Requirements Eng (2016) 21:553–572
DOI 10.1007/s00766-015-0232-4

ORIGINAL ARTICLE

Systems performance prediction using requirements quality attributes classification

John L. Dargan[1] · James S. Wasek[1] · Enrique Campos-Nanez[1]

Received: 27 August 2014 / Accepted: 19 June 2015 / Published online: 7 July 2015

Abstract Poor requirements definition can adversely impact system cost and performance for government acquisition programs. This can be mitigated by ensuring requirements statements are written in a clear and unambiguous manner with high linguistic quality. This paper introduces a statistical model that uses requirements quality factors to predict system operational performance. This work explores four classification techniques (Logistic Regression, Naïve Bayes Classifier, Support Vector Machine, and K-Nearest Neighbor) to develop the predictive model. This model is created using empirical data from current major acquisition programs within the federal government. Operational Requirements Documents and Operational Test Reports are the data sources, respectively, for the system requirements statements and the accompanying operational test results used for model development. A commercial-off-the-shelf requirements quality analysis tool is used to determine the requirements linguistic quality metrics used in the model. Subsequent to model construction, the predictive value of the model is confirmed through execution of a sensitivity analysis, cross-validation of the data, and an overfitting analysis. Lastly, Receiver Operating Characteristics are examined to determine the best performing model. In all, the results establish that requirements quality is indeed a predictive factor for end-system operational performance, and the resulting statistical model can influence requirements development based on likelihood of successful operational performance.

✉ John L. Dargan
jdargan40@gmail.com

[1] School of Systems Engineering and Applied Science, George Washington University, Washington, DC 20052, USA

Keywords Poor requirements · Requirements definition · Requirements engineering · Requirements quality attributes · Natural language requirements

1 Introduction

1.1 Problem statement

Requirements definition and quality have historically been problematic areas within the systems engineering process [1]. This is further compounded as requirements statements are more and more constructed in natural language sentences, leading to increased potential for ambiguity and misinterpretation. There is ample research in the technical literature, indicating that errors, gaps, and vagueness in requirements contribute to system deficiencies, incomplete system test plans, and unsatisfactory system performance. In addition, a litany of Government Accountability Office (GAO) reports have been written highlighting the preponderance of poor requirements development and management in government acquisition programs, and, moreover, this has been such a significant issue that the 2009 Weapon Systems Acquisition Reform Act explicitly requires the Department of Defense (DoD) to address and improve its performance requirements. Despite the well-documented and widespread acknowledgment that poor requirements quality leads to "downstream" issues with defects and performance, the problem remains. This problem, however, could be better managed if there were a means to predict the probability of successful end-system operational performance following requirements development. This would enable quick identification of deficient requirements needing remedy based on their adverse impact on performance. Hence, this research addresses the hypothesis that

end-system operational performance can be determined through use of predictive modeling based on requirements quality factors.

1.2 Related research

The current technical literature is replete with discussion regarding the challenges with natural language requirements and the need for metrics and tools to assess requirements quality. The following are three specific areas within the body of knowledge related to the research presented in this paper:

- Requirements quality defects.
- Ambiguity identification tools.
- Defect prediction techniques.

1.2.1 Requirements quality defects

Park et al. [2] discuss the costly nature of software rework based on requirements defects and note as much as 85 % of software defects originate with ambiguous and incomplete requirements. Moreover, the cost of fixing requirements-related defects can be up to 50 times greater if the defects are corrected in the latter development stages.

Fantechi et al. [3] discuss the inherent ambiguity in natural language requirements and propose that associated linguistic defects can be categorized into three areas: expressiveness (issues with the understanding of use cases by humans), consistency (presences of semantics contradictions and structural incongruities), and completeness (lack of necessary contents). The authors' research shows deriving metrics to assess requirements ambiguity mitigation can be achieved using natural language processing techniques. However, these techniques are not sufficient to address the other defect areas of consistency and completeness of natural language requirements.

In a similar note, Nigam et al. [4] identify three types of ambiguity as potential sources of misinterpretation: lexical (a word has several meanings), syntactic (use of prepositional phrases that enable phrases to be connected in several ways to form a correct sentence), and syntax (omission of proper punctuation). The authors assessed four open-source software requirements specifications against the three types of ambiguity and determined that the preponderance of ambiguities is syntactic due to use of vague words in the requirements statements.

Yang et al. [5] assert that while natural language requirements lead to unwanted ambiguities, not all ambiguities lead to misinterpretations by users and stakeholders. Thus, the authors propose focusing on anaphoric ambiguity (occurs when readers disagree on how pronouns should be interpreted) as a key requirements quality defect that is commonly found in requirements specifications and can readily lead to misinterpretations by various users. The authors also developed a Naïve Bayes Classifier to predict when anaphoric ambiguity will lead to misunderstanding.

The software test community recognizes the importance of natural language requirements quality and offers that linguistic analysis and testing techniques could be combined to help testers better understand the relations between natural language requirements and improved test planning [6, 7]. Moreover, redundancies and implicit relationships in requirements specifications can be exploited to reduce the total testing effort against the requirements specification.

1.2.2 Ambiguity identification tools

Gonzalo et al. [6] suggest addressing requirements quality issues on the front end of the development process is necessary for improved software quality. Therefore, they proposed development of an automated tool to obtain low-level quality indicators in textual requirements, such as ambiguity and overlap, to identify requirements defects and enable higher quality requirements to be drafted. They also reference an independent survey of requirements quality tools that ranked the authors' tool as the most promising of eight available tools.

Lami and Ferguson [7] emphasize there is little evidence in the literature that the techniques and tools available for assessing natural language requirements have been empirically validated. Consequently, their research aims to empirically show that an automated natural language requirements analysis tool is more advantageous when compared to a manual review process by human subject matter experts. The authors conducted an empirical study involving a real-world industrial project led by a global telecommunications company that performs analysis of natural language requirements documents through use of external consultants that identify and report defects. The study results show that use of an automated tool can significantly improve the requirements analysis process. However, human inspections cannot be fully replaced and should be complemented by the automation.

Gnesi et al. [8] address the topic of automation of the analysis of natural language requirements and provide a specific exploration of one such analysis tool called Quality Analyzer for Requirement Specifications (QuARS). The authors' methodology entails using the tool in case studies for the automotive, banking, and aerospace domains to empirically illustrate the benefits the tool affords automated natural language requirements analysis. The study results demonstrate that QuARS can provide effective support for the natural language requirements analysis and is highly adaptable to different application domains.

Kiyavitskaya et al. [9] propose a tool that uses a two-step approach to addressing requirements specification ambiguities where the first step identifies those requirements statements with potential ambiguity and the second step provides the specific sources of ambiguities in the flagged ambiguous statements. The authors tested the tool using several small requirements specifications and compared the results to a human review by a subject matter expert. They concluded the tool, although promising, did not accurately identify some of the severe ambiguity issues in the specifications. Based on the results, the authors state they are updating the tool for better ambiguity identification performance.

1.2.3 Defect prediction techniques

As discussed in [2], the authors use regression analysis to develop a model to predict defects based on requirements specification attributes before initiating code development. The results illustrated there are requirements attributes with a statistically significant correlation with the number of defects.

Bibi et al. [10] propose use of Regression via Classification to predict software defects. Their intent is to exploit the benefits of classification algorithms and mitigate the issues of incomprehensibility associated with regression algorithms. The research showed that Regression via Classification could provide a means for identifying causes of faults in addition to the relationships between attributes and defects.

Tian and Noore [11] describe use of an innovative Support Vector Machine application to software failure time for reliability prediction. Their schema involves the network learning and recognizing software failure sequencing. Experimental results show the Support Vector Machine application is effective for next-step-predictability and out performs current neural network approaches.

Malhotra and Jain [12] conducted a literature search on fault detection prediction to determine how broadly statistical and machine learning methods were utilized. They observed that the trend is moving from traditional statistical methods to machine learning approaches, and evermore studies are providing evidence that machine learning approaches provide better classification performance compared to statistical methods.

Rawat and Dubey [13] performed a literature study on software defect prediction models to characterize the respective advantages and disadvantages of the regression and machine learning models. While they concluded that prediction models depend on the volume of available defect data and accuracy of the classifier, they acknowledged machine learning techniques have an appeal based on their support for model calibration and classification accuracy.

Overall, the contemporary technical literature contains numerous discussions on natural language quality defects, automated tools for requirements analysis, and an inventory of software defect and reliability prediction methods. However, there is limited discussion on the impact of requirements linguistic quality on system-level operational performance. The research presented in this paper leverages current ambiguity identification tools and defect forecasting techniques for application to system performance prediction. This effort is intended to bridge the gap in the prevailing body of knowledge by providing empirical evidence of the predictive relationship between linguistic requirements quality and end-system operational performance.

1.3 Contribution summary

This research on the predictive relationship between requirements quality and system performance provides the following major contributions:

- Predictive Modeling Development Methodology.
- Statistical Significance of Requirements Quality Relationship to System Performance.
- Comparison of alternative Machine Learning modeling results.
- Additional Areas of Research for Predictive Modeling.

The remaining sections of this paper are outlined as follows: Sect. 2 introduces the methodology for the predictive model construction and validation based on empirical data from current federal government acquisition programs and describes the data and the tools used for requirements and statistical analysis. In Sect. 3, four predictive models are established. Binary Logistic Regression is employed first followed by three machine learning techniques. These techniques include a Naïve Bayes Classifier, Support Vector Machine, and K-Nearest Neighbor. In Sect. 4, results from Binary Logistic Regression are then compared to the machine learning techniques to determine best predictive performance. This comparison includes model validation, sensitivity analysis, assessment of model generalization, and examination of Receiver Operating Characteristics. Finally, summary and conclusions are discussed in Sect. 5.

2 Methodology

2.1 Overview

Statistical modeling based on empirical performance data is the approach used to address the research hypothesis presented in this paper. The model development and

validation methodology involve the following sequential steps as illustrated in Fig. 1.

1. Obtain Operational Requirements Document (ORD) Key Performance Parameters (KPPs) and Operational Test Report results from DoD and Department of Homeland Security (DHS) acquisition databases.
2. Analyze KPPs for linguistic quality.
3. Develop predictive model using the KPP linguistic quality metrics and the associated operational test results.
4. Conduct sensitivity analysis for classification accuracy.
5. Validate model for predictive performance.
6. Assess model for data overfitting: a condition that constructs a statistical model with too many degrees of freedom and results in overoptimism in the prediction accuracy of the model.
7. Evaluate model performance based on Receiver Operating Characteristics.

2.2 Data

A total of 242 KPP requirements statements and associated operational test results served as the data used for development and validation of the model. It should be acknowledged that while this data set size is considered small for classification purposes, it is believed sufficient given the nature of the problem addressed in this research. The data were obtained from DoD and DHS major acquisition program ORDs and Operational Test Reports. A sample of the data used in this research is provided in Table 1. Five of the 242 KPP requirements statements are included in this table along with the associated results of the linguistic requirements analysis and the operational test performance results. Using KPP number 1 in the table as an exemplar, the operational test performed by the federal government revealed the performance for this requirement was met; however, the linguistic analysis performed in this research showed the requirement had two "Vague" and one "Compound Requirements" linguistic quality defects—note, further description of the linguistic quality factors is provided in Sect. 2.3. This linguistic analysis methodology was replicated for all 242 KPP requirements statements and served as inputs for construction of the predictive model.

Lastly, an aggregate profile of the data used for this research is captured in Table 2 and shows the majority of

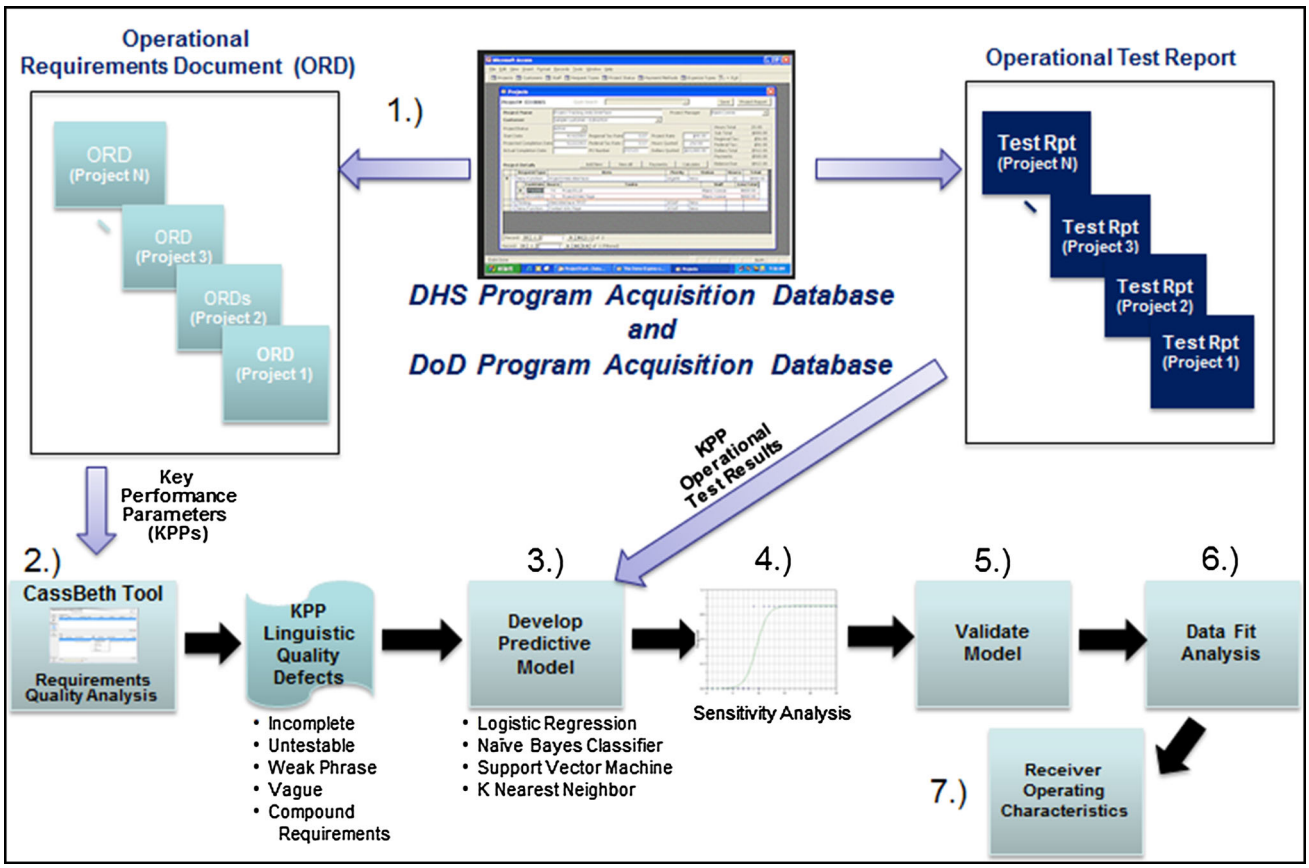

Fig. 1 Methodology overview

Table 1 Sample requirements and operational test results data

KPP no.	KPP requirement statement (documented in operational requirements document)	KPP linguistic quality defects (CassBeth tool analysis results)					KPP operational test results (documented in government test report)
		Incomplete	Untestable	Weak phrase	Vague	Compound requirements	KPP performance met
1.	All aircraft modifications shall meet all FAA Supplemental Type Certificate and airworthiness requirements	0	0	0	2	1	Yes
2.	The selected radar must be able to detect a maneuvering, all aspect, target, approximately the size of a Cessna 172 in a look-down, medium clutter environment at a distance of 50 nm	0	0	1	1	0	No
3.	The HPA Sensor and Cockpit Modernization shall replace analog flight instruments with a fully STC compliant Electronic Flight Instrumentation System (EFIS) capable of, and integrated to insure correct display of all sensor, flight, navigation, and associated data	0	0	0	1	3	Yes
4.	The HPA Sensor and Cockpit Modernization program shall replace the APG-66 radar installed with a more capable, fully digital, multi-mode radar weighing no more than the APG-66 and fully compatible with the HPA's existing cooling and electrical power generation specifications (T)	0	0	0	0	3	Yes
5.	The HPA Sensor and Cockpit Modernization program shall replace the AN/AAS-36 IR system with an EO/IR system combining a laser rangefinder (T), NVG compatible laser pointer/illuminator (O), CCD TV (T), Camera Spotter scope (O), and infrared camera (T) weighing no more than the currently installed EO/IR payload (T)	0	0	0	0	2	No

Table 2 Data profile for predictive model development

Agency	DHS (%)	DoD (%)	DHS + DoD (%)
Program type			
Information technology	32	2	34
Aircraft programs	14	25	39
Munition programs	0	16	16
Ship programs	2.5	0	2.5
Satellite programs	0	2.5	2.5
Other programs	6	0	6
Total	54.5	45.5	100

data were sourced from DHS information technology programs and DoD aircraft programs.

2.3 Model construction and validation

A predictive model is developed to demonstrate that end-system operational performance can be projected based on the requirements linguistic quality factors as defined below:

- Incomplete—requirement statement does not fully capture requirement parameters.
- Untestable—requirement statement lacks an objective means to verify requirement has been satisfied.
- Weak Phrase—Unquantified or indefinite requirements statement.
- Vague—requirement statement has a continuum of possible interpretations.
- Compound Requirements—statement contains multiple needs.

The linguistic quality factors listed above, while not exhaustive, represent the most common defective attributes of the requirements statements. They have been selected for use in this research since elimination of these defects is generally accepted as characteristics of good requirements [14].

2.3.1 Modeling approaches overview

Several machine learning approaches such as regression analysis, clustering, and classification are available for

developing the predictive model [15, 16]. Given (1) the requirements linguistic quality factor data listed above are categorical and serve as independent or explanatory variables for the model, and (2) the dependent response variable, "operational performance met," is dichotomous, a classification approach would be most appropriate [17]. As such, four machine learning algorithms within the classification domain are chosen to develop predictive models and compare performance results. The following criteria are used to select the algorithms:

- Fast classification speed.
- Reduced danger of data overfitting.
- Ability to handle smaller training set.
- High model accuracy.

Based on the above criteria, Binary Logistic Regression (BLR), Naïve Bayes Classifier (NBC), Support Vector Machine (SVM), and K-Nearest Neighbor (KNN) machine learning techniques are selected [18–20]. Note, while BLR is a more traditional statistical modeling technique, as per Wu's data mining survey [21], NBC, SVM, and KNN are high-performing, modern algorithms considered to be within the top 10 most influential for classification. An overview of each algorithm is provided in the subsequent paragraphs.

Binary Logistic Regression is a statistical classification model that predicts the probability of an outcome of a binary response variable based on one or more explanatory input variables. The Logistic Regression model can be expressed as follows:

$$\text{Logit}(p) = a + bx$$

- p is probability.
- Logit (p) is the natural logarithm of the odds ratio, $\frac{p}{1-p}$.

BLR effectively handles correlated input variables and provides a probabilistic framework that enables quick incorporation of more training data as it becomes available. The underlying distribution of the BLR model is binomial, and the parameters are estimated using the method of maximum likelihood.

The Naïve Bayes Classifier leverages Bayes' theorem and assumes independence among the predictor variables. NBC is a relatively simple model that often outperforms more sophisticated models. The NBC algorithm is expressed as follows:

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

- $P(c|x)$ is the posterior probability of class given predictor.
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

The Support Vector Machine performs classification by determining the hyperplane that maximizes the distance between the two classes. This technique yields one of the most robust and accurate classification algorithms. The SVM attempts to maximize the following function with respect to **w** and b:

$$Lp = \frac{1}{2} \| \boldsymbol{w} \| - \sum_{i=1}^{t} \alpha_i y_i (\boldsymbol{w} \cdot \boldsymbol{x} + b) + \sum_{i=1}^{t} \alpha_i$$

- Lp is the Lagrangian.
- α_i are nonnegative numbers such that derivatives of Lp with respect to α_i are zero.
- t is the number of training samples.
- $\boldsymbol{w}$ and b define the hyperplane.

K-Nearest Neighbor is one of the simplest and easy-to-implement machine learning algorithms, yet has good performance, especially as the size of the training set increases. KNN finds a group of K objects in the training set that are closest to the test object. The class assigned to the test object is based on the majority of its nearest neighbors. The algorithm uses one of several distance functions (Euclidean, Euclidean squared, city block, and Chebychev) to determine the K-Nearest Neighbors. The Euclidean distance function is shown below:

$$\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

- k is the number of neighbors.
- x_i is the query point.
- y_i is a case from the training sample.

2.3.2 Predictive model development process

Each of the four approaches described in the preceding section is employed to develop and validate their respective predictive models as described below:

1. Use of empirical data is critical for development of the predictive model. As such, the DHS and DoD program acquisition databases are accessed to provide the ORD KPPs and the KPP operational test results that serve as the data utilized for this research. A total of 242 key performance parameter requirements statements and the associated operational test results are used for model creation.
2. These requirements statements are imported into an automated requirements analysis tool to calculate the linguistic quality metrics for each statement. In

addition, the operational test reports associated with each ORD are reviewed to determine whether performance was met for each key performance parameter used in the model.

3. These data are then used to determine the classifier, or model, for predicting system performance. Four approaches are utilized. First, a Binary Logistic Regression model is developed. Using the same data, a Naïve Bayes Classifier is developed followed by a Support Vector Machine and K-Nearest Neighbor approach.
4. Next, a sensitivity analysis is conducted on each of the classifiers using the following parameters:

   - Number of independent variables.
   - Training Data Sample Size.
   - Distance Measure (KNN only).
   - Number of Nearest Neighbors (KNN only).

   The objective of this analysis is to assess how these parameters impact classification accuracy [22]. This allows determination of the amount of change in model behavior given a change in the parameter as well as which parameter has the most substantive impact on performance. The results of the analysis are used to adjust or calibrate the classifier as necessary.

5. K-fold cross-validation [23, 24], a common technique for confirming predictive performance, is used to assess how the classifier will generalize to an independent data set. This method involves dividing the data into K subsets and executing the cross-validation over K iterations. One of the K subsets will be used as the test set, and the other K-1 subsets will form the training set for each iteration. The value of K is set to 10 for this analysis. The advantage of this approach is that it does not depend heavily on which data points are in the training set and which ones are in the test set.
6. Subsequent to cross-validation, an analysis is conducted to determine how well the data fit the model, with particular emphasis on overfitting. This analysis will compare the training data set performance to test data set performance to determine the point where test set accuracy reaches a steady-state value. Overfitting constructs a statistical model with too many degrees of freedom in the process and results in overly optimistic performance. In other words, while the data are well described in the model, the predictions do not generalize to new data outside the study sample [25].
7. Lastly, Receiver Operating Characteristics (ROC) are examined for each classifier. ROC graph examination allows visualization of the trade-off between successful classification and false alarm rate for each of the classifiers [26]. In general, the classifier located in the most northwest position on the ROC graph has the better performance.

2.4 Analysis tools

Requirements analysis can be accomplished through expert judgment, techniques such as case-based reasoning [27], or use of automated analysis tools [28, 29]. According to a recent empirical study completed by Lami [7], automated requirements analysis tools can be more effective than human reviews for finding defective requirements. Hence, the research presented in this paper utilizes an automated tool to determine linguistic quality metrics. As described by Lami [7], there are numerous requirements analysis tools such as the QuARS, Requirements Quality Analyzer (RQA), TigerPro, and the CassBeth Specification Analysis Tool. A feature comparison of the aforementioned analysis tools illustrated in Table 3 shows all have pertinent linguistic analysis capability, utilize common input file formats, and can quickly produce analysis results. While a government or commercial organization responsible for large development projects with a substantial amount of requirements specifications may want to consider a tool such as RQA, the focused scope of this research lends itself to employment of the CassBeth tool. Specifically, the CassBeth Specification Analysis Tool was selected for use in this research based on the following:

- Simple Web interface that enables access anywhere.
- Accepts Microsoft Word or text file inputs.
- Adjustable analysis parameters.
- Low purchase cost.

CassBeth [30] provides an automated method to review requirements specifications and identify linguistic faults in the requirements statements. These faults include poorly worded statements that are Vague, Untestable, Weak Phrases, Compound Requirements, and Incomplete. In addition, the tool contains a complete service library with the following options:

- Requirements Text Analysis.
- Key Requirements Analysis.
- Duplicate Objects Analysis.
- Domain Capabilities Analysis.
- Generic Capabilities Analysis.

The research discussed in this paper uses the CassBeth tool's Requirement Text Analysis service to assess linguistic quality of the key performance parameter statements found in the DHS and DoD ORDs. This service utilizes a default rule set to analyze word and phrase

Table 3 Requirements analysis tools feature comparison

Features	Quality analyzer of requirement specifications (QuARS)	Requirements quality analyzer (RQA)	TIGER Pro	CassBeth specification analysis tool
Lexical analysis	✔	✔	✔	✔
Syntactical analysis	✔	✔	✔	✔
Statistical analysis	✔	✔	✔	✔
Graphical analysis	✔	✔	✔	×
Adjustable analysis rules	✔	✔	×	✔
Real-time analysis results	✔	✔	✔	✔
Defect summary report	✔	✔	✔	✔
Dynamic object oriented requirements system (DOORS) interface	×	✔	✔	×
Input file type	.txt, .dat	.xls, DOORS	.txt, csv, DOORS	.txt, .doc
HW/operating system platform	Not specified	Client/Server Win 2008/Win 7	PC Windows 7	PC Web Interface
Software purchase cost	Not commercially available	$500+	$50	$20

patterns that typically result in poorly worded requirements statements. However, the default rule set can be adjusted if necessary to encompass user-specified instructions. Following the import of the key performance parameter statements, the tool generates a report that identifies the linguistic quality defects, if any, in each statement. The tool can enumerate the number of instances of the identified defects in the statements, and it can also treat the defects as binary. In other words, each key performance parameter statement has a quality metric (Vague, Weak Phrase, Compound Requirements, Untestable, Incomplete) that is either defective or non-defective. In turn, these continuous and binary results serve as the independent variables for the predictive model. A screen shot of the CassBeth tool user interface control panel is shown in Fig. 2. The control panel allows the user to select a file to input for analysis, choose the service library for the desired analysis, run the analysis, and view the analysis results. In addition, a representative screen shot of a linguistic quality metrics report for data utilized in this research is provided in Fig. 3.

The data analysis for this research is facilitated through use of the STATISTICA statistical software tool. Requirements linguistic quality metrics determined by the CassBeth Specification Analysis Tool are input into STATISTICA for statistical analysis. Specifically, STATISTICA performs the Binary Logistic Regression on the metrics data to determine the linear predictor model, statistical significance, and the odds ratio. STATISTICA is also used to perform the NBC, SVM, and KNN analyses to develop and validate the respective classifiers, conduct the sensitivity analysis, and perform data fit analysis.

3 Results

3.1 Linguistic analysis results

As described in Sect. 2.2 of this paper, the predictive model is developed based on the linguistic quality factors (Incomplete, Untestable, Weak Phrase, Vague, and Compound Requirements) of the 242 KPP requirements statements. Based on the results from the CassBeth Specification Analysis Tool, there were a total of 284 linguistic defects identified in the KPP requirements statements. The graph in Fig. 4 depicts the distribution of linguistic quality defects in the requirements statements. It should be noted that some of the requirement statements had more than one type of linguistic quality defect. None of the requirements statements were assessed as Incomplete, and there were only two instances of Untestable requirements statements. The majority of the defects were assessed as Weak Phrase, Vague, and Compound Requirements, where Compound Requirements was the largest single linguistic defect with 210 instances.

The linguistic quality defects distribution is also shown for each of the acquisition program types: information technology, aircraft, munitions, ships, satellites, and other, respectively, in Figs. 5, 6, 7, 8, 9, and 10. Note, the information technology, aircraft, munitions, and other programs reflect a similar defect profile to the overall profile shown in Fig. 4. The ship programs are the only program type having more linguistic defect instances of Weak Phrase and Vague than Compound Requirements, and the satellite programs also have higher instances of vague linguistic defects compared to weak phrase defects.

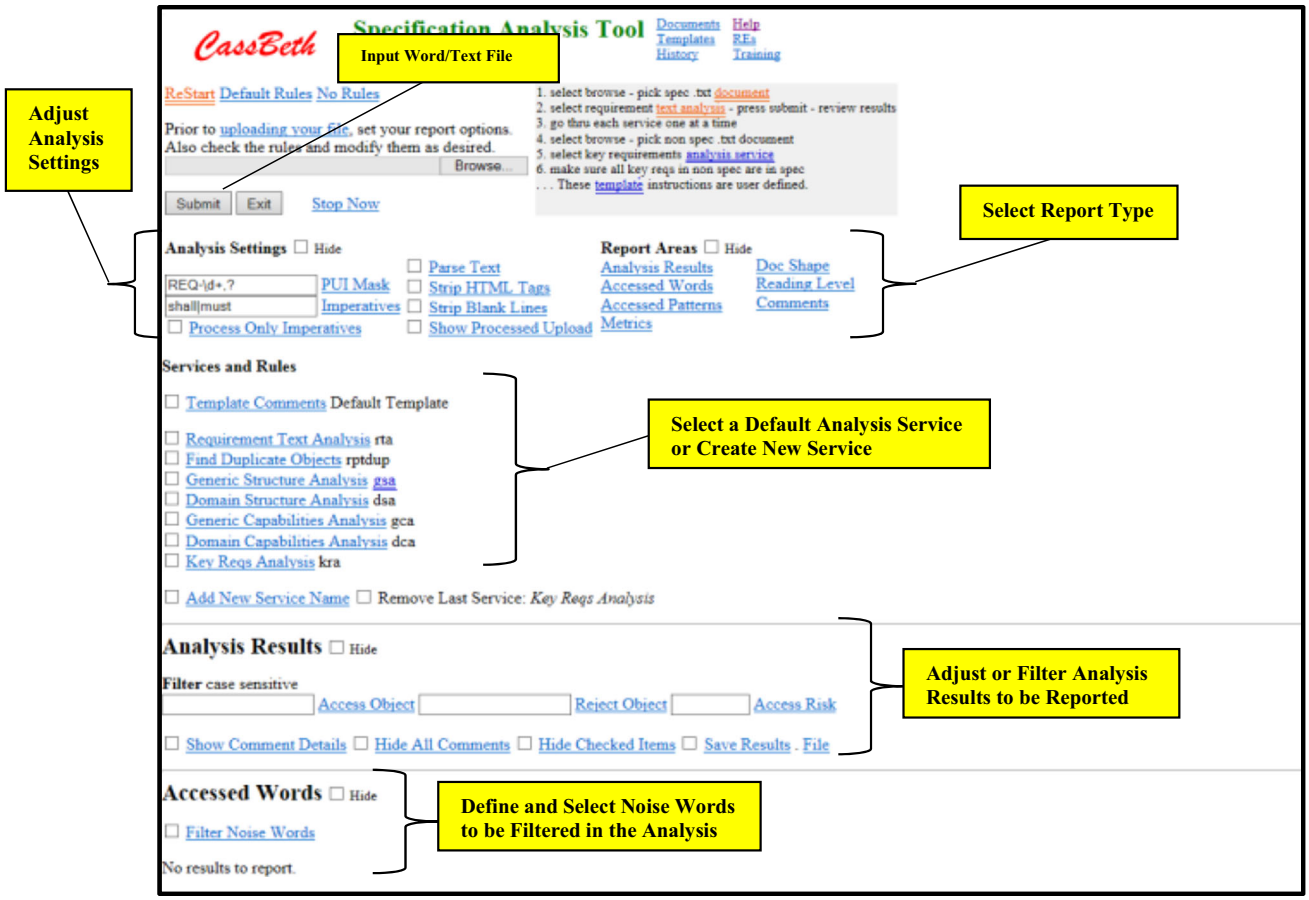

Fig. 2 Specification analysis tool user interface control panel

3.2 Predictive modeling results

As described in Sect. 2.3, four predictive models are established to describe the relationship between requirements linguistic quality attributes and end-system operational performance. Binary logistic Regression is employed first followed by three machine learning techniques: Naïve Bayes Classifier, Support Vector Machine, and K-Nearest Neighbor. Results for each model are described in the succeeding paragraphs.

3.3 Binary Logistic Regression

STATISTICA 12.0 was employed to execute Binary Logistic Regression on the KPP linguistic quality attributes and the associated operational test results. Recall, five linguistic quality attributes (Incomplete, Untestable, Weak Phrase, Vague, and Compound Requirements) are used in the analysis. These attributes are the independent variables for the model, and the associated operational test results serve as the dependent variable, Performance Met. An iterative attribute elimination process was used to determine the statistically significant contributing attributes, where statistical significance is defined as ρ value less than or equal to 0.05 for this analysis. The following statistics are key indicators of the quality of the Logistic Regression model and were assessed as part of the attribute elimination process:

Wald Statistic—this statistic has a Chi-square distribution and is used to assess statistical significance of the attributes, or independent variables. If significance values are less than 0.05, the null hypothesis should be rejected as the attribute does make a significant contribution [31].

Hosmer–Lemeshow Goodness-of-Fit Statistic—this statistic indicates how well the model prediction fits the observed data. If the Hosmer–Lemeshow goodness-of-fit test statistic is greater than 0.05, as needed for well-fitting models, the null hypothesis that there is no difference between observed and model-predicted values should be rejected. This implies the model's estimates fit the data at an acceptable level [32].

Nagelkerke R^2—this descriptor has a value from zero to one and indicates the amount of variation in the dependent variable that is explained by the logistic model [28]. A number closer to one indicates a strong relationship between the predictors and the prediction.

Metrics Hide

Save Metrics with analysis run Test Spec_4.txt 05/21/014 22:22:45 Appended Metrics File

Total Lines: 232
Blank Lines: 65
Non Blank Lines: 167
Imperatives: 100
Shalls: 94
Wills:
IsReq:

Message: This document is 59% imperatives suggesting you have a specification. But, you should move some of the descriptive text to your analysis or design documents.

Item	Risk	Count	Children	% lines	% imperative	% shall	% will	% isreq	% Norm
Buzz Words rta	High								
Compound Req rta	Low	60		25.86	60	63.82			
Directive rta	Low	4		1.72	4	4.25			
Fragment rta	Low								
Incomplete rta	High	1		0.43	1	1.06			
Internal Reference rta	High	3		1.29	3	3.19			
Multiple Imperatives rta	Med								
Not Standalone rta	Low								
Options rta	High	1		0.43	1	1.06			
Unbounded rta	Low	9		3.87	9	9.57			
Undefined rta	Low	3		1.29	3	3.19			
Unsure rta	High								
Untestable rta	High								
Weak Phrases rta	Med	9		3.87	9	9.57			
Weak Words rta	Med	11		4.74	11	11.7			
z Mined Objects		72		31.03	72	76.59			

Rules Total 16
Rules Triggered 10
Rules Not Triggered 6
Percent of Rules Triggered 62%

Fig. 3 Specification analysis tool sample metrics report

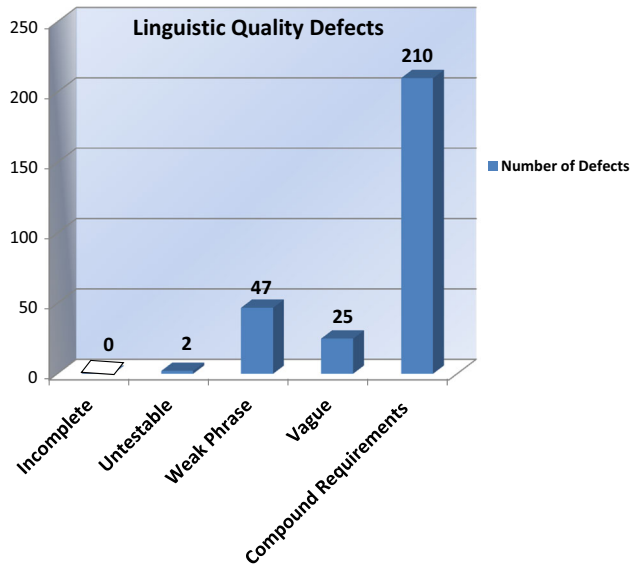

Fig. 4 Linguistic quality defects graph (overall)

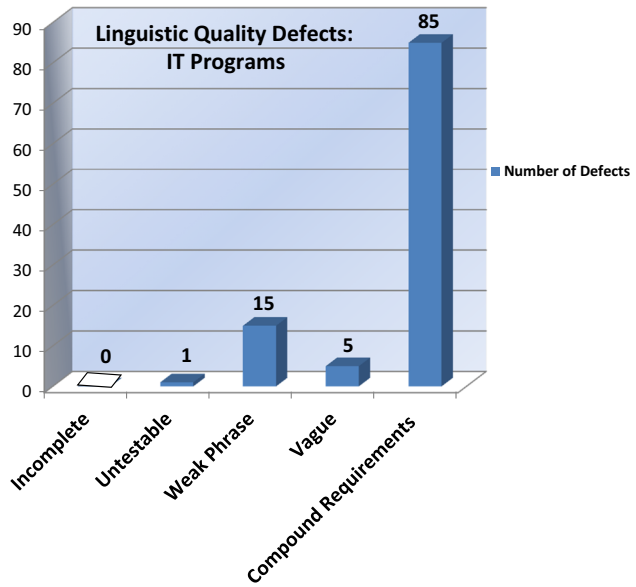

Fig. 5 Linguistic quality defects—IT programs

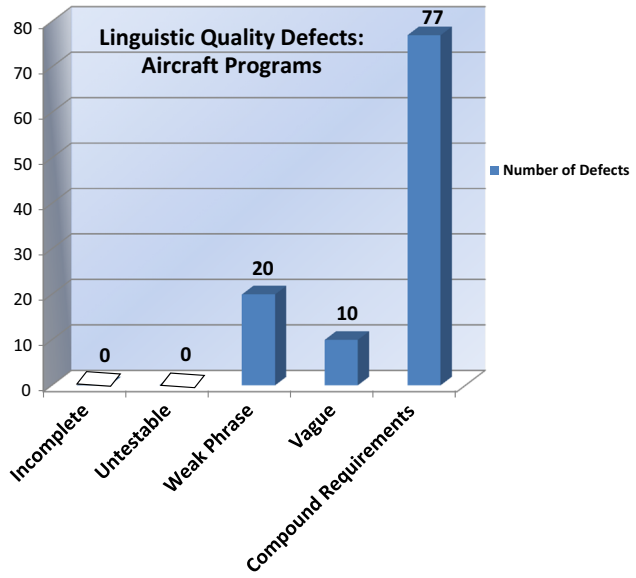

Fig. 6 Linguistic quality defects—aircraft programs

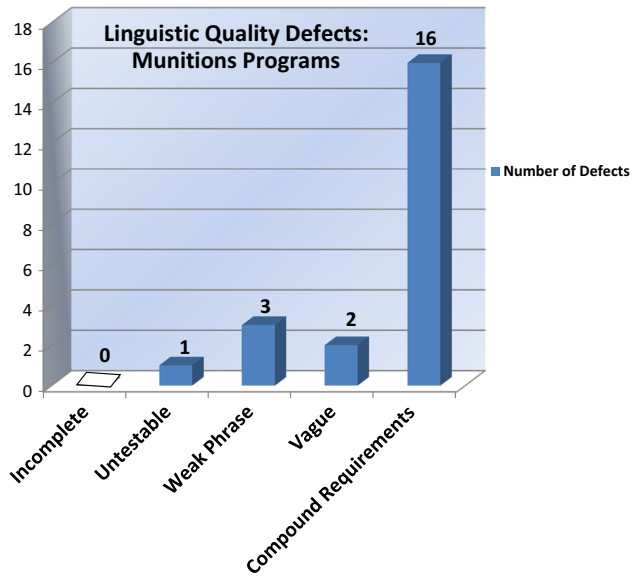

Fig. 7 Linguistic quality defects—munitions programs

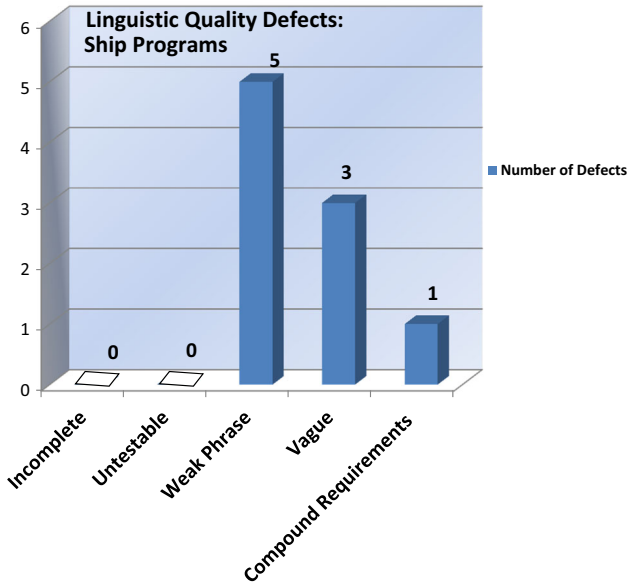

Fig. 8 Linguistic quality defects—ship programs

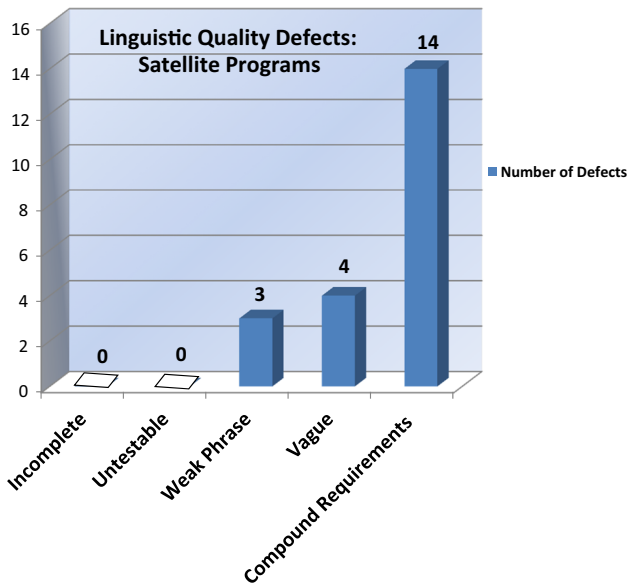

Fig. 9 Linguistic quality defects—satellite programs

Odds Ratio—this ratio defines the level of influence the attributes, or independent variables, have on the output [33].

Table 2 presents the attribute statistics for the initial Logistic Regression model. Note, the "Incomplete" attribute was removed from the initial model since there were no instances of "Incomplete" defects in the empirical data used for this analysis. Use of the remaining four attributes in the Logistic Regression model shows that only one attribute, Compound Requirements, has significance, although the overall classification accuracy is 74 % for the validation data and 79 % for the training data as shown in Tables 4 and 5, respectively. In addition, the Nagelkerke R^2 value of 0.106031 indicates only about 11 % of the variation in the dependent variable is explained by the logistic model. This indicates the initial attributes do not provide a statistically significant, well-fitting Logistic Regression model. Henceforth, attributes are eliminated one by one in an effort to find a statistically significant, effective model. The interim analysis results are not presented in this paper for the purpose of brevity. However, the analysis culminates with a model containing two attributes, Vague and Compound Requirements, as shown in Table 6. In particular, the model has significance values

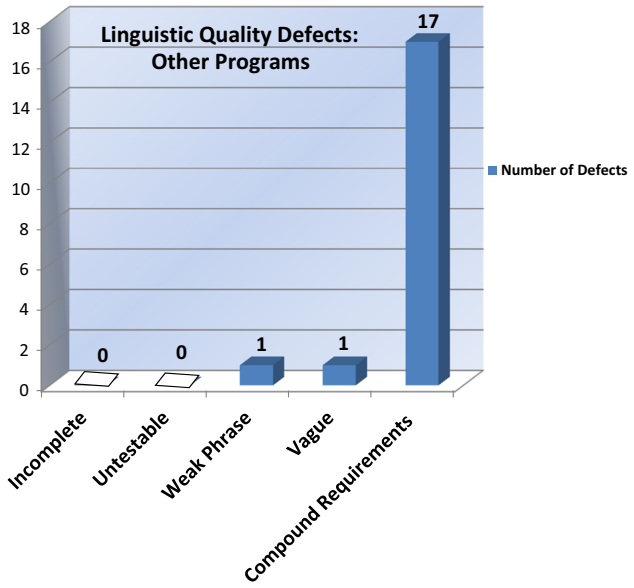

Fig. 10 Linguistic quality defects—other programs

of 0.0478 and 0.006, respectively, for the Vague and Compound Requirements attributes. The Nagelkerke R^2 value of 0.986 indicates there is a strong relationship between the predictors and the prediction, and the Hosmer–Lemeshow p value of 0.1874 (>0.05) indicates this is a well-fitting model. Lastly, as depicted in Tables 7 and 8, the model has an overall classification accuracy of 74 % against the validation data and 78 % against the training data. The final logistic model is shown in Table 9 as follows:

$$\text{Logit}\ (p) = CR^*\,(-0.256) + V^*\,(-0.504) + 1.129$$

where P probability performance is met, CR Compound Requirement, V Vague Requirement

A sensitivity analysis varies one model parameter while holding the other parameters constant, and this is repeated for each parameter in the model. The objective is to assess how each parameter impacts classification accuracy. This sensitivity analysis is conducted against two parameters for the Logistic Regression model: the number of independent variables and the training data sample size. Note, the "Incomplete" attribute was not observed in the empirical data. Therefore, the sensitivity analysis for number of independent variables was conducted on the remaining four attributes (Untestable, Weak Phrase, Vague, and Compound Requirements). The sensitivity analysis results are shown in Tables 10 and 11. As reflected in the tables, the model is insensitive (has little or no change) to the number of attributes and training sample size with respect to classification accuracy.

Next, a data fit analysis is conducted to assess how well the model predictions generalize to new data. The goal is to ensure that the model is not overfitted: a condition that can lead to overoptimism in the predictive accuracy of the model. The plot shown in Fig. 11 compares the training set classification accuracy and the test set classification accuracy. The optimal amount of training occurs at the point test classification accuracy reaches steady state while the training classification accuracy continues to improve [30]. If training is continued at this point, overfitting occurs and the generalization of the Logistic Regression model is decreased [22]. This point occurs at 30 training samples for the final Logistic Regression model. Note, as per Hosmer–Lemeshow [35], there should be 10–20 samples per inde-

Table 4 Initial logistic regression model

Effect	Performance met—parameter estimates Distribution: BINOMIAL, link function: LOGIT Modeled probability that performance met = yes							
	Statistic	Estimate	SE	Wald statistic	Odds ratio	Lower CL 95.0 %	Upper CL 95.0 %	p value
Intercept		1.16425	0.286724	16.48775				
Untestable		−1.16425	1.482462	0.61677	0.312158	−2.59341	3.217730	0.432251
Weak Phrase		0.13604	0.359457	0.14323	1.145726	0.44120	1.850249	0.705092
Compound requirements		−0.28620	0.115733	6.11559	0.751109	0.52428	0.977942	0.013399
Vague		−0.47608	0.258438	3.39346	0.385908	−0.12062	0.892438	0.065455
Scale		1.00000	0.000000	1.000000				
Nagelkerke R^2	0.106031							
Hosmer–Lemeshow	1.2838							0.732973
Loglikelihood	−99.42647							

Table 5 Classification accuracy (test)—initial logistic regression model

Test data	Classification of cases Odds ratio: 0.000000 Log odds ratio: infinity		
	Predicted: yes	Predicted: no	Percent correct
Observed: yes	31	1	96.875
Observed: no	10	0	0
Overall classification accuracy			73.809

Table 6 Classification accuracy (training)—initial logistic regression model

Training data	Classification of cases Odds ratio: 6.333333 Log odds ratio: 1.845827		
	Predicted: yes	Predicted: no	Percent correct
Observed: yes	152	3	98.0645161
Observed: no	40	5	11.1111111
Overall classification accuracy			78.50

Table 7 Final logistic regression model

Effect	Performance met—parameter estimates Distribution: BINOMIAL, link function: LOGIT Modeled probability that performance met = yes							
	Statistic	Estimate	SE	Wald statistic	Odds ratio	Lower CL 95.0 %	Upper CL 95.0 %	p value
Intercept		1.129429	0.278965	16.39151				
Compound requirements		−0.255608	0.094214	7.36070	0.774445	0.589789	0.959101	0.006666
Vague		−0.504189	0.254816	3.91502	0.364810	−0.134620	0.864240	0.047856
Scale		1.000000	0.000000		1.000000			
Nagelkerke R^2	0.986491							
Hosmer–Lemeshow	1.7378							0.187414
Loglikelihood	−24.11865							

Table 8 Classification accuracy (test)—final logistic regression model

Test data	Classification of cases Odds ratio: 0.000000 Log odds ratio: infinity		
	Predicted: yes	Predicted: no	Percent correct
Observed: yes	31	1	96.875
Observed: no	10	0	0
Overall classification accuracy			73.809

pendent variable. Since the final Logistic Regression model is comprised of two explanatory variables, 30 samples are within the recommended range.

3.3.1 Naïve Bayes Classifier

Next, an NBC is developed applying the same attribute data used to develop the Binary Logistic Regression model in the previous section. STATISTICA 12.0 was also employed to develop the NBC based on the KPP linguistic quality attributes and the associated operational test results. The NBC results using four attributes (Untestable, Weak Phrase, Compound Requirements, and Vague) and 200 training samples reveal a classification accuracy of 86 % using test data and 77 % using training data. A sensitivity analysis against the number of independent factors and

Table 9 Classification accuracy (training)—final logistic regression model

Training data	Classification of cases Odds ratio: 4.943089 Log odds ratio: 1.597991		
	Predicted: yes	Predicted: no	Percent correct
Observed: yes	152	3	98.0645161
Observed: no	41	4	8.889
Overall classification accuracy			78.00

Table 10 Logistic regression sensitivity analysis: number

No. of attributes	No. of significant	Classification accuracy (validation) (%)	Classification accuracy (training)
Logistic regression sensitivity analysis: Wald statistic (number of attributes)			
4	1	74	79
3	2	74	78
2	2	74	78
1	1	74	78

Table 11 Logistic regression sensitivity analysis: training sample size

Training sample size	Classification accuracy (test) (%)	Classification accuracy (training) (%)
200 samples	74	78
181 samples	74	80
121 samples	72	79
60 samples	74	78
30 samples	78	67

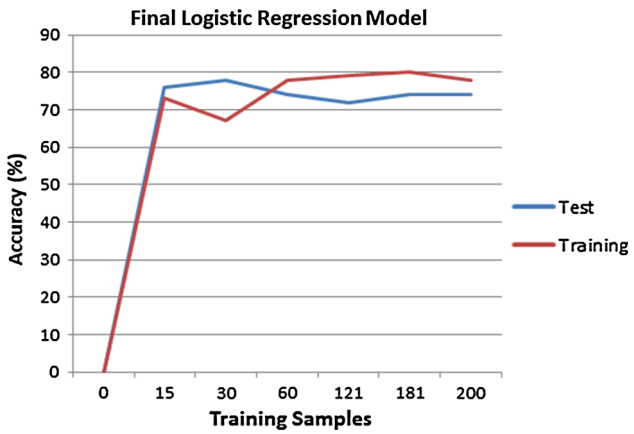

Fig. 11 Training versus test accuracy

training data sample size was also accomplished; results are provided in Tables 12 and 13. These results show the NBC classifier accuracy is insensitive to the number of independent factors for the training set and shows only a slight sensitivity for the validation set. The NBC classifier exhibited a slight improvement in classification accuracy with increased training data sample size. Specifically, increasing the training samples from the smallest size (60 samples) to the largest size (200 samples) was a 233 % increase in sample size, but resulted in just an 8 % increase in classification accuracy for the test set and 2 % increase in accuracy for the training set.

Following sensitivity analysis, a data fit analysis is conducted on the NBC classifier as was done previously for the Logistic Regression model. This analysis assesses how well the model predictions generalize to new data. The plot shown in Fig. 12 compares the training set classification accuracy and the test set classification accuracy. The optimal amount of training occurs at the point test classification accuracy reaches steady state while the training classification accuracy continues to improve [34]. This point occurs around 200 training samples for the NBC classifier. Use of additional training samples would result in overfitting, and model generalization would decrease [19].

3.3.2 Support Vector Machine

Subsequent to development of the NBC, an SVM machine learning approach is used applying the same attribute data used to develop the Binary Logistic Regression model and the Naïve Bayes Classifier in the previous sections. STATISTICA 12.0 is utilized again to develop the SVM based on the KPP linguistic quality attributes and the associated

Table 12 NBC sensitivity analysis: number of factors

No. of factors	Classification accuracy (test) (%)	Classification accuracy (training) (%)
NBC sensitivity analysis: independent factors		
4	82	77
3	82	77
2	86	77
1	85	77

Table 13 NBC sensitivity analysis: training sample size

Training sample size	Classification accuracy (test) (%)	Classification accuracy (training) (%)
NBC sensitivity analysis: training sample size		
200 samples	86	77
181 samples	82	77
121 samples	82	74
60 samples	78	75
30 samples	78	71

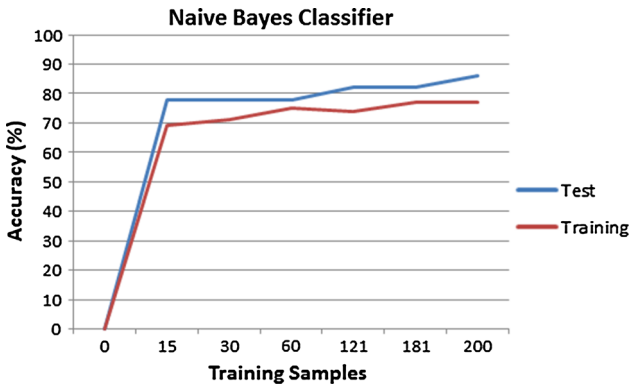

Fig. 12 Training versus test accuracy

operational test results. The SVM results are obtained using 181 training samples, each with four attributes (Untestable, Weak Phrase, Compound Requirements, and Vague) and are as follows:

- Number of support vectors = 88 (80 bounded).
- Cross-validation accuracy (%) = 76.243.
- Support vectors per class: 43 (No), 45 (Yes),
- Classification accuracy (%) = 76.243 (Train), 80.328 (Test).

A sensitivity analysis against the number of independent factors and training data sample size was also accomplished; results are shown in Tables 14 and 15. These results show the SVM classifier accuracy is insensitive to the number of independent factors and exhibited a slight improvement in classification accuracy with increased training data sample size. Specifically, increasing the training samples from the smallest size (60 samples) to the largest size (200 samples) was a 233 % increase in sample size, but resulted in just a 4 % increase in classification accuracy for the validation set and 1 % increase in accuracy for the training set.

Next, a data fit analysis is conducted on the SVM classifier as was done previously for the Logistic Regression model and NBC classifier to assesses how well the model predictions generalize to new data. The plot shown in Fig. 13 compares the training set classification accuracy and the test set classification accuracy. The optimal amount of training occurs around 200 training samples for the SVM classifier.

3.3.3 K-Nearest Neighbor

Finally, the KNN machine learning technique was applied utilizing the same attribute data used to develop the Binary Logistic Regression model, Naïve Bayes Classifier, and Support Vector Machine in the previous sections. STATISTICA 12.0 is utilized again to develop the KNN classifier based on the KPP linguistic quality attributes and the associated operational test results. The KNN results using four attributes (Untestable, Weak Phrase, Compound Requirements, and Vague) and 181 training samples are as follows:

- Number of nearest neighbors = 5.
- Distance measure: Euclidean.
- Averaging: uniform.
- Cross-validation accuracy (%) = 74.03315.
- Classification accuracy (%) = 81.967 (Test).

Sensitivity analyses were accomplished against the KNN distance measure, number of nearest neighbors, number of independent factors and training data sample size. Note, classification accuracy for the examples set was

Table 14 SVM sensitivity analysis: number of factors

No. of factors	Classification accuracy (test) (%)	Classification accuracy (training) (%)
SVM sensitivity analysis: independent factors		
4	80	76
3	80	76
2	80	76
1	80	76

Table 15 SVM sensitivity analysis: training sample size

Training sample size	Classification accuracy (validation) (%)	Classification accuracy (training) (%)
SVM sensitivity analysis: training sample size		
200 samples	81	77
181 samples	80	76
121 samples	76	76
60 samples	77	76
30 samples	78	71

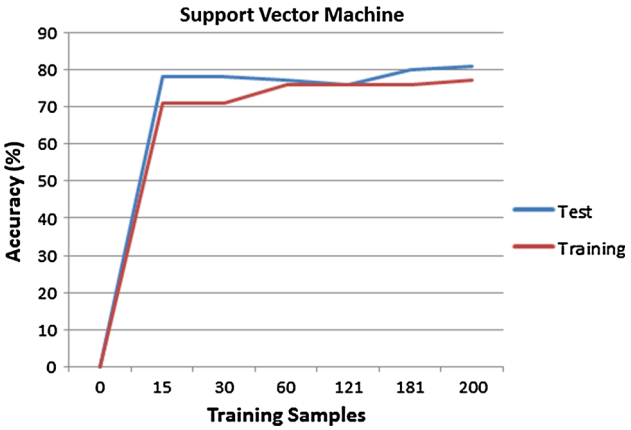

Fig. 13 Training versus test accuracy

not calculated given the example sample is used by KNN for predicting the testing sample.

The first sensitivity analysis assessed impact of classification accuracy based on use of four different distance measures: Euclidean, Euclidean Squared, City Block, and Chebychev, and is summarized in Table 16. The results showed there was no change in classification accuracy with a change in distance measure.

The next sensitivity analysis assessed the impact of the number of nearest neighbors on cross-validation accuracy. The plot in Fig. 14 shows cross-validation accuracy improves with an increase in the number of nearest neighbors.

A sensitivity analysis against the number of independent factors and training data sample size was accomplished next; results are shown in Tables 17 and 18. These results show the KNN classifier accuracy is insensitive to the number of independent factors and exhibited an improvement in classification accuracy with increased training data sample size.

Table 16 KNN sensitivity analysis: distance measure

Distance measure	Classification accuracy (validation) (%)
KNN sensitivity analysis: distance measure	
Euclidean	82
Euclidean squared	82
City block	82
Chebychev	82

Specifically, increasing the training samples from the smallest size (60 samples) to the largest size (200 samples) was a 233 % increase in sample size, but resulted in a 7 % increase in classification accuracy for the validation set.

4 Discussion

The four classification approaches, Logistic Regression, NBC, SVM, and KNN, developed in the previous section are now compared to determine best overall performance. A performance summary for the classifiers is provided in Table 19. All classifiers were validated and were insensitive to changes in the number of independent factors in the model, and they all showed only slight changes in classification accuracy as the training sample size increased. This indicates the classifiers are stable with respect to random variation from the selection of training data [36]. Moreover, they all had good classification accuracy that generalized well to new data. In all, this shows that multiple classification techniques have established that requirements quality can be used as a predictive factor for end-system operational performance.

Given that all four approaches had satisfactory classification performance, it would appear the final classifier

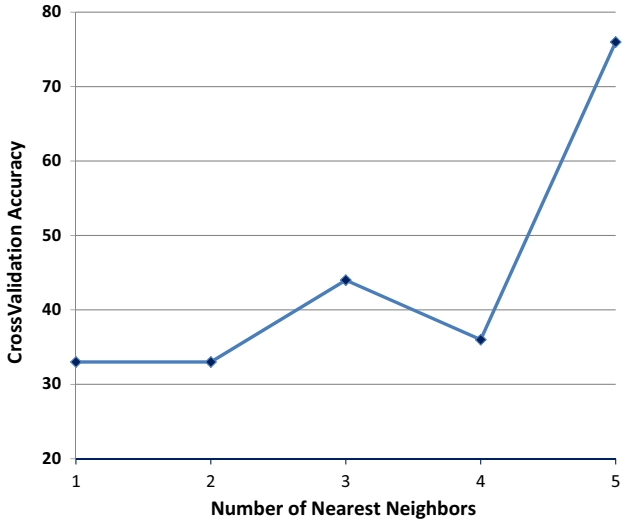

Fig. 14 Number of nearest neighbors versus cross-validation accuracy

selection should be based on overall classification accuracy. However, an effective means for classifier comparison is through use of ROC curves. Specifically, ROC space graphs are two-dimensional graphs that depict classifier performance by plotting true-positive rate against false-positive rate for the classifiers [37]. One classifier is better than another if its point in ROC space is to the northwest of the first [38]. The ROC space graph shown in Fig. 15 is derived from the confusion matrices for the four classifiers. The graph shows that performance for Logistic Regression and SVM is not much better than classifying by chance. In other words, these classifiers will get the positive classifications correct nearly all the time, but have a high false-positive rate. However, KNN and NBC show solid performance in the upper left quadrant of the ROC graph. Moreover, given the KNN location on the ROC graph has the most northwest position of the four classifiers, KNN has the superior classification performance. This result is not surprising since a general machine learning technique feature comparison between KNN and NBC as described by Kotsiantis [19] indicates that KNN has slightly better general classification accuracy. Therefore, given the nature of the data set used in this research, it is clear that KNN is the model of choice based on its classification accuracy against the validation set, insensitivity to the number of model parameters, and performance shown on the ROC graph.

5 Summary and conclusions

5.1 Summary

The first section of this paper provided background on the need for higher quality requirements statements given the "downstream" issues with defects and performance due to poor requirements, and set the stage for this research hypothesis that end-system operational performance can be determined through use of predictive modeling based on requirements quality attributes. This section also provided a litany of related research and described the intent of this research to bridge the gap in the prevailing body of knowledge by providing empirical evidence of the predictive relationship between requirements quality and end-system operational performance.

The second section of this paper described the model development and validation methodology and provided the sources and profile of the data used for model construction. This section defined the linguistic quality factors (Vague, Weak Phrase, Compound Requirements, Untestable, and Incomplete), outlined the regression analysis and machine learning techniques (Logistic Regression, Naïve Bayes Classifier, Support Vector Machine, and K-Nearest Neighbor), and described the analysis tools used for this research.

The third section of this paper provided results on the linguistic defects in the requirements statement data used for this research and showed that by far, Compound Requirements is the most significant linguistic defect in the data. Next, results were presented from development of the Logistic Regression model and the following machine learning approaches, Naïve Bayes Classifier, Support Vector Machine, and K-Nearest Neighbor. This included validation results, sensitivity analysis, data fit analysis, and ROC graph analysis results. A summary and comparison of all results were then presented in the fourth section of the paper.

Table 17 KNN sensitivity analysis: number of factors

No. of factors	Classification accuracy (validation) (%)
KNN sensitivity analysis: independent factors	
4	82
3	82
2	84
1	84

Table 18 KNN sensitivity analysis: training sample size

Training sample size	Classification accuracy (validation) (%)
KNN sensitivity analysis: training sample size	
200 samples	85
181 samples	84
121 samples	80
60 samples	79
30 samples	78

Table 19 Classification summary table

Metric	Classification accuracy (validation set) (%)	Sensitivity analysis (wrt classification accuracy)		Generalization
		Factors	Sample size	
Model				
Logistic Regression	74	Insensitive	Slight sensitivity	Yes
Naïve Bayes Classifier	86	Insensitive	Slight sensitivity	Yes
Support Vector Machine	81	Insensitive	Slight sensitivity	Yes
K-Nearest Neighbor	86	Insensitive	Slight sensitivity	N/A

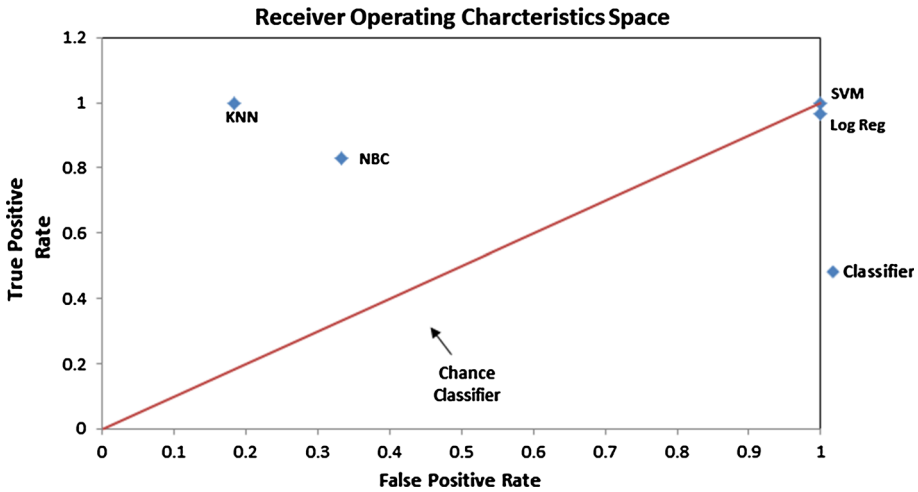

Fig. 15 Classifier receiver operating characteristics space

Table 20 Future research summary

Research area	Methodology
Model enhancement	Enhance the models with additional data points
	Consider using data from other government agencies such as NASA and the FAA
Alternative modeling approaches	Use alternate prediction models
	Consider Decision Trees and Neural Networks
	Consider combining modeling techniques
Piloting	Pilot use of the model in a new systems engineering project
	Utilize model as front-end risk management tool and assess how its use impacts improvements in requirements statements

5.2 Conclusions

This paper introduced four statistical classifiers that use requirements quality factors to predict system operational performance and were created using empirical data from current major acquisition programs in the federal government. The first approach used Logistic Regression. Following an iterative factor elimination process, a significant, well-fitted model was developed containing two independent variables, Compound Requirements and Vague. The overall classification accuracy of this model was 74 %.

Next, three machine learning approaches: Naïve Bayes Classifier, Support Vector Machine, and K-Nearest Neighbor, were used to create classifiers utilizing the same data used for developing the Logistic Regression model. All three machine learning approaches generalized well to new data and had overall classification accuracies ranging from 81 to 86 %, which exceeded accuracy of the Logistic Regression model.

The performances of the four classifiers were depicted on a ROC graph that plotted true-positive rate against false-positive rate. The graph illustrated that the performance of SVM and Logistic Regression was effectively random classification. However, KNN and NBC showed good performance in the upper left quadrant of the graph, with KNN having the better performance based on its position

as the most northwest on the ROC graph. Given the nature of the data set used in this research, it is evident the KNN Classifier has overall superior performance and is selected as the preferred model.

In all, the results from the four classification approaches used in this research confirm the hypothesis that requirements quality can be used as a predictive factor for end-system operational performance.

5.3 Future research

There are several areas of prospective research to further the investigation described in this paper. These areas include model enhancement, alternative modeling approaches, and piloting. Table 20 further describes these research areas.

Conduct of the future research described in this section will enhance model validity and provide additional verification that insight into probability of successful operational performance based on predictive modeling can influence requirements decisions on the front end of the systems engineering process.

References

1. Sutcliffe AG, Economou A, Markis P (1999) Tracing requirements errors to problems in the requirements engineering process. Requir Eng 4(3):134–151
2. Park S, Maurer F, Eberlein A, Fung T.-S (2010) Requirements attributes to predict requirements related defects. In: Proceedings of the 2010 conference of the center for advanced studies on collaborative research—CASCON'10, p 42
3. Fantechi A, Gnesi S, Lami G, Maccari A (2003) Applications of linguistic techniques for use case analysis. Requir Eng 8(3):161–170
4. Nigam A, Arya N, Nigam B, Jain D (2012) Tool for automatic discovery of ambiguity in requirements. Int J Comput Sci Issues 9(5):350–356
5. Yang H, Roeck A, Gervasi V, Willis A, Nuseibeh B (2011) Analysing anaphoric ambiguity in natural language requirements. Requir Eng 16(3):163–189
6. Génova G, Fuentes JM, Llorens J, Hurtado O, Moreno V (2011) A framework to measure and improve the quality of textual requirements. Requir Eng 18(1):25–41
7. Lami RW (2007) Giuseppe and Ferguson, an empirical study on the impact of automation on the requirements analysis process. J Comput Sci Technol 22(3):338–347
8. Lami G, Gnesi S, Fabbrini F, Fusani M, Trentanni G (2004) An Automatic tool for the analysis of natural language requirements. Information Science Technology Institute, Pisa, pp 1–21
9. Kiyavitskaya N, Zeni N, Mich L, Berry DM (2008) Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requir Eng 13(3):207–239
10. Bibi S, Tsoumakas G, Stamelos I, Vlahavas I (2006) Software defect prediction using regression via classification. In: IEEE international conference on computer systems and applications, pp 330–336
11. Tian L, Noore A (2005) Dynamic software reliability prediction: an approach based on support vector machines. Int J Reliab Qual Saf Eng 12(04):309–321
12. Malhotra R, Jain A (2012) Fault prediction using statistical and machine learning methods for improving software quality. J Inf Process Syst 8(2):241–262
13. Rawat MS, Dubey SK (2012) Software defect prediction models for quality improvement: a literature study. Int J Comput Sci 9(5):288–296
14. Turk W (2006) Writing requirements. IET Eng Manag J 16(3):20–24
15. Hall T, Beecham S, Bowes D, Gray D, Counsell S (2012) A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng 38(6):1276–1304
16. Elish K, Elish M (2008) Predicting defect-prone software modules using support vector machines. J Syst Softw 81(5):649–660
17. Saed Sayad (2015) An introduction to data mining. *Data mining in engineering, business, and medicine*. [Online]. Available: http://www.saedsayad.com/data_mining_map.htm. Accessed 15 March 2015
18. Cerpa N, Bardeen M, Kitchenham B, Verner J (2010) Evaluating logistic regression models to estimate software project outcomes. Inf Softw Technol 52(9):934–944
19. Kotsiantis SB (2007) Supervised machine learning: a review of classification techniques. Informatica 31:249–268
20. Kuntz L, Meyer G, Carnahan B (2003) Comparing statistical and machine learning classifiers: alternatives for predictive modeling in human factors research. Hum Factors J 45(3):408–423
21. Wu X, Kumar V, Ross Quinlan J, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, Zhou Z-H, Steinbach M, Hand DJ, Steinberg D (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
22. Gondra I (2008) Applying machine learning to software fault-proneness prediction. J Syst Softw 81(2):186–195
23. Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International joint conference on artificial intelligence, pp 0–6
24. Rodríguez JD, Pérez A, Lozano JA (2010) Sensitivity analysis of kappa-fold cross validation in prediction error estimation. IEEE Trans Pattern Anal Mach Intell 32(3):569–575
25. Karystinos GN, Pados DA (2000) On overfitting, generalization, and randomly expanded training sets. IEEE Trans Neural Netw 11(5):1050–1057
26. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
27. Hajar Mat Jani SAM (2011) Implementing case-based reasoning technique to software requirements specifications quality analysis. Int J Adv Comput Technol 3(1):23–31
28. Natt Och Dag J, Regnell B, Carlshamre P, Andersson M, Karlsson J (2002) A feasibility study of automated natural language requirements analysis in market-driven development. Requir Eng 7(1):20–33
29. Georgiades A, Marinos, Andreou (2011) A novel software tool for supporting and automating the requirements engineering process with the use of natural language. Int J Comput Sci Technol 4333(i):591–597
30. Mathison S, Hurworth R (2006) Document analysis. CassBeth document analysis manual. Cherry Hill
31. Seibel JS, Mazzuchi TA, Sarkani S (2006) Same vendor, version-to-version upgrade decision support model for commercial off-the-shelf productivity applications. Syst Eng 9(4):296–312
32. Bewick V, Cheek L, Ball J (2005) Statistics review 14: logistic regression. Crit Care 9(1):112–118
33. Halloran PSO (2005) Lecture 10: logistical regression II—multinomial data. Columbia University, New York, pp 1–73

34. Ye F (2010) What you see may not be what you get—a brief introduction to overfitting the problem of overfitting
35. Gortmaker SL, Hosmer DW, Lemeshow S (1994) Applied logistic regression. Contemp Sociol 23(1):159
36. Dietterich T (1998) Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput 10(7):1895–1923
37. Majnik M, Bosni Z (2013) ROC analysis of classifiers in machine learning: a survey. Intell Data Anal 17:531–558
38. Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159