p>Aksoy-yurdagul, Dilan, Rullani, Francesco, Rossi-lamastra, Cristina, 2014 [185]</p> <p>Lampe, Cliff, 2013 [186]</p> <p>Wiggins, Andrea, Howison, James, Crowston, Kevin, 2009 [187]</p> <p>Van Antwerp, Matthew, Madey, Greg, 2010 [188]</p> <p>Madey, Greg, 2010 [189]</p>

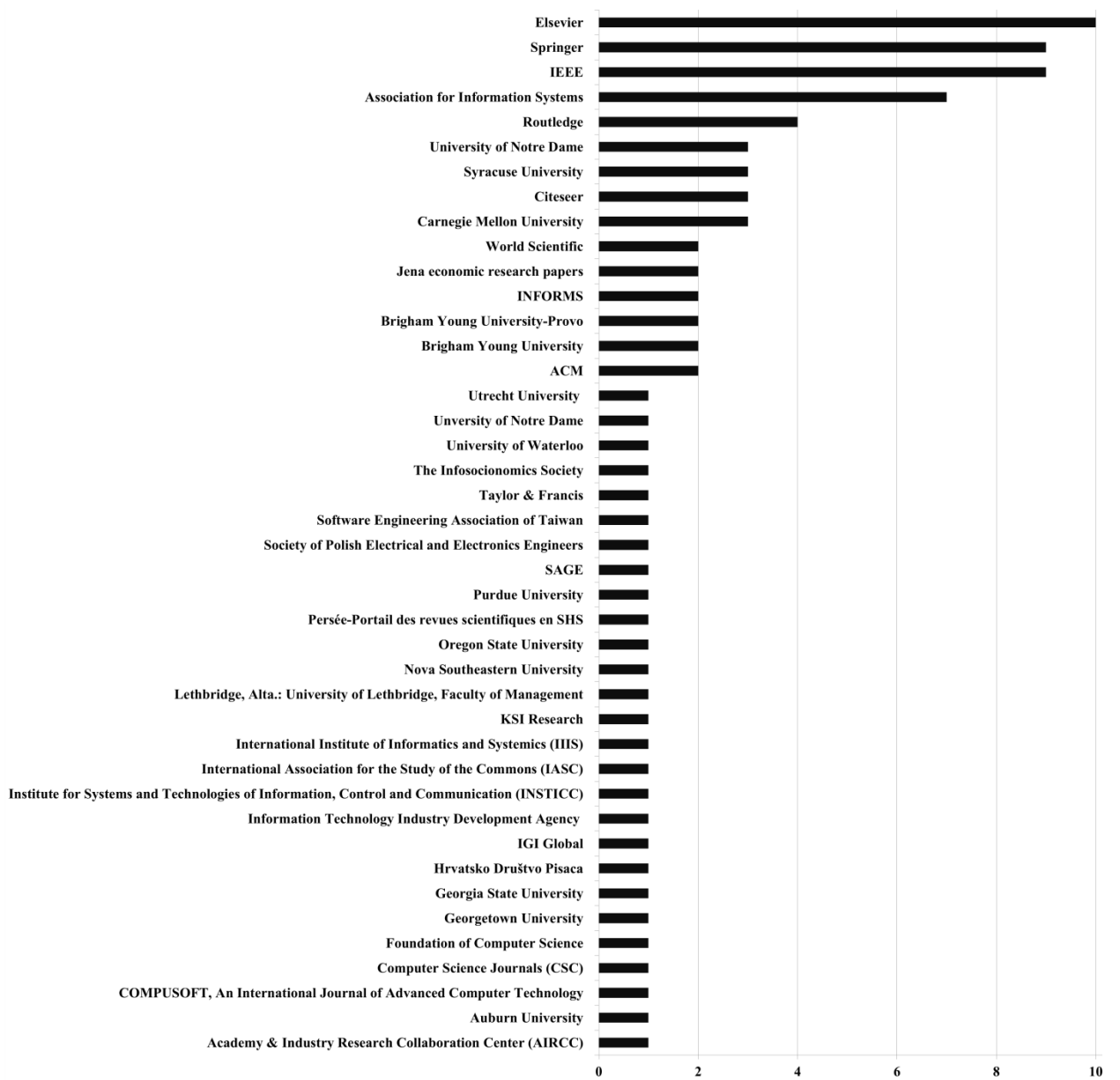
Research focus	Reference
	<p>Anjos, Eudisley, Brasileiro, Jansepetrus, Silva, Danielle, Zenha-Rela, Mário, 2016 [190]</p> <p>Bolici, Francesco, Howison, James, Crowston, Kevin, 2009 [191]</p> <p>Smit, Coach Drs R, 2009 [192]</p> <p>Van Antwerp, Matthew, 2010 [193]</p> <p>Skopik, Florian, Schall, Daniel, Dustdar, Schahram, 2012</p>
<b>OSS Adoption</b>	<p>Lakka, Spyridoula, Stamati, Teta, Michalakelis, Christos, Anagnostopoulos, Dimosthenis, 2015 [194]</p>
<b>OSS Process</b>	<p>Foushee, Brandon, Krein, Jonathan L, Wu, Justin, Buck, Randy, Knutson, Charles D, Pratt, Landon J, MacLean, Alexander C, 2013 [195]</p> <p>Brindescu, Caius, Codoban, Mihai, Shmarkatiuk, Sergii, Dig, Danny, 2014 [196]</p> <p>Wen, Wen, Forman, Chris, Graham, Stuart JH, 2010 [197]</p> <p>Vlas, Radu E, Vlas, Cristina, 2011 [198]</p> <p>Hosack, Bryan, Sagers, Glen, 2011 [199]</p> <p>Wagle, Damodar P Shenvi, 2011 [200]</p> <p>MacLean, Alexander C, 2012 [201]</p> <p>Pratt, Landon J, 2013 [202]</p> <p>Foushee, Brandon D, 2013 [203]</p> <p>Daniel, Sherae, Stewart, Katherine, Darcy, David, 2009 [179]</p> <p>Gupta, Anu, Singla, RK, 2014 [204]</p> <p>Garousi, Vahid, Leitch, James, 2010 [205]</p> <p>Sen, Ravi, Singh, Siddhartha S, Borle, Sharad, 2012 [45]</p> <p>Ghapanchi, Amir Hossein, 2015 [206]</p> <p>Daniel, Sherae, Stewart, Katherine, 2016 [207]</p> <p>Kanwal, Preet, Gupta, Anu, Singla, Ravinder Kumar, 2014 [208]</p> <p>Gonzalez-Barahona, Jesus M, Izquierdo-Cortazar, Daniel, Squire, Megan, 2010 [209]</p> <p>Abdou, Tamer, 2014 [210]</p> <p>Ramos, Eduardo Costa, Santoro, Flávia Maria, Baião, Fernanda Araujo, 2011 [211]</p> <p>Schweik, Charles, English, Robert, 2013 [212]</p> <p>Abdou, Tamer, Grogono, Peter, Kamthan, Pankaj, 2013 [213]</p> <p>Lemnar, Alexandru, 2013 [214]</p> <p>Blekh, Aleksandr, 2015 [215]</p> <p>Codoban, Mihai, 2015 [216]</p> <p>Rossi-Lamastra, Cristina, Rullani, Francesco, Piva, Evila, 2011 [217]</p> <p>Vernet, Antoine, Kilduff, Martin, Salter, Ammon, 2013 [218]</p> <p>Rozenberg, Leonard, Kieruzel, Magdalena, 2016 [219]</p> <p>Garousi, Vahid, 2009 [220]</p> <p>Garousi, Vahid, 2009 [221]</p>

Research focus	Reference
	Wiggins, Andrea, Crowston, Kevin, 2010 [222] Van Antwerp, Matthew, Madey, Greg, 2010 [223] Ramos, Eduardo Costa, Santoro, Flavia Maria, Baião, Fernanda, 2011 [224] Jensen, Chris, Scacchi, Walt, 2008 [225] Crowston, Kevin, Østerlund, Carsten, Howison, James, Bolici, Francesco, 2011 [226] Schweik, Charles M, English, Robert, Paienjton, Qimti, Haire, Sandy, 2010 [227] Gao, Yuchen, 2016 [228] Van Antwerp, Matthew, Madey, Greg, 2008 [64] Syed, SAS, 2013 [229] Schweitzer, Frank, Nanumyan, Vahan, Tessone, Claudio J, Xia, Xi, 2014 [230]
<b>Software Languages</b>	Krein, Jonathan L, 2011 [231] Vlas, Radu, 2012 [232] MacLean, Alexander C, Pratt, Landon J, Krein, Jonathan L, Knutson, Charles D, 2010 [233] Vlas, Radu, Robinson, William N, 2011 [234] Vlas, Radu, Robinson, William N, 2013 [235] Vlas, Radu E, Robinson, William N, 2012 [236]

## B.2. Yearly Cumulative Publications – SRDA



### B.3. Publications by Publisher



## APPENDIX C: DATA RESULTS

### C.3. Weka 3.8.1 Machine Learning Algorithm – Results

#### C.3.1. Random Forest

=== Run information ===

```
Scheme:      weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V
0.001 -S 1
Relation:    testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:   18019
Attributes:  7
             Rank-quartile
             Download-quartile
             AvgTime-Quartile
             page_views-Quartile
             msg_posted-Quartile
             Total_D_P_S-Quartile
             Total_Nreq-Quartile
Test mode:   split 66.0% train, remainder test
```

Time taken to build model: 2.13 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 1.37 seconds

=== Summary ===

Correctly Classified Instances	4818	78.6484 %
Incorrectly Classified Instances	1308	21.3516 %
Kappa statistic	0.7154	
Mean absolute error	0.1376	
Root mean squared error	0.2653	
Relative absolute error	36.6927 %	
Root relative squared error	61.2685 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	0.836	0.131	0.680	0.836	0.750	0.661	0.937	0.820
High	0.587	0.071	0.737	0.587	0.654	0.560	0.916	0.787
Low	0.798	0.058	0.821	0.798	0.809	0.748	0.955	0.890
Very Low	0.928	0.025	0.924	0.928	0.926	0.902	0.992	0.975
Weighted Avg.	0.786	0.071	0.790	0.786	0.784	0.717	0.950	0.868

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1281	200	50	2	a = Very High
521	910	114	5	b = High
74	124	1213	109	c = Low
8	0	101	1414	d = Very Low



### C.3.2. Meta Bagging

=== Run information ===

```
Scheme:          weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 10 -W
weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0
Relation:        testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:       18019
Attributes:       7
                  Rank-quartile
                  Download-quartile
                  AvgTime-Quartile
                  page_views-Quartile
                  msg_posted-Quartile
                  Total_D_P_S-Quartile
                  Total_Nreq-Quartile
Test mode:       split 66.0% train, remainder test
```

Time taken to build model: 1.52 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.38 seconds

=== Summary ===

Correctly Classified Instances	4813	78.5668 %
Incorrectly Classified Instances	1313	21.4332 %
Kappa statistic	0.7143	
Mean absolute error	0.1484	
Root mean squared error	0.2696	
Relative absolute error	39.5636 %	
Root relative squared error	62.255 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	0.852	0.141	0.669	0.852	0.749	0.661	0.936	0.798
High	0.559	0.061	0.757	0.559	0.643	0.556	0.912	0.778
Low	0.797	0.057	0.822	0.797	0.810	0.748	0.944	0.872
Very Low	0.938	0.028	0.918	0.938	0.928	0.904	0.992	0.971
Weighted Avg.	0.786	0.072	0.791	0.786	0.782	0.716	0.946	0.855

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1306	176	49	2	a = Very High
554	866	127	3	b = High
84	102	1212	122	c = Low
8	0	86	1429	d = Very Low

### C.3.3. J48 Decision Tree

=== Run information ===

```
Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:   18019
Attributes:  7
              Rank-quartile
              Download-quartile
              AvgTime-Quartile
              page_views-Quartile
              msg_posted-Quartile
              Total_D_P_S-Quartile
              Total_Nreq-Quartile
Test mode:   split 66.0% train, remainder test
```

=== Classifier model (full training set) ===

Time taken to build model: 0.37 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.06 seconds

=== Summary ===

Correctly Classified Instances	4804	78.4198 %
Incorrectly Classified Instances	1322	21.5802 %
Kappa statistic	0.7123	
Mean absolute error	0.1493	
Root mean squared error	0.2764	
Relative absolute error	39.8099 %	
Root relative squared error	63.8351 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	0.832	0.136	0.672	0.832	0.743	0.652	0.926	0.755
High	0.575	0.071	0.734	0.575	0.645	0.550	0.898	0.740
Low	0.785	0.051	0.835	0.785	0.809	0.749	0.934	0.859
Very Low	0.948	0.030	0.912	0.948	0.930	0.906	0.989	0.947
Weighted Avg.	0.784	0.072	0.788	0.784	0.781	0.714	0.936	0.825

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1275	207	50	1	a = Very High
536	892	116	6	b = High
79	116	1193	132	c = Low
8	1	70	1444	d = Very Low

### C.3.4. Decision Table

=== Run information ===

```

Scheme:      weka.classifiers.rules.DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D
1 -N 5"
Relation:    testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:   18019
Attributes:   7
              Rank-quartile
              Download-quartile
              AvgTime-Quartile
              page_views-Quartile
              msg_posted-Quartile
              Total_D_P_S-Quartile
              Total_Nreq-Quartile
Test mode:   split 66.0% train, remainder test

```

Time taken to build model: 0.94 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.19 seconds

=== Summary ===

Correctly Classified Instances	4748	77.5057 %
Incorrectly Classified Instances	1378	22.4943 %
Kappa statistic	0.7001	
Mean absolute error	0.1858	
Root mean squared error	0.2833	
Relative absolute error	49.5433 %	
Root relative squared error	65.4318 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	0.834	0.146	0.656	0.834	0.734	0.640	0.929	0.791
High	0.575	0.080	0.708	0.575	0.635	0.533	0.903	0.748
Low	0.765	0.048	0.841	0.765	0.801	0.741	0.942	0.846
Very Low	0.929	0.027	0.921	0.929	0.925	0.900	0.988	0.964
Weighted Avg.	0.775	0.075	0.781	0.775	0.773	0.703	0.940	0.837

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1278	212	36	7	a = Very High
554	892	102	2	b = High
95	149	1163	113	c = Low
20	6	82	1415	d = Very Low

### C.3.5. K-Nearest Neighbor

=== Run information ===

```
Scheme:      weka.classifiers.lazy.KStar -B 20 -M a
Relation:    testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:   18019
Attributes:  7
              Rank-quartile
              Download-quartile
              AvgTime-Quartile
              page_views-Quartile
              msg_posted-Quartile
              Total_D_P_S-Quartile
              Total_Nreq-Quartile
Test mode:   split 66.0% train, remainder test
```

Time taken to build model: 0.01 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 128.18 seconds

=== Summary ===

Correctly Classified Instances	4715	76.967 %
Incorrectly Classified Instances	1411	23.033 %
Kappa statistic	0.693	
Mean absolute error	0.1909	
Root mean squared error	0.2897	
Relative absolute error	50.9178 %	
Root relative squared error	66.8943 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	0.865	0.156	0.650	0.865	0.742	0.652	0.930	0.791
High	0.515	0.056	0.757	0.515	0.613	0.529	0.903	0.752
Low	0.778	0.062	0.805	0.778	0.791	0.724	0.932	0.836
Very Low	0.924	0.033	0.901	0.924	0.913	0.884	0.987	0.958
Weighted Avg.	0.770	0.077	0.778	0.770	0.764	0.696	0.938	0.834

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1326	165	41	1	a = Very High
582	799	161	8	b = High
105	88	1182	145	c = Low
28	3	84	1408	d = Very Low

### C.3.6. Multi-Layer Perceptron

=== Run information ===

```
Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.3 -N 500 -V 0 -S 0 -E
20 -H 5 -G -R
Relation:    testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:   18019
Attributes:  7
              Rank-quartile
              Download-quartile
              AvgTime-Quartile
              page_views-Quartile
              msg_posted-Quartile
              Total_D_P_S-Quartile
              Total_Nreq-Quartile
Test mode:   split 66.0% train, remainder test
```

Time taken to build model: 88.14 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.28 seconds

=== Summary ===

Correctly Classified Instances	1533	25.0245 %
Incorrectly Classified Instances	4593	74.9755 %
Kappa statistic	0	
Mean absolute error	0.375	
Root mean squared error	0.433	
Relative absolute error	99.9908 %	
Root relative squared error	99.992 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	1.000	1.000	0.250	1.000	0.400	0.000	0.659	0.365
High	0.000	0.000	0.000	0.000	0.000	0.000	0.574	0.276
Low	0.000	0.000	0.000	0.000	0.000	0.000	0.494	0.238
Very Low	0.000	0.000	0.000	0.000	0.000	0.000	0.742	0.539
Weighted Avg.	0.250	0.250	0.063	0.250	0.100	0.000	0.617	0.354

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1533	0	0	0	a = Very High
1550	0	0	0	b = High
1520	0	0	0	c = Low
1523	0	0	0	d = Very Low

### C.3.7. Iterative Classifier

=== Run information ===

```
Scheme:          weka.classifiers.meta.IterativeClassifierOptimizer -W
weka.classifiers.meta.LogitBoost -L 50 -P 1 -E 1 -I 1 -F 10 -R 1 -metric RMSE -S 1 -- -P 100 -L
-1.7976931348623157E308 -H 1.0 -Z 3.0 -O 1 -E 1 -S 1 -I 10 -W
weka.classifiers.trees.DecisionStump
Relation:        testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:       18019
Attributes:      7
                Rank-quartile
                Download-quartile
                AvgTime-Quartile
                page_views-Quartile
                msg_posted-Quartile
                Total_D_P_S-Quartile
                Total_Nreq-Quartile
Test mode:       split 66.0% train, remainder test
```

Time taken to build model: 6.18 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.08 seconds

=== Summary ===

Correctly Classified Instances	4130	67.4176 %
Incorrectly Classified Instances	1996	32.5824 %
Kappa statistic	0.5656	
Mean absolute error	0.2243	
Root mean squared error	0.3297	
Relative absolute error	59.8224 %	
Root relative squared error	76.1498 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	0.701	0.140	0.625	0.701	0.661	0.541	0.896	0.672
High	0.481	0.140	0.537	0.481	0.508	0.354	0.804	0.532
Low	0.664	0.107	0.672	0.664	0.668	0.560	0.861	0.702
Very Low	0.853	0.047	0.857	0.853	0.855	0.807	0.966	0.907
Weighted Avg.	0.674	0.109	0.672	0.674	0.672	0.564	0.881	0.702

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1075	433	25	0	a = Very High
501	746	295	8	b = High
96	206	1010	208	c = Low
48	3	173	1299	d = Very Low

### C.3.8. Naïve-Bayes

=== Run information ===

```
Scheme:      weka.classifiers.bayes.NaiveBayes
Relation:    testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:   18019
Attributes:  7
             Rank-quartile
             Download-quartile
             AvgTime-Quartile
             page_views-Quartile
             msg_posted-Quartile
             Total_D_P_S-Quartile
             Total_Nreq-Quartile
Test mode:   split 66.0% train, remainder test
```

Time taken to build model: 0.07 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.38 seconds

=== Summary ===

Correctly Classified Instances	3890	63.4998 %
Incorrectly Classified Instances	2236	36.5002 %
Kappa statistic	0.5135	
Mean absolute error	0.202	
Root mean squared error	0.3567	
Relative absolute error	53.8642 %	
Root relative squared error	82.3777 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Very High	0.763	0.161	0.613	0.763	0.680	0.564	0.885	0.594
High	0.347	0.129	0.476	0.347	0.401	0.244	0.764	0.465
Low	0.616	0.136	0.599	0.616	0.607	0.475	0.831	0.634
Very Low	0.818	0.061	0.817	0.818	0.818	0.757	0.959	0.900
Weighted Avg.	0.635	0.122	0.626	0.635	0.626	0.509	0.859	0.647

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1170	318	45	0	a = Very High
623	538	347	42	b = High
94	253	936	237	c = Low
21	21	235	1246	d = Very Low

### C.3.9. AdaBoost M1

=== Run information ===

```
Scheme:          weka.classifiers.meta.AdaBoostM1 -P 100 -S 1 -I 10 -W
weka.classifiers.trees.DecisionStump
Relation:        testfile4b-weka.filters.unsupervised.attribute.Remove-R1-10,18-32
Instances:       18019
Attributes:      7
                 Rank-quartile
                 Download-quartile
                 AvgTime-Quartile
                 page_views-Quartile
                 msg_posted-Quartile
                 Total_D_P_S-Quartile
                 Total_Nreq-Quartile
Test mode:       split 66.0% train, remainder test
```

Time taken to build model: 0.14 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.02 seconds

=== Summary ===

Correctly Classified Instances	2807	45.8211 %
Incorrectly Classified Instances	3319	54.1789 %
Kappa statistic	0.2757	
Mean absolute error	0.2974	
Root mean squared error	0.3856	
Relative absolute error	79.2955 %	
Root relative squared error	89.0498 %	
Total Number of Instances	6126	

=== Detailed Accuracy By Class ===

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Class	0.000	0.000	0.000	0.000	0.000	0.000	0.664	0.332
Very High	1.000	0.670	0.336	1.000	0.503	0.333	0.665	0.336
High	0.000	0.000	0.000	0.000	0.000	0.000	0.554	0.270
Low	0.825	0.055	0.834	0.825	0.829	0.773	0.885	0.731
Very Low	0.458	0.183	0.292	0.458	0.333	0.276	0.692	0.417
Weighted Avg.								

=== Confusion Matrix ===

a	b	c	d	<-- classified as
0	1533	0	0	a = Very High
0	1550	0	0	b = High
0	1269	0	251	c = Low
0	266	0	1257	d = Very Low