Quantitative Assessment of the Software Maintenance Process

Joel Henry, Sallie Henry, Dennis Kafura, and Lance Matheson

TR 93-04

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

February 1, 1993

Quantitative Assessment of the Software Maintenance Process

Joel Henry, Sallie Henry, Dennis Kafura, Lance Matheson

Abstract: *This paper describes analysis techniques used to quantitatively assess the software maintenance process of a large military contractor, and the results obtained. The analysis techniques make use of basic data collected throughout the maintenance process. The data collected are extensive and allow the effects of adding a set of functional enhancements to be traced to process activities and product impact. The analysis techniques include data snooping, applied to gain insight into relationships and trends in the data. Simple nonparametric statistical techniques are then applied to test relationships between data items. The results provide valuable information for predicting process and product characteristics, and assessing the maintenance process.*

Biographical Sketches:

Joel Henry is in the final stages of his Ph.D. dissertation in Computer Science at Virginia Polytechnic Institute & State University. He is currently Assistant Professor of Computer Science at Dickinson College in Carlisle Pennsylvania, and has worked on software development projects for Digital Equipment Corporation and General Electric. Mr. Henry earned the BS and MS degrees in Computer Science from Montana State University in 1985 and 1986 respectively. His current research interests include the software process, software metrics and software engineering methodologies.

Dr. Sallie Henry is Associate Professor of Computer Science at Virginia Polytechnic Institute & State University. Dr. Henry received the BS in Mathematics from University of Wisconsin-La Crosse in 1972, and the MS in Computer Science from Iowa State University in 1977, and the Ph.D. also in Computer Science from Iowa State University in 1979. Her current research interests include software complexity metrics, software design methodologies, and software engineering.

Dr. Dennis Kafura received his Ph.D. degree in Computer Science from Purdue University in 1974 and has been an Associate Professor of Computer Science at Virginia Tech since 1982. He is interested in both software engineering and operating systems. In addition to the work reported in this paper, his previous research work has included deterministic scheduling, paging strategies and software metrics. His current research efforts focusses on the synthesis of concurrency and object-oriented programming. Two projects resulting from this work are the ACT++ project, investigating object-oriented concurrency control, and the Synergy project, combining object-oriented programming with the OSI protocols as a basis for realizing distributed forms of concurrency.

Lance A. Matheson is Assistant Professor of Management Science at Virginia Polytechnic Institute and State University. He received the Ph.D. in Management Science, M.B.A., and B.S. in Chemistry from the University of Washington. Dr. Matheson also has been an engineer and production manager for several electronics and metal finishing companies. His primary research interests are the economic analysis of quality control procedures and integrating quality control into the design of production systems. He has served as Associate Program Chair for the Southeastern Chapter of the Decision Sciences Institute and Information Systems Specialist for the 1992 National Decision Sciences Institute meeting. He has published in IIE Transactions, and Mathematical and Computer Modeling. He is a member of TIMS, DSI, ASQC and APICS.

Quantitative Assessment of the Software Maintenance Process

Joel Henry, Sallie Henry, Dennis Kafura and Lance Matheson

Abstract: *This paper describes analysis techniques used to quantitatively assess the software maintenance process of a large military contractor, and the results obtained. The analysis techniques make use of basic data collected throughout the maintenance process. The data collected are extensive and allow the effects of adding a set of functional enhancements to be traced to process activities and product impact. The analysis techniques include data snooping, applied to gain insight into relationships and trends in the data. Simple nonparametric statistical techniques are then applied to test relationships between data items. The results provide valuable information for predicting process and product characteristics, and assessing the maintenance process.*

I. INTRODUCTION

Most large software systems have long lifetimes during which the software undergoes significant change. Software maintenance is defined as the set of activities performed to change a software product after the software product is delivered to the customer [PRER87]. These activities, plus the tools and methods used to maintain software are referred to as the maintenance process. Changes to existing software include adding functionality to the software, correcting defects discovered in the software system, adapting the software to changes in the environment, and changing the software to support future maintenance or operation. The variety of changes made to software and the fact that most maintenance personnel were not involved in the development effort add significantly to the difficulties encountered while performing software maintenance.

Often viewed as an iceberg because only the very tip of the maintenance effort is visible [CANR72], maintenance activities account for over 60% of all effort expended by a development organization [PRER87]. Furthermore, one kind of maintenance, functional enhancement, comprises more than 50% of total maintenance costs [PRER87]. These experiences support the view that software maintenance, and functional enhancements in particular, are important challenges currently facing most software organizations.

In recent years the software process (including both development and maintenance) has received a great deal of attention [HUMW87] [HUMW89] [BOLT91] because the process used to develop and maintain software significantly impacts the cost, quality and timeliness of software products. The impact

is so significant that software process improvement is seen as the most important approach to software product improvement [HUMW89].

While software development typically refers to the creation of new software, software maintenance is performed for a variety of reasons. The four types of software maintenance activities are:

1. Corrective - changes made to correct defects in software
2. Adaptive - changes needed to adapt existing software to a changing environment
3. Perfective - enhancements to software which provide additional functionality or modify existing functionality
4. Preventative - changes which improve future maintainability, reliability or support future enhancements

The tasks employed during perfective maintenance are very similar to those applied during development: specify, design, code, and test. Thus, the first step in perfective maintenance is to obtain a written specification of the functionality to be added. The written specification is given by changes and additions to the documentation specifying the functionality of the existing software. In principle the written specification is given completely and is never changed during the ensueing maintenance effort. In practice, however, these specifications are corrected and refined throughout the maintenance process. The changing of functional specifications during maintenance and development is referred to as requirements volatility. Requirements volatility has been cited as the leading problem in a field study of software managers [THAR82]. Changing requirements adversely affects the design, coding and testing of software. An acute need exists to quantitatively assess the maintenance process and the impact of requirements volatility on the maintenance process.

The focus of this paper is assessment of perfective maintenance activities driven by changes to the specification documents of existing software. Analysis of the maintenance process and requirements volatility attempts to answer three general questions:

1. Can the maintenance process and the product characteristics affected by maintenance activities be assessed and predicted?
2. What is the impact of requirements volatility on the maintenance process?

3. What is the impact of requirements volatility on the software product?

These questions are answered using a new process assessment methodology [HENJ92b] integrating the principles of Total Quality Management [JABJ91] and the Process Maturity Framework developed at the Software Engineering Institute [HUMW89]. The assessment methodology involves process investigation based on the work of SEI and supplemented by Henry [HENJ91a]. A process modeling technique employing control flow diagrams is used to define an organization's maintenance process [HENJ92a]. Process and product data, collected throughout the development or maintenance process, are analyzed using nonparametric statistical techniques. The statistical analysis results and the process model are used to assess an organization's software process.

This paper describes the statistical techniques used in a two year study conducted at a large commercial software organization to assess the maintenance process and the impact of requirements volatility on the maintenance process. The assessment illustrates relationships between process and product characteristics, and the impact of requirements volatility on both the maintenance process and the delivered software product.

The remainder of this paper is divided into four sections. Section II describes the commercial software organization, the software product, and the process used to maintain the software product. Section III presents the statistical techniques used to analyze product and process data. Section IV presents analysis results in five distinct areas, each illustrating the use of one or more of the analysis techniques described in Section III. Section V outlines conclusions and the direction of future work.

II IMPLEMENTATION OF THE ASSESSMENT METHODOLOGY

Implementation of the new integrated assessment methodology described in [HENJ92b] involves four significant steps: investigation, modeling, instrumentation, and analysis. Results of the first step in the integrated assessment methodology, the investigation step, are described by in [HENJ91b]. The modeling step, documenting the GE software process, is presented in [HENJ92a] and the results discussed in [HENJ92c]. The third step in the integrated assessment methodology, instrumentation, took place

throughout the project. This paper presents the results of the analysis step of the integrated assessment methodology.

Recognizing the importance of each step, and the need for the assessment team to be present throughout the assessment period, the assessment team specified the following requirements for an implementation organization:

1. The organization must be committed to process assessment and improvement.
2. The organization must allow the assessment team on-site access throughout the assessment period.
3. The organization must support the collection of data throughout the assessment period.

The Computer Program Projects Group within the Government Electronic Systems Division (GESD) of General Electric (GE) met these requirements. In an effort to improve both the software process and the software products produced, GESD formed the Product Improvement Committee in 1990. The Committee includes four full-time members and over fifty part-time members from throughout GESD. The Computer Program Projects Group agreed to fund an assessment team member for six months per year, over a two year period, for on-site work. In addition, the Group committed to gathering data as a normal part of the software process.

The assessment conducted in this study concentrated on perfective maintenance activities for three reasons. First, the manpower, resources and time required to assess the entire GE software process were beyond those available in this study. Second, requirements changes, omissions and misunderstandings are recognized as a major problem in the commercial software industry [BOEB91] [CHML90] [THAR82]. Third, GE recognizes the large amount of effort and resources expended on maintenance. A project level savings of only five percent would translate into hundreds of thousands of dollars.

The organization developed, and the Computer Program Projects Group manages the maintenance of, a large radar based weapons system for the U.S. Navy. The system consists of both hardware and software and enjoys a long legacy of successful operation. The weapons system is controlled by more than ten different computer programs executing on separate hardware systems. The systems accept input from external sensors and operators. Extensive communication between the programs via a high speed network takes place during operation. The weapons system has undergone extensive maintenance during

it's twenty-two year lifetime. The maintenance efforts are largely aimed at incorporating new functionality and correcting defects.

The computer programs comprising the radar based weapons system are released in baselines. A baseline is implemented under a contract which specifies a set of functional enhancements. Each enhancement is referred to as an upgrade item. Upgrade items typically require changes to the Program Performance Specification (PPS), the Interface Design Specification (IDS) or both. The process used to create new baselines is shown in Figure 1.

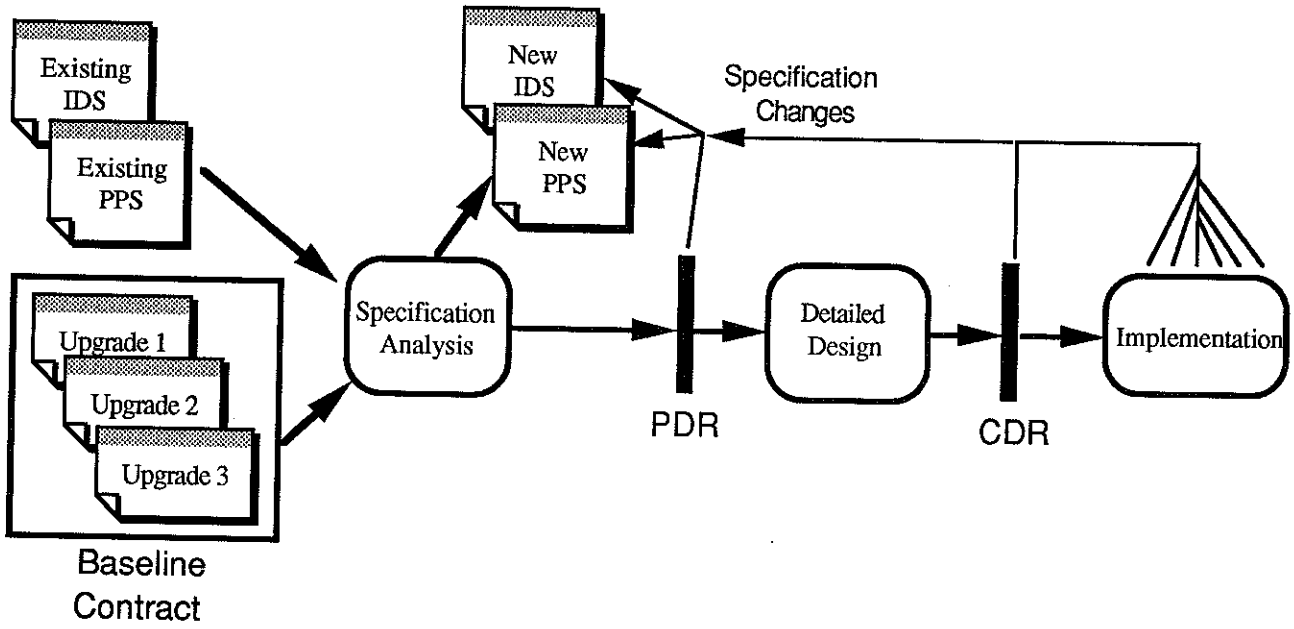

Figure 1. Baseline Development Process

System engineers analyze the impact and interpret the precise meaning of each upgrade item during the system specification and software specification phases, depicted together as Specification Analysis in Figure 1. This analysis is documented in the new PPS and the new IDS, which are reviewed at the Preliminary Design Review (PDR). The new PPS and new IDS are the basis for implementing the new baseline. Hereafter we will simply say PPS and IDS to refer to the new versions of these documents.

Detailed design of upgrade items takes place following the PDR. Detailed analysis and design of the functionality documented in the PPS and IDS typically require additions and changes to the PPS and the

IDS. The IDS and PPS pages requiring change and the detailed design are reviewed at the Critical Design Review (CDR). Successful completion of the CDR, as judged by the customer representatives and management, initiates the implementation phase.

The new versions of the PPS and the IDS presented at the PDR and the CDR are modified at the PDR, the CDR, and post-CDR. These specification changes are documented using a specification change form and implemented following the CDR.

Specification changes are the driving force behind each baseline. Specification changes are a major source of difficulty in producing a new baseline because

- the majority of specification changes occur after CDR
- most specification changes require code changes
- implementation requires coordination of system engineers, project managers, subcontractors, testers and technical writers
- the timing of specification changes requires a functionality to be fit into an already designed product

These issues make specification changes a serious concern of each baseline effort.

The PPS and the IDS are written, maintained and controlled by the system engineering group. This group determines the need for a specification change and originates a change using a specification change form. The completed specification change form has attached to it copies of the PPS or IDS pages requiring change. The specification change form and attached pages are reviewed and approved by both management and the customer before implementation is begun.

Specification changes submitted at the PDR and the CDR are evaluated and directed during these reviews. A specification change may be directed for implementation (approved), assigned to a future baseline (deferred) or rejected (withdrawn). Specification changes approved at the PDR and the CDR are implemented following the CDR.

Specification changes originating after the CDR are submitted to a review board. The board consists of representatives of project management, system engineering, configuration management and the customer. A specification change may be approved, withdrawn or deferred to another baseline at the

direction of the board. The board is supplied with estimates of cost and schedule impact as well as the estimated impact on computer programs.

Specification changes approved for implementation at the PDR, the CDR and by the review board are assigned a computer program change number. This number is entered as a comment on each line of code added or changed to implement the associated specification change. In addition, the programmers implementing specification changes record the number of lines of code added, deleted and changed. The specification change and computer program change number provide traceability of each specification change. Detailed information about implementation effort, test results, and final status is also recorded for each specification change.

A large amount of data about upgrade items and upgrade specification changes impacting the largest computer program in the radar based weapons system, the control program, was captured during a specific baseline. Data was collected during the baseline effort by the project management organization, the software subcontractor, the metrics group, testing groups and upper level management. Validation was done by cross referencing data, performing an automated analysis of the delivered source code, and interviewing personnel.

Validated data is stored and analyzed using the Exstatixs statistical analysis software package. This package allows data to be input, manipulated, graphed and statistically analyzed in many different ways. The results produced by the software package were verified using a second statistical package.

III ANALYSIS TECHNIQUES

The analysis techniques presented in this paper use nonparametric statistics. Parametric statistical techniques assume data is drawn from a population adhering to a specific type of distribution (e.g., normal). Nonparametric techniques do not assume any distribution and are valid for all populations [IMAR83]. Nonparametric techniques are advantageous because

- the assumptions are less demanding,
- the affect of noisy data is reduced,

- nominal (assigned to a specific class, e.g. high vs. low) and ordinal (objects are ordered by a greater than or less than relationship) scales may be used, and
- the results are as powerful and rigorous as the results of parametric techniques [SCHN92].

One goal of analysis is to predict future product and process characteristics using data available at critical points in the baseline effort. Linear and nonlinear regression determines the ability to predict one variable from one or more other variables. The predicted variable (**y**) is referred to as the dependent variable and the variables used to form the prediction equation are called independent variables (**x**). Simple linear regression assumes the dependent variable increases or decreases linearly with respect to one independent variable. Simple linear regression is described by an equation of the form:

$$y = a x + b$$

Multiple linear regression assumes the dependent variable behaves in a linear fashion, but is influenced by more than one independent variable. Multiple linear regression is described by an equation of the form:

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots b$$

The ability to predict the dependent variable from the independent variables is evaluated in two ways. Associated with each regression analysis is an R^2 value and a P-value. R^2 is the square of the linear correlation coefficient between the actual values and the predicted values. R^2 values range from 0 to 1.0 and represent the amount of variability in the dependent variable that is explained by the independent variables. A large R^2 provides confidence that the prediction equation explains a significant portion of the variability in the dependent variable. The P-value of a test is defined as:

- the smallest probability leading to rejection of the test hypothesis

In the case of linear regression, the test hypothesis is that prediction of the dependent variable using the independent variable is possible. The P-value quantifies the probability of asserting that linear prediction is possible when in fact linear prediction is not possible. Very small P-values represent a low probability of mistakenly assuming prediction is possible. Accurate and confident prediction equations are characterized by both a large R^2 value and a small P-value.

Another goal of analysis is to compare the means of samples drawn from two or more independent populations to detect population differences. The Wilcoxen-Mann-Whitney rank sum test and the Kruskal-

Wallis test compare means of samples taken from two or more populations [OTTL88]. The test hypothesis in sample mean comparisons is that the sample means are equal. Both tests produce a P-value evaluating the probability of assuming no difference exists between sample means. In this case a small P-value indicates there is ample evidence to support rejection of the test hypothesis. That is, a small P-value indicates a small probability that the populations have identical means.

The association between data values is also of interest. Association is analyzed using linear and rank correlation. Linear correlation measures the linear relationship between two variables. Rank correlation does not assume a linear relationship but rather tests the relationship based on ranks assigned to data values. The rank of a data values is the position of the data value in an ordered list of all data values. In the case of both linear and rank correlation, correlation values range from -1.0 to +1.0. Correlation approaching -1.0 and +1.0 demonstrate a strong association between variables.

A simple classification of software products based on data available early in the maintenance process which predicts future product characteristics is also valuable. For example, classifying software components into categories of error-proneness based on their size is an important classification scheme if a statistically significant association exists between size and number of defects per component. Contingency tables are used to measure the statistical significance of such associations.

In a contingency table the rows classify items (e.g. components) using one criteria (e.g. size) and the columns classify items using a second criteria (e.g. number of defects). Contingency tables can be used to test the hypothesis that row classification is independent of column classification. This hypothesis is tested using a P-value derived from the classification of actual data. The P-value indicates the likelihood that row and column classification are independent. A small P-value causes the test hypothesis to be rejected in favor of assuming row classification and column classification are dependent. In our example, a small P-value would indicate that large components should be classified as error-prone.

The statistical techniques described here are basic nonparametric tests available on most commercially available statistical software packages. The results of these statistical tests are easily interpreted and presented. Our experience has shown that simple statistical techniques applied to basic data are much more useful in the commercial world than elegant techniques applied to complex metrics.

IV ASSESSMENT RESULTS

While significant results have been obtained in seven areas, only five areas are described here because of proprietary concerns. Each of the following five subsections describes the results of statistical analysis. Results are prefaced by discussion of the specific portion of the process under analysis, the data used in the analysis, the area of interest, and the statistical techniques used.

UPGRADE ITEMS

Thirteen upgrade items were defined in the contract for the control program for the baseline under investigation. The contract also contains an estimate of the effort, in person months, needed to implement each upgrade item. The estimates are produced using a GE proprietary procedure.

Table 1 shows the contract estimates of the person months needed to implement each upgrade item as a percentage of the actual person months expended. For example, the estimate of the number of person months needed to implement upgrade item number 3 was 48.04% of the actual person months expended. In this instance the estimate represents less than half the actual person months needed to implement the upgrade.

Upgrade Item	Estimate of Person Months as a Percentage of Actual
1	43.36%
2	27.05%
3	48.04%
4	53.13%
5	53.21%
6	252.56%
7	101.37%
8	46.51%
9	43.31%
10	106.75%
11	15.59%
12	22.37%
13	14.61%

Table 1. Upgrade Item Effort Estimates as Percentage of Actual

The contract estimates for each upgrade item need to be revised during the maintenance process using data from the current baseline and historical data from previous baseline efforts. As each baseline effort progresses, data becomes available about the progress and status of the current effort. This data, used in conjunction with historical information, could provide valuable information about the current baseline effort.

One goal of the analysis of upgrade items was to determine if the actual effort required to implement an upgrade item could be predicted at two critical points in the maintenance process: immediately following PDR and imediately following CDR. If predictions could be made, the accuracy and timeliness must also be determined.

Applying the statistical technique of multiple linear regression, two predictive equations were derived, the first using data available at the PDR and the second using data available at the CDR. Table 2 presents the results obtained using multiple regression for both of these sets of data. The data items included in the regression equations were chosen using stepwise regression techniques which select the most statistically significant independent variables from the set of available independent variables.

Prediction Time	Dependent Variable	Independent Variables	R^2	P-value
PDR	Actual Person Months	• Contract Estimate of Person Months • Number of PDR Spec Changes	.82	.0011
CDR	Actual Person Months	• Contract Estimate of Person Months • Number of PDR Spec Changes • CDR Estimate of Changed SLOCs	.91	.0001

Table 2. Results of Regression Analysis

The prediction equations include three independent variables, two available at PDR and the third available at CDR. The estimate of person months in the contract and the number of specification changes approved at the PDR for each upgrade item are used in the PDR prediction equation. These same independent variables and the estimate of source lines of code (SLOCs) needed to be changed to implement

each upgrade are included in the CDR prediction equation. The dependent variable in both cases is the actual number of person months needed to implement an upgrade item.

Cross validation was performed to insure that the regression equations were not overly tailored to the data used to construct them. Cross validation is performed by randomly selecting half the data, deriving a regression equation, then testing the ability of the derived regression equation to predict the dependent variable using the remaining data. Cross validation of both PDR and CDR regression equations was successful, indicating the predictive equations are able to accurately predict the actual person months needed to implement upgrade items.

Analysis of the prediction errors (residuals) was performed to determine if prediction intervals could be constructed. The residuals of each regression equation are normally distributed. The residuals of each equation were divided at the median predicted value. The variance of residuals from above the mean and below the mean found to be statistically equivalent for both equations. These assumptions are necessary to construct prediction intervals around the regression equations. Such prediction intervals have been constructed, but are not presented here for proprietary reasons.

Cross validation and the ability to construct prediction intervals increase organizational confidence in the regression equations. Prediction intervals are in use at this time. Project management uses the intervals as a window of variability of actual upgrade item effort.

POST-CDR UPGRADE SPECIFICATION CHANGES

The defined process used to maintain computer programs allows coding to begin following the CDR. Theoretically, the requirements are stable before coding begins. In reality requirements continue to evolve and change throughout the maintenance effort. Evolution and change of requirements are documented as specification changes.

Post-CDR specification changes are recognized throughout the organization as having a significant impact on both the process and the product. There were more than twice as many specification changes after the CDR for the analyzed baseline than at the CDR and PDR. Post-CDR specification changes must be distributed to, and their implementation coordinated among, many different groups (i.e. system

engineering, project management, subcontractors, quality assurance, testing groups and configuration management).

The goal of analyzing upgrade specification changes was to quantitatively evaluate their impact on both software products and the maintenance process. Specifically, the analysis sought to determine if PDR and CDR upgrade specification changes differed in product and product impact from post-CDR upgrade specification changes. If differences existed, quantification of these differences was required.

Ninety-one upgrade specification changes were written during the baseline. These specification changes were separated by origination time period as PDR, CDR or post-CDR. The characteristics of each group are shown in Table 3.

Origination Period	Number of Spec. Changes	Avg. Changed SLOCs per Spec. Change	Avg. Modules Changed per Spec. Change	Avg. Changed SLOCs per Module per Spec. Change	Avg. SLOCs/Personday as a Percentage of CDR Spec. Change Avg. SLOCs/Personday
PDR	13	1371.5	7.2	342.6	99.5%
CDR	17	1012.2	9.7	137.1	100%
Post CDR	61	84.4	1.8	44.9	72.4%

Table 3. Specification Change Characteristics by Origination Period

Two significant observations were made from Table 3, namely:

1. The productivity (measured by SLOCs/Personday) realized implementing post-CDR upgrade specification changes was 72.4% of the CDR specification change productivity.
2. The post-CDR specification changes have less impact on the computer programs, as measured average SLOCs changed, average modules changed, and average changed SLOCs per module, than PDR and CDR specification changes.

The Wilcoxon-Mann-Whitney rank sum test determined that a statistically significant difference existed between the average SLOCs changed per personday, SLOCs changed and SLOCs changed per module for post-CDR specification changes, and the same data for PDR and CDR specification changes. The data suggests that the size of the change might be responsible for the observed decrease in productivity.

Contingency tables were constructed to analyze the association between changed SLOCs and productivity for all upgrade specification changes. The tables were created using the median values of changed SLOCs per personday, changed SLOCs, and changed SLOCs per module to classify the data. The contingency tables are shown in Tables 4 and 5. The P-values for these tables are less than 0.005 for both tables allowing the rejection of the hypothesis that changed SLOCs per personday is independent of changed SLOCs and changed SLOCs per module.

	<= Median Changed SLOCs	> Median Changed SLOCs
<= Median SLOCs Changed per Personday	33	13
> Median SLOCs Changed per Personday	4	41

Table 4. Changed SLOCs per Personday vs. Changed SLOCs

	<= Median Changed SLOCs per Module	> Median Changed SLOCs per Module
<= Median SLOCs Changed per Personday	42	3
> Median SLOCs Changed per Personday	4	42

Table 5. Changed SLOCs per Personday vs. Changed SLOCs per Module

Based on the information in Tables 4 and 5, and an analysis of the organization's process, it appeared that the overhead required to process and document specification changes, which had small impact on the control program, significantly decreased the productivity. A regression equation predicting persondays from changed SLOCs was fitted using all upgrade specification changes. The R^2 of this equation is .94 and the P-value is less than 0.0001. The y-intercept of this equation, representing the number of persondays expended for each specification change regardless of size, was quite high. This intercept indicates that each specification change requires a significant amount of overhead. This overhead decreases the productivity of specification changes which require a small number of changed SLOCs.

Quantifying the overhead incurred by major activities in the maintenance process is an important step to understanding and managing the process. While small specification changes must be performed, an organization must account for this overhead to effectively assess, plan and perform software maintenance.

MODULE IMPACT

All SLOCs added or changed to implement a specification change are commented with a computer program change number. In addition, all lines of source code changed to correct errors in the control program are similarly commented. These computer program change numbers provide a way to trace the effect of specification changes and error corrections to control program source code.

The control program is comprised of modules which are made up of procedures. Procedures consist of source code and comments. Computer program change numbers entered on each changed or added line of code allow modifications to be traced to, and accumulated by, procedure and module.

The relationship between upgrade items, upgrade specification changes and errors corrected per module in the control program was investigated. Specifically, we analyzed the impact of upgrade items and upgrade specification changes on the reliability of modules within the control program. Reliability in this context means the error-proneness of modules and was measured by errors corrected per module. If relationships existed, the relationships were to be quantified.

The number of errors corrected per module was captured as well as the number of upgrade items and upgrade specification changes affecting each module. The percentage of the module changed and the number of persondays expended per module to implement upgrade specification changes were also captured.

Simple graphs were constructed to compare the impact of upgrade items and upgrade specification changes to the reliability of software modules. The graphs indicated nonlinear relationships existed, suggesting rank order correlation be applied.

Rank correlation was applied to determine the association between errors corrected per module and the impact of upgrade items per module. Impact on modules is measured by the number of upgrades affecting a module, the number of upgrade specification changes affecting a module, and the percentage of the module changed to implement upgrade specification changes. Impact on the maintenance process is

measured by the number of persondays expended implementing upgrade specification changes per module. Strong rank correlation existed between errors corrected per module and upgrade item impact, as shown by the rank correlation coefficients in Table 6.

A simple classification of modules according to upgrade impact which predicts modules as having more or less than a specific number of errors was also desired. Contingency tables were created using median values of errors corrected, number of upgrades affecting a module and the number of upgrade specification changes affecting a module. The contingency tables are shown in Tables 7 and 8, and both have P-values of less than 0.005.

	Number of Upgrades	Number of Upgrade Spec Changes	Percentage of Module Changed	Number of Persondays
Number of Errors Corrected	.803	.855	.767	.808

Table 6. Correlation Between Errors Corrected and Upgrade Impact

	<= Median Number of Upgrade Items Impacting	> Median Number of Upgrade Items Impacting
<= Median Number of Errors Corrected	22	5
> Median Number of Errors Corrected	2	26

Table 7. Number of Defects vs. Upgrade Item Impact

The rank correlation coefficient shows significant correlation exists between errors corrected and upgrade item impact. Strong correlation suggests modules can be ranked using upgrade impact. This ranking predicts the ranking of modules based on errors corrected. Modules predicted to be error-prone can be selected for module inspection following the coding phase in an attempt to detect errors prior to lengthy and expensive test phases. Module ranking can also be used to select modules for additional unit and module integration testing prior to system testing.

	<= Median Number of Upgrade Spec Changes Impacting	> Median Number of Upgrade Spec Changes Impacting
<= Median Number of Errors Corrected	25	3
> Median Number of Errors Corrected	6	22

Table 8. Number of Defects vs. Upgrade Specification Change Impact

Table 7 shows that selecting modules based on upgrade item impact greater than the median number would include 26 of the 28 modules (93%) with error rates above the median number of errors. Of the 31 modules selected with upgrade impact greater than the median value, 26 of them (84%) actually had more than the median number of errors per module.

The analysis quantitatively evaluates the impact of upgrade items on modules. Modules can be rank ordered based on upgrade item impact or the historical median value of upgrade item impact can be used to classify modules as more or less error-prone. Upgrade impact information is currently being used within the organization to select modules for additional code walkthroughs or more intensive testing.

ENGINEERING TEST PHASE

Weapons systems programs are written by a software subcontractor then delivered to the contracting organization. The delivered programs undergo two different test phases following delivery. The first test phase is the engineering test phase in which upgrade items are tested to detect errors and insure the specified functionality is achieved. The second test phase qualifies the entire system for demonstration to the customer.

The engineering test phase is an important phase in the maintenance process. It is the first phase which provides visibility to the customer of the ongoing maintenance effort. The number of errors found and the number of upgrades successfully completing the engineering test phase are documented and reported to the customer.

The goal of analyzing the engineering test phase was to determine the characteristics of upgrade items which influence the engineering test results, namely the number of errors detected and the testing failure rate. The influence of the characteristics was to be quantitatively specified.

Rank correlation and linear correlation analysis were applied to determine the relationship between upgrade item characteristics and engineering test results. The correlation of the number of tests failed, number of tests run, and number of errors found are shown in Table 9. The correlation of upgrade item characteristics with either tests failed or errors detected is very low. The correlation of total tests run is statistically significant, suggesting a relationship exists between the upgrade characteristics of modules changed, person days, SLOCs changed and total tests.

Upgrade Item Characteristics	Rank Correlation			Linear Correlation
	Tests Failed	Errors Detected	Total Tests	
Modules Changed	-0.031	0.439	0.594	0.633
Person Days	0.087	0.152	0.703	0.737
Spec. Changes	-0.366	0.084	0.257	0.027
SLOCs Changed	0.207	0.495	0.740	0.860

Table 9. Engineering Test Phase Result Correlation

Further investigation of the engineering test phase showed very low first test failure rates for all upgrade items affecting the control program for the baseline under analysis. Both historical data and management input supported the view that first test failure rates are typically much higher for the control program. The failure rates for the baseline under investigation are shown in Figure 2.

Analysis of the organization's maintenance process provided the key information needed to explain the quantitative results of Table 10 and Figure 2. In an effort to detect errors earlier in the maintenance process, the engineering test group supplied the software subcontractor with the testbed to be executed during the engineering test phase. The software subcontractor used these tests to verify the computer program before the engineering test phase began.

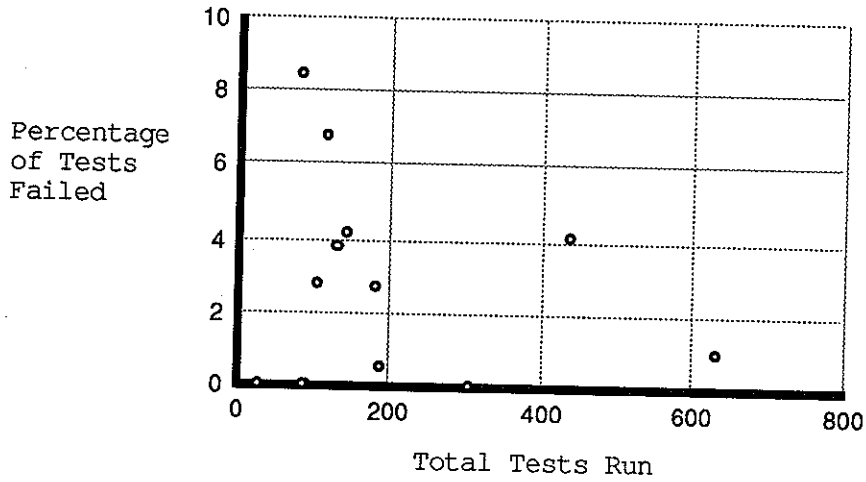

Figure 2. Percentage of First Tests Failed

Quantitative analysis of the subcontractor's test phase supports the assertion that errors were detected earlier in the baseline effort. The rank correlation between the number of upgrades and errors detected in the subcontractor test phase is .72. This correlation suggests the subcontractor detected and repaired the errors uncovered using the tests supplied by GE.

The number of errors detected during the engineering test phase does not correlate with any upgrade item characteristic. This is explained by the fact that the tests run during the engineering test phase have been run by the subcontractor in the immediately preceding phase of the maintenance process. The duplication of testing is expensive and unneeded. The organization is now in the process of changing the role of the engineering test group and the activities performed during the engineering test phase. The new focus of engineering test activities will be baseline to baseline regression.

INCOMPLETE COMPUTER PROGRAM CHANGES

Changes to computer programs are generated in response to errors and specification changes. A unique computer program change number is assigned to implement each change to computer programs and is used to track changes to source computer programs. Implementation information about computer

program changes is stored in a database and tracked using specific codes to describe the status of each change. A change may be open (under development), or closed (coded and successfully tested).

Computer program changes remaining open against a computer program at the completion of the product baseline represent unfixed errors or unimplemented specification changes. These changes and errors are deferred to future baselines. The number of open changes is a concern to both the organization and the customer at the delivery of the product baseline.

The goal of analyzing open changes is to determine if the number of open changes can be predicted using data available during development. The accuracy and timeliness of the predictions must also be determined. If such predictions can be accurately produced, the predictions could be used to limit the number of changes approved for a baseline and thus reduce the number of open changes at the delivery of a baseline.

Dependent Variable	Independent Variables	R^2	P-Value
Open Computer Program Changes	• Spec Changes Impacting Code • Spec Changes not Impacting Code • Interface Change Requests	.99	.0016

Table 10. Results of Open Computer Program Change Regression Analysis

Applying the technique of multiple regression, prediction equations were fitted using data available throughout the maintenance process. Table 10 presents the results of using multiple regression to predict the number of open computer program changes per computer program in the weapons system.

The independent variables used in the equation are generated throughout the maintenance process. The equation generated can be applied with the available data to provide predictions at specific points during the implementation effort. These predictions can be used by project management to quantitatively estimate the number of open computer program changes at the conclusion of the current baseline. The R^2 of .99 and the P-value of .0016 provide a great deal of statistical confidence in the regression equation. The data sets are small in this case (13 different computer programs), making further analysis across baselines necessary.

V CONCLUSIONS

The results of our analysis have been reviewed and interpreted by development personnel and project management, yielding suggestions for improvements in the maintenance process. These improvements are being inserted into the maintenance process. In addition, a central database is being developed to store maintenance information and track the impact of specification changes on the software products.

The data analysis techniques described here make use of simple statistical tests commonly used in the fields of management science and industrial engineering. The techniques discover relationships between maintenance activities and process and product characteristics. The relationships provide visibility into the maintenance process and quantify the impact of requirements volatility on the software product.

The application of nonparametric statistical techniques to basic data collected during the software process provide valuable insight into the process and the impact of the process on the software product. The statistical results quantify the effectiveness of process activities as well as the impact of process activities on the software product.

Statistical results used in conjunction with an accurate process model provide a powerful method for assessing the software process. The process model supports specifying where in the process to collect data and when prediction, correlation and association are most useful. The two distinct types of information provided by statistical analysis and content analysis provide unique opportunities to assess and improve the software process.

Future analysis will focus on corrective maintenance activities and specific phases within the organization's maintenance process (such as unit test and integration test). The results obtained to date are being actively used to improve the organization's ability to produce higher quality software, on time and within budget.

REFERENCES

[BOEB91] Boehm, B. W., "Software Risk Management: Principles and Practices," *IEEE Software*, January 1991.

[BOLT91] Bollinger, T.B. and McGowen, C., "A Critical Look at Software Capability Evaluations," *IEEE Software*, July 1991.

[CANR72] Canning, R., "The Maintenance 'Iceberg,'" *EDP Analyzer*, October 1972.

[CHML90] Chmura, L. J., Norico, A. F., and Wicinski, T. J., "Evaluating Software Design Processes by Analyzing Change Data Over Time,"*IEEE Transactions on Software Engineering*, Vol. 16, No. 7, July 1990.

[HENJ91a] Henry, J. E., Keenan, S., Keenan, M. and Henry, S., "An Assessment Method for Small Organizations," *Proceedings of the 1st International Software Quality Assurance Conference*, October 1991.

[HENJ91b] Henry, J. E. and Dublanica, W., "Improving GESD Computer Program Development Processes," General Electric Corporation, Government Electronic System Division, November 1991.

[HENJ92a] Henry, J. E., Dublanica, W., Kubeck, J. and McMullen, B., "A Process Modeling Technique and Application Results," *Proceedings of the Software Tools and Techniques Symposium*, March 1992.

[HENJ92b] Henry, J. E.and Henry, S., "An Integrated Approach to Software Process Assessment," *Proceedings of the Pacific Northwest Software Quality Conference*, to appear, October 1992.

[HENJ92c] Henry, J. E.and Blasewitz, R. E., "Process Modeling: Theory and Reality," *IEEE Software*, November 1992.

[HUMW87] Humphrey, W.S. and Sweet, W.L. "A Method for Assessing the Software Engineering Capability of Contractors," Software Engineering Institute, Carnegie Mellon University, September 1987.

[HUMW89] Humphrey, W.S., *Managing the Software Process*, Addison-Wesley, 1989.

[IMAR83] Iman, R. L., and Conover W. J., *Modern Business Statistics*, Wiley, 1983.

[JABJ91] Jablonski, J., R., *Implementing Total Quality Management: An Overview*, Pfeiffer, 1991.

[OTTL88] Ott, L., *An Introduction to Statistical Methods and Data Analysis*, PWS-Kent, 1988.

[PRER87] Pressman, R. S., *Software Engineering: A Practitioners Approach*, McGraw-Hill, 1987.

[SCHN92] Schneidewind, N. F., "Methodology for Validating Software Metrics," *IEEE Transactions on Software Engineering*, May 1992.

[THAR82] Thayer, R. H., Pyster, P. and Wood, R. C., "Validating Solutions to Major Problems in Software Engineering Project Management," *IEEE Computer*, August 1982.