Colorado Technical University

Addressing Software Volatility in the System Life Cycle

A Dissertation Submitted To
The Graduate Council In Partial Fulfillment Of
The Requirement For The Degree Of
Doctor of Computer Science

Department of Computer Science

By

Glenn Gerard Butcher

M.S. Florida Institute of Technology, Melbourne, FL, 1985

B.S. Arizona State University, Tempe, AZ, 1980

Colorado Springs, Colorado

September 1997

Addressing Software Volatility in the System Life Cycle

By

Glenn Gerard Butcher

THE DISSERTATION IS APPROVED

Dr. Charles N. Schroeder

Dr. Mary Jane Willshire

Dr. Richard Fairley

Date Approved

Abstract

Software maintenance costs are well-documented as the largest component of software life cycle cost. While significant research attention has been paid to the characterization of maintainability, program understanding, and software organization for maintainability, there is little work published to date on the a priori identification and encapsulation of a software system's volatile points. This dissertation proposes a methodology for identifying the most probable points in software to be developed to experience the highest frequency of change, and use this postulated information as guidance in developing tools, organization, and techniques to make these points easy to change. In order to validate the fundamental behavior of enhancive software volatility, a volatility-oriented maintenance life cycle cost model is presented to describe the relationship between frequency, cost-to-change, encapsulation strategies, and the resulting life cycle cost. The behavior of this model is then correlated against historical change data from systems of a single domain.

Acknowledgments

The success of this dissertation is directly due to the members of my committee, Drs. Richard Fairley, Mary Jane Willshire, and particularly my chairman, Dr. Charles Schroeder. Their guidance and suggestions turned this “half-baked” idea into a viable approach for addressing software volatility.

A number of people directly contributed their time in facilitating access to the data essential to accomplishing this study, including Lt. Col. Richard Pope, Maj. Dennis Kaip, 1Lt. Geoff Bednarsky, Sydney Rodriguez, Bob Klotz, James Benavedes, Robert Philp, and Roy Garcia of the Space and Warning Systems Center, Peterson Air Force Base, Colorado. The assistance of David Erickson, Joseph Kailey, and Edward Schulz from the Lockheed-Martin Corporation enabled collection of data for the new Granite Sentry.

Special thanks go to Nicholas Zvegintzov, who, through his writings and conversation, provided invaluable insight into the philosophy and process of software maintenance. He also provided incisive comment on the original research proposal.

A debt of gratitude is due to the members of Operating Location SG, Air Force Operational Test and Evaluation Center, Buckley Air National Guard Base, Colorado. These people are my co-workers, and their patience with me during this test of persistence will not soon be forgotten.

I dedicate this effort to Shandra, Eric, and Matthew, hoping it makes even just a small difference in the world whose reins they will take.

But foremost, I owe my personal success in this endeavor to my wife Sherry. You fully and unwaveringly supported me in this endeavor from beginning to end, and now it is time for me to return that support. Particularly, for all that I've learned here, none of it matches the importance of these words that you live and have shown me:

"If I speak in the tongues of men and of angels, but have not love, I am only a resounding gong or a clanging cymbal. If I have the gift of prophesy and can fathom all mysteries and all knowledge, and if I have a faith that can move mountains, but have not love, I am nothing. If I give all I possess to the poor and surrender my body to the flames, but have not love, I gain nothing.

Love is patient, love is kind. It does not envy, it does not boast, it is not proud. It is not rude, it is not self-seeking, it is not easily angered, it keeps no record of wrongs. Love does not delight in evil but rejoices with the truth. It always protects, always trusts, always hopes, always perseveres. Love never fails." [25]

Thank you, Sherry; I love you...

Contents

List of Tables

Chapter 1

Introduction

The maintenance phase of the software life cycle has been amply described as the crux of the “software crisis” [20]. And when one examines the nature of the effort devoted to maintenance, it is determined that the majority of that effort is devoted to changes made to accommodate new or evolving user requirements. It then follows that initiatives targeted to reducing the cost of maintenance, particularly the cost of adaptations, would provide more cost reduction per unit of effort spent than efforts targeted at reducing any other cost. Many cost reducing initiatives proposed in the past few years have purported to address maintenance costs through better software organization (object orientation, structured programming), increasing program understanding, and quantification of attributes such as software size and complexity. But few have directly addressed the costs of change, or in other words the effort required to implement specific changes. Particularly, there is very little documented research to date on the feasibility or effectiveness of predicting change in software yet to be developed. Countless techniques for encapsulating software change points exist in both research and practical application, but there exists no methodology to target their application to the most volatile points of a software system. This dissertation sets out to validate the concepts of software volatility prediction and encapsulation, opening the door for specification of processes to support both activities through the software life cycle.

Two definitions that conceptualize software volatility are given. First, *software volatility* is defined as the propensity for software to change over time in response to evolving requirements. Later in this dissertation, an operational definition for software volatility will be presented. *Software changeability* refers to the ease with which a given software artifact can be changed in response to evolving requirements. The first definition deals with the situation of change; the second with the response to the situation. Note that the definitions consider only enhancive changes, per the categorizations of change presented by Van Horn [30]; corrective volatility, restructuring volatility and, to an extent, adaptive volatility respond to dynamics outside of the realm of evolving user requirements. It is the intent of this dissertation to validate the fundamental behavior of software volatility in order to substantiate the cost-effectiveness of predicting and encapsulating volatile points in software. The long range goal of this research is to "operationalize" software volatility by inserting techniques in the software requirements definition and design processes to make software more malleable - promoting software changeability targeted toward anticipated requirements volatility.

1.1 Statement of the Problem

Relatively speaking, all software is "easy" to change. The name "software" implies a manifestation that is malleable; receptive to modification. The software-based functions of a system do take much less intellectual and physical effort to change than the hardware based functions, both in the design and implementation of the change as well as its promulgation among multiple copies. The architectural concepts presented by John Von Neumann for stored program computers resulted in a class of machines with flexibility to handle a much broader range of problems, in ways tailored to individual preferences, than any other machines introduced previously.

But, while the physical act of modification is simple, making software have its intended effect is less so. Programmers usually are too busy getting the program to work as intended the first time to worry about making the program easy to change later. And, once a program is put into use, subsequent changes to accommodate changing user needs may be simple or complex, depending a lot on the nature of the change and how the implementing analyst and/or programmer constructed the area(s) affected.

It should be clear that there is great cost-saving potential in any advancement that encourages the design of software to accommodate change. One rationale to support this assertion is the same used in code profiling: execution of the program under a code profiler shows where the program spends its computer cycles - the areas that occupy most of the execution time become prime targets for optimization. However, most research on maintainability to date is on its characterization, in the hope that, by showing good and bad characteristics, maintainability researchers will encourage developers to design and code more maintainable software. It is for this reason that the term “changeability” is preferred in order to distance the proactive approach of building in changeability advocated in this work from the after-the-fact characterization of maintainability.

Actually, the research collected to date suggests three hierarchical levels that, combined, serve to characterize a software system's ability to accommodate volatility. From lowest in potential to highest, they are:

1. Maintainability: The general aspects of software structure - modularity, object orientation, coupling, cohesion, etc. - that facilitate all aspects of maintenance, from defect correction to enhancement.

2. Changeability: Requirements analysis and design undertaken to specifically accommodate anticipated enhancive change.

3. Domain Engineering: The top-to-bottom design of a software system as a foundational architecture of components common to the application's domain, capped by an integrating layer that allows the use of the components to be tailored to fit specific applications. This requires specification of an interaction model for the domain's components that covers most possible application situations.

While all three of these aspects are important, it is upon the second of these three that this inquiry will focus: validation of the life-cycle benefit of attempting to predict the most volatile points in a software system, and encapsulation of these points with mechanisms that will lower their cost-to-change.

1.2 Hypotheses

The following hypotheses have been selected as the key indicators of the validity of incorporating software volatility point prediction and encapsulation techniques into software development processes:

H1: Distribution of frequency of enhancive change in a system is not uniform.

This hypothesis asserts that a relatively small proportion of a software system experiences frequent “revisit” to implement enhancive change, validating the benefit of making these parts as easy to change as economically possible. This assertion parallels the economic argument for reuse, in that extra effort to encapsulate a change point has the potential for positive amortization over the reduced unit cost of a large number of anticipated changes during maintenance.

H2: Within a domain, the most volatile objects of change are common to all systems.

This hypothesis asserts that common forces of enhancive change exert upon all systems of a domain, validating the effort to use change histories of existing systems to predict patterns of change in new systems of the same domain.

H3: Effort expended in software development to encapsulate a volatile point reduces the cost to change the encapsulation object in maintenance.

This hypothesis captures the essential motivation for conducting volatility prediction and encapsulation - life cycle cost reduction. Its basis is the demonstrated tendency of volatility to cluster in a relatively small number of modules of a system, which correspondingly map to a certain few change drivers that occur with frequency greater than 1 during the maintenance life cycle. If there are no change drivers that occur more than once in a software system, then expending effort to predict and encapsulate volatile points would not be worth the effort saved in making changes.

1.3 Assumptions

The following assumptions serve to focus this inquiry on those aspects of software volatility that show the most promise in reducing life cycle costs:

1. The effort required to make software changeable will compete with the effort required to implement the required functionality and performance. Given that most software development efforts have finite resources, the framework for incorporating changeability should target the most "volatile" parts of the system. "Enhancive volatility" is defined with respect to software change as the propensity for a software system to change over time in response to changing user requirements. Consequently, "volatile points" are specific places in a software system that are expected to change during the life cycle. This definition is used in postulating modifications to the software development process to incorporate changeability.

2. Every concept of software organization described to date implicitly supports changeability. Structured programming, encapsulation, abstraction, objects and their polymorphism and inheritance -- all of these, when practiced to

the letter and intent of their authors, help to reduce the effort required to implement subsequent changes. Well-organized software is relatively easier to change. However, the focus of this research will be on application of mechanisms targeted to facilitating specific changes.

3. The techniques to predict and encapsulate software volatility are independent of the particular software development methodology used. Indeed, these techniques should enhance the cost-effectiveness of any methodology by targeting its design phase(s) toward implementing change mechanisms that cut the cost of change.

1.4 Conceptual Framework

There are two fundamental activities required to make a software component easier to change: 1) identifying the need to make it changeable, and 2) encapsulating the component in an appropriate change mechanism. The first step is rooted in the requirements gathering and analysis process, the second in the design and implementation process. Therefore the focus of incorporating the concept of volatility in the software development process is on the identification, prioritization, and encapsulation of volatility points in a software system.

Identification of volatility points is a lot like identifying risk in software development: you don't know if you were right until you get there. Like lottery numbers, the actual change history of a software system isn't precisely known until it happens.

Sorting software systems into the following categories provides a rough idea of the certainty that can be expected in volatility point identification:

1. Existing Systems - Software that is currently in use. It has a history of change, which is maintained formally or at least in the minds of the maintainers, that can be captured and analyzed to identify volatility points.

2. Replacement Systems - Software to be developed to replace an existing system. This software can also make use of the existing system’s change history. But replacement software may provide functionality that changes the expectations of its users, consequently altering the enhancive volatility from that expected based on analysis of the previous system.

3. “Brand-New” Systems - Software to be developed to meet a new need. It is for these systems that volatility point identification is most problematic. Approaches could include user interviews to identify anticipated changes in requirements, and identification of existing systems in the same domain and/or architectural category whose change histories can be examined for correlations.

Volatility identification for all three categories of systems is an exercise in prediction based on prior events. The categories present a continuum of uncertainty based on decreasing correlation of past experience.

Using the change histories of existing systems to anticipate volatility of new systems in the same domain requires a transformation from the system-specific volatile points to a general specification of the driving requirement. For this reason, the objective of existing system volatility analysis must be the identification of *change drivers,* which are descriptions of the user requirement driving the enhancive modification. Change drivers should generically describe the data and/or behavior to be changed in the system. Generic specification should allow change drivers to be applied to software systems across the target domain.

Prioritization of change drivers must account for a number of factors that will influence the selection of the most cost-effective volatility points to encapsulate with appropriate change mechanisms. This prioritization must take into account the frequency of anticipated change, along with the criticality of the change, the required response time, and identification of volatile points that are affected by more than one change driver. Prioritization should support the encapsulation of the volatile points that result in the biggest positive impact to the system's life cycle cost, given that we can't afford to inflict the same level of treatment to all identified change drivers.

All of the means available to encapsulate volatile points must be understood in terms of their "bang for the buck" -- how much they cost relative to their potential benefit. This knowledge is essential to match prioritized volatility points with appropriate encapsulation mechanisms. Other factors influence this pairing, such as the nature of the operational environment, required responsiveness, and configuration management constraints, but cost-benefit is the overriding consideration.

It must be recognized that a given volatile point will have multiple encapsulation alternatives, and the selection of a volatile point for encapsulation may well depend on identifying a cost-palatable mechanism. Therefore, the software implementation process must include an activity to identify <u>all</u> appropriate encapsulation mechanisms for each candidate volatility point so that cost-benefit tradeoff decisions can be made with full knowledge.

1.4.1 Definitions

Prior to presenting the methodological and cost models, it is necessary to propose operational definitions of key terms and concepts:

1. Software Volatility: generically defined to be the ratio of system elements changed to the total number of system elements for a given period of time. Software volatility is expressed as a percentage. This definition can be instantiated in a number of ways:

    a. with respect to the foundational metric of "system element." This element can be lines of code or number of modules. Modules could be further discriminated into categories, i.e. data vs. code.

    b. with respect to the time period. Volatility can be calculated for a release, or for a time interval, or for a life cycle, depending on the comparative analysis to be conducted.

c. with respect to change type. The standard enhancive, adaptive, corrective, and perfective categories can be used. This dissertation will make particular use of characterization of enhancive volatility.

This measure of software volatility meets the definition of a ratio scale in that its values express an interval relationship and the scale has a proper zero value, 0/total elements = no volatility.

2. Volatility Topology: This is the characterization of historical volatility for a specific system, expressed using Pareto analysis of module change frequency for a specific time interval.

3. Volatility Exposure: In the manner of risk exposure per Boehm [9], volatility exposure is a combinatorial measure whose components are frequency of change and cost to change for a given change driver. Both components represent the key considerations in encapsulating a volatile point in a software system, and their product provides a value for prioritization in the methodological model.

4. Volatile Point: A place in software where change occurs. At this time, its locality can be as broad as a module or as specific as a single identifier. Its primary purpose is for identification of encapsulation candidates.

5. Change Driver: A requirement that changes, thus stimulating software volatility. A change driver is identified by performing a "bottom up" analysis of historical change, identifying the most frequently changed modules and

determining the requirements that drive their change. Given the differences in specific systems, volatility analysis must produce change drivers in order to provide relevant information that is comparable across systems of a domain. This dissertation particularly concentrates on enhancive change drivers, those that are rooted in changed user expectations.

6. Encapsulation: More widely recognized as an attribute of object orientation, in this dissertation encapsulation refers to the specific act of constructing a change mechanism around a volatile point to reduce the cost of making a change. Encapsulation can take any manner of form, to include parameterization, interactive tools, and documentation, either singly or in combinations.

The following sections describe the two models of software volatility used in this dissertation. The cost model describes the enhancive software maintenance life cycle from the perspective of software volatility, parameterizing the key volatility attribute of change frequency in its role in driving life cycle cost. The methodological model proposes a process for addressing software volatility in the development phase of the software life cycle based on change frequency. Validation of the cost model will in turn validate the utility of the methodological model in lowering life cycle cost.

1.4.2 A Volatility-Oriented Enhancive Maintenance Life Cycle Cost Model

A parametric cost model captures the fundamental behavior of a given effort environment and allows the application of cost drivers with known behavior to influence the fundamental behavior. Development of a volatility-oriented cost model for enhancive software maintenance was undertaken to provide a formal vehicle for describing volatility behavior and to support the proposed process for volatility analysis and encapsulation. This model has no mechanisms for calibration, and is thus not intended for use in estimating costs of actual enhancive maintenance programs. The cost model consists of two equations described below.

1. Life cycle cost of a single enhancive change driver ($SM_{\Delta LCD}$): This equation takes the cost to perform a single change and extends it through the maintenance life cycle based on anticipated frequency and postulated frequency behavior. This equation is used to represent the life cycle cost of an identified change driver, a recurring requirement for change:

$$SM_{\Delta LCD} = \sum_{i=1}^{y} \left(SM_{\Delta C} \cdot F \right)_i \qquad \text{(Eq. 1)}$$

Where:

$SM_{\Delta C}$	=	Cost to inflict an instance of the change driver;
F	=	Change Frequency;
y	=	Life Cycle in Years.

2. Enhancive Maintenance Life Cycle cost (SM_{MLC}): This equation relies on the identification and prioritization of change drivers for the system to be developed. The change drivers are then each modeled using equation 1, and the results summed in equation 2 as follows:

$$SM_{MLC} = \frac{\sum_{j=1}^{m_p} \left(SM_{\Delta LCD}\right)_j}{p} \qquad \text{(Eq. 2)}$$

Where:

p	=	Overall percentage of effort to be incorporated in the significant change drivers expressed by *m*.
m_p	=	Number of significant change drivers for a given percentage of effort *p*.

The determination of *m* is a central question of this dissertation. *m* characterizes the tendency of volatility to cluster in a few change drivers; in other words, a certain few changes are made regularly. Specifically *m* in the model is the number of “significant” change drivers of the domain. The first research objective, characterization of software volatility, has as its central question, “How many change drivers are significant?” Expressed in percentages, a Pareto analysis of change drivers rank-ordered by frequency describes the relationship, “A% of the change drivers are responsible for B% of the total change effort.” The assertion of this dissertation is that a statistically significant A%-B% relationship can be identified for <u>all</u> software supporting a given domain, and that significant values of *m* can be reliably specified for each domain at a user-specified percentage of maintenance effort.

1.4.3 Methodological Model

The specific activities undertaken in software development to address adaptive volatility should produce a system with lower maintenance costs, owing to the system's built-in flexibility. Toward this end, the following methodology is proposed to target development resources toward encapsulating the volatile points with the most "bang for the buck:" those that result in the greatest maintenance cost savings. This model is not dependent on any particular development methodology; instead its steps relate to the canonical activities of requirements discovery, design, and implementation, present in some form in all development methodologies. Each sequential activity is listed below and described in detail in subsequent sections:

1. Identification of all anticipated change drivers. This activity would rely primarily on the demonstrated volatility of existing systems, obtained from analysis of release histories and user interviews.

2. Analysis of each change driver to assess its volatility exposure in terms of its frequency of occurrence and magnitude of effort.

3. Prioritization of the change drivers by their anticipated volatility exposure.

4. Input of the subset of change drivers that are anticipated to account for the bulk of adaptive change into a volatility oriented cost model to produce trade studies of encapsulation alternatives in terms of their relative cost-benefit.

5. From the model-based analysis, development of a prioritized list of encapsulations for pursuit in system design and development

6. System developers implement as many of the encapsulations, from the top of the list to the bottom, as the available resources will permit.

1.4.3.1 Change Driver Identification

This is the most problematic aspect of addressing software volatility, for the essential act is one of predicting future events. The only information that provides any chance of supporting reliable predictions of software volatility is the history of change in similar systems serving similar customers. Within this continuum, the best circumstance is the identification of volatile points within an existing system for attention with perfective maintenance; the next-best circumstance is developing the replacement of a system with a well-established change history and a stable customer base. The first circumstance provides the most risk-free environment for volatility identification, followed closely by the second. The element of risk added in the second circumstance comes primarily from the change in customer expectations with the introduction of a new tool for doing their business. This element of risk rises as the opportunity to study change histories of similar systems decreases, as suggested by the categories proposed previously in this section. The concept of domain serves the task of identifying similar systems quite well, with the expectation that systems serving in the same application domain will suffer similar pressures of change. However, assuming that systems serving the same domain have similar volatility topologies is overly simplistic, considering that the same customers may have different evolution requirements for each system. So, any effort to predict software volatility should address both historical change and the specific expectations of the customer base.

The analysis of historical change in a system establishes its volatility topology. This topology expresses a number of characteristics of the system's change history, to include the frequency of releases and a Pareto analysis of change frequency by module. Successful determination of a volatility topology assumes the existence of release documentation that identifies when the release occurred, what change requests the release satisfied, and the modules "touched" to satisfy each change request.

The volatility topology is then used to identify the system attributes subjected to change. These attributes will have names such as "system users," "withholding rules," or "map displays" identifying data or behavior subject to change. This attribute identification is accomplished by performing semantic analysis of the titles and descriptions of the change requests that drove change of the volatile modules. Also identified in this step are the types of changes the attributes undergo, such as "delete," "add," "update," or "move." The combination of the attribute with the types of changes inflicted comprises a change driver for the system.

1.4.3.2 Change Driver Analysis

The intent of this step in the model is to identify the volatility exposure of the system to the change drivers identified in the previous step. The concept of exposure is similar to the one defined for risk analysis by Boehm [9]. The two components that contribute to volatility exposure are the frequency of occurrence and the cost to accomplish the change. The list of change drivers identified in the previous step of the methodological model carry frequency information established in the process of developing the volatility topology. However, establishing the cost to perform a change can be difficult in light of the sparse efforts to collect such data among maintenance organizations. In the absence of historical cost data, the number of modules "touched" to effect a change driver can serve as a coarse indicator of effort required.

Calculation of a single-valued exposure measure in the manner of risk exposure (RE):

$$RE = P(UO) \times L(UO) \quad [9]$$

Where:

UO - Undesirable outcome;

P - Probability of occurrence;

L - Magnitude of loss if risk occurs.

suffers from ambiguity of relative magnitude between two close values. It is proposed that volatility exposure be presented as an ordered pair (f,c) where f = frequency of change and c = the cost to implement the change.

1.4.3.3 Change Driver Prioritization

Rank-ordering of the identified change drivers is essential to ensuring that attention is applied in development to the software components most likely to change. The volatility exposures identified in the previous step form an important starting point in this effort, but the actual ordering is best established in a review process that involves the system engineers, project managers, and customer representatives. A number of outcomes besides a rank-ordered list come from such a process. First, the identified change drivers undergo a “sanity check” from review in the context of relative importance by the participants. Second, the participants develop an ownership in the list and its intended purpose of directing encapsulation of likely volatile points in the system to be developed.

The prioritized list deserves capture and control as a configuration item so that subsequent changes are effected only after a directed and documented process involving exposure reassessment and review by the original list participants.

1.4.3.4 Trade Studies

This step coincides with design activities in the software development methodology. The first activity in this step is to determine how many change drivers will be addressed with encapsulation mechanisms, *m* in the cost model (Eq. 2). While the ultimate determination of effort to be devoted to encapsulation will be based on the resources available, this determination should also depend on how many change drivers account for most of the anticipated change.

The second activity is the development of encapsulation alternatives for each change driver. Intuitively, a tradeoff between the cost to develop the alternative versus the cost savings as a function of cost to make a single change times the anticipated frequency is the prime concern; however, characterization of this tradeoff is beyond the scope of this dissertation.

1.4.3.5 Selection of Encapsulation Initiatives

In this step, the population of identified encapsulation alternatives is considered for implementation. This is proposed as another review process conducted with the same participants as the change driver prioritization. Multiple encapsulation alternatives for each change driver are to be considered, both among themselves and in their relation to the alternatives for the other change drivers. It is in this review that synergistic relationships between encapsulation alternatives can be identified and consideration given to their combination into single mechanisms. Alternatives involving data parameterization are probably the most amenable to combination under one change mechanism. Also, the resources available for encapsulation efforts are to be considered. The product of this activity is a list of volatility encapsulation initiatives to be incorporated into the software development effort, rank-ordered in priority of cost-benefit based on the preceding volatility analysis

1.4.3.6 Encapsulation-Oriented Development

In this step, volatility encapsulation mechanisms are developed hand-in-hand with the functional core of the software system, in the priority order specified in the previous step.

1.5 Research Method

This research pursued two main objectives: 1) validation of the fundamental behavior of software volatility, and 2) validation of the fundamental relationships described by the volatility-oriented maintenance life cycle cost model. The domain of integrated tactical warning/attack assessment correlation systems supporting aerospace defense of the North American continent was chosen as the universe of discourse for these two inquiries. Each inquiry objective was to be met using software change data collected from the maintenance efforts of each system in the target domain. For the two objectives, if the objective could not be met in the target domain, then it was to be concluded that the postulations of the objective could not be universally applied across all domains. The research conducted in support of this dissertation was considered to be complete when sufficient data to support assessment of the proof criteria specified below was collected from the subject domain. Each of the two inquiries is described in the following sections.

1.5.1 Volatility Validation

One of the fundamental objectives of this inquiry was to validate the propensity of adaptive software change to cluster in a relatively small percentage of the code. Without evidence of this elemental tendency, performing volatility prediction and encapsulation would not be worth the effort. In the cost model (Eq. 2), this parameter is captured by *m*, the number of significant change drivers. The supporting research methodology proscribed collection of software release data from the following systems in the universe of discourse:

1. Command and Control Processing and Display System (CCPDS) and its replacement (CCPDS-R);

2. Space Defense Operations Center (SPADOC);

3. Granite Sentry, both the Phase III system and its replacement.

This data was used to establish each system's *volatility topology*, in the manner described in the methodological model. From the volatility topology, each system's change drivers were identified and prioritized, again according to the methodological model. Then, for this inquiry, correlation between the lists was established, both with regard to the presence of common change drivers and their historical frequency of occurrence.

The objective of this inquiry was to show that volatility does cluster in certain modules, both as a function of frequency in a single system and as a correlation of high-volatility change drivers among systems of the same domain. The parameter *m* was also defined for the subject domain, the number of change drivers significant to enhancive maintenance in integrated tactical warning and attack assessment correlation systems.

Proof Criterion 1: Given D = the subject domain, S = a given system, n_f = the number of modules experiencing frequency of impact f: For D(S), $n_f > n_{f+1}$.

(For a given system of the subject domain, the number of modules experiencing a given change frequency is greater than the number of modules experiencing the next higher change frequency.)

Proof Criterion 2: Given D = the subject domain, Sn = a given system, and COL_{Sn} = Change Object List of system Sn: For all D(Sx, Sy), r_s (COL_{Sx}, COL_{Sy}) is significant at 99%.

(For all combinations of Sx and Sy of the subject domain, the coefficient of correlation of the systems' change object lists is significant at 99%.)

1.5.2 Cost Model Validation

The objective of this inquiry was to validate the fundamental behavior of the volatility-oriented maintenance life cycle model by testing the hypothesis. The hypothesis was tested by comparing projected maintenance life cycle costs of both the operational and final Granite Sentry systems. The following criterion was established:

Proof Criterion 3: Given ΔSD = Change Cost in Staff-Days: $\Delta SD_{GSold} - \Delta SD_{GSnew}$ is influenced by volatility encapsulation.

(The difference in the cost to change the old Granite Sentry from the cost to change the new Granite Sentry is influenced by volatility encapsulation)

1.6 Summary

This chapter presented an overview of the concept of software volatility, to include three hypotheses that direct the research, a conceptual framework for software volatility that includes both cost and methodological models, and a research method designed to support validation of the three hypotheses. Chapter 2 presents an analysis of the relevant literature that forms the basis for the concept of software volatility, Chapter 3 outlines the prosecution of the research method, Chapter 4 presents the analysis of the research, and Chapter 5 asserts the outcomes of validating the hypotheses and presents both recommendations for incorporating the methodological model in software processes and directions for further research.

Chapter 2

Analysis of the Literature

In this chapter, the following areas are discoursed with the intent of establishing the foundation for the research presented herein: 1) motivational research, 2) previous direct research, both with regard to software volatility as defined in Chapter 1 and software maintenance cost modeling, and 3) related research with regard to software volatility prediction and encapsulation.

Central to the discussion of any aspect software maintenance is the definition of categories of changes conducted during maintenance. Originally, Swanson [29] proposed the categories of corrective (response to failures), adaptive (changes in data or processing environments) and perfective (increasing efficiency, performance, or maintainability). Lientz and Swanson later surveyed 120 organizations and reported a significant subcategory within perfective maintenance, enhancements (user demands for enhancements and extensions) [22]. Van Horn, in [30], offers a concise definition of enhancement: “modification to meet new or unrecognized user requirements.” These four categories of maintenance serve as the basis for discovery in this study, with prime attention paid to enhancements.

2.1 Motivational Research

In order to understand the need to address evolvabilty of software, it is necessary to visit the foundational research into its nature. The basic need for addressing the cost of maintenance has been presented many times, supported primarily by the proportion of software activity devoted to maintenance. This percentage has been measured at anywhere from 50% to 80%. Further, the proportion of effort devoted to enhancements has been variously measured and surveyed in the range of 39% to 59% (see Dekleva and Zvegintzov [12] or Hops and Sherif [19] for compendia of studies), always the largest proportion in any of the reports. It should be clear that attempts to reduce the cost of enhancive changes to software are targeted at the category of changes with the most potential for cost savings.

Belady and Lehman [6] first described a sub-discipline they named, "program evolution dynamics" in 1974, proposing three laws of evolutionary behavior based on statistical analysis of releases of the IBM OS/360 operating system. The first rule sums up the elemental state of software volatility as follows: "A system that is used undergoes continuing change until it is judged more cost effective to freeze and recreate it." Parnas published a paper in 1979 [24] where he clearly defined the circumstances that call for "design for change." It was intuitively obvious to these researchers early in the evolution of software development into "programming in the large" that anticipated change should be a key motivator in requirements analysis

Van Horn, in [30], calls for “evolvability as a design criterion,” and the “(preservation of) evolvability during evolution.” However, he then asserts that preserving evolvability is to be done through periodic restructuring of software. He then implies that, “we need not be so concerned with having the best structure when the software is created. Any flaws in structure can be healed as the software evolves.” This thinking runs counter to the use of program structure as a tool of evolvability, presented in Section 2.2.

The need for software evolvability has also surfaced in management and policy circles. Horowitz [20] provides examples (mainly from DoD) that illustrate the benefit of timeliness’ independence from cost, the increasing cost of maintenance in the later years of the life cycle, and the importance of specifying and adhering to a structured architecture in the subsequent incorporation of major system changes, such as porting to new machines and upgrading system services. He also points out the tendency for software developers to spend much effort on implementing a customer’s functional and performance requirements as understood at the time, with little concern with how they will change during the life cycle. Most telling is his cite of an unnamed survey that asked 123 businesses what they thought were the government’s most important concerns when awarding software contracts - ease of maintenance and maintenance cost ranked 8th and 9th out of 10. His statement: “The basic problem is the lack of a strong requirement for modifiability that facilitates software maintenance.” However, the basic need for changeability engineering is well-recognized by DoD software managers, substantiated in the recommendations of the Software Logistics Workshop [26] which refers to "quantify(ing) the propensity for change in weapon system(s)," "(developing) a model for determining the optimum level of maintenance at which software changes should occur." Horowitz [20] provides a concise summation of the problem: "Software does provide flexibility, but it must be designed from the start with an architecture that allows it to do so."

2.2 Previous Research

This section presents the foundational research for this dissertation. Both the process model and the cost model are substantiated by significant prior work.

2.2.1 Software Volatility

The first significant discussion of software volatility was presented by Belady and Lehman in [6]. They called it "complexity" and defined it as, "the fraction of the released system modules that were handled during the course of the release." Their notion of volatility forms the basis for the operational definition described in the previous chapter.

A few researchers have pondered the phenomenon of volatility in software, and have proposed methods to deal with it directly related to the models presented herein. Land [21] provided the following fundamental concepts:

1. Uncertainty of potential software changes grows greater the further in time volatility point prediction is attempted, to the point where a cost-feasible design cannot be conceived to meet the range of expected changes. This future time is called the *forecasting horizon.*

2. Limitations in tools and techniques prevent development of systems with infinite flexibility.

3. "It is generally cheaper to build a dedicated, highly specific than a generalised, flexible one."

4. However, the cost of a flexible package may be cheaper if its cost can be distributed among multiple customers.

Land's concepts provide important constraints to expectations of volatility analysis and encapsulation, particularly with regard to the forecasting horizon. It should be evident that this horizon will be shorter than most software life cycles, and that a certain amount of volatility cannot be anticipated.

Land provides guidelines for designing systems that meet changing user needs, to include:

1. Use analysis, design and evaluation techniques that involve user participation to achieve as accurate a model of the real world as possible.

2. Use design methods which incorporate experimentation and prototyping.

3. Attempt to distinguish between the stable and volatile aspects of the system.

4. Avoid early commitment to a particular design.

5. Adopt designs that can cope with a range of possible futures.

6. Use future analysis to craft viable predictions. Land offers a methodology for future analysis as an appendix to his paper.

7. Build flexibility into the system.

8. Use new hardware and software technology to develop small systems that are easily replaced.

Land's treatise provides an excellent bounding of the scope of the problem, that of dealing with a system's ability to meet multiple possible futures. His particular emphasis on user involvement can help to address volatility, but the onus is still on developers to present volatility as a concern to involved users if their feedback in that area is expected.

Podger [27], "postulates that any system can be divided into:

1. An inner zone of basic values and principles, which it would take a revolution to change.

2. An intermediate zone of general procedures which are subject to change but where the lead time between the change being formulated and required is quite long.

3. An outer zone of specific procedures, subject to more rapid and frequent change." (from Land [21])

Podger's topology implies a continuum of volatility requirements based on the needed response for a change. This response time continuum forms a component for determining the prioritization of change drivers.

However, it is with Hager [16, 17] and Bækgaard [5] that the fundamental precedent research in software volatility is presented. Both describe specifically a methodology of volatility analysis that is fundamentally composed of an identification activity and a prioritization activity. Hager proposes encapsulating volatility with program structure incorporating information hiding and abstraction per Parnas, while Bækgaard advocates a combination of program structure and parameterization. Bækgaard also offers the following questions to drive volatility analysis:

1. Which system properties are likely to change?

2. Who should be enabled to make the changes?

3. How are volatile properties to be bound to the software to facilitate change?

The second question implies an important demarcation in determining the manner of addressing a volatile component of a system: that of deciding whether a particular change mechanism is to be manipulated by programmers or system users. At this demarcation, the cost to effect a given change will drop significantly when responsibility is migrated from the hands of the programmers and their maintenance process to the users. Correspondingly, the implementation of a change mechanism usable by system users will cost more than a change mechanism targeted to the maintenance process. This dissertation proposes to extend their conceptual definitions to a practical, results-oriented methodology suitable for incorporation into any software development endeavor.

A few “real world” software development efforts have attempted to directly address software volatility and the need to predict where software changes will occur in maintenance. Hager, in [16, 17], illustrated the previously described concept of hiding volatile system properties to promote changeability. Floyd [14] describes the efforts of the F-22 Advanced Technology Fighter (ATF) contractor, Lockheed-Martin Corp., in the identification of historical volatility trends in other software-dependent fighter aircraft for use as a guide in the design of F-22 flight software. Floyd, et. al., determined that, at the Computer Software Configuration Item (CSCI) level, the information required to effectively target change mechanisms was the probability of change, the size of the change, and the category of the change (Corrective, Enhancive, or Adaptive). They also presented survey results from software design leads on the F-16 C/D program on the “predictive volatility” of the various software components of this highly volatile program (many different software versions to support changing missions, foreign military sales, etc.). A more direct experience with software volatility is the Granite Sentry program, one of six hardware/software upgrade programs that comprise the Cheyenne Mountain Upgrade, a $1.6 billion overhaul of the missile, air, and space warning data correlation systems of North American Aerospace Defense Command (NORAD) and United States Space Command. This multi-phased program used the change experience from over 20 years of software releases to the operational warning systems in Cheyenne Mountain, plus their own experience with three previous phases of Granite Sentry development, to attempt a prediction and encapsulation of volatility points in the final system, to be delivered for operational use in the latter part of 1996. The experiences of these programs provide valuable insight toward the development of volatility prediction and encapsulation methodologies and the need for a concise methodology that can be easily and consistently applied during the event of software development. The Granite Sentry program was chosen to contribute to the data collected to support this dissertation.

2.2.2 Software Maintenance Cost Modeling

Parametric cost estimation provides an unambiguous venue for the presentation of cost relationships. Mathematically, parametric models are sets of processes where system characteristics are mapped to appropriate ranges of cost [11]. Application of parametric models to software development rides a significant body of research, most of it intended to provide cost estimation tools for software development practitioners. However, the number of non-linear influences of software development keep it from being parametrically modeled without use of post-model modifiers. Still, the discipline offers a concise method for describing cost relationships for use in process model validation, the intent of this dissertation.

To date, only a few extremely fundamental cost relationships are presently understood with regard to software. The two most widely recognized cost relationships, as evidenced in their application in a large number of cost models, is the linear relationship between software size and the effort to produce it, and the distribution of effort over development phases based on the Rayleigh model. Despite their coarseness, these two relationships provide a foundation upon which the variability of software cost estimation can be baselined. Probably the most significant influence upon estimation variability is that of the subject domain, as evidenced in its visibility in the modifiers and structure of most cost estimation models. Most manifestations of domain in cost models are in the form of "complexity," with the continuum of scale ranging from information system applications "to" real-time and embedded applications. COCOMO's "mode" [8] is the most well-known manifestation, with contributing attributes of organizational understanding, experience with related systems, conformance with pre-established requirements and interface specifications, concurrent hardware and procedure development, need for innovation, and product size contributing to the selection of effort and schedule equations. Note that identification of these attributes is essential to proper application of COCOMO's mode, for there is no universally accepted definition of complexity. One of the fundamental assertions of this dissertation is that identification of domain is essential for the categorization of software volatility.

The effort to define cost relationships related to software maintenance is long-running. However, most efforts to develop maintenance life-cycle cost models have not reached a level of granularity below the software release, and few are supported with validation based on actual project costs. This previous research does provide valuable insight into the primary influences of maintenance cost for use in developing a change-level, volatility-oriented cost model.

Holchin [18] provides a summary of the primary maintenance cost estimating relationships (CERs) described in the literature. The most granular of these is the maintenance/development effort ratio, which describes maintenance life-cycle cost as a percentage of the overall life cycle cost. Boehm [8] describes this relationship as follows:

$$E_M = (M/D)E_D$$

Where:

M/D	=	The maintenance/development ratio;
E_D	=	The effort required for development;
E_M	=	The effort required for maintenance.

The maintenance/development ratio is applied to the effort required for development to obtain the life-cycle effort required for maintenance. Holchin relates estimates for the proportion of life-cycle cost devoted to maintenance from 40% to 82%, which is consistent with other observations in the literature. From this variability, it should be apparent that this relationship serves no practical purpose in describing maintenance cost behavior and should serve only as a rough indicator of the magnitude of the problem.

Holchin also describes a level-of-effort relationship based on programmer productivity in thousands of source instructions per programmer and provides one observation of maintenance coverage related to real-time and aerospace software - 8K - 10K SLOC per programmer per month. This relationship's significant qualifier, like that of most software cost estimation relationships, is dependent on the subject domain.

But probably the most well-recognized maintenance cost attribute in the published models is that of annual change traffic. Usually expressed as a percentage of code changed in a year, this attribute is a direct application of the operational definition of software volatility in costing maintenance effort. It must be recognized that this attribute is dependent on software size as a scaling factor. Widespread use of this volatility-oriented attribute is evidenced in an excellent comparison of the life cycle support capabilities of the most well-known software cost modeling tools by Ferens [13].

Probably the most well-known maintenance cost model was described by Boehm [8] as part of COCOMO:

$$MM_A = (MM_{Nom})(ACT)(P_M)$$

Where MM_A is the annual man-months, MM_{Nom} is the man-month estimate from the nominal intermediate equation, ACT is the annual change traffic (the fraction of the code changed per year), and P_M is the product of maintenance multipliers. This model recognizes the fundamental contribution of volatility to maintenance life-cycle cost as ACT. It is interesting to note that the current proposal for COCOMO 2.0 [1] abrogates both volatility and the life cycle cost approach by addressing maintenance as reuse. REVIC [3] is an automated version of COCOMO that uses new project data to calibrate the original COCOMO equations and provides additional risk and phase distribution data; it takes the COCOMO maintenance equation and allocates annual cost to a fifteen-year life cycle. It also accelerates annual cost in the first three years to account for resolution of residual errors from development. For both COCOMO and REVIC, no validation is presented of the behaviors of volatility or residual errors using actual project data.

It should be clear from this review of the state of cost estimation that there is a need to carry software maintenance cost estimation to a more detailed level, and to validate the resulting model with actual project results. This dissertation proposes a new level of granularity based on software volatility.

2.3 Related Research

In this section is presented research that, while not directly related to the topic of software volatility, contributes perspective, definitions and techniques.

2.3.1 Volatility Identification

While the methodological model proposed herein for addressing software volatility has its roots in the methods specified by Hager [16] and Bækgaard [5], many of its specifics come from the techniques of risk analysis. Boehm in [9] provides a concise overview. Use of the risk process model as the basis for the volatility analysis methodological model provides a well-understood process framework that captures the need to prioritize volatility encapsulation alternatives among themselves. This allows project resources "robbed" from implementing functionality and performance requirements to be directed where they will do the most good and gives volatility encapsulation the best chance to survive in software design and implementation.

2.3.2 Volatility Encapsulation

The research to date that directly addresses the engineering of software changeability has mostly focused on generic attributes of modifiability without regard to their cost tradeoffs. One publication, [4], taxonomizes the generally understood attributes that make software more maintainable, but it does not address the specific process of making software more changeable. Another study, [10], specifies software quality factors in three categories: performance, design, and adaptation. The adaptation category contains the following factors: expandability, flexibility, interoperability, portability, and reusability. Of note is their chart that describes positive and negative interrelationships between the factors in all three categories. In particular, the chart shows the negative effect of increases in both expandability and flexibility on virtually every performance factor, which emphasizes the need to prioritize changeability requirements in order to properly trade them off against performance. The methodological model provides two places where such prioritization is undertaken.

Parnas, in [24], provides a key distinction in encapsulation, that between software generality and software flexibility. He defines generality as the ability for software to "...be used *without change* in a variety of situations," and flexibility as the ability for software to be "*...easily changed* to be used in a variety of situations." Parnas' issue between the two was the "run-time cost to be paid for generality" vs. "the design-time cost to build flexibility." It should be recognized that the fundamental difference between generality and flexibility is where lies the responsibility for adapting the software, in the users or the developers. Parnas further stated that, "the decision (between generality and flexibility) should be a conscious one." Volatility analysis provides the context in which to make these decisions, based on anticipated frequency of change and the cost to change.

The research topology on the role of software tools in supporting change is comparable to that for compiler optimization: many good ideas, no unifying theme. In some cases the purpose of a given tool has nothing to do with its eventual use as a change mechanism; data base management systems, for example, play a significant role as a repository for configuration and behavior parameters in some software systems. Martin [23] provides extensive discussion on design techniques to facilitate maintenance, to include source code organization and structuring and use of data base management systems and fourth generation languages; however, he does not discuss the process of predicting and encapsulating volatility. A software construction technique called "parameterized programming" espoused the encapsulation of software parameters deemed likely to change in various mechanisms to facilitate ease of change. Goguen provides a complete treatment of parameterized programming in [15] , calling for language facilities to support the parameterization of both data and algorithms. Experience revealed that parameterization did not necessarily make changing the software less complex [28], and that the effort spent encapsulating every software parameter easily surpassed the marginal cost-benefit of changeability [7]. It should be clear that volatility analysis would serve to focus the application of parameterization and other encapsulation techniques to the places where they would produce the greatest benefit for the cost.

In summary, volatility-oriented methodological and cost models offer a unified process for comprehensively addressing the concerns and concepts identified in the preceding research. The need for such a framework has been clearly and repeatedly stated, and the models provided herein offer a practical application of the preceding research for use by software practitioners to identify and prioritize encapsulation mechanisms in terms of their benefit versus cost. Additionally, the research conducted to validate the hypothesis forms the framework for establishing the volatility topologies of other domains in subsequent research.

Chapter 3

Research Method

The research conducted to support the hypotheses consists of two components: 1) validation of the fundamental behavior of enhancive volatility, and 2) confirmation of the positive cost-benefit relationship of enhancive volatility encapsulation and maintenance cost. These two components were supported by an analysis of historical change within a chosen domain to determine volatility topology and common change drivers, and then a cost-benefit analysis of a rudimentary volatility identification and encapsulation effort conducted by one of the programs within the chosen domain, respectively. The subsequent sections describe the grounding concept of software maintenance, the target domain for this study, the data collection process, and the manipulation methodologies used to yield relevant information for analysis.

3.1 The Fundamental Process of Software Maintenance

In order to provide a framework for the data collection and analysis undertaken in this dissertation, the software maintenance process is described in this section. This description is essential to understanding the implications of the correlations asserted in this inquiry.

The on-going act of software maintenance involves the identification of the need for changes to software in use, and the controlled infliction of these changes. This identification can and usually does occur at almost any time during the maintenance life cycle, but the corresponding infliction is usually accomplished in groups at planned intervals. The delivery for operational use of software in which a set of identified changes has been incorporated is commonly known as a release. While there are instances of software maintenance life cycles where identified changes are inflicted dynamically without grouping, it was important to select a subject domain for this inquiry where regularly scheduled releases were implemented. Regular releases provide a data collection point around which to establish frequency of occurrence, the essential characteristic of software volatility.

An identified need for software change will ultimately result in the modification of one or more of the software system's constituent objects. The nature of this modification is important, for it results in a new system that is somehow different. Most discussions of software modification are content to deal with the software states, that is, the before- and after-entities surrounding the act of change. But the word "modification" is a verb, implying activity that has characteristics of interest in this inquiry. The range of activity involving modification can be completely described in terms of software size by the acts of addition, change, and deletion. That is, the possible range of activities available to a software maintainer are to add new code to the software, change code already in the software, or delete existing code from the software. It may be argued that change of code is elementally the deletion of code followed by its replacement with new, similar code. However, inspection of change histories of software reveals the significance of change as a distinct category. In fact, a significant sub-categorization of changes becomes apparent:

1) appending new instances of previously existing objects, 2) alteration of existing instances of objects, and 3) deletion of instances of objects. There are significant differences in magnitude of effort and impact to the software between the addition of a new map display to a library of map displays and the incorporation of the capability to display maps where none previously existed. So, for the purpose of describing the nature of an individual volatile act, the following verbs will be used through the rest of this document:

1. add: incorporate a new capability that did not exist before.

2. append: incorporate a new instance of a previously existing capability.

3. alter: modify an existing instance of a capability.

4. delete: take out an instance of a capability; the fundamental capability, and perhaps other instances, remain.

5. remove: take out all aspects of a capability.

At the beginning of this inquiry, it was determined that establishing the frequency of change of the constituent objects would be the basis for subsequently determining the existence of regularly occurring change drivers. The reason for this determination was that software objects such as modules are unambiguously identifiable due to the need for development tools to recognize them. Therefore, the "module" was chosen as the entity for identification of change points. In the systems of the subject domain, various types of modules were identified, containing such objects as code, data, and, configuration information. For this inquiry, no attempt was made to establish meaningful correlations based on module type, although this effort will have significance in subsequent research on volatility encapsulation. The maintenance processes for the systems of the subject domain all captured information on the modules affected for each constituent change of a release, which added to the significance of selecting modules as change points. It must be recognized that the level of change point granularity presented by a module can vary from system to system; some programming languages encourage more collection of related entities in their compilable objects than others. However, the collection of change information at levels lower than the lowest identifiable configuration item is problematic; indeed, there are few maintenance organizations where the collection of change information down to the module level is possible.

Based on the above considerations, selection of a subject domain for this inquiry was based primarily on the existence of histories of regularly scheduled releases, with release documentation including named changes and the constituent affected modules.

3.2 Description of the Domain

The domain chosen as the target of this inquiry is that of integrated tactical warning and attack assessment (ITW&AA) correlation systems. ITW&AA is the primary mission of the North American Aerospace Defense Command (NORAD), a combined command consisting of American and Canadian military forces. Headquartered at Peterson Air Force Base, Colorado Springs, Colorado, NORAD is responsible for defending the North American continent from air, missile and space threats posed from hostile foreign countries. Its primary command center for coordinating these operations is located at Cheyenne Mountain Air Force Base, south of Colorado Springs. ITW&AA correlation systems take event messages from a variety of sensors located around the world regarding air, missile, and space threats to the North American continent and “fuse” the information into a coherent depiction for use by senior military decision makers.

Each of the three mission segments has its own correlation system: 1) Granite Sentry for air correlation; 2) Command Center Processing and Display System - Replacement (CCPDS-R) for missile correlation; and 3) Space Defense Operations Center (SPADOC) for space correlation. Procurement of each of these systems was commenced separately, then later consolidated under the $1.8B Cheyenne Mountain Upgrade (CMU) program. However, program consolidation did not result in architectural cross-fertilization, and the three systems were essentially developed independently. They do exchange event information according to well-defined criteria using a common inter-mission bus. The following sections overview each of the three systems.

3.2.1 Granite Sentry

Granite Sentry provides correlation support to the personnel who staff the Air Defense Operations Center (ADOC) in Cheyenne Mountain Air Force Base. The operational version of Granite Sentry was delivered to the Air Force in 1992 as Phase III of an incremental development. The Granite Sentry program was originally developed in-house by military programmers, but development was later transferred to the Lockheed-Martin Corporation (LMCO). The operational version is scheduled to be replaced in early

1997 with a “final capability” version. The software consists of approximately 300,000 source lines of code that execute on at least two Digital Equipment Corporation VAX 8550 computers (total delivered: 5) and networked VAX workstations located in the mission work centers. Granite Sentry software is written almost entirely in Ada, although command scripts and graphical definition files are often touched in enhancive maintenance.

Besides providing historical change data to support the domain analysis, an effort by Lockheed Martin to identify and encapsulate volatility in the final version will be analyzed to validate the cost-benefit of addressing software volatility.

3.2.2 Command Center Processing and Display System - Replacement (CCPDS-R)

CCPDS-R correlates inputs from land- and space-borne sensors and provides display systems to assist NORAD in its mission to provide U.S. and Canadian decision makers with unambiguous warning information of strategic missile attacks. The CCPDS-R program replaced correlation systems located at Cheyenne Mountain Air Force Base, Offutt Air Force Base in Omaha, Nebraska, and at the National Military Command Center in Washington, D.C. and provided display terminals to U.S. and allied command centers worldwide. The target of this study is the CCPDS-R “Common” correlation system located in Cheyenne Mountain. Its software consists of 700,000 lines of Ada code running on 2 networked Digital Equipment VAX 6000-530 computers.

3.2.3 Space Defense Operations Center (SPADOC)

SPADOC correlates the inputs of radar and optical sensors to support the cataloging and tracking of space-borne objects, as well as providing information to support warning of attack from space.

Granite Sentry and CCPDS-R were both developed in the Ada programming language by Lockheed-Martin Corporation and TRW, respectively. SPADOC was developed in FORTRAN by Loral Corporation.

3.3 The Data Collection Subject: The Space and Warning Systems Center (SWSC)

The SWSC is responsible for maintenance of command and control and communications systems that support NORAD and U.S. Space Command operations in Cheyenne Mountain. It is a directorate of the Space Systems Support Group (SSSG), a unit of the Air Force Materiel Command. The SWSC has performed the above mission for over 25 years, and has developed a well-behaved process to coordinate the maintenance of the hundreds of sensor, communication, correlation, and display software systems comprising the integrated tactical warning and attack assessment (ITW&AA) network. It is through the good graces of this organization and the integrity of its maintenance process that this inquiry into software volatility is supported with complete and concise change history data. The following is a description of this maintenance process, provided with the intent of communicating a perspective from which the data presented in this section is analyzed.

All software systems that belong to the ITW&AA network participate in a coordinated process of maintenance updates referred to as the “vertical release process.” While each system’s maintenance organizations are free to make whatever “stand alone” changes they deem necessary, all systems must coordinate changes that affect other systems in a series of review boards. In the case of a coordinated change, a “generic” standard change form (SCF) is written by the requiring party, which is then presented to a board for impact analysis. Each system that identifies an impact based on the generic SCF is responsible for writing a “corollary” SCF to instigate their part of the change. Another board decides on the content of vertical releases consisting of generic and accompanying corollary SCFs, nominally delivered every six months. Vertical releases are named using the last two digits of the year in which they are delivered, appended with a sequence number, e.g. “95-1”. Inclusion of a generic-corrollary SCF set in a release gives each participating maintenance organization a common delivery date to aim for to ensure that all systems remain interoperable after the release. If any organization subsequently determines that it cannot meet the release date with its corollary, the entire SCF may be backed out of the release in a coordinated fashion.

3.4 Method: Establishing The Nature of Volatility

The first objective of this part of the inquiry was to determine how to characterize volatility. The fundamental meaning of this characterization should ultimately support decisions by program managers on where to best target resources to mitigate volatility. However, even before characterization is undertaken, it must be recognized that there must exist a certain fundamental behavior of software volatility, that of regularity of change in a small percentage of the software. If, over time, the change inflicted upon a software system is uniformly spread across a large proportion of the system, then the cost-benefit of expending resources to encapsulate volatility is questionable. The data collected and presented in this part of the inquiry was targeted to establishing meaningful measures of volatility and identifying its non-uniform distribution in the subject domain.

3.4.1 Change History Data Collection

In order to characterize the nature of volatility, maintenance releases from software-based systems with well-documented histories of change were studied to determine what proportions of the software were revisited for enhancive modifications by frequency over a contiguous period of time. Absence of major developmental incursions such as incremental deliveries was determined to be an important constraint to the series of releases considered, for incremental deliveries introduce new parts of the system with no previous volatility experience. These new parts do not experience the same volatility potential for the studied period of time as do the rest of the system. Characterization of their volatility cannot thus be compared to the rest of the system under analysis. This proved to be a major limitation in data collection, for at least two of the ITW&AA systems in the subject domain had experienced recent incremental deliveries that reduced the amount of contiguous time under maintenance considerably.

For this inquiry, a goal of five releases worth of contiguous data was established, but was only met for one system. The data dictionary established to support data collection is given in Table 3.1:

			Source		
Level	**Nomenclature**	**Definition**	**Granite Sentry**	**CCPDS-R**	**SPADOC**
Release	Title	Designator used to identify the release	Code Turnover Report (CTR)	CTR	Version Description Document (VDD)
Release	Operational Date	Date release was first put into operational use	SCF Data Base	SCF Data Base	SCF Data Base
Change	UCN	Alpha-numeric designator assigned to each change	CTR	CTR	VDD
Change	Title	Descriptive title assigned to each release	SCF Data Base	SCF Data Base	VDD
Change	Mod/Fix	Modification/ Fix Categorization	CTR (Second to last letter of UCN)	CTR (Second to last letter of UCN)	CTR (Second to last letter of UCN)
Module	Name	Name of module as known to the compiler, usually a file name	CTR	CTR	VDD
Module	Category	Type of module, e.g. package specification, data file, etc.	CTR (Module name extension)	CTR (Module name extension)	VDD

Table 3.1: Change History Data Dictionary

The two major sources of affected module data were Code Turnover Reports (CTRs) (CCPDS-R, Granite Sentry) and Version Description Documents (VDDs) (SPADOC). There is a fundamental difference in the submission of these two products that renders an inconsistency in their interpretation. CTRs are developed for each software build delivered during the release development. Builds are an intermediate release given to the testers. Given that errors can (and usually do) exist in the initial implementation of a change, modules that appear in the first build of a release for a given change can appear again for that same change in subsequent builds. CTRs thus yield different touch counts than VDDs, which report the modules affected by a given change reported once, no matter how many builds contain the changed module. The same document (VDDs or CTRs) was not available for all three systems, so a consistent definition for touch count could not be applied.

The number of releases required to achieve significance in the tabulation will be addressed in the section on analysis. All three systems use the same release title nomenclature, so the data coverage is presented in Table 3.2.

System	94-1	94-2	95-1	AOC-2	95-2	96-1	96-2	97-1
Granite Sentry				N/A	▨ x	▨ x	▨ x	
CCPDS-R		▨ x	▨ x	▨ x	▨ x	▨ x	▨ x	▨ x
SPADOC	▨	▨		N/A		▨ x	▨ x	▨ x

Table 3.2: Module Volatility Release Coverage

▨	Data Available
x	Used in module analysis

The data described above was provided by the SWSC in hard copy. OCR equipment and text editing software was utilized to produce spreadsheets of the data suitable for analysis. The SPADOC 94-1 and 94-2 releases were not used in this analysis due to the delivery of an “acquisition release” in the intervening period that disturbed the continuity of the system with major modifications that would have invalidated the module volatility metrics.

3.4.2 Change History Data Reduction

In order to perform the analysis required to substantiate the proof criteria, change frequency tabulations were compiled for each system of the subject domain. Each change frequency tabulation consists of a sorted list of all the modules touched for each change made in a contiguous series of releases. This tabulation is presented as a list of modules sorted in decreasing order of frequency of change. It is important to understand the structure of this presentation, for it forms the basis upon which all of the subsequent volatility relationships are established. For a given system, the modules at the top of the list are impacted by more change over time than modules that appear lower on the list. Any touch count greater than 1 is an indicator of volatility, signifying that such a module has relevance in at least two or more enhancive changes.

The first reduction accomplished was to cull out the “fix” changes in order to limit the scope of the analysis to enhancive maintenance, categorized in the subject domain as “modifications.” The essential indicator of volatility is frequency of change, so initially the module occurrences across all releases were tabulated for each system. Subsequent inspection of the data from the subject domain revealed the case where a module was touched multiple times by multiple changes in one release, and not at all in the others. It was recognized that this case rendered no value in terms of predicting change frequency, so the count of the number of **releases** where a module was touched was substituted. Specifically, if a module was touched at least once in a release, then a count of “1” was established for that module in that release. This method provided a count that reflected a frequency of change, based on the fact that releases were regularly scheduled in all three subject domain systems. Change frequency tabulations were developed in the following format for each of the three systems:

Module	Release 1	Release 2	Release 3	Touch Count	Release Count
MODULEA	3	2	4	9	3
MODULEB	2	1	3	6	3
MODULEC	2	1	0	3	2
MODULED	1	0	0	1	1

Table 3.3: Sample Change Frequency Tabulation

It must be recognized that the touch count can have significance along with the release count in characterizing volatility of a given module. However, for this inquiry, inconsistencies in the means of reporting affected modules between systems of the subject domain prevent a meaningful interpretation. Therefore, release count was identified as the primary metric for the characterization of volatility.

The module change frequency tabulations developed for each of the systems of the subject domain are presented in Appendices A (SPADOC), B (CCPDS-R), and C (Granite Sentry). Inspection of the data revealed the following summary metrics, useful in characterizing module-level volatility at a glance:

Metric	**SPADOC**	**CCPDS-R**	**Granite Sentry**
Number of Modules in System		3768	4353
Number of Modules Touched > 0	288	749	259
Number of Modules Touched > 1	90	162	21
Number of Modules Touched > 50%	90	14	21
Maintenance Focus	64%	20%	6%
Volatility Focus	20%	4%	0.5%
Volatility Concentration	31%	22%	8%

Table 3.4: Summary Volatility Metrics

The module counts were chosen for their potential significance: ">0" represents modules touched in at least one release of the subject period, which provides information on how much of the system receives attention in maintenance. ">1" represents modules touched in more than one release during the subject period, which represents the threshold of volatility according to the operational definition. Modules touched in more than one release during the subject period experienced a phenomenon termed "revisit" for this inquiry. That is, a volatile module is one that has received attention in two or more releases separated by the significant time between releases. ">50%" represents modules touched in more than 50% of the releases of the subject period. This percentage was arbitrarily selected as an threshold of significant volatility per the operational definition; due to the small number of releases available for SPADOC and Granite Sentry, this threshold has no significance. For CCPDS-R however, the seven releases of change history did present a significantly smaller number of modules with release counts > 50% for consideration.

The percentage metrics represent fundamental relationships between the counts described above. Each metric is defined as follows:

1. Maintenance Focus: The percent of the system touched for maintenance during the subject period (Number of modules touched >0 / Number of modules in system).

2. Volatility Focus: The percent of the system touched for maintenance in more than one release during the subject period (Number of modules touched > 1 / Number of modules in system).

3. Volatility Concentration: The percent of the system touched for maintenance that was touched in more than one release (Number of modules touched > 1 / Number of modules touched > 0). This metric provides a normalized volatility proportion suitable for comparison amongst the systems of the subject domain.

The most important analysis supported by module volatility was the establishment of the concentration of frequency of change in a relatively small part of the systems of the subject domain. This was accomplished by tabulating the number (n) of modules with each change frequency (f) in all three systems; for a given n_f, $n_f > n_{f+1}$:

Number of Releases (f)	CCPDS-R	Granite Sentry	SPADOC
5	1	-	-
4	13	-	-
3	44	-	45
2	104	21	45
1	587	238	198

Table 3.5: Change Frequency Summary

With the exception of $n_2 = n_3$ for SPADOC, all other change frequencies exhibit the condition $n_f > n_{f+1}$.

3.5 Method: Identifying Cross-System Volatility Correlations

An activity essential to providing information to software developers on what attributes of software to encapsulate is the identification of common change drivers across systems of the domain. “Commonality” has two dimensions, 1) the degree of occurrence in systems of the domain, and 2) the frequency of occurrence in the systems where the change driver occurs. To determine change driver commonality, it is necessary to develop the same sort of topology as developed for software components.

3.5.1 Cross-System Volatility Data Collection

The original intent of this dissertation was to determine change drivers for changes of the subject domain using information from the primary change document, the SCF. In this way, the change driver identification technique could be developed to be applied to systems maintained with a nominal degree of process. However, the act of change documentation is highly dependent on the individual judgment of programmers and analysts regarding the proper descriptive words, resulting in a wide variety in the application of the English language in change titles and short descriptions. Additionally, it was revealed during inspection of the changes from the subject domain that a few were actually containers for multiple related changes. For example, one SCF in the Granite Sentry collection held the title “Database Changes.” This SCF turned out to encompass changes to three distinct databases, a fact not derivable from the SCF form.

Consequently, other documentation was eventually identified to support the change driver analysis. The SWSC directs the development of a set of documents during the analysis, design, implementation, integration, and testing of approved changes. The Requirements Analysis/Design Intent (RADI) document is developed for each change selected to be incorporated into an up-coming release, and contains detailed descriptions of the modified system requirements, tagged for traceability to the original system requirements document and the appropriate test cases. The change descriptions used to accomplish change driver identification were taken from the introductions of these documents. The following table identifies the releases from which RADI change descriptions were obtained for change driver analysis:

System	94-1	94-2	95-1	AOC-2	95-2	96-1	96-2	97-1
Granite Sentry				N/A	▨	▨	▨	
CCPDS-R					▨	▨	▨	▨
SPADOC				N/A	▨	▨	▨	

Table 3.6: Change Driver Release Coverage

▨ Data Available and used in change driver analysis

3.5.2 Cross-System Volatility Data Reduction

Even with the concise descriptions extracted from the RADI documents, the wording was sufficiently inconsistent to render automated semantic analysis impossible. A scoring technique was developed to identify the constituent change drivers of an SCF in a two-step process. First, the generic changes accomplished in the SCF were identified as verb-object pairs from a small set, specifically the one described in Table 3.6:

object / verb	input	process	output
add			
append	x		
alter		x	
delete			
remove			

Table 3.7: Change Driver Generic Pairs Example

The verbs are the same as the categories of modification described in paragraph 3.1. The object is simply to categorize the modified entity as either input, processing, or output. In this task, it is essential to describe the change in the system's behavior, not necessarily the mechanism changed. For instance, a change in the border width for a text box may be accomplished on the subject system by changing a numeric value in a configuration file; this change would be more appropriately describe as "alter output" referring to the modified display, rather than "alter input" referring to the modified data file. For the first step, the short SCF description from the RADI was scanned sentence by sentence, and each applicable verb-object pair was identified. In the second step, each verb-object was revisited in the description and a one-word name was identified for the entity modified under the verb-object pair. Focusing the selection of a generic object name for each change driver by first categorizing it as input/output/data served to increase the probability of identifying consistent names that could be compared across systems.

In the same manner as that accomplished for module volatility, change driver volatility was tabulated for each system with both occurrence counts and release counts. These tabulations are presented in the following table:

Category	CCPDS-R	Granite	SPADOC	Total	Percent of Total
alter output	120	21	200	341	35%
alter input	42	2	136	180	19%
alter process	39	12	120	171	18%
append output	57	1	52	110	11%
append input	21	0	40	61	6%
append process	10	0	24	34	4%
delete output	24	1	1	26	3%
delete input	12	1	4	17	2%
add output	10	0	1	11	1%
delete process	1	1	4	6	1%
add process	4	0	1	5	1%
add input	0	0	1	1	≈ 0%
remove input	0	0	0	0	0%
remove output	0	0	0	0	0%
remove process	0	0	0	0	0%

Table 3.8: Change Category Tabulation

Correlation among the three systems was established using Spearman's coefficient of rank correlation as follows:

H_0: $r_s = 0$; H_1: $r_s > 0$

Critical value of t = 2.650 ($\alpha = .01$, df = 13, one-tailed test)

	Granite Sentry	CCPDS-R	SPADOC
Granite Sentry	-	-	-
CCPDS-R	$r_s = .79$ $t = 4.70$	-	-
SPADOC	$r_s = .81$ $t = 5.07$	$r_s = .87$ $t = 6.37$	-

Table 3.9: Change Categorization Correlation

H_0 is therefore rejected for all three combinations, establishing a significant correlation among the clustering of change categories.

Establishing the most prevalent category of change was accomplished by a ranking by both frequency within systems as well as occurrence in more than one system. The following table lists the change objects that occurred in two or more of the systems, along with their total exposure:

Object	Occurrence	CCPDS-R	Granite	SPADOC	Total Exposure
display	3	112	21	144	277
message	3	104	3	84	191
menu	2	20	0	26	46
satellite	2	2	0	14	16
COTS	2	3	0	12	15
command	2	3	0	6	9
site	2	0	1	7	8
alarm	2	2	2	0	4
threat	2	0	1	1	2
count	2	1	0	1	2
keyboard	2	1	0	1	2

Table 3.10: Predominant Change Objects

The sort performed on the above table used “Occurrence” as the first sort key and “Total Exposure” as the second. In this instance, the exposure rankings conform to the primary sort on the number of systems in which the change object appears; this does not always have to be the case. A large number of occurrences in one system can push a change driver with a lower occurrence into the territory of objects which occur in more systems. The objective of this particular analysis is to determine “predominant change objects,” where the first condition of predominance is the occurrence of the object in many systems of the domain. Sorting first by occurrence serves this first condition; sorting second by exposure serves to rank the objects within their “buckets” of occurrence.

It should be noted that, for the subject domain, there is a high degree of correlation between occurrence in many systems and change volume. Correlation of occurrence among the systems and the exposure in each system was established using Spearman's coefficient of rank correlation as follows:

H_0: $r_s = 0$; H_1: $r_s > 0$

Critical value of t = 2.358 ($\alpha = .01$, df $\approx$ 60, one-tailed test)

	Granite Sentry	CCPDS-R	SPADOC
r_s	0.8263851	0.652381	0.6147092
t	11.555496	6.7778233	6.136548

Table 3.11: Change Object Correlation

3.6 Method: Asserting the Benefit of Addressing Volatility

3.6.1 Cost Comparison Data Collection

Granite Sentry offered an opportunity for comparison of change costs between a system where volatility was not formally addressed in its design with the equivalent system where volatility was analyzed and encapsulated. Particularly, for vertical release 95-2, the organization developing the replacement system incorporated SCFs that the operational system also incorporated. To support cost comparison, the rough-order-of-magnitude (ROM) cost estimates for 12 of 13 identical modifications made to both systems were obtained from the responsible organizations. Additionally, impacted module tabulations were also obtained for 5 of the 13 releases. It was determined from interviews with engineers from the development team that most of the system attributes determined to be volatile were encapsulated in .BCP files and imported into the data base management system for subsequent use by the executing system.

3.6.2 Cost Comparison Data Reduction

It was first determined that the cost estimates for effecting each change to the new system was on average 53% less than the costs estimated to make the equivalent changes on the old system. To attempt to determine the contributory benefit of volatility encapsulation, the cost difference was correlated against the difference in number of Ada modules impacted and the number of .BCP files impacted in the new system. Using this computed data from the 5 changes with module data, a regression analysis was performed. At the .05 significance level, it was determined that the selected independent variables do not have the capability to predict difference in change cost between the new and old Granite Sentry systems, making it impossible to attribute any component of cost benefit directly to the encapsulation method used.

Chapter 4

Analysis

4.1 Analysis: Establishing The Nature of Volatility

Development of volatility tabulations for the systems of the subject domain provided interesting insight into the nature of software evolution. However, cross-comparison between systems of module-oriented volatility metrics is of limited value for the following reasons:

1. Differences in reported entities alter the level of granularity with which volatility is identified. In the cases of CCPDS-R and Granite Sentry, the reported entity was the Ada package or program. For SPADOC, the reported entity was the Computer Program Component (CPC), which encompassed all the FORTRAN routines supporting a system function. SPADOC's CPC represented a level of granularity comparable to the Computer Software Component (CSC) used in CCPDS-R and Granite Sentry, which contain a large number of modules. So, for every component impacted by a change in SPADOC, a larger percentage of the total system was counted.

2. The releases comprising the subject period for Granite Sentry did not include the number of changes that would normally be considered due to the impending arrival of the replacement system. This fact correlates to Granite

Sentry’s small volatility metrics compared to CCPDS-R and SPADOC. Even the normalized volatility concentration (8%) exhibits a significant difference from the SPADOC (31%) and CCPDS-R (22%) numbers to which attribution to a significant difference in change traffic cannot be discounted.

For the above reasons, module volatility metrics cannot be reliably used for developing domain characterizations. However, the module volatility topology represented by the tabulation of module touch counts can provide information of significant utility to maintainers of the system. For instance, for the set of corrective changes made during a series of releases, identification of modules visited the most points to specific places in the system with reliability problems. Additionally, the volatility focus metric can be used as an indicator of the proportion of the overall system with reliability problems. But the focus of this inquiry is to establish the validity of characterizing volatility; the essential contribution of module volatility analysis is to validate the fundamental nature of volatility. Toward this end, the module change frequencies tabulated for all three systems illustrate the phenomenon that change happens over time in a relatively small proportion of the total source code, evidenced by the frequency counts of all three systems meeting the condition $n_f > n_{f+1}$.

4.2 Analysis: Identifying Cross-System Volatility Correlations

It then becomes imperative to engineer a contrivance to produce consistent manifestations of the behavior of volatility across systems. Simple categorization of change according to the five verbs and three objects described in Chapter 3 may seem trivial, but these descriptors are universally applicable to all software systems. It may eventually come to pass that enough categorization of change in this general manner is gathered to characterize volatility universally, providing motivation to developers in all domains to concentrate on, say, making “alter output” easy to accomplish in their system because it is the most prevalent type of change. With regard to specific domains, the second step of using the change category to identify a generic object shows significant promise, if only to categorize domain change with respect to inputs and outputs. Processing appears to be too specific to individual systems to drive out common change objects using this mechanism. The percent contribution of each of the identified change objects to the overall volatility exposure of the domain is listed in the table below:

Object	**Occurrence (# systems)**	**CCPDS-R**	**Granite Sentry**	**SPADOC**	**Average Percent**
display	3	37.33%	63.64%	38.20%	46.39%
message	3	34.67%	9.09%	22.28%	22.01%
menu	2	6.67%	0.00%	6.90%	4.52%
satellite	2	0.67%	0.00%	3.71%	1.46%
COTS	2	1.00%	0.00%	3.18%	1.39%
command	2	1.00%	0.00%	1.59%	0.86%
site	2	0.00%	3.03%	1.86%	1.63%
alarm	2	0.67%	6.06%	0.00%	2.24%
threat	2	0.00%	3.03%	0.27%	1.10%
count	2	0.33%	0.00%	0.27%	0.20%
keyboard	2	0.33%	0.00%	0.27%	0.20%
file	1	0.00%	0.00%	5.57%	1.86%
dialog	1	0.00%	0.00%	3.71%	1.24%
alert	1	4.00%	0.00%	0.00%	1.33%
rules	1	3.33%	0.00%	0.00%	1.11%
data	1	2.00%	0.00%	0.00%	0.67%
DIM	1	0.00%	12.12%	0.00%	4.04%
report	1	1.33%	0.00%	0.00%	0.44%
time	1	1.33%	0.00%	0.00%	0.44%
element set	1	0.00%	0.00%	1.06%	0.35%

Table 4.1: Change Object Volatility Contribution

Object	Occurrence (# systems)	CCPDS-R	Granite Sentry	SPADOC	Average Percent
notification	1	0.00%	0.00%	1.06%	0.35%
orbit	1	0.00%	0.00%	1.06%	0.35%
security	1	0.00%	0.00%	1.06%	0.35%
status	1	1.00%	0.00%	0.00%	0.33%
timer	1	1.00%	0.00%	0.00%	0.33%
baud rate	1	0.00%	0.00%	0.53%	0.18%
options	1	0.00%	0.00%	0.53%	0.18%
weapon	1	0.00%	3.03%	0.00%	1.01%
error	1	0.33%	0.00%	0.00%	0.11%
failover	1	0.33%	0.00%	0.00%	0.11%
form	1	0.33%	0.00%	0.00%	0.11%
message field	1	0.33%	0.00%	0.00%	0.11%
message filter	1	0.33%	0.00%	0.00%	0.11%
scenario	1	0.33%	0.00%	0.00%	0.11%
summary	1	0.33%	0.00%	0.00%	0.11%
table	1	0.33%	0.00%	0.00%	0.11%
thresholding	1	0.33%	0.00%	0.00%	0.11%
track	1	0.33%	0.00%	0.00%	0.11%
applications	1	0.00%	0.00%	0.27%	0.09%
ASAT	1	0.00%	0.00%	0.27%	0.09%
buffer	1	0.00%	0.00%	0.27%	0.09%
catalog	1	0.00%	0.00%	0.27%	0.09%
CIS	1	0.00%	0.00%	0.27%	0.09%
cluster	1	0.00%	0.00%	0.27%	0.09%
collision avoidance	1	0.00%	0.00%	0.27%	0.09%
consistency	1	0.00%	0.00%	0.27%	0.09%
decay dates	1	0.00%	0.00%	0.27%	0.09%
edit	1	0.00%	0.00%	0.27%	0.09%
Ephemeris	1	0.00%	0.00%	0.27%	0.09%
extrapolation DC	1	0.00%	0.00%	0.27%	0.09%
folder	1	0.00%	0.00%	0.27%	0.09%
interface	1	0.00%	0.00%	0.27%	0.09%
IRONs	1	0.00%	0.00%	0.27%	0.09%
keys	1	0.00%	0.00%	0.27%	0.09%
maneuver	1	0.00%	0.00%	0.27%	0.09%
msg	1	0.00%	0.00%	0.27%	0.09%
multipliers	1	0.00%	0.00%	0.27%	0.09%
observation	1	0.00%	0.00%	0.27%	0.09%
printer	1	0.00%	0.00%	0.27%	0.09%

Table 4.1: Change Object Volatility Contribution (continued)

Object	Occurrence (# systems)	CCPDS-R	Granite Sentry	SPADOC	Average Percent
projection	1	0.00%	0.00%	0.27%	0.09%
screen print	1	0.00%	0.00%	0.27%	0.09%
solar	1	0.00%	0.00%	0.27%	0.09%
vote	1	0.00%	0.00%	0.27%	0.09%
window	1	0.00%	0.00%	0.27%	0.09%

Table 4.1: Change Object Volatility Contribution (continued)

The remaining change objects contributed 1% or less each and/or occurred in only one system. With respect to the subject domain, the two most volatile object types, displays and messages, occurred in all three systems and accounted for an average of 68% of the total exposure to change. The next most volatile object in at least two of the systems is the menu, accounting for an average of 5% of the total exposure to change across two of the three systems. Given these measured proportions of volatility, an assertion of significant values of *m* for corresponding *p* can be made. For the subject domain, significant values of *m* at *p*=46%, is 1; *p*=68%, is 2; *p*=73%, is 3. Higher values of *m* do not contribute significantly to the total domain volatility. The significance of this assertion is that, for the domain of integrated tactical warning/attack assessment correlation systems, developers of these systems can accommodate approximately 68% of the anticipated volatility during maintenance by encapsulating aspects of first displays, then messages, with change mechanisms. The traditional “80% - 20%” rule of Pareto analysis was also considered, where the number of change objects that account for 80% of the enhancive maintenance was determined. This value, $m_{80\%}$ = 8, includes five change objects who contribute only an average of 1.5% each to the total enhancive effort.

However, $m_{80\%}$ incorporates 13% of all identified change objects, indicating an even higher concentration of change than the "80% - 20%" pattern.

It should be noted that there were change objects that contributed to the total enhancive effort of the domain on the order of menus (i.e., DIM processing, 4%), but occurred in only one system (SPADOC). As with the number of releases a change object was touched, the number of systems it appears in has to be considered separately from its change traffic in order to give proper weight to its prevalence among systems over any high volatility within a single system. So, for any volatility analysis, be it of modules within a single system or change objects among systems, prioritization should occur first by release or system (module or change object volatility respectively), then by impact count. This sorting "bucketizes" change traffic according to its prevalence.

It can be argued that the input/output-oriented change objects comprising displays and messages have many more instances within a specific system than the specific processing identified in the analysis. Indeed, each of the three systems of the subject domain contain components corresponding to tens if not hundreds of displays and messages, where the processing objects identified usually consisted of a single instance. But the proportions of change categories tabulated for the subject domain support the imperative to attend to displays and messages, for all processing changes accounted for only 22% of the total change analyzed in the subject domain versus 51% attributed to outputs.

4.3 Analysis: Asserting the Benefit of Addressing Volatility

Based on the available data, it was impossible to attribute a specific component of reduced change cost to the new Granite Sentry's encapsulation method. Indeed, many of the same people who developed the old Granite Sentry were involved in developing its replacement and may very well have engaged in other improvements of code organization and structure that resulted in more efficient change.

The analysis presented in this chapter establishes the nature of enhancive software volatility and manifests the potential to identify common change drivers in systems of a domain. The next chapter discusses the implications of this analysis with respect to its constructive impact on the process of software development.

Chapter 5

Results and Conclusions

5.1 Summary of Results

This dissertation set out to validate the concepts of volatility identification and encapsulation as beneficial influences on system life cycle costs. The foundation for this validation was the proof of the three hypotheses presented in Chapter 1. The summary of the decisions regarding the hypotheses are:

H1: Distribution of frequency of enhancive change in a system is not uniform:

Proven - Frequency of change is not uniformly distributed.

H2: Within a domain, the most volatile objects of change are common to all systems:

Proven - Systems of a domain can experience common change drivers.

H3: Effort expended in software development to encapsulate a volatile point reduces the cost to change the encapsulation object in maintenance:

Not Proven - Insufficient data to assert a correlation between volatility encapsulation and cost savings.

While the cost-benefit of addressing volatility was not proven, the behavior of volatility exhibited within systems and correlated among systems of a domain shows significant potential for cost reduction through encapsulation. First, significant rates of revisit in certain modules of a system, as proven by H1, validates the assertion that the extra cost of encapsulation can be amortized over multiple instances of change cost reduction. So, the following relationship must hold for a specific encapsulation effort:

$$SM_E < SM_{\Delta C} \times EF$$

Where:

SM_E	=	Cost to encapsulate in staff-months
$SM_{\Delta C}$	=	Cost to inflict a single instance of a change driver (from Equation 1)
EF	=	Expected instances of change during life cycle

The selection of change drivers to encapsulate in new software development can be assisted by the selection of an appropriate value of *m* for a given percentage of anticipated volatility. Given that the equation describing the Pareto curve of volatility topology is logarithmic,

$$y = C\ln(x) + b\,,$$

the second derivative,

$$\frac{d^2y}{dx^2} = -\frac{C}{x^2}\,,$$

is a potential indicator of the point on the curve where the most volatile change drivers are captured. However, the selection of a particular *m* is dependent not only on the contribution of each change driver to the overall anticipated volatility, but also to the cost to effectively encapsulate the change driver. Pragmatic consideration of the available resources may preclude engaging in a particularly costly encapsulation, even if the change driver's volatility contribution to *p* is high. By this insight, it should be apparent that the selection of *m* based on a mathematical criterion such as significant transitions of the second derivative would be unnecessarily arbitrary.

The second contribution is that there is a consistent way to identify change objects in systems of a domain such that common objects can be identified in all the systems. Further, for the subject domain, certain common change objects exhibited high volatility in all systems, proven by H2, to the point that two of the objects (displays and messages) account for an average of 68% of the enhancive volatility in all three systems. So, for the ITW&AA correlation systems domain, encapsulating changeable aspects of displays has the potential of addressing as much as 46% of the expected enhancive volatility in a new system, and doing the same for messages will potentially address another 22%. In fact, the first order characterization (add, append, alter, delete, remove; input, processing, output) has potential for categorizing patterns of change across all software systems. It should be noted that developers in the ITW&AA domain may wish to tabulate the change objects with the associated first-order verbs to further define the prevalent types of change inflicted upon the change objects. The representation of volatility as an ordered pair also constituted a significant contribution, recognizing the time-ordered distribution of change as the primary indicator of volatility worth addressing, followed by the total activity over the period under study.

It should be recognized that, no matter how incisive the information presented by a volatility analysis, the effort expended on it will amount to naught unless there is motivation within a software development manager to expend resources specifically targeted to encapsulating potentially volatile points. Proof of the third hypothesis is eventually required to provide the compelling evidence that volatility encapsulation is a viable life cycle alternative. In the near term, the effort of the SWSC to specify the process of domain engineering within the context of their organization offers the opportunity to inject volatility identification and encapsulation into the roles and responsibilities of specific individuals. In the SWSC’s Domain Engineering guidebook [2], the domain focus is captured in the “product line organization,” which specifies roles and responsibilities for the functions required to build and maintain a domain architecture supporting a “product line” of specific, related systems. Specifically, module-level volatility analysis provides product line managers an incisive set of metrics with which to identify and prioritize perfective modifications to their software charges to support lower change costs. Change driver metrics also have value to product line managers, supporting their efforts to characterize volatility across all systems of their product line. Change driver analysis may also provide value to product line domain analysis as an alternate source of identification for objects of the service layer.

5.2 Conclusions and Further Study

The main contribution of this dissertation was to establish a methodological model for directly addressing software volatility in new systems development. The proposed model also provides a framework for further validation of the effort to address software volatility. Based on the demonstrated tendency of change to cluster with respect to frequency in a relatively small part of a software system, the fundamental behavior of volatility was established, providing the fundamental motivation to address it in development. It was also shown that categorizations of change form a promising means of identifying common change drivers in software systems of a domain, providing the essential information for addressing volatility in new development. Yet to be shown is the fundamental economic cost benefit of addressing volatility; based on the complex nature of real-world software development and all its influences, the best approach for demonstrating cost-benefit is probably through a controlled experiment.

With respect to characterization, other domains should be subjected to the same analysis to discover consistent patterns of volatility. Also, the results of change driver analysis should be analyzed across domains to attempt discovery of change patterns common to all software systems.

The methodology of change driver characterization deserves further refinement, especially with regard to its potential contribution to domain analysis. In particular, the use of a controlled language for writing change descriptions would provide significant data to volatility research, allowing effective comparison of change activity across systems of a domain. It could also provide useful data to other endeavors, such as change impact analysis and reengineering initiatives.

Finally, this dissertation focused on only one of the two fundamental activities of addressing volatility: identification. The activity of encapsulation warrants further definitional effort and development of a coherent taxonomy based on implementation costs incurred and change costs saved. Particularly, an encapsulation break-even equation would provide software developers with a tool to evaluation encapsulation efforts with respect to anticipated volatility and the software's expected useful life. With these contributions, the tools to address software volatility during new system development will provide a coherent and effective approach to reducing software life cycle costs.

Appendix 1

Volatility Characterization - SPADOC

a. SPADOC Module Volatility Tabulation

Module	96-1	96-2	97-1	Grand Total	Module Type	Release Count	Release Percent
ODFD	10	18	10	38	DB	3	100%
ORFI	6	19	10	35	DB	3	100%
ODFC	6	20	6	32	DB	3	100%
ODFO	7	18	7	32	DB	3	100%
OMOP	8	15	4	27	DB	3	100%
ORUD	5	15	7	27	DB	3	100%
OSPC	3	14	8	25	DB	3	100%
ORQI	5	17	2	24	DB	3	100%
OHER	6	6	5	17	DB	3	100%
AMFD	2	7	6	15	DB	3	100%
OPRF	4	8	2	14	DB	3	100%
AMDF	3	5	4	12	DB	3	100%
CT08	4	5	3	12	DB	3	100%
MNTPMP	2	3	6	11	APP	3	100%
AMMP	1	5	4	10	DB	3	100%
AMST	2	4	4	10	DB	3	100%
MNTPMM	1	3	6	10	APP	3	100%
MNTMLB	3	4	2	9	APP	3	100%
OATI	4	4	1	9	DB	3	100%
EM05	2	3	3	8	DB	3	100%
ORVV	2	2	4	8	DB	3	100%
ADSP	2	2	3	7	DB	3	100%
AMHVPP	2	3	2	7	APP	3	100%
BLDRUA	3	3	1	7	APP	3	100%
SPTCTN	3	3	1	7	APP	3	100%
LCHEVN	1	4	1	6	APP	3	100%
ACDA	1	3	1	5	DB	3	100%
CTLGSV	1	3	1	5	APP	3	100%
DIADSV	2	2	1	5	APP	3	100%
LCHNOM	1	3	1	5	APP	3	100%
MNTPFU	1	3	1	5	APP	3	100%
OSIOUT	3	1	1	5	APP	3	100%
DIADGN	2	1	1	4	APP	3	100%
DIAGIN	2	1	1	4	APP	3	100%
EM01	1	2	1	4	DB	3	100%
LCHLCM	1	2	1	4	APP	3	100%
MNTVAR	1	2	1	4	APP	3	100%
SPTLBR	1	2	1	4	APP	3	100%
BLDCNF	1	1	1	3	APP	3	100%
SPTRRD	1	1	1	3	APP	3	100%
STTGBO	1	1	1	3	APP	3	100%
STTGMO	1	1	1	3	APP	3	100%
STTGRO	1	1	1	3	APP	3	100%

Module	**96-1**	**96-2**	**97-1**	**Grand Total**	**Module Type**	**Release Count**	**Release Percent**
STTOBC	1	1	1	3	APP	3	100%
STTSTL	1	1	1	3	APP	3	100%
OCOM	3	7	0	10	DB	2	67%
ACMT	0	3	6	9	DB	2	67%
MNTMDI	0	4	4	8	APP	2	67%
BLDBLB	0	4	3	7	APP	2	67%
AMHLIB	3	3	0	6	APP	2	67%
LCHLLB	3	3	0	6	APP	2	67%
MNTMDC	0	2	4	6	APP	2	67%
AMHGVO	0	4	1	5	APP	2	67%
BLDRUC	1	4	0	5	APP	2	67%
CT03	2	3	0	5	DB	2	67%
CT16	0	3	2	5	DB	2	67%
MNTGB3	3	0	2	5	APP	2	67%
SPED	0	3	2	5	DB	2	67%
AMHOMP	0	3	1	4	APP	2	67%
CRIALP	0	3	1	4	APP	2	67%
CSICSP	0	1	3	4	APP	2	67%
CTLSSV	1	3	0	4	APP	2	67%
CTLSUH	0	3	1	4	APP	2	67%
SPTFEL	0	3	1	4	APP	2	67%
AMHAMP	1	2	0	3	APP	2	67%
AMHFCV	0	2	1	3	APP	2	67%
BLDMPC	1	2	0	3	APP	2	67%
CTLDST	0	2	1	3	APP	2	67%
CTLSHT	1	2	0	3	APP	2	67%
FELD	0	1	2	3	DB	2	67%
MNTAOB	0	2	1	3	APP	2	67%
OSIPBL	0	2	1	3	APP	2	67%
OSIUTL	0	2	1	3	APP	2	67%
SPTRNS	2	0	1	3	APP	2	67%
SSEC	0	2	1	3	DB	2	67%
TSKRTK	1	2	0	3	APP	2	67%
ASTRPC	0	1	1	2	APP	2	67%
BLDBSM	1	1	0	2	APP	2	67%
BLDBUC	1	1	0	2	APP	2	67%
CRSCSP	0	1	1	2	APP	2	67%
CTLSTH	1	1	0	2	APP	2	67%
DIAGOU	1	0	1	2	APP	2	67%
DIAGPH	1	1	0	2	APP	2	67%
MNTAOD	1	0	1	2	APP	2	67%
MNTMOB	0	1	1	2	APP	2	67%
MNTTTC	0	1	1	2	APP	2	67%
PRDRPT	1	1	0	2	APP	2	67%
RTAVFI	1	1	0	2	APP	2	67%
STTCRS	1	0	1	2	APP	2	67%
STTCRX	1	1	0	2	APP	2	67%

Module	96-1	96-2	97-1	Grand Total	Module Type	Release Count	Release Percent
AMHEXP	0	4	0	4	APP	1	33%
CRIHIO	0	4	0	4	APP	1	33%
CTLOPI	0	4	0	4	APP	1	33%
MNTPSP	0	4	0	4	APP	1	33%
SPTEEL	0	4	0	4	APP	1	33%
CRISVR	0	3	0	3	APP	1	33%
CRSALP	0	3	0	3	APP	1	33%
CRSCIF	0	3	0	3	APP	1	33%
CTLSRH	0	3	0	3	APP	1	33%
EMSEMC	0	3	0	3	APP	1	33%
LCHLDM	0	3	0	3	APP	1	33%
LETD	0	3	0	3	DB	1	33%
OSISEQ	0	3	0	3	APP	1	33%
PMERER	0	3	0	3	APP	1	33%
SMARTS	0	3	0	3	APP	1	33%
SMAWMG	0	3	0	3	APP	1	33%
CRICSP	0	2	0	2	APP	1	33%
CRISLP	0	2	0	2	APP	1	33%
CTLREH	0	2	0	2	APP	1	33%
CTLROT	0	2	0	2	APP	1	33%
EMSSXR	0	2	0	2	APP	1	33%
LCHISD	0	2	0	2	APP	1	33%
LCHSEG	0	2	0	2	APP	1	33%
LFOL	0	2	0	2	DB	1	33%
MGFL	0	2	0	2	DB	1	33%
MNTACC	0	2	0	2	APP	1	33%
MNTACI	0	2	0	2	APP	1	33%
MNTMGM	0	2	0	2	APP	1	33%
OSIGOC	0	2	0	2	APP	1	33%
OSISAM	0	2	0	2	APP	1	33%
RTXRTX	0	2	0	2	APP	1	33%
SAEI	0	2	0	2	DB	1	33%
SATI	0	2	0	2	DB	1	33%
SCSD	0	2	0	2	DB	1	33%
SMA	0	0	2	2	APP	1	33%
SMAALA	0	2	0	2	APP	1	33%
SMAMSD	0	2	0	2	APP	1	33%
SP08	0	0	2	2	DB	1	33%
SPTP	0	2	0	2	DB	1	33%
STXT	0	2	0	2	DB	1	33%
TSKTUT	0	2	0	2	APP	1	33%
AARF	0	1	0	1	DB	1	33%
ALA	0	0	1	1	APP	1	33%
AMFC	0	1	0	1	DB	1	33%
AMHMST	0	1	0	1	APP	1	33%
AMHRCV	0	1	0	1	APP	1	33%
AMHSOM	0	1	0	1	APP	1	33%

Module	96-1	96-2	97-1	Grand Total	Module Type	Release Count	Release Percent
AMHTAP	1	0	0	1	APP	1	33%
AMHVDI	0	1	0	1	APP	1	33%
ASTALB	1	0	0	1	APP	1	33%
ASTCPS	1	0	0	1	APP	1	33%
ASTDCI	0	1	0	1	APP	1	33%
ASTDCX	0	1	0	1	APP	1	33%
ASTEQM	0	0	1	1	APP	1	33%
ASTESO	0	0	1	1	APP	1	33%
ASTGRT	1	0	0	1	APP	1	33%
ASTIOM	0	1	0	1	APP	1	33%
ASTLAG	0	0	1	1	APP	1	33%
ASTMAD	0	0	1	1	APP	1	33%
ASTMDT	1	0	0	1	APP	1	33%
ASTOBS	0	0	1	1	APP	1	33%
ASTSEN	0	0	1	1	APP	1	33%
ASTUPM	0	1	0	1	APP	1	33%
BBLB26	0	0	1	1	APP	1	33%
BC3MNT	0	0	1	1	APP	1	33%
BCTL	0	1	0	1	DB	1	33%
BEISAV	0	0	1	1	APP	1	33%
BLBDCX	0	0	1	1	APP	1	33%
BLDBCC	1	0	0	1	APP	1	33%
BLDBEI	0	0	1	1	APP	1	33%
BLDREP	0	0	1	1	APP	1	33%
BSTR	0	1	0	1	DB	1	33%
CRSSIF	0	1	0	1	APP	1	33%
CRSSLP	0	1	0	1	APP	1	33%
CRSSVR	0	1	0	1	APP	1	33%
CSIMGA	0	1	0	1	APP	1	33%
CT17	0	1	0	1	DB	1	33%
CT18	0	1	0	1	DB	1	33%
CTLACS	0	1	0	1	APP	1	33%
CTLAPA	0	1	0	1	APP	1	33%
CTLASH	0	1	0	1	APP	1	33%
CTLDOH	0	1	0	1	APP	1	33%
CTLEHR	0	1	0	1	APP	1	33%
CTLEVT	0	1	0	1	APP	1	33%
CTLLOF	0	1	0	1	APP	1	33%
CTLRTG	0	1	0	1	APP	1	33%
CTLSQH	0	1	0	1	APP	1	33%
CTLTSH	0	1	0	1	APP	1	33%
DBAONI	0	0	1	1	APP	1	33%
DBAPRS	1	0	0	1	APP	1	33%
DBMAVT	0	0	1	1	APP	1	33%
DBMNTL	0	0	1	1	APP	1	33%
DIAPSS	0	0	1	1	APP	1	33%
DISCSC	0	1	0	1	APP	1	33%

Module	96-1	96-2	97-1	Grand Total	Module Type	Release Count	Release Percent
EATINP	0	1	0	1	APP	1	33%
EM05	0	0	1	1	DB	1	33%
EMSSRS	0	1	0	1	APP	1	33%
GBNV	0	0	1	1	DB	1	33%
LPRM	0	1	0	1	DB	1	33%
MARM	0	1	0	1	DB	1	33%
MN05	0	0	1	1	DB	1	33%
MNTBOB	0	0	1	1	APP	1	33%
MNTCOW	0	0	1	1	APP	1	33%
MNTEOD	0	0	1	1	APP	1	33%
MNTEPH	1	0	0	1	APP	1	33%
MNTMNB	0	0	1	1	APP	1	33%
MNTOBT	0	0	1	1	APP	1	33%
MNTPDC	0	1	0	1	APP	1	33%
MNTPOL	0	0	1	1	APP	1	33%
MNTTAC	0	1	0	1	APP	1	33%
NSEC	0	0	1	1	DB	1	33%
OCOV	0	1	0	1	DB	1	33%
ODSC	0	0	1	1	DB	1	33%
OPADAL	1	0	0	1	APP	1	33%
OPADOF	1	0	0	1	APP	1	33%
OPAITO	0	0	1	1	APP	1	33%
OPAJNC	1	0	0	1	APP	1	33%
OPAJWE	1	0	0	1	APP	1	33%
OPAOIT	0	0	1	1	APP	1	33%
OPARCV	1	0	0	1	APP	1	33%
OPARIT	1	0	0	1	APP	1	33%
OPASEL	1	0	0	1	APP	1	33%
OPASFC	1	0	0	1	APP	1	33%
OPASMF	1	0	0	1	APP	1	33%
OPATSC	1	0	0	1	APP	1	33%
OPAXMT	1	0	0	1	APP	1	33%
OPSDAR	1	0	0	1	APP	1	33%
OPSJSA	1	0	0	1	APP	1	33%
OPSRMC	1	0	0	1	APP	1	33%
OPSSEA	1	0	0	1	APP	1	33%
OPSSMR	1	0	0	1	APP	1	33%
ORPI	1	0	0	1	DB	1	33%
OSIDIP	0	1	0	1	APP	1	33%
OSIDOC	0	1	0	1	APP	1	33%
OSIINC	0	1	0	1	APP	1	33%
OSIINR	0	1	0	1	APP	1	33%
OSIOAP	0	1	0	1	APP	1	33%
OSIOSC	0	1	0	1	APP	1	33%
OSIOSD	0	1	0	1	APP	1	33%
OSISC	0	1	0	1	APP	1	33%
OSISRV	0	1	0	1	APP	1	33%

Module	96-1	96-2	97-1	Grand Total	Module Type	Release Count	Release Percent
PMEDCN	0	1	0	1	APP	1	33%
PMESST	0	1	0	1	APP	1	33%
PRDDAV	1	0	0	1	APP	1	33%
PRDEFM	1	0	0	1	APP	1	33%
PRDESI	1	0	0	1	APP	1	33%
PRDNCP	0	1	0	1	APP	1	33%
PRDNCU	0	1	0	1	APP	1	33%
PRDVPR	0	1	0	1	APP	1	33%
RDDS	1	0	0	1	DB	1	33%
RTAIPR	0	1	0	1	APP	1	33%
RTARTA	0	0	1	1	APP	1	33%
RTX	1	0	0	1	APP	1	33%
SATL	0	0	1	1	DB	1	33%
SAVREM	0	0	1	1	APP	1	33%
SCPA	0	0	1	1	DB	1	33%
SCPL	0	0	1	1	DB	1	33%
SCR	0	0	1	1	APP	1	33%
SDIR	0	0	1	1	DB	1	33%
SDTDPG	0	1	0	1	APP	1	33%
SFMWKD	0	1	0	1	APP	1	33%
SMAAEA	0	1	0	1	APP	1	33%
SMADEA	0	1	0	1	APP	1	33%
SMAGSA	0	1	0	1	APP	1	33%
SMAMCD	0	1	0	1	APP	1	33%
SMAMSA	0	1	0	1	APP	1	33%
SMASAV	0	1	0	1	APP	1	33%
SMASAW	0	1	0	1	APP	1	33%
SMASPS	0	1	0	1	APP	1	33%
SP07	0	0	1	1	DB	1	33%
SP09	0	0	1	1	DB	1	33%
SP10	0	0	1	1	DB	1	33%
SP11	0	0	1	1	DB	1	33%
SP12	0	0	1	1	DB	1	33%
SPTASF	0	1	0	1	APP	1	33%
SPTEQP	0	0	1	1	APP	1	33%
SPTIOD	0	1	0	1	APP	1	33%
SPTIOT	0	0	1	1	APP	1	33%
SPTRAF	1	0	0	1	APP	1	33%
SPTRDW	1	0	0	1	APP	1	33%
SPTRRI	1	0	0	1	APP	1	33%
SPTRSW	1	0	0	1	APP	1	33%
SPTSVP	0	1	0	1	APP	1	33%
SRFC	0	0	1	1	DB	1	33%
SRNK	0	0	1	1	DB	1	33%
SSTD	1	0	0	1	DB	1	33%
STRF	0	0	1	1	DB	1	33%
STTGTO	0	1	0	1	APP	1	33%

Module	96-1	96-2	97-1	Grand Total	Module Type	Release Count	Release Percent
STTMGR	1	0	0	1	APP	1	33%
STTMGS	1	0	0	1	APP	1	33%
STTSTT	0	1	0	1	APP	1	33%
TPRM	0	0	1	1	DB	1	33%
TSKSAN	0	1	0	1	APP	1	33%
TSKTFM	0	1	0	1	APP	1	33%
TSKTIN	0	1	0	1	APP	1	33%
TSKTMG	0	1	0	1	APP	1	33%
TSSN	0	0	1	1	DB	1	33%
WMGREM	0	0	1	1	APP	1	33%

b. SPADOC Change Driver Tabulation

Change Object	94-1	96-1	96-2	97-1	Grand Total	Release Count	Release Pct	Exposure	Exposure Percent
display	5	6	22	3	36	4	100%	144	38%
message	11	0	11	6	28	3	75%	84	22%
file	2	2	3	0	7	3	75%	21	6%
menu	1	0	12	0	13	2	50%	26	7%
dialog	3	4	0	0	7	2	50%	14	4%
satellite	0	6	0	1	7	2	50%	14	4%
COTS	0	3	0	3	6	2	50%	12	3%
command	2	0	1	0	3	2	50%	6	2%
element set	0	1	1	0	2	2	50%	4	1%
notification	0	1	0	1	2	2	50%	4	1%
orbit	0	0	1	1	2	2	50%	4	1%
security	0	1	0	1	2	2	50%	4	1%
site	0	7	0	0	7	1	25%	7	2%
baud rate	0	0	2	0	2	1	25%	2	1%
options	0	2	0	0	2	1	25%	2	1%
applications	0	0	1	0	1	1	25%	1	0%
ASAT	0	0	1	0	1	1	25%	1	0%
buffer	0	0	1	0	1	1	25%	1	0%
catalog	1	0	0	0	1	1	25%	1	0%
CIS	0	0	1	0	1	1	25%	1	0%
cluster	0	1	0	0	1	1	25%	1	0%
collision avoidance	1	0	0	0	1	1	25%	1	0%
consistency	0	0	1	0	1	1	25%	1	0%
count	0	1	0	0	1	1	25%	1	0%
decay dates	1	0	0	0	1	1	25%	1	0%
edit	0	1	0	0	1	1	25%	1	0%
ephemeris	0	1	0	0	1	1	25%	1	0%
extrapolation DC	0	0	1	0	1	1	25%	1	0%
folder	0	1	0	0	1	1	25%	1	0%
interface	0	0	1	0	1	1	25%	1	0%
IRONs	0	0	1	0	1	1	25%	1	0%
keyboard	0	0	0	1	1	1	25%	1	0%

Change Object	**94-1**	**96-1**	**96-2**	**97-1**	**Grand Total**	**Release Count**	**Release Pct**	**Exposure**	**Exposure Percent**
keys	0	0	1	0	1	1	25%	1	0%
manuever	0	0	1	0	1	1	25%	1	0%
msg	0	1	0	0	1	1	25%	1	0%
multipliers	0	0	1	0	1	1	25%	1	0%
observation	0	0	0	1	1	1	25%	1	0%
printer	0	0	0	1	1	1	25%	1	0%
projection	0	1	0	0	1	1	25%	1	0%
screen print	1	0	0	0	1	1	25%	1	0%
solar	1	0	0	0	1	1	25%	1	0%
threat	0	0	1	0	1	1	25%	1	0%
vote	0	1	0	0	1	1	25%	1	0%
window	0	1	0	0	1	1	25%	1	0%
Grand Total	29	42	65	19	155				

Appendix 2

Volatility Characterization - CCPDS-R

a. CCPDS-R Module Volatility Tabulation

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
DGN_MENU_FORMS	0	4	2	1	2	1	0	10	.ADB	5	71%
ENUMERATED_TYPES	0	0	1	1	7	2	0	11	.DAT	4	57%
USER_CONTROL	0	1	3	4	1	0	0	9	.ADB	4	57%
CMP_PRECALCULATED_AOI_DATA	3	0	1	0	2	2	0	8	.DAT	4	57%
CMP_PRECALCULATED_THREAT_FAN	3	0	1	0	2	1	0	7	.DAT	4	57%
BUILD_DISPLAY_DATABASE	0	0	1	2	1	2	0	6	.COM	4	57%
CMP_LOFT_ANGLE_DATA	1	0	1	0	2	2	0	6	.DAT	4	57%
SSP_MISSION_DATA_COUNTS	0	0	0	1	2	1	2	6	.ADB	4	57%
CMP_GRAZE_BUFFER_PARAMETERS	1	0	1	0	2	1	0	5	.ADB	4	57%
CMP_RELATIVE_MULTIPLE_MIDPOINT_TIC	1	0	1	0	2	1	0	5	.DAT	4	57%
CMP_STRATEGIC_ADAPTATION	0	0	1	2	1	1	0	5	.DAT	4	57%
DGN_GEOGRAPHIC	0	1	2	0	1	1	0	5	.ADB	4	57%
SSP_SEWS_ML_PROCESSING	1	0	0	0	1	1	2	5	.ADB	4	57%
CSD_SYSTEM_CONTROL_PROCEDURES	0	0	1	1	1	1	0	4	.ADB	4	57%
DGN_SCREEN_MANAGER	0	4	5	1	0	0	0	10	.ADB	3	43%
SGI_BASIC_TYPES_	0	0	0	1	6	3	0	10	.ADS	3	43%
SGI_TABLE_IX_	0	0	0	3	3	2	0	8	.ADS	3	43%
SGI_DISPLAY_TEXT_	0	0	0	2	4	1	0	7	.ADS	3	43%
SCN_SYSC_CONNECTIVITY_SITES	0	0	1	0	3	2	0	6	.DAT	3	43%
CMP_DCO_AOI_DB	1	0	1	0	3	0	0	5	.DAT	3	43%
CMP_NMP_AOI_DB	1	0	1	0	3	0	0	5	.DAT	3	43%
CMP_RMP_AOI_DB	1	0	1	0	3	0	0	5	.DAT	3	43%
CMP_SSP_AOI_DB	1	0	1	0	3	0	0	5	.DAT	3	43%
CMT_TOOL_TYPES_	0	0	0	2	2	1	0	5	.ADS	3	43%
CSD_INIT	0	0	0	2	2	1	0	5	.COM	3	43%
FOM_FOE_OUTPUT_NON_DISCRETE_MESSAGES	0	0	0	1	3	1	0	5	.ADB	3	43%
SCN_OMP_MESSAGE_CATEGORY_FILE	0	0	0	1	3	1	0	5	.DAT	3	43%
SGI_CSSR_FORMAT_IDS_	0	0	0	2	2	1	0	5	.ADS	3	43%
SGI_DISPLAY_TYPES_	0	0	0	2	2	1	0	5	.ADS	3	43%
SGI_RECORD_ASSIGNMENT	0	0	0	2	1	0	2	5	.ADB	3	43%
SSP_BASIC_TYPES_	0	0	0	1	2	0	2	5	.ADS	3	43%
SSP_SEWS_DATABASE_	0	0	0	1	2	0	2	5	.ADS	3	43%
CCO_MESSAGE_INDEX_NUMBERS	0	0	0	1	2	1	0	4	.DAT	3	43%
CCO_MESSAGE_PRIORITIES	0	0	0	1	2	1	0	4	.DAT	3	43%
CMP_AOI_DATA	1	0	1	0	2	0	0	4	.DAT	3	43%
CMP_CMP_COMPLEX_DB	1	0	1	0	2	0	0	4	.DAT	3	43%
CMP_DCO_COMPLEX_DB	1	0	1	0	2	0	0	4	.DAT	3	43%
CMP_MISSILE_TYPING	1	0	1	0	2	0	0	4	.DAT	3	43%
CMP_WORLD_LAUNCH_COMPLEXES	1	0	1	0	2	0	0	4	.DAT	3	43%
CMT_VIM_BODY_TOOL	0	0	0	1	1	0	2	4	.ADB	3	43%
CSD_RECONFIGURATION_PROCEDURES	0	0	0	1	1	2	0	4	.ADB	3	43%
DGN_FDB_FORMAT_BUILD_UTILITIES	0	0	1	2	1	0	0	4	.ADB	3	43%
FOM_FORMAT_PDS_M1A	0	0	1	1	2	0	0	4	.ADB	3	43%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
MESSAGE_INPUT_FILE	0	0	0	1	2	1	0	4	.DAT	3	43%
PIM_VIM_MISSION_MESSAGE_DATA	0	0	1	0	1	0	2	4	.ADB	3	43%
SDG_MESSAGE_FORMATTER	1	0	0	1	2	0	0	4	.ADB	3	43%
SGI_CSSR_MESSAGE_IDS_	0	0	0	1	2	1	0	4	.ADS	3	43%
CCO_SUMMARY_MESSAGE_FORMAT_IDS	0	0	0	1	1	1	0	3	.DAT	3	43%
CMP_SENSOR_LOCATIONS	1	0	0	0	1	1	0	3	.DAT	3	43%
CSD_ERM_ALARM_PROCEDURES	0	0	1	1	1	0	0	3	.ADB	3	43%
CSD_OPERATIONS	0	0	0	1	1	1	0	3	.ADB	3	43%
FOM_CSSR_MESSAGE_HEADER_UTILITIES	0	0	0	1	1	1	0	3	.ADB	3	43%
FOM_FOE_PERIODIC_TYPES_	0	0	0	1	1	1	0	3	.ADS	3	43%
FOM_FOE_PROCESS_PERIODIC_MESSAGES	0	0	0	1	1	1	0	3	.ADB	3	43%
SCN_SERVICES_PROCESSING	0	0	0	1	1	1	0	3	.ADB	3	43%
SGI_ERROR_CODES	0	0	1	1	1	0	0	3	.ADB	3	43%
USER_CURRENT_SITE_STATUS_POPUP	0	0	1	1	1	0	0	3	.RAW	3	43%
USER_SYSTEM_CONTROL_INTERFACE	0	0	1	1	0	1	0	3	.ADB	3	43%
FOM_FORMAT_ATAMS_STATUS	0	0	0	0	5	3	0	8	.ADB	2	29%
DGN_ALARM_PROCESSOR	0	4	3	0	0	0	0	7	.ADB	2	29%
VMSP_PERF_INTERFACE	0	0	0	0	3	3	0	6	.COM	2	29%
USER_ALARM_PROCESSING	0	0	4	0	0	1	0	5	.ADB	2	29%
USER_WORKSTATION_STATUS	0	0	0	2	0	3	0	5	.ADB	2	29%
FOM_FOE_ACTIONS	0	0	0	1	3	0	0	4	.ADB	2	29%
SCN_PROCESS_KEEPALIVE_MESSAGE	0	0	1	3	0	0	0	4	.ADB	2	29%
SCN_VIM_PROCESSING	0	0	1	3	0	0	0	4	.ADB	2	29%
CMT_LTD_LOAD_DCO_DISPLAY	0	0	0	2	1	0	0	3	.ADB	2	29%
CSD_INITIALIZATION	0	0	0	1	2	0	0	3	.ADB	2	29%
DGN_CONFIGURATION	0	0	0	2	0	1	0	3	.ADB	2	29%
DGN_MENU_TYPES_	0	0	0	2	1	0	0	3	.ADS	2	29%
FDB_UTILITIES	0	0	0	1	0	0	2	3	.ADB	2	29%
FOM_FORMAT_FORWARD_USERS_AN26	0	0	1	0	2	0	0	3	.ADB	2	29%
FOM_FORMAT_PDS_M2	0	0	0	2	1	0	0	3	.ADB	2	29%
NMP_BASIC_TYPES_	0	0	0	0	1	0	2	3	.ADS	2	29%
NMP_NUDET_DATABASE_	0	0	0	0	1	0	2	3	.ADS	2	29%
NMP_SATELLITE_NUDET_PROCESSING	0	0	0	0	1	0	2	3	.ADB	2	29%
PIM_VIM_KEEPALIVE_PROCESSING	0	0	1	0	2	0	0	3	.ADB	2	29%
PIM_VIM_SEWS_ML_EVENT_MESSAGE	0	0	1	0	0	0	2	3	.ADB	2	29%
PIM_VIM_SEWS_MLU_EVENT_MESSAGE	0	0	1	0	0	0	2	3	.ADB	2	29%
RMP_RADAR_DATABASE	0	0	2	0	1	0	0	3	.ADB	2	29%
RMP_RMP_RADAR_TASK_COMP	0	0	0	2	1	0	0	3	.ADB	2	29%
SDG_LOAD_DATABASE_FILES	1	0	0	0	2	0	0	3	.ADB	2	29%
SGI_FUSED_DATABASE_	0	0	0	0	2	1	0	3	.ADS	2	29%
SSP_INITIALIZE_IDB_DATA	0	0	0	1	0	0	2	3	.ADB	2	29%
SSP_REPAIR_THREAD	0	0	0	1	0	0	2	3	.ADB	2	29%
SSP_SEWS_DATABASE	0	0	0	1	0	0	2	3	.ADB	2	29%
SSP_SEWS_R_PROCESSING	0	0	0	0	0	1	2	3	.ADB	2	29%
STARTUP_SHADOW_TWAA1	0	0	0	0	1	2	0	3	.RCF	2	29%
STARTUP_SHADOW_TWAA2	0	0	0	0	1	2	0	3	.RCF	2	29%
USER_ALARM_ASSIGNMENT_SUMMARY_MWC	0	0	1	0	0	2	0	3	.RAW	2	29%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
ARCH_MONITOR_LOG_PROCEDURES	0	0	0	1	1	0	0	2	.ADB	2	29%
ARCH_MONITOR_LOG_PROCEDURES_	0	0	0	1	1	0	0	2	.ADS	2	29%
CCPDSR_SOFTWARE_VERSION	1	1	0	0	0	0	0	2	.DAT	2	29%
CCS_GEOGRAPHIC_SEARCH	1	0	0	0	1	0	0	2	.ADB	2	29%
CMP_LORIG_CHAR_DB	1	0	0	0	1	0	0	2	.DAT	2	29%
CMP_LORIG_LATITUDE_DB	1	0	0	0	1	0	0	2	.DAT	2	29%
CMP_LORIG_LONGITUDE_DB	1	0	0	0	1	0	0	2	.DAT	2	29%
CMP_MISSILE_INVENTORY	0	0	1	1	0	0	0	2	.DAT	2	29%
CONI_CONFIGURATION_PROCESSING	0	0	0	1	0	1	0	2	.ADB	2	29%
CREATE_VXT_TERMINAL_ACCOUNT	0	0	0	0	1	1	0	2	.COM	2	29%
CSD_CONFIGURE_SYSTEM	0	0	0	1	1	0	0	2	.ADB	2	29%
CSD_CREATE_SUBSYSTEM_SENSOR_STATUS_MENU	0	0	1	0	1	0	0	2	.ADB	2	29%
CSD_INTERNAL_STRUCTURES_	0	0	0	1	1	0	0	2	.ADS	2	29%
CSD_MONITOR_LOG_PROCEDURES	0	0	0	1	0	1	0	2	.ADB	2	29%
CSD_SYSTEM_CONTROL_PROCEDURES_	0	0	0	1	1	0	0	2	.ADS	2	29%
DGN_FIELD_CONTROL_IO	0	0	0	0	1	1	0	2	.ADB	2	29%
DGN_FORMAT_BUILD_TYPES_	0	0	0	1	1	0	0	2	.ADS	2	29%
DGN_GEOGRAPHIC_	0	1	0	0	1	0	0	2	.ADS	2	29%
DGN_IR_COVERAGES	0	0	0	0	1	1	0	2	.DAT	2	29%
DGN_MAP_GENERATOR	0	1	0	0	1	0	0	2	.ADB	2	29%
DGN_MENU_UTILITIES	0	0	1	0	1	0	0	2	.ADB	2	29%
DISP_TASK_INIT	0	0	0	1	1	0	0	2	.COM	2	29%
FIL_CSSR_FORMAT_IDS_	0	0	0	1	1	0	0	2	.ADS	2	29%
FIL_PDS_FORMAT_IDS_	0	0	0	1	1	0	0	2	.ADS	2	29%
FIL_SAC_FORMAT_IDS_	0	0	0	1	1	0	0	2	.ADS	2	29%
FIL_UPDATE_SCENARIO_TIMES	0	0	0	1	1	0	0	2	.ADB	2	29%
FOM_FOE_BASIC_TYPES_	0	0	0	1	0	1	0	2	.ADS	2	29%
FOM_FORMAT_ATAMS_STATUS_	0	0	0	0	1	1	0	2	.ADS	2	29%
FOM_FORMAT_FORWARD_USERS_AU09	0	0	1	0	1	0	0	2	.ADB	2	29%
FOM_FORMAT_PDS_M17	0	0	0	1	0	1	0	2	.ADB	2	29%
FOM_FORMAT_PDS_M1B	0	0	0	1	0	1	0	2	.ADB	2	29%
FOM_FORMAT_PDS_M3	0	0	0	1	1	0	0	2	.ADB	2	29%
FOM_FORMAT_PDS_M5B	0	0	1	1	0	0	0	2	.ADB	2	29%
FOM_FORMAT_PDS_N2	0	0	0	1	1	0	0	2	.ADB	2	29%
FOM_FORMAT_PDS_T1	0	0	1	1	0	0	0	2	.ADB	2	29%
GAC_PROCESS_TEMPLATE	0	0	0	1	0	1	0	2	.ADB	2	29%
GAC_TYPES_	0	0	0	0	1	1	0	2	.ADS	2	29%
INJP_MESSAGE_INJECTION	0	0	1	0	0	1	0	2	.ADB	2	29%
M2_MISSILE_LAUNCH_SUMMARY_CM07	0	0	0	1	1	0	0	2	.OUT	2	29%
MCIO_PUT_BASIC_UTILITIES	0	0	0	0	1	1	0	2	.ADB	2	29%
PIM_VIM_ACTION_A	0	0	1	1	0	0	0	2	.ADB	2	29%
PIM_VIM_START_UP	0	0	1	0	1	0	0	2	.DAT	2	29%
QPR_ANALYSIS_SQL_SUPPORT	0	0	0	1	1	0	0	2	.SQLMOD	2	29%
QPR_SENSOR_SQL_SUPPORT	0	0	0	1	1	0	0	2	.SQLMOD	2	29%
QPR_SQL_SUPPORT	0	0	0	1	1	0	0	2	.SQLMOD	2	29%
RESUME_REAL_EXTERNAL_INPUTS	0	0	0	1	0	1	0	2	.INC	2	29%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
RMP_TIME_TO_IMPACT	0	0	0	0	1	1	0	2	.ADB	2	29%
SAS_TASKS	0	0	0	1	0	1	0	2	.DAT	2	29%
SCN_APPLICATION_PROCESSING	0	0	1	0	0	1	0	2	.ADB	2	29%
SCN_SYSC_DATA_	0	0	0	0	1	1	0	2	.ADS	2	29%
SCN_SYSC_NETWRK_CONTROL_TASK_COMP	0	0	0	0	1	1	0	2	.ADB	2	29%
SGI_MCIO_STATUS_MESSAGE_	0	0	0	0	1	1	0	2	.ADS	2	29%
SHUTDOWN_NETWORK	0	0	0	1	0	1	0	2	.RCF	2	29%
SPM_TASK_INIT	0	0	0	1	1	0	0	2	.COM	2	29%
SSP_READ_MISSILE_TYPING	1	0	0	1	0	0	0	2	.ADB	2	29%
SSP_TASK_INIT	0	0	0	1	1	0	0	2	.COM	2	29%
SUSPEND_REAL_EXTERNAL_INPUTS	0	0	0	1	0	1	0	2	.INC	2	29%
TAS_INPUT_FILE	0	0	0	1	1	0	0	2	.DAT	2	29%
USER_CMAFB_SYSTEM_STATUS_RELATED	0	0	1	0	1	0	0	2	.RAW	2	29%
USER_DISP_DISPLAYS_TASK_COMP	0	0	0	1	0	1	0	2	.ADB	2	29%
USER_INTEGRATED_ATTACK_SUMMARY_RELATED	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_LAUNCH_ORIGIN_SUMMARY	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_MISSILE_ATTACK_SUMMARY_RELATED	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_NORAD_NUDET_ASSESSMENT_RELATED	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_NORAD_REFINED_RADAR_COMPOSIT_F92	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_NUDET_ASSESSMENT_RELATED	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_NUDET_SITUATION_RELATED	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_OPCC_SYSTEM_STATUS_RELATED	0	0	1	0	1	0	0	2	.RAW	2	29%
USER_REFINED_RADAR_COMPOSITE_RELATED	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_SENSOR_SITUATION_RELATED	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_STRATEGIC_MOB_SUMMARY	0	0	0	1	1	0	0	2	.RAW	2	29%
USER_STRATEGIC_SUMMARY	0	0	0	1	1	0	0	2	.RAW	2	29%
PIM_VIM_RADAR_CAPABILITY	0	0	0	4	0	0	0	4	.ADB	1	14%
SDG_EVENT_MESSAGE_GENERATOR	0	0	0	0	4	0	0	4	.ADB	1	14%
CONI_PROCESS_GSM_INJ_MESSAGE	0	0	0	3	0	0	0	3	.ADB	1	14%
DGN_ALARM_PROCESSOR_	0	3	0	0	0	0	0	3	.ADS	1	14%
DGN_SCREEN_MANAGER_	0	3	0	0	0	0	0	3	.ADS	1	14%
GENB_MC_CONSTRUCT_ASCII_DATA	0	0	0	3	0	0	0	3	.ADB	1	14%
SDG_MESSAGE_TIME_ORDERING	0	0	0	0	3	0	0	3	.ADB	1	14%
SDG_STATISTICS_POST_PROCESSOR	0	0	0	0	3	0	0	3	.ADB	1	14%
USER_DIRECT_RADAR_SITE_STATUS	0	0	0	3	0	0	0	3	.RAW	1	14%
VXT_TERM_LOGIN	0	0	0	0	3	0	0	3	.COM	1	14%
CDT_THREAT_FAN_GRAPHICS	0	0	0	0	2	0	0	2	.ADB	1	14%
CMP_LORIG_ADAPTATION	0	0	0	0	2	0	0	2	.DAT	1	14%
CMT_VIM_SPEC_TOOL	0	0	0	0	0	0	2	2	.ADB	1	14%
CONR_RUNTIME_CONTROL	0	0	2	0	0	0	0	2	.ADB	1	14%
COR_STATUS_ALARM	0	0	0	0	0	2	0	2	.ADB	1	14%
CSO_LOGIN	0	0	0	0	2	0	0	2	.COM	1	14%
DECW$CSO_TERM	0	0	0	0	2	0	0	2	.DAT	1	14%
DECW$ENDSESSION	0	0	0	0	2	0	0	2	.DAT	1	14%
DECW$MWM	0	0	0	0	2	0	0	2	.DAT	1	14%
DECW$MWM_RC	0	0	0	0	2	0	0	2	.DAT	1	14%
DRD_VIEW_REAL_MESSAGE	0	0	0	0	0	0	2	2	.ADB	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
FOM_FORMAT_PDS_M11	0	0	0	2	0	0	0	2	.ADB	1	14%
GDS_KEEP_ALIVE_YZ31	0	0	0	0	2	0	0	2	.INB	1	14%
GENB_MC_CONSTRUCT_MESSAGE_CONTENTS	0	0	0	2	0	0	0	2	.ADB	1	14%
GENB_MC_RETRIEVE_ASCII_DATA	0	0	0	2	0	0	0	2	.ADB	1	14%
GENB_MC_RETRIEVE_MESSAGE_CONTENTS	0	0	0	2	0	0	0	2	.ADB	1	14%
GENB_MC_RETRIEVE_MESSAGE_DEFAULTS	0	0	0	2	0	0	0	2	.ADB	1	14%
GENB_MESSAGE_CONTENTS	0	0	0	2	0	0	0	2	.ADB	1	14%
GENB_PROCESS_GSM_INJ_MESSAGE	0	0	0	2	0	0	0	2	.ADB	1	14%
GENU_EP_PROCESS_DISTRIBUTED_CONTENTS	0	0	0	2	0	0	0	2	.ADB	1	14%
MCIO_PUT_QUEUE_UTILITIES	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTION_CK	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTION_CR	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTION_IT	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTION_K	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTION_KR	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTION_T	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTIONS	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ACTIONS_	0	0	0	0	0	0	2	2	.ADS	1	14%
PIM_VIM_CHECK_INBOUND_TASK_COMP	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_DISPOSITION_PROCESSING	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_DUP_MESSAGES	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_DUP_MESSAGES_	0	0	0	0	0	0	2	2	.ADS	1	14%
PIM_VIM_ICADS_EVENT_MESSAGE	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_ICADS_EVENT_MESSAGE_	0	0	0	0	0	0	2	2	.ADS	1	14%
PIM_VIM_MESSAGE_HANDLER	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_MESSAGE_HANDLER_	0	0	0	0	0	0	2	2	.ADS	1	14%
PIM_VIM_NUDET_EVENT_MESSAGE	0	0	0	0	0	0	2	2	.ADB	1	14%
PIM_VIM_NUDET_EVENT_MESSAGE_	0	0	0	0	0	0	2	2	.ADS	1	14%
PIM_VIM_SEWS_ML_EVENT_MESSAGE_	0	0	0	0	0	0	2	2	.ADS	1	14%
PIM_VIM_SEWS_MLU_EVENT_MESSAGE_	0	0	0	0	0	0	2	2	.ADS	1	14%
PMP_PROCESS_FIELD_BINARY	0	0	0	0	0	2	0	2	.ADB	1	14%
PMP_WRITE_MESSAGE_LOG	0	0	0	0	0	0	2	2	.ADB	1	14%
QPR_ANALYSIS_INTERFACE	0	0	0	2	0	0	0	2	.ADB	1	14%
QPR_ANALYSIS_SQL_SUPPORT_	0	0	0	2	0	0	0	2	.ADS	1	14%
QPR_CREATE_VIEWS	0	0	0	2	0	0	0	2	.ADB	1	14%
QPR_CREATE_VIEWS_SQL_	0	0	0	2	0	0	0	2	.ADS	1	14%
QPR_DATA_ANALYSIS	0	0	0	2	0	0	0	2	.ADB	1	14%
QPR_DATA_REDUCTION	0	0	0	2	0	0	0	2	.ADB	1	14%
QPR_DATA_REDUCTION_INTERFACE_	0	0	0	2	0	0	0	2	.ADS	1	14%
QPR_INTERNAL_STRUCTURES_	0	0	0	2	0	0	0	2	.ADS	1	14%
QPR_OPERATIONS	0	0	0	2	0	0	0	2	.ADB	1	14%
QPR_PROCESS	0	0	0	2	0	0	0	2	.ADB	1	14%
QPR_SQL_SUPPORT_	0	0	0	2	0	0	0	2	.ADS	1	14%
RTR_RECORDING_TYPES	0	0	0	0	0	0	2	2	.ADB	1	14%
RTR_RTAD_RECORDING_TASK_COMP	0	0	0	0	0	0	2	2	.ADB	1	14%
RTR_RTRD_RECORDING_TASK_COMP	0	0	0	0	2	0	0	2	.ADB	1	14%
SCN_OMP_ACTION_C	0	0	0	0	0	0	2	2	.ADB	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
SCN_OMP_GLOBAL_VARIABLES	0	0	0	0	0	2	0	2	.ADB	1	14%
SCN_SYSC_NETWRK_CONTROL_TASR_COMP	0	0	0	0	0	2	0	2	.ADB	1	14%
SCN_SYSC_TYPES_	0	0	0	0	0	2	0	2	.ADS	1	14%
SDG_LOAD_DATABASE_FILES_	0	0	0	0	2	0	0	2	.ADS	1	14%
SDG_MISSION_GENERATOR	0	0	0	0	2	0	0	2	.ADB	1	14%
SGI_BASIC_TYPES	0	0	2	0	0	0	0	2	.ADB	1	14%
SGI_CSSR_INTERFACE_MESSAGE_	0	0	0	0	0	0	2	2	.ADS	1	14%
SGI_DISPLAY_TEXT	0	0	2	0	0	0	0	2	.ADB	1	14%
SGI_RESET_DUPLICATE_MESSAGE_	0	0	0	0	0	0	2	2	.ADS	1	14%
SGI_TABLE_IX	0	0	2	0	0	0	0	2	.ADB	1	14%
SSP_READ_AREA_OF_INTEREST_POINTS	0	0	0	0	2	0	0	2	.ADB	1	14%
SSP_SSP_SEWS_TASK_COMP	0	0	0	2	0	0	0	2	.ADB	1	14%
START_LOGIN	0	0	0	0	2	0	0	2	.COM	1	14%
TDBA_DATABASE_ACCESS_	0	0	0	2	0	0	0	2	.ADS	1	14%
USER_SUMMARY_DATA_MENU	0	0	0	2	0	0	0	2	.RAW	1	14%
USER_SYSTEM_SUMMARY	0	0	0	0	0	2	0	2	.RAW	1	14%
ARCH_CONTROL_PROCEDURES	0	0	0	1	0	0	0	1	.ADB	1	14%
ARCH_CONTROL_PROCEDURES_	0	0	0	1	0	0	0	1	.ADS	1	14%
ARCH_TASK_INIT	0	0	0	1	0	0	0	1	.COM	1	14%
ARCHIVE_CONTROL_TASK_COMP	0	0	0	1	0	0	0	1	.ADB	1	14%
ARCN_INTERNAL_STRUCTURES_	0	0	0	1	0	0	0	1	.ADS	1	14%
CCO_APCC_ADDRESSES_OPERATIONAL	0	0	0	0	1	0	0	1	.DAT	1	14%
CCO_CMAFB_ADDRESSES_OPERATIONAL	0	0	0	0	1	0	0	1	.DAT	1	14%
CCO_MAP_DESTINATION_TO_LL	0	0	0	0	1	0	0	1	.DAT	1	14%
CCO_TDTC_ADDRESSES_OPERATIONAL	0	0	0	0	1	0	0	1	.DAT	1	14%
CCPDSR_STATUS_KS01	0	0	0	0	1	0	0	1	.OUT	1	14%
CCPDSR_STATUS_REEP_ALIVE_KS02	0	0	0	0	0	1	0	1	.OUT	1	14%
CCS_GEOGRAPHIC_SEARCH_	0	0	0	0	1	0	0	1	.ADS	1	14%
CDT_AOI_BUILD_PROCEDURES	0	0	0	0	1	0	0	1	.ADB	1	14%
CDT_AOI_MAPPING	1	0	0	0	0	0	0	1	.DAT	1	14%
CFG_NAS_MESSAGE_INFORMATION_	0	0	0	1	0	0	0	1	.ADS	1	14%
CMN_DEV_TO_OPERATIONAL	0	0	0	1	0	0	0	1	.COM	1	14%
CMP_LORIG_APE_DB	1	0	0	0	0	0	0	1	.DAT	1	14%
CMP_TRD_COORDINATE_DB	0	0	0	0	1	0	0	1	.DAT	1	14%
CMP_TRD_COUNTRY_DB	0	0	0	0	1	0	0	1	.DAT	1	14%
CMP_TRD_SEGMENT_DB	0	0	0	0	1	0	0	1	.DAT	1	14%
CMT_LTD_LOAD_FIELD_DEFINIT1ON	0	0	0	1	0	0	0	1	.ADB	1	14%
CMT_LTD_LOAD_FIELD_DEFINITION_	0	0	0	1	0	0	0	1	.ADS	1	14%
CMT_SGI_WRITE_SPEC_FOM	0	0	0	0	1	0	0	1	.ADB	1	14%
CMT_TOOL_VIM_MAKE_ENUMERATION_BOTH	0	0	0	1	0	0	0	1	.ADB	1	14%
CMT_VIM_ROUTINES	0	0	0	1	0	0	0	1	.ADB	1	14%
COMPLEX	0	0	0	0	1	0	0	1	.INPUT	1	14%
CONI_CONI_INITIATION_TASK_COMP	0	0	1	0	0	0	0	1	.ADB	1	14%
CONI_CONI_INITIATION_TASR_COMP	0	0	0	0	0	1	0	1	.ADB	1	14%
CONI_SCENARIO_INITIATION	0	0	1	0	0	0	0	1	.ADB	1	14%
CONI_START_PROCESSING	0	0	1	0	0	0	0	1	.ADB	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
CONR_BUILD_EXECUTION_CONTROL_DISPLAY	0	0	1	0	0	0	0	1	.ADB	1	14%
CONR_REPOSITION_SCENARIO	0	0	1	0	0	0	0	1	.ADB	1	14%
COR_CONTROL_PROCESSING	0	0	0	0	0	1	0	1	.ADB	1	14%
COR_DATA_MAINTENANCE	0	0	0	0	0	1	0	1	.ADB	1	14%
COR_ROUTING	0	0	0	0	0	1	0	1	.ADB	1	14%
COR_TYPES_	0	0	0	0	0	1	0	1	.ADS	1	14%
CSD_CANCEL_DATA_RECORDING_CHANGES	0	0	0	0	1	0	0	1	.ADB	1	14%
CSD_CANCEL_MESSAGE_PRINTING_CHANGES	0	0	0	0	1	0	0	1	.ADB	1	14%
CSD_CONFIRM_TAPE_NOTIFICATION	0	0	0	1	0	0	0	1	.ADB	1	14%
CSD_CREATE_HARDWARE_STATUS_MENU	0	0	0	0	1	0	0	1	.ADB	1	14%
CSD_CREATE_HARDWARE_STATUS_SUBMENU	0	0	0	0	1	0	0	1	.ADB	1	14%
CSD_CREATE_PRINTABLE_MESSAGES_MENU	0	0	0	0	1	0	0	1	.ADB	1	14%
CSD_CREATE_RECORDABLE_DATA_MENU	0	0	0	0	1	0	0	1	.ADB	1	14%
CSD_ERM_ALARM_PROCEDURES_	0	0	0	1	0	0	0	1	.ADS	1	14%
CSD_INITIALIZATION_	0	0	0	0	1	0	0	1	.ADS	1	14%
CSD_MENUS	0	0	0	0	1	0	0	1	.ADB	1	14%
CSD_MENUS_	0	0	0	1	0	0	0	1	.ADS	1	14%
CSD_MONITOR_LOG_PROCEDURES_	0	0	0	0	1	0	0	1	.ADS	1	14%
CSD_OPERATIONS_	0	0	0	1	0	0	0	1	.ADS	1	14%
CSD_PROCESS_KEYBOARD_INPUT	0	0	0	1	0	0	0	1	.ADB	1	14%
CSD_RECONFIGURATION_PROCEDURES_	0	0	0	1	0	0	0	1	.ADS	1	14%
DECW$5MB_WINDOW_COLOR	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$CSSO_TERM	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$DRD_TERM	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$IWO_TERM	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$SMB_BACKGROUND	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$SMB_BACKGROUND_COLOR	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$SMB_WINDOW	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$VMSP_TERM	0	0	0	0	1	0	0	1	.DAT	1	14%
DECW$WSO_TERM	0	0	0	0	1	0	0	1	.DAT	1	14%
DGN_BOUNDARY_POINTS	0	0	0	0	1	0	0	1	.DAT	1	14%
DGN_BUFFER_TYPES_	0	1	0	0	0	0	0	1	.ADS	1	14%
DGN_CONFIGURATION_	0	0	0	1	0	0	0	1	.ADS	1	14%
DGN_DISPLAY_DATABASE_IO	0	0	0	0	1	0	0	1	.ADB	1	14%
DGN_FDB_DATA_STRING	0	0	0	1	0	0	0	1	.ADB	1	14%
DGN_FORMAT_BUILD	0	0	0	0	1	0	0	1	.ADB	1	14%
DGN_FORMAT_BUILD_TYPES	0	0	1	0	0	0	0	1	.ADB	1	14%
DGN_GKS_ESCAPES_	0	0	0	0	1	0	0	1	.ADS	1	14%
DGN_GRAPHICS_INTERFACE	0	1	0	0	0	0	0	1	.ADB	1	14%
DGN_GRAPHICS_INTERFACE_	0	1	0	0	0	0	0	1	.ADS	1	14%
DGN_IR_COVERAGE	1	0	0	0	0	0	0	1	.DAT	1	14%
DGN_MAP_TYPES_	0	0	0	0	1	0	0	1	.ADS	1	14%
DGN_MAPFIL	0	0	0	0	1	0	0	1	.DAT	1	14%
DGN_MENU_FORMS_	0	1	0	0	0	0	0	1	.ADS	1	14%
DGN_MOVE_DATABASE_VALUE	0	0	0	1	0	0	0	1	.ADB	1	14%
DGN_POLFIL	0	0	0	0	1	0	0	1	.DAT	1	14%
DGN_PROCESS_FIELD_DATA_LIST	0	0	0	1	0	0	0	1	.ADB	1	14%

Module	**94-2**	**95-1**	**95-2**	**96-1**	**96-2**	**97-1**	**AOC 2A**	**Grand Total**	**Module Type**	**Release Count**	**Release Percent**
DGN_RECORD_TEXT_IO	0	0	0	0	1	0	0	1	.ADB	1	14%
DGN_SINO_SOVIET_REGIONS	0	0	0	0	1	0	0	1	.DAT	1	14%
DGN_SMGR_SCREEN_TASK_COMP	0	0	0	1	0	0	0	1	.ADB	1	14%
DGN_SYMBOL_TABLE	0	1	0	0	0	0	0	1	.ADB	1	14%
DGN_SYMBOL_TABLE_	0	1	0	0	0	0	0	1	.ADS	1	14%
DGN_TRD_POINTS	0	0	0	0	1	0	0	1	.DAT	1	14%
DGN_WINDOW_MANAGEMENT	0	1	0	0	0	0	0	1	.ADB	1	14%
DGN_WINDOW_TYPES_	0	1	0	0	0	0	0	1	.ADS	1	14%
DMG_RETRIEVAL_SERVICES	0	0	0	0	1	0	0	1	.ADB	1	14%
DRD_DO_ACTIONS	0	0	0	0	1	0	0	1	.ADB	1	14%
DRD_GLOBALS_	0	0	0	0	1	0	0	1	.ADS	1	14%
DRD_MENUS	0	0	0	0	1	0	0	1	.ADB	1	14%
DRD_MENUS_	0	0	0	0	1	0	0	1	.ADS	1	14%
DRD_OPERATIONS	0	0	0	0	1	0	0	1	.ADB	1	14%
DRD_PROCESS	0	0	0	0	1	0	0	1	.ADB	1	14%
DRD_PROCESS_OPERATOR_ACTION_LOG	0	0	0	0	0	1	0	1	.ADB	1	14%
DRD_PROCESS_SYSTEM_REPORT	0	0	0	0	1	0	0	1	.ADB	1	14%
DRD_PROCESS_SYSTEM_REPORT_	0	0	0	0	1	0	0	1	.ADS	1	14%
ERM_DATABASE	0	0	0	1	0	0	0	1	.ADB	1	14%
ERM_TYPES_	0	0	0	0	1	0	0	1	.ADS	1	14%
FDB_INITIALIZE_DATABASE	0	0	0	0	1	0	0	1	.ADB	1	14%
FDB_INITIALIZE_REMOTE_PROCESS	0	0	0	1	0	0	0	1	.ADB	1	14%
FDB_MASTER_SERVICES	0	0	0	1	0	0	0	1	.ADB	1	14%
FDB_MESSAGES	0	0	0	1	0	0	0	1	.ADB	1	14%
FDB_MESSAGES_	0	0	0	1	0	0	0	1	.ADS	1	14%
FDB_READ_SERVICES	0	0	0	1	0	0	0	1	.ADB	1	14%
FDB_READ_SERVICES_	0	0	0	1	0	0	0	1	.ADS	1	14%
FDB_RECEIVE_FDB_UPDATE	0	0	0	1	0	0	0	1	.ADB	1	14%
FDB_WRITE_SERVICES	0	0	0	1	0	0	0	1	.ADB	1	14%
FDB_WRITE_SERVICES_	0	0	0	1	0	0	0	1	.ADS	1	14%
FOM_FOE_ACTION_M	0	0	0	0	0	1	0	1	.ADB	1	14%
FOM_FOE_ACTION_T	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FOE_ACTIONS_X	0	0	0	0	1	0	0	1	.ADB	1	14%
FOM_FOE_COMMON_UTILITIES	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FOE_PROCESS_DISCRETE_CHANGES	0	1	0	0	0	0	0	1	.ADB	1	14%
FOM_FOE_PROCESS_NON_DISCRETE_CHANGES	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FOE_READ_FDB_CHANGES	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FOE_READ_FDB_NON_DISCRETE	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_FORWARD_USERS_AU08	0	0	1	0	0	0	0	1	.ADB	1	14%
FOM_FORMAT_FORWARD_USERS_AUO9	0	0	0	0	1	0	0	1	.ADB	1	14%
FOM_FORMAT_FORWARD_USERS_MT02	0	0	0	0	0	1	0	1	.ADB	1	14%
FOM_FORMAT_PDS_A1	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_A2	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_C1C	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_C3	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_C4	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_I1	0	0	0	1	0	0	0	1	.ADB	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
FOM_FORMAT_PDS_I4	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M10	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M11_	0	0	0	1	0	0	0	1	.ADS	1	14%
FOM_FORMAT_PDS_M12	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M13	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M14	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M15	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M16	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M4	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M5A	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M6	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_M78?	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_N3	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_N5	0	0	0	1	0	0	0	1	.ADB	1	14%
FOM_FORMAT_PDS_T2	0	0	0	1	0	0	0	1	.ADB	1	14%
GENB_MC_INJECT_MESSAGE	0	0	0	1	0	0	0	1	.ADB	1	14%
GENU_CR_RETRIEVE_RECORDED_MESSAGES	0	0	0	1	0	0	0	1	.ADB	1	14%
INITIALIZE_NETWORK	0	0	0	0	0	1	0	1	.INI	1	14%
INITIALIZE_NETWORR	0	0	0	0	0	1	0	1	.INI	1	14%
INJP_INJECTION_PROCESSING	0	0	1	0	0	0	0	1	.ADB	1	14%
INM_CONFIGURE_SYSTEM	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_CREATE_MENU_DISPLAY	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_DISPLAY_COLORS	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_DISPLAY_COLORS_	0	0	0	0	1	0	0	1	.ADS	1	14%
INM_ERM_PROCEDURES	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_INITIALIZATION	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_MENU_DEFINITION	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_MENU_PROCESSING	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_NMO_MENUS	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_OPERATIONS	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_RECONFIGURATION_PROCEDURES	0	0	0	0	1	0	0	1	.ADB	1	14%
INM_USI_PROCEDURES_	0	0	0	0	1	0	0	1	.ADS	1	14%
INW_MENUS	0	0	1	0	0	0	0	1	.ADB	1	14%
ITC_NODE_MANAGER	0	0	0	1	0	0	0	1	.ADB	1	14%
ITC_WATCHDOG_PROCEDURES	0	0	0	1	0	0	0	1	.ADB	1	14%
ITC_WATCHDOG_PROCEDURES_	0	0	0	1	0	0	0	1	.ADS	1	14%
LAUNCH_SUMMARY_COUNTS_CC10	0	0	0	1	0	0	0	1	.CCC	1	14%
LAUNCH_SUMMARY_COUNTS_CC1O	0	0	0	1	0	0	0	1	.CCC	1	14%
M11_DETAILED_MOB_CM15	0	0	0	1	0	0	0	1	.OUT	1	14%
M17_REENTRY_REPORT_CM21	0	0	0	0	0	1	0	1	.OUT	1	14%
M3_MISSILE_ATTACK_SUMMARY_CM08	0	0	0	0	1	0	0	1	.OUT	1	14%
MCIO_MCS_BASIC_UTILITIES	0	0	0	0	0	1	0	1	.ADB	1	14%
MCIO_MCS_MAS_REAL_ACTION_ROUTINES	0	0	0	0	0	1	0	1	.ADB	1	14%
MCIO_PUT_BASIC_UTILITIES_	0	0	0	0	0	1	0	1	.ADS	1	14%
MCIO_PUT_DI_ACTION_ROUTINES	0	0	0	0	0	1	0	1	.ADB	1	14%
MCIO_PUT_GLOBAL_DATA_	0	0	0	0	0	1	0	1	.ADS	1	14%
MCIO_PUT_GLOBAL_TYPES_	0	0	0	0	1	0	0	1	.ADS	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
N2_NUDET_SUMMARY_CM31	0	0	0	0	1	0	0	1	.OUT	1	14%
NAS_INSTALL	0	0	0	0	0	1	0	1	.COM	1	14%
NETWORK_PROCESS_DEFINITION	0	0	0	0	0	1	0	1	.INC	1	14%
NMP_NMP_NUDET_TASK_COMP	0	0	0	1	0	0	0	1	.ADB	1	14%
NMP_NUDET_DATABASE	0	0	0	0	1	0	0	1	.ADB	1	14%
NUT_SCREEN_MANAGEMENT	0	0	0	0	1	0	0	1	.ADB	1	14%
NUT_SCREEN_MANAGEMENT_	0	0	0	0	1	0	0	1	.ADS	1	14%
OMP_TASK_INIT	0	0	0	0	0	1	0	1	.COM	1	14%
OMP_TASR_INIT	0	0	0	0	0	1	0	1	.COM	1	14%
OPERATIONAL_A9_REPUBLIC_SORTED	0	0	0	0	1	0	0	1	.INPUT	1	14%
OPERATIONAL_ALL_INTEGRITY_SORTED	0	0	0	0	1	0	0	1	.INPUT	1	14%
OPERATIONAL_BLUE_LAUNCHES_SCENARIO	0	0	0	0	1	0	0	1	.INPUT	1	14%
OPERATIONAL_STATUS_CC03	0	0	1	0	0	0	0	1	.CCC	1	14%
PIM_VIM_ACTION_E	0	0	1	0	0	0	0	1	.ADB	1	14%
PIM_VIM_ACTION_R	0	0	1	0	0	0	0	1	.ADB	1	14%
PIM_VIM_GET_SENSOR_SOURCE_	0	0	0	0	1	0	0	1	.ADS	1	14%
PIM_VIM_GET_SOURCE	0	0	1	0	0	0	0	1	.ADB	1	14%
PIM_VIM_ITC_UTILITIES	0	0	0	1	0	0	0	1	.ADB	1	14%
PIM_VIM_ITC_UTILITIES_	0	0	0	1	0	0	0	1	.ADS	1	14%
PIM_VIM_QUICK_ALERT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
PIM_VIM_QUICK_LOOK_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
PIM_VIM_RADAR_CAPABILITY_	0	0	0	1	0	0	0	1	.ADS	1	14%
PIM_VIM_SEWS_MS_EVENT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
PIM_VIM_SEWS_R_EVENT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
PMP_FORMAT_IDS_FOR_TIME_VAL	0	0	0	0	1	0	0	1	.DAT	1	14%
PMP_MUT_UNIQUE_VALIDATION	0	0	0	0	0	1	0	1	.ADB	1	14%
PMP_MUT_UNIQUE_VALIDATION_	0	0	0	0	0	1	0	1	.ADS	1	14%
PROCESS_WATCHDOG_LOGICALS	0	0	0	1	0	0	0	1	.COM	1	14%
PSDC$SCHEDULE	0	0	0	0	0	1	0	1	.DAT	1	14%
QPR_BUILD_SQL	0	0	0	0	1	0	0	1	.COM	1	14%
QPR_SENSOR_SQL_SUPPORT_	0	0	0	1	0	0	0	1	.ADS	1	14%
QPR_SQL_CREATE	0	0	0	0	1	0	0	1	.SQLMOD	1	14%
RADAR_LAUNCU_EVENT_CC25	0	0	0	1	0	0	0	1	.CCC	1	14%
RMP_BASIC_TYPES_	0	0	0	0	1	0	0	1	.ADS	1	14%
RMP_RADAR_DATABASE_	0	0	0	0	1	0	0	1	.ADS	1	14%
RMP_THREAT_NONTHREAT	0	0	1	0	0	0	0	1	.ADB	1	14%
RTR_CALCULATE_STORAGE	0	0	0	0	1	0	0	1	.ADB	1	14%
RTR_RECORDING_TYPES_	0	0	0	0	1	0	0	1	.ADS	1	14%
RTR_TERMINATE_RECORDABLE	0	0	0	0	1	0	0	1	.ADB	1	14%
SAS_ETC_LINK	0	0	0	1	0	0	0	1	.CMNLNK	1	14%
SAS_NODES	0	0	0	0	0	1	0	1	.DAT	1	14%
SAS_PROCESSES	0	0	0	1	0	0	0	1	.DAT	1	14%
SCN_ERROR_PROCESSING	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_OMP_ACTION_A	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_OMP_ACTION_B	0	0	0	0	0	1	0	1	.ADB	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
SCN_OMP_ACTION_G	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_OMP_BANNER	0	0	0	1	0	0	0	1	.DAT	1	14%
SCN_OMP_GLOBAL_VARIABLES_	0	0	0	0	0	1	0	1	.ADS	1	14%
SCN_OMP_PRINT_COMMAND	0	0	0	0	1	0	0	1	.COM	1	14%
SCN_OMP_PRINT_MESSAGE_TASR_COMP	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_OMP_PRINT_MESSAGES_TASK_COMP	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_PROCESS_FLOW_CONTROL	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_PROCESS_MCIO_STATUS	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_PROCESS_SYSTEM_MODE	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_PROCESS_TEST_CONTROL	0	0	0	0	0	1	0	1	.ADB	1	14%
SCN_SYSC_DATA	0	0	1	0	0	0	0	1	.ADB	1	14%
SCN_SYSC_READ_CONNECTIVITY_SITES	0	0	1	0	0	0	0	1	.ADB	1	14%
SDG_2XJCS	0	0	0	0	1	0	0	1	.INPUT	1	14%
SDG_BASIC_TYPES	0	0	0	0	1	0	0	1	.ADB	1	14%
SDG_DATABASE_DEFINITIONS_	0	0	0	0	1	0	0	1	.ADS	1	14%
SDG_LORIG_ALGORITHM_LIBRARY	0	0	0	0	1	0	0	1	.ADB	1	14%
SDG_LORIG_ALGORITHM_LIBRARY_	0	0	0	0	1	0	0	1	.ADS	1	14%
SDG_LORIG_DATABASE_DEFINITIONS_	1	0	0	0	0	0	0	1	.ADS	1	14%
SDG_PHYSICAL_CONSTANTS	0	0	0	0	1	0	0	1	.ADB	1	14%
SDG_SAT_SENSOR_CORR	0	0	0	0	1	0	0	1	.DAT	1	14%
SEC	0	0	0	0	0	1	0	1	.COM	1	14%
SEC_ADD_USER	0	0	0	0	1	0	0	1	.COM	1	14%
SEC_IMTERNAL_STRUCTURES_	0	0	0	0	1	0	0	1	.ADS	1	14%
SEC_INSTALL_TEMPLATES	0	0	0	0	1	0	0	1	.COM	1	14%
SECRET_SCREEN	0	0	0	0	1	0	0	1	.DAT	1	14%
SEWS_R_EVENT_CC20	0	0	0	0	0	1	0	1	.CCC	1	14%
SGI_CC_TOTAL_RADAR_OBJECTS_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_CCPDSR_REEP_ALIVE_MESSAGE_	0	0	0	0	0	1	0	1	.ADS	1	14%
SGI_CSO_SYSC_PROCESS_STATUS_MESSAGE_	0	0	0	0	1	0	0	1	.ADS	1	14%
SGI_DISPLAY_TYPES	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_FDB_INVENTORY_MESSAGE_	0	0	0	1	0	0	0	1	.ADS	1	14%
SGI_FDB_MISSILE_LAUNCH_MESSAGE_	0	0	0	1	0	0	0	1	.ADS	1	14%
SGI_FDB_TYPES_	0	0	0	1	0	0	0	1	.ADS	1	14%
SGI_FUNCTION_KEY_INPUT_MESSAGE_	0	1	0	0	0	0	0	1	.ADS	1	14%
SGI_FUSED_DATADASE_	0	0	0	1	0	0	0	1	.ADS	1	14%
SGI_KEEPALIVE_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_MCIO_BASIC_TYPES_	0	0	0	0	0	1	0	1	.ADS	1	14%
SGI_MENU_TYPES_	0	0	0	1	0	0	0	1	.ADS	1	14%
SGI_PDS_I4_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_PDS_M14_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_PDS_M15_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_PDS_M17_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_PDS_M1A_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_PDS_S1_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_PERFORMANCE_LOG_MESSAGE_	0	0	0	0	1	0	0	1	.ADS	1	14%
SGI_PRINT_REQUEST_MESSAGE_	0	0	0	0	1	0	0	1	.ADS	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
SGI_QUICK_ALERT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_QUICK_LOOK_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_RECORDING_TYPES_	0	0	0	0	1	0	0	1	.ADS	1	14%
SGI_RTRD_PRIMARY_SHADOW_COORD_MESSAGE_	0	0	0	0	0	1	0	1	.ADS	1	14%
SGI_SEWS_ML_EVENT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_SEWS_MLU_EVENT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_SEWS_MS_EVENT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_SEWS_R_EVENT_MESSAGE	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_SPRF_TIMED_WRITE_MESSAGE_	0	0	0	0	1	0	0	1	.ADS	1	14%
SGI_SYSC_TYPES_	0	0	0	0	0	1	0	1	.ADS	1	14%
SGI_TAPE_DATABASE_MESSAGE_	0	0	0	1	0	0	0	1	.ADS	1	14%
SGI_TAS_TYPES	0	0	1	0	0	0	0	1	.ADB	1	14%
SGI_TAS_TYPES_	0	0	0	0	1	0	0	1	.ADS	1	14%
SHUTDOWN_VDSI1	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_VDSI2	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_VDSI3	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_WSC06	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_WSC07	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_WSC08	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_WSC09	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_WSC10	0	0	0	0	0	1	0	1	.RCF	1	14%
SHUTDOWN_WSCll	0	0	0	0	0	1	0	1	.RCF	1	14%
SITE_STATUS_SM03	0	0	0	1	0	0	0	1	.INB	1	14%
SPM_INPUT_PARAMETERS	0	0	0	0	1	0	0	1	.DAT	1	14%
SPRF_SPM_PERF_MONITR_TASK_COMP	0	0	0	0	1	0	0	1	.ADB	1	14%
SPRF_SPM_PERFORM_SPM_PERFORMANCE	0	0	0	0	1	0	0	1	.ADB	1	14%
SPRF_SPM_SERVICES	0	0	0	0	1	0	0	1	.ADB	1	14%
SPRF_SPM_SERVICES_	0	0	0	0	1	0	0	1	.ADS	1	14%
SPRF_WSC06_MONITORED_DEVICES	0	0	0	0	0	1	0	1	.DAT	1	14%
SPRF_WSC07_MONITORED_DEVICES	0	0	0	0	0	1	0	1	.DAT	1	14%
SPRF_WSC08_MONITORED_DEVICES	0	0	0	0	0	1	0	1	.DAT	1	14%
SPRF_WSC09_MONITORED_DEVICES	0	0	0	0	0	1	0	1	.DAT	1	14%
SPRF_WSC10_MONITORED_DEVICES	0	0	0	0	0	1	0	1	.DAT	1	14%
SPRF_WSCll_MONITORED_DEVICES	0	0	0	0	0	1	0	1	.DAT	1	14%
SSDCWS	0	0	0	0	1	0	0	1	.INPUT	1	14%
SSP_BASIC_TYPES	1	0	0	0	0	0	0	1	.ADB	1	14%
SSP_TIME_TO_IMPACT	0	0	0	0	1	0	0	1	.ADB	1	14%
STARTUP_ARCHIVE_PROCESS	0	0	0	1	0	0	0	1	.COM	1	14%
STARTUP_VDSI1	0	0	0	0	0	1	0	1	.RCF	1	14%
STARTUP_VDSI2	0	0	0	0	0	1	0	1	.RCF	1	14%
STARTUP_VDSI3	0	0	0	0	0	1	0	1	.RCF	1	14%
STARTUP_WSC06	0	0	0	0	0	1	0	1	.RCF	1	14%
STARTUP_WSC07	0	0	0	0	0	1	0	1	.RCF	1	14%
STARTUP_WSC08	0	0	0	0	0	1	0	1	.RCF	1	14%
STARTUP_WSC09	0	0	0	0	0	1	0	1	.RCF	1	14%
STARTUP_WSC10	0	0	0	0	0	1	0	1	.RCF	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
STARTUP_WSCll	0	0	0	0	0	1	0	1	.RCF	1	14%
T1_SITE_STATUS_REPORT_CM38	0	0	1	0	0	0	0	1	.OUT	1	14%
TAS_DATABASE	0	0	0	1	0	0	0	1	.RDO	1	14%
TAS_THROTTLE	0	0	1	0	0	0	0	1	.DAT	1	14%
TDBA_READ_FIELD_LIST	0	0	0	1	0	0	0	1	.SQLADA	1	14%
TERMINATE_TEST_EXERCISE	0	0	0	0	0	1	0	1	.FCF	1	14%
TESTCTL_INIT	0	0	1	0	0	0	0	1	.COM	1	14%
THIRTY1	0	0	0	0	1	0	0	1	.INPUT	1	14%
TIFP_TIFP_INTERFACE_TASK_COMP	0	0	0	0	0	1	0	1	.ADB	1	14%
TMBL_BUILD_INJECTION_MESSAGE	0	0	0	1	0	0	0	1	.ADB	1	14%
TMBL_MC_CONVERT_MESSAGE	0	0	0	1	0	0	0	1	.ADB	1	14%
TOTAL_RADAR_OBJECTS_CC05	0	0	1	0	0	0	0	1	.CCC	1	14%
USER-WS03_SUITE_O1	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_ALARM_ASSIGNMENT	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_ALARM_ASSIGNMENT_SUMMARY_NCC	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_ANTARCTIC_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_ASSESSMENT_AND_VALIDATION_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_CINCNORAD_WARNING_SUMMARY	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_CINCNORAD_WARNING_SUMMARY_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_CMAFB_SYSTEM_STATUS_GRAPHIC	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_CONTROL_	0	0	0	1	0	0	0	1	.ADS	1	14%
USER_CONUS_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_CONUS_LEFT	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_CORRELATED_CONN_STATUS	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_CORRELATED_CONN_STATUS_PREFORMAT	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_CORRELATED_CONN_STATUS_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_CORRELATED_SCIS_STATUS_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_DEFCON_LERTCON_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_DETAILED_ICADS_LIST_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DETAILED_MOB	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DETAILED_MOB_PREFORMAT	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DETAILED_MOB_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DETAILED_NUDET_LIST_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DETAILED_RADAR_EVENTS	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DIRECT_CS2_STATUS_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_DIRECT_CS3_STATUS_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_DIRECT_DDC_STATUS_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_DIRECT_DSP_SITE_STATUS	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_DIRECT_DSP_SITE_STATUS_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_DIRECT_LAUNCH_SUMMARY_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DIRECT_NUDET_SUMMARY_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DIRECT_RADAR_IMPACT_SUMM_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DIRECT_RADAR_LAUNCH_SUMM_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_DIRECT_RADAR_SITE_STATUS_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_EUROPE_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_EUROPE_ICADS_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
USER_EUROPE_NUDET_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_ICADS_ASSESSMENT_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_ICADS_SITUATION_RELATED_	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_ICADS_SITUATION_THEATER_MAP_2698	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_IR_LAUNCH_COMPOSITE_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_IR_LAUNCH_LISTING_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_IR_SIIUATION_RELATED	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_IR_SITUATION_MAP_OVERLAYS	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_IR_SITUATION_THEATER_MAP_OVERLAYS	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_KOREA_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_KOREA_ICADS_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_KOREA_NUDET_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_MIDEAST_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_MIDEAST_ICADS_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_MIDEAST_NUDET_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_MINUTES_TO_REPORT_FOED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_MINUTES_TO_REPORT_FOED_MAIN_MENU	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_MINUTES_TO_REPORT_FOED_PREFORMAT	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_MISSILE_ATTACK_SUMMARY	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_MISSILE_WARNING_SUMMARY	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_MISSILE_WARNING_SUMMARY_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_MOB_SUMMARY	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_MOB_SUMMARY_PREFORMAT	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_MOB_SUMMARY_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_MTF_DB	0	0	1	0	0	0	0	1	.DAT	1	14%
USER_NA_ICADS_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_NA_NUDET_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_NONINTERACTIVE	0	0	1	0	0	0	0	1	.ADB	1	14%
USER_NORAD_IR_LAUNCH_COMPOSITE_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_NORAD_PRELIM_RADAR_COMPOSITE_B8E	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_NORTH_AMERICA_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_NORTH_AMERICAN_WARNING_PREFORMAT	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_NORTH_AMERICAN_WARNING_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_NORTH_AMERICAN_WARNING_SUMMARY	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_NORTH_POLE_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_NUDET_SITUATION_THEATER_MAP_A889	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPCC_SYSTEM_STATUS_GRAPHIC	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_01	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_02	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_03	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_04	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_05	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_06	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_07	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_08	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_09	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_OPERATOR_FORMATTED_MAP_10	0	0	0	0	1	0	0	1	.RAW	1	14%

Module	**94-2**	**95-1**	**95-2**	**96-1**	**96-2**	**97-1**	**AOC 2A**	**Grand Total**	**Module Type**	**Release Count**	**Release Percent**
USER_PACIFIC_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_PACIFIC_ICADS_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_PACIFIC_NUDET_SITUATION_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_PERIODS_OF_INTEREST_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_PRELIM_RADAR_COMPOSITE_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_PREVIEW_MAP_01	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_PREVIEW_MAP_02	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_RADAR_SITUATION_MAP_OVERLAYS	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_RADAR_SITUATION_RELATED	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_SCENARIO_EXECUTION_CONTROL_ID	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_SENSOR_DETECTIONS_LEFT	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_SENSOR_DETECTIONS_RIGHT	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_SENSOR_MAINTENANCE_RELATED	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_SHIP_SUB_LOCATIONS_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_SINO_SOVIET_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_SINO_SOVIET_IR_SITUATION_OVERLAYS	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_SINO_SOVIET_IR_SITUATION_PREFORMAT	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_SO_ATLANTIC_ICADS_SIT_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_SO_ATLANTIC_NUDET_SIT_RELATEED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_SOUTH_ATLANTIC_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_STRATEGIC_MOB_SUMMARY_PREFORMAT	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_STRATEGIC_MOB_SUMMARY_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_STRATEGIC_SUMMARY_PREFORMAT	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_STRATEGIC_SUMMARY_RELATED	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_SUMMARY_DAT_MENU	0	0	0	1	0	0	0	1	.RAW	1	14%
USER_SYSTEM_MODE_CONTROL	0	0	0	1	0	0	0	1	.ADB	1	14%
USER_SYSTEM_STATUS_MENU	0	0	1	0	0	0	0	1	.RAW	1	14%
USER_WORLD_CENTER	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS02_SUITE_01	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS02_SUITE_03	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS02_SUITE_04	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS02_SUITE_05	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS02_SUITE_06	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS03_SUITE_03	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS03_SUITE_04	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS03_SUITE_05	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS04_SUITE_02	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS04_SUITE_03	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS04_SUITE_04	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS04_SUITE_05	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS04_SUITE_06	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS04_SUITE_O1	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WS05_SUITE_02	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS05_SUITE_03	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS05_SUITE_04	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS05_SUITE_05	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS05_SUITE_06	0	0	0	0	0	1	0	1	.RAW	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
USER_WS05_SUITE_51	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS06_SUITE_01	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS06_SUITE_02	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS06_SUITE_03	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS06_SUITE_04	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS06_SUITE_05	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS06_SUITE_06	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS07_SUITE_01	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS07_SUITE_02	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS07_SUITE_03	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS07_SUITE_06	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS08_SUITE_01	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS08_SUITE_02	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS08_SUITE_03	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS08_SUITE_04	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS08_SUITE_05	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WS08_SUITE_06	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSC_AMWC_SECURITY_CLASS1_AREA	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSC_AMWC_SECURITY_CLASS2_AREA	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO1_SUITE_02	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WSO1_SUITE_03	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WSO1_SUITE_04	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WSO1_SUITE_05	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WSO1_SUITE_06	0	0	0	0	1	0	0	1	.RAW	1	14%
USER_WSO10_SUITE_01	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO10_SUITE_02	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO10_SUITE_03	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO10_SUITE_04	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO10_SUITE_05	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO10_SUITE_06	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO7_SUITE_04	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO7_SUITE_05	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO9_SUITE_01	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO9_SUITE_02	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO9_SUITE_03	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO9_SUITE_04	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO9_SUITE_05	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSO9_SUITE_06	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSOll_SUITE_01	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSOll_SUITE_02	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSOll_SUITE_03	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSOll_SUITE_04	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSOll_SUITE_05	0	0	0	0	0	1	0	1	.RAW	1	14%
USER_WSOll_SUITE_06	0	0	0	0	0	1	0	1	.RAW	1	14%
USI_CREATE_MENU_DISPLAY	0	0	0	0	1	0	0	1	.ADB	1	14%
USI_MENU_DEFINITION	0	0	0	0	1	0	0	1	.ADB	1	14%
USI_MENU_PROCESSING	0	0	0	0	1	0	0	1	.ADB	1	14%

Module	94-2	95-1	95-2	96-1	96-2	97-1	AOC 2A	Grand Total	Module Type	Release Count	Release Percent
USI_PROCEDURES_	0	0	0	0	1	0	0	1	.ADS	1	14%
VIM_INPUT_FILE	0	0	0	0	1	0	0	1	.DAT	1	14%
VMSP_COMMAND	0	0	0	0	1	0	0	1	.COM	1	14%
VMSP_LOGIN	0	0	0	0	1	0	0	1	.COM	1	14%
VRP_TWAA_SYSUAF_QUOTAS	0	0	0	0	1	0	0	1	.DAT	1	14%
VRP_WSC_SYSUAF_QUOTAS	0	0	0	0	1	0	0	1	.DAT	1	14%
VUE$MASTER	0	0	0	0	1	0	0	1	.DAT	1	14%
VUE$PROFILE.VUE$	0	0	0	0	1	0	0	1	.DAT	1	14%

b. CCPDS-R Change Driver Tabulation

Change Object	95-2	96-1	96-2	97-1	Grand Total	Release Count	Release Percent	Exposure	Exposure Percent
display	7	6	14	1	28	4	100%	112	37%
message	1	2	20	3	26	4	100%	104	35%
menu	0	9	1	0	10	2	50%	20	7%
alert	0	2	4	0	6	2	50%	12	4%
rules	1	0	4	0	5	2	50%	10	3%
data	1	0	0	2	3	2	50%	6	2%
time	1	0	1	0	2	2	50%	4	1%
report	0	0	4	0	4	1	25%	4	1%
command	0	3	0	0	3	1	25%	3	1%
COTS	0	0	3	0	3	1	25%	3	1%
status	0	0	3	0	3	1	25%	3	1%
timer	0	0	3	0	3	1	25%	3	1%
alarm	2	0	0	0	2	1	25%	2	1%
satellite	0	0	2	0	2	1	25%	2	1%
count	1	0	0	0	1	1	25%	1	0%
error	1	0	0	0	1	1	25%	1	0%
failover	0	1	0	0	1	1	25%	1	0%
form	0	0	0	1	1	1	25%	1	0%
keyboard	1	0	0	0	1	1	25%	1	0%
message field	0	1	0	0	1	1	25%	1	0%
message filter	0	1	0	0	1	1	25%	1	0%
scenario	1	0	0	0	1	1	25%	1	0%
summary	0	0	1	0	1	1	25%	1	0%
table	0	0	1	0	1	1	25%	1	0%
thresholding	1	0	0	0	1	1	25%	1	0%
track	0	0	0	1	1	1	25%	1	0%
Grand Total	18	25	61	8	112				

Appendix 3

Volatility Characterization - Granite Sentry

a. Granite Sentry Module Tabulation

Module	95-2	96-1	96-2	Grand Total	Module Type	Release Count	Release Percent
AIR_PUSH_CALLBACKS	3	1	0	4	.ADB	2	67%
MESSAGE_C167	2	2	0	4	.ADB	2	67%
MESSAGE_C17R	2	2	0	4	.FRM	2	67%
C170_TRACK_REPORT	1	2	0	3	.ADB	2	67%
FORM_ECTAR_REPORT	2	1	0	3	.UIL	2	67%
MESSAGE_K169	1	2	0	3	.FRM	2	67%
AIR_LIB	1	1	0	2	.ADB	2	67%
AUTOMATIC_MESSAGE_GENERATION	1	1	0	2	.ADB	2	67%
FIGHTER_STATUS_PKG	1	1	0	2	.ADB	2	67%
FORM_ECTAR_REPORT_PKG	1	1	0	2	.ADB	2	67%
FORMAT_N1X	1	1	0	2	.ADB	2	67%
IR_MISSILE_KIND_TC	1	0	1	2	ADB	2	67%
IR_MISSILE_KIND_TC_	1	0	1	2	.ADS	2	67%
MESSAGE_C169	1	1	0	2	.FRM	2	67%
MESSAGE_CALLS	1	1	0	2	.ADB	2	67%
N016_TRACK_REPORT	1	1	0	2	.ADB	2	67%
OLD_USER_PROFILE_DATA_STRUC_	1	1	0	2	.ADS	2	67%
OPLAN_DB_TRANSACTIONS	1	1	0	2	.ADB	2	67%
USER_PROFILE_DEFAULTS	1	1	0	2	.DAT	2	67%
WD_IR_LAUNCH_EST	1	0	1	2	.ADB	2	67%
WD_IR_LAUNCH_EST_	1	0	1	2	.ADS	2	67%
AIR_BUILD_ROCC_SOCC_SUMMARY_PULL_RIGHT	3	0	0	3	.ADB	1	33%
CHECK_AIR_DIALOG_B0XES	3	0	0	3	.ADB	1	33%
COMMON_RESET_PKG	3	0	0	3	.ADB	1	33%
GS_MSG_COMMON_TYPES_	0	3	0	3	.ADS	1	33%
GSW_AIR_MENU_TYPES_	3	0	0	3	.ADS	1	33%
GSW_AIR_WIDGETS_	3	0	0	3	.ADS	1	33%
MESSAGE_N016	0	3	0	3	.ADB	1	33%
AIR_BUILD_REQUEST_TABLES_MENU	2	0	0	2	.ADB	1	33%
AIR_BUILD_ROTHR_M_SUMMARY_DIALOG_BOXES	2	0	0	2	.ADB	1	33%
AIR_BUILD_ROTHR_T_SUMMARY_DIALOG_BOXES	2	0	0	2	.ADB	1	33%
AIR_BUILD_ROTHR_V_SUMMARY_DIALOG_BOXES	2	0	0	2	.ADB	1	33%
AIR_PUSH_CALLBACKS_	2	0	0	2	.ADS	1	33%
AIRBASE_LOOKUP_TABLE_PKG_	2	0	0	2	.ADS	1	33%
BUILD_OTHB_TRACK_MESSAGE	0	2	0	2	.ADB	1	33%
BUILD_REGION_SECTOR_TRACK_SUMMARY_TABLE	2	0	0	2	.ADB	1	33%
BUILD_TRACK_MESSAGE	0	2	0	2	.ADB	1	33%
CREATE_RADAR_OUTLINES	2	0	0	2	.ADB	1	33%
ELEMENT_TAB_PKG	0	2	0	2	.ADB	1	33%
FORM_AIRBASE_LIST	2	0	0	2	.UIL	1	33%
FORM_COMMAND_CONTROL_ID	2	0	0	2	.UIL	1	33%
FORM_RADAR_SITE_STATUS	2	0	0	2	.UIL	1	33%
GEN_CHANNEL_STATUS_UPDATE	0	2	0	2	.ADB	1	33%

Module	95-2	96-1	96-2	Grand Total	Module Type	Release Count	Release Percent
GET_CHANGE_REQUEST	0	2	0	2	.ADB	1	33%
GET_INIT_REQUEST	0	2	0	2	.ADB	1	33%
GSW_DISPLAYS_	2	0	0	2	.ADS	1	33%
GSW_SET_AIR_REQ_TABLE_DEFAULTS	2	0	0	2	.ADB	1	33%
MESSAGE_C170	0	2	0	2	.ADB	1	33%
PRINT_REGION_SECTOR_TRACK_SUMMARY	2	0	0	2	.ADB	1	33%
PROJECTION_CONSTANTS_	2	0	0	2	.ADS	1	33%
ROTHR_MONTANA_AIR_TOGGLE_CALLBACKS	2	0	0	2	.ADB	1	33%
ROTHR_MONTANA_AIR_TOGGLE_CALLBACKS_	2	0	0	2	.ADS	1	33%
ROTHR_TEXAS_AIR_TOGGLE_CALLBACKS	2	0	0	2	.ADB	1	33%
ROTHR_TEXAS_AIR_TOGGLE_CALLBACKS_	2	0	0	2	.ADS	1	33%
ROTHR_VIRGINIA_AIR_TOGGLE_CALLBACKS	2	0	0	2	.ADB	1	33%
ROTHR_VIRGINIA_AIR_TOGGLE_CALLBACKS_	2	0	0	2	.ADS	1	33%
SOCC_BOUNDARIES	2	0	0	2	.GEO	1	33%
TRACK_SUMMARY_PKG	2	0	0	2	.ADB	1	33%
WD_POLYGON_BOUNDARY	2	0	0	2	.ADB	1	33%
ADIZ	1	0	0	1	.GEO	1	33%
AIR_DEFENSE_DB_DEFINITION	0	1	0	1	.ADB	1	33%
AIR_MAP_PKG	1	0	0	1	.ADB	1	33%
AIR_MAP_PKG_	1	0	0	1	.ADS	1	33%
AIRBASE_FIGHTER_STATUS_PKG	0	1	0	1	.ADB	1	33%
AIRBASE_PKG	1	0	0	1	.ADB	1	33%
AIRCRAFT_MOVEMENT_PKG	0	1	0	1	.ADB	1	33%
ALARM_DEFINITIONS_	0	1	0	1	.ADS	1	33%
ALARM_DISPLAY_CONTROL_PKG	0	1	0	1	.ADB	1	33%
AM_EMR_SEC_SCREEN	1	0	0	1	.FRM	1	33%
AWS_GRAPHIC_PKG_	0	1	0	1	.ADS	1	33%
AWS_HEADER_PKG	1	0	0	1	.ADB	1	33%
BEE_SUBSURFACE	0	1	0	1	.GKSM	1	33%
BEE_SURFACE	0	1	0	1	.GKSM	1	33%
BLUE_SUB_SHIP_ECM	0	1	0	1	.GKSM	1	33%
BLUE_SUB_SHIP_SPAWNING	0	1	0	1	.GKSM	1	33%
BUILD_DEFCON_CHANGE_TABLE	1	0	0	1	.ADB	1	33%
BUILD_EMR_REPORT	1	0	0	1	.ADB	1	33%
BUILD_FIGHTER_STATUS	1	0	0	1	.ADB	1	33%
BUILD_HEADER_SEG_2	1	0	0	1	.ADB	1	33%
BUILD_ICON_DISPLAY	0	1	0	1	.ADB	1	33%
BUILD_MTR_REPORT	0	1	0	1	.ADB	1	33%
BUILD_NORAD_ROCC_SOCC_STATUS_TABLE	1	0	0	1	.ADB	1	33%
BUILD_SUB_SHIP_ICONS	0	1	0	1	.ADB	1	33%
BUILD_TAB_MESSAGE	1	0	0	1	.ADB	1	33%
C001_WEATHER_REPORT	1	0	0	1	.ADB	1	33%
C167_INTERCEPTOR_STATUS_REPORT	0	1	0	1	.ADB	1	33%
C169_ECTAR_REPORT	1	0	0	1	.ADB	1	33%
C171_RADAR_SITE_STATUS	1	0	0	1	.ADB	1	33%
C172_E3A_STATUS_REPORT	1	0	0	1	.ADB	1	33%

Module	95-2	96-1	96-2	Grand Total	Module Type	Release Count	Release Percent
CANADA_BOUNDS	1	0	0	1	.GKSM	1	33%
CANADA_EAST_BOUNDS	1	0	0	1	.GKSM	1	33%
CHANGE_DATA_PUSH_CB	0	1	0	1	.ADB	1	33%
CHARACTER_UTILITIES_PKG	1	0	0	1	.ADB	1	33%
CHARACTER_UTILITIES_PKG_	1	0	0	1	.ADS	1	33%
CHECK_ICON_TEXT	1	0	0	1	.ADB	1	33%
CLIPPING_PKG	1	0	0	1	.ADB	1	33%
CLIPPING_PKG_	1	0	0	1	.ADS	1	33%
CONUS_BOUNDS	1	0	0	1	.GKSM	1	33%
CONV_RADAR	0	1	0	1	.ADB	1	33%
CONVERT_TRACK_DATA_TO_STRING	0	1	0	1	.ADB	1	33%
CREATE_EXERCISE_HEADER_2	1	0	0	1	.ADB	1	33%
CREATE_HEADER_2	1	0	0	1	.ADB	1	33%
CREATE_MAP_BACKGROUNDS	1	0	0	1	.ADB	1	33%
CREATE_SUB_SHIP_ECM_ICONS	0	1	0	1	.ADB	1	33%
CREATE_SUB_SHIP_SPAWNING_ICONS	0	1	0	1	.ADB	1	33%
CREATE_SUB_SHIP_SPLASHED_ICONS	0	1	0	1	.ADB	1	33%
CREATE_SUBSURFACE_ICONS	0	1	0	1	.ADB	1	33%
CREATE_SURFACE_ICONS	0	1	0	1	.ADB	1	33%
DATA_REDUCTION_COMMON_TYPES_	1	0	0	1	.ADS	1	33%
DEAD_RECKON	0	1	0	1	.ADB	1	33%
DEFCON_CHANGE_PKG	1	0	0	1	.ADB	1	33%
DEFCON_CHANGE_PKG_	1	0	0	1	.ADS	1	33%
DELETE_ITEM	1	0	0	1	.ADB	1	33%
DISPLAY_LIST_PKG_	1	0	0	1	.ADS	1	33%
E_3_RP_STATUS_PKG	1	0	0	1	.ADB	1	33%
E_3_RP_STATUS_PKG_	1	0	0	1	.ADS	1	33%
ELEMENT_TAB_PKG_	0	1	0	1	.ADS	1	33%
EXERCISE_HEADER_2	1	0	0	1	.GKSM	1	33%
FAKELT_SURFACE	0	1	0	1	.GKSM	1	33%
FAKER_SUBSURFACE	0	1	0	1	.GKSM	1	33%
FORM_AIRBASE_FIGHTER_OVERALL_PKG_	1	0	0	1	.ADS	1	33%
FORM_AIRBASE_FIGHTER_TYPE	0	1	0	1	.UIL	1	33%
FORM_AIRBASE_FIGHTER_TYPE_PKG_	1	0	0	1	.ADS	1	33%
FORM_AIRBASE_SELECTION_1	1	0	0	1	.UIL	1	33%
FORM_AIRBASE_SELECTION_2	1	0	0	1	.UIL	1	33%
FORM_AIRBASE_SELECTION_3	1	0	0	1	.UIL	1	33%
FORM_AIRBASE_WEATHER_PKG_	1	0	0	1	.ADS	1	33%
FORM_AWACS_STATUS	1	0	0	1	.UIL	1	33%
FORM_AWACS_STATUS_PKG	1	0	0	1	.ADB	1	33%
FORM_AWACS_STATUS_PKG_	1	0	0	1	.ADS	1	33%
FORM_COMMAND_CONTROL_ID_PKG	1	0	0	1	.ADB	1	33%
FORM_COMMAND_CONTROL_ID_PKG_	1	0	0	1	.ADS	1	33%
FORM_ECSUM_REPORT	1	0	0	1	.UIL	1	33%
FORM_ECSUM_REPORT_PKG	1	0	0	1	.ADB	1	33%
FORM_ECSUM_REPORT_PKG_	1	0	0	1	.ADS	1	33%

Module	95-2	96-1	96-2	Grand Total	Module Type	Release Count	Release Percent
FORM_ECTAR_REPORT_PKG_	1	0	0	1	.ADS	1	33%
FORM_ECTAR_REPORT_RESET_PROC	1	0	0	1	.ADB	1	33%
FORM_FUNCTION_STATUS	1	0	0	1	.UIL	1	33%
FORM_FUNCTION_STATUS_PKG	1	0	0	1	.ADB	1	33%
FORM_FUNCTION_STATUS_PKG_	1	0	0	1	.ADS	1	33%
FORM_RADAR_SITE_STATUS_PKG	1	0	0	1	.ADB	1	33%
FORM_RADAR_SITE_STATUS_PKG_	1	0	0	1	.ADS	1	33%
FORM_REGION_FIGHTER_OVERALL	1	0	0	1	.UIL	1	33%
FORM_REGION_FIGHTER_OVERALL_PKG	1	0	0	1	.ADB	1	33%
FORM_REGION_FIGHTER_OVERALL_PKG_	1	0	0	1	.ADS	1	33%
FORM_REGION_SECTOR_ACE_C2_ID	1	0	0	1	.UIL	1	33%
FORM_REGION_SECTOR_ACE_C2_ID_PKG	1	0	0	1	.ADB	1	33%
FORM_REGION_SECTOR_RADAR	1	0	0	1	.UIL	1	33%
FORM_REGION_SECTOR_RADAR_PKG	1	0	0	1	.ADB	1	33%
FORM_REGION_SECTOR_RADAR_PKG_	1	0	0	1	.ADS	1	33%
FORM_REGION_SECTOR_STATUS	1	0	0	1	.UIL	1	33%
FORM_REGION_SECTOR_STATUS_PKG	1	0	0	1	.ADB	1	33%
FORM_REGION_SECTOR_STATUS_PKG_	1	0	0	1	.ADS	1	33%
FORM_REGION_SELECTION_1	1	0	0	1	.UIL	1	33%
FORM_REGION_SELECTION_2	1	0	0	1	.UIL	1	33%
FORM_REGION_SELECTION_3	1	0	0	1	.UIL	1	33%
FORM_ROCC_SOCC_EMER_ACTIONS	1	0	0	1	.UIL	1	33%
FORM_SCRAMBLE_ORDER_PKG_	1	0	0	1	.ADS	1	33%
FORM_STATUS_REPORT_E3A	1	0	0	1	.UIL	1	33%
FORM_STATUS_REPORT_E3A_PKG	1	0	0	1	.ADB	1	33%
FORM_TRACK_REPORT	0	1	0	1	.UIL	1	33%
FORM_TRACK_REPORT_PKG	0	1	0	1	.ADB	1	33%
FORM_TRACK_REPORT_PKG_	0	1	0	1	.ADS	1	33%
FORMAT_MESSAGE_DATA	1	0	0	1	.ADB	1	33%
FRIENDLY_SUBSURFACE	0	1	0	1	.GKSM	1	33%
FRIENDLY_SURFACE	0	1	0	1	.GKSM	1	33%
GET_A_C170	0	1	0	1	.ADB	1	33%
GKS_SETUP	0	1	0	1	.ADB	1	33%
GKS_STOP	0	1	0	1	.ADB	1	33%
GREEN_SUB_SHIP_ECM	0	1	0	1	.GKSM	1	33%
GREEN_SUB_SHIP_SPAWNING	0	1	0	1	.GKSM	1	33%
GS_ALARM_PKG_	0	1	0	1	.ADS	1	33%
GS_CSSO	1	0	0	1	.UIL	1	33%
GS_CSSO_PKG_	1	0	0	1	.ADS	1	33%
GS_MSG_FMTS_	0	1	0	1	.ADS	1	33%
GSW_EVENT_HANDLER	0	1	0	1	.ADB	1	33%
HEADER_1	1	0	0	1	.GKSM	1	33%
HEADER_2	1	0	0	1	.GKSM	1	33%
HOSTILE_SUBSURFACE	0	1	0	1	.GKSM	1	33%
HOSTILE_SURFACE	0	1	0	1	.GKSM	1	33%
ICON	0	1	0	1	.GKSM	1	33%

Module	**95-2**	**96-1**	**96-2**	**Grand Total**	**Module Type**	**Release Count**	**Release Percent**
INITIALIZE_GLOBAL	1	0	0	1	.ADB	1	33%
INTELLIGENCE_SUBSURFACE	0	1	0	1	.GKSM	1	33%
INTELLIGENCE_SURFACE	0	1	0	1	.GKSM	1	33%
IR_LAUNCH_AREA_TC_	1	0	0	1	.ADS	1	33%
LAT_LONS	1	0	0	1	.GEO	1	33%
LOAD_DISPLAY_LIST	1	0	0	1	.ADB	1	33%
LOAD_ECM_ICONS	0	1	0	1	.ADB	1	33%
LOAD_SPAWNING_ICONS	0	1	0	1	.ADB	1	33%
LOAD_SYMBOLS	0	1	0	1	.ADB	1	33%
ME55AGE_C169	0	1	0	1	.FRM	1	33%
MESSAGE_C162	1	0	0	1	.FRM	1	33%
MESSAGE_C171	1	0	0	1	.FRM	1	33%
MESSAGE_K162	1	0	0	1	.FRM	1	33%
MESSAGE_N008	1	0	0	1	.FRM	1	33%
MESSAGE_N030	0	1	0	1	.FRM	1	33%
MESSAGE_TO_TEXT_CONVERSIONS_	1	0	0	1	.ADS	1	33%
MICROWAVE_SENSORS	0	1	0	1	.DAT	1	33%
MOVE_A_TRACK	0	1	0	1	.ADB	1	33%
N014_AWACS_STATUS	0	1	0	1	.ADB	1	33%
NORAD_ROCC_SOCC__STATUS__PKG_	1	0	0	1	.ADS	1	33%
NORTH_AMERICA_BOUNDS	1	0	0	1	.GKSM	1	33%
NORTH_EAST	1	0	0	1	.GKSM	1	33%
NORTH_EAST_ADIZ	1	0	0	1	.GKSM	1	33%
NORTH_EAST_BOUNDS	1	0	0	1	.GKSM	1	33%
NORTH_WEST_BOUNDS	1	0	0	1	.GKSM	1	33%
OPLAN_DB_TRANSACTIONS_	0	1	0	1	.ADS	1	33%
OTH_EAST_1_BOUNDS	1	0	0	1	.GKSM	1	33%
OTH_EAST_2_BOUNDS	1	0	0	1	.GKSM	1	33%
OTH_EAST_3_BOUNDS	1	0	0	1	.GKSM	1	33%
OTH_WEST_4_BOUNDS	1	0	0	1	.GKSM	1	33%
OTH_WEST_5_BOUNDS	1	0	0	1	.GKSM	1	33%
OTH_WEST_6_BOUNDS	1	0	0	1	.GKSM	1	33%
PENDING_SUBSURFACE	0	1	0	1	.GKSM	1	33%
PENDING_SURFACE	0	1	0	1	.GKSM	1	33%
POLAR_PROJECTION_BOUNDS	1	0	0	1	.GKSM	1	33%
PRINT_DEFCON_CHANGE	1	0	0	1	.ADB	1	33%
PRINT_MESSAGE_REPORT	0	1	0	1	.ADB	1	33%
PRINT_NORAD_ROCC_SOCC_STATUS	1	0	0	1	.ADB	1	33%
PROCESS_ROCC_SOCC	1	0	0	1	.ADB	1	33%
PROCESS_SYSTEM_STATUS_ALARM	0	1	0	1	.ADB	1	33%
PROCESS_TRACK_MESSAGE	0	1	0	1	.ADB	1	33%
PUSH_CALLBACKS	0	1	0	1	.ADB	1	33%
RADAR_OUTLINES	1	0	0	1	.GKSM	1	33%
RADAR_PKG	1	0	0	1	.ADB	1	33%
RADARS	0	1	0	1	.IN	1	33%
RED_SUB_SHIP_ECM	0	1	0	1	.GKSM	1	33%

Module	95-2	96-1	96-2	Grand Total	Module Type	Release Count	Release Percent
RED_SUB_SHIP_SPAWNING	0	1	0	1	.GKSM	1	33%
ROCC_SOCC_PKG	1	0	0	1	.ADB	1	33%
SHOW_A_SHIP_SUB	0	1	0	1	.ADB	1	33%
SHOW_DISPLAY	1	0	0	1	.ADB	1	33%
SHOW_ROCC_SOCC	1	0	0	1	.ADB	1	33%
SOUTH_EAST	1	0	0	1	.GKSM	1	33%
SOUTH_EAST_ADIZ	1	0	0	1	.GKSM	1	33%
SOUTH_EAST_BOUNDS	1	0	0	1	.GKSM	1	33%
SOUTH_WEST_BOUNDS	1	0	0	1	.GKSM	1	33%
SPECIAL_SUBSURFACE	0	1	0	1	.GKSM	1	33%
SPECIAL_SURFACE	0	1	0	1	.GKSM	1	33%
STRING_UTILITIES_PKG	1	0	0	1	.ADB	1	33%
STRING_UTILITIES_PKG_	1	0	0	1	.ADS	1	33%
SUBSURFACE_SPLASHED	0	1	0	1	.GKSM	1	33%
SURFACE_SPLASHED	0	1	0	1	.GKSM	1	33%
TDA_ARRAY_SPEC_	0	1	0	1	.ADS	1	33%
TDA_LOOKUP_PKG	0	1	0	1	.ADB	1	33%
TRACK_CHANGE	0	1	0	1	.FRM	1	33%
TRACK_INITIALIZATION	0	1	0	1	.FRM	1	33%
TRACK_STRING_CONV_PKG	1	0	0	1	.ADB	1	33%
TRACR_PKG	0	1	0	1	.ADB	1	33%
UNKNOWN_SUBSURFACE	0	1	0	1	.GKSM	1	33%
UNKNOWN_SURFACE	0	1	0	1	.GKSM	1	33%
USER_PROFILE_DATA_STRUC_	1	0	0	1	.ADS	1	33%
WD_TARGETS	1	0	0	1	.ADB	1	33%
WESTERN_HEMISPHERE_BOUNDS	1	0	0	1	.GKSM	1	33%
WHITE_SUB_SHIP_ECM	0	1	0	1	.GKSM	1	33%
WHITE_SUB_SHIP_SPAWNING	0	1	0	1	.GKSM	1	33%
WORLD_05	1	0	0	1	.GEO	1	33%
WORLD_BOUNDS	1	0	0	1	.GKSM	1	33%
YELLOW_SUB_SHIP_ECM	0	1	0	1	.GKSM	1	33%
YELLOW_SUB_SHIP_SPAWNING	0	1	0	1	.GKSM	1	33%

b. Granite Sentry Change Driver Tabulation

Change Object	95-2	96-1	96-2	Grand Total	Release Count	Release Percent	Exposure	Exposure Percent
display	3	3	1	7	3	100%	21	64%
DIM	1	0	1	2	2	67%	4	12%
message	0	3	0	3	1	33%	3	9%
alarm	0	2	0	2	1	33%	2	6%
site	0	1	0	1	1	33%	1	3%
threat	1	0	0	1	1	33%	1	3%
weapon	0	1	0	1	1	33%	1	3%

Change Object	95-2	96-1	96-2	Grand Total	Release Count	Release Percent	Exposure	Exposure Percent
Grand Total	5	10	2	17				

Bibliography

[1] _______, “COCOMO 2.0 Model User's Manual,” University of Southern California 1996.

[2] _______, “Domain Engineering Guidebook,” Space and Warning Systems Center, <http://www.asset.com/stars/loral/domain/guide/home.html>, 22 June 1995.

[3] _______, “REVIC User's Manual,” U.S. Air Force Cost Center, 1991.

[4] AFOTEC, *Software Maintainability - Evaluation Guide*, vol. 3. Kirtland AFB, NM: HQ Air Force Operational Test and Evaluation Center, 1991.

[5] L. Bækgaard, “Designing Adaptable Software - Parameterization of Volatile Properties,” presented at Conference on Software Maintenance, San Diego, CA, 1990.

[6] L. A. Belady and M. M. Lehman, “A model of large program development,” *IBM Systems Journal*, vol. 15, no. 3, pp. 225-252, 1976.

[7] K. Bennett, “Re: Software Volatility,” Personal correspondence, 1996.

[8] B. W. Boehm, *Software Engineering Economics*: Prentice Hall, 1981.

[9] B. W. Boehm, “Software Risk Management: Principles and Practices,” *IEEE Software*, vol. 8, no. 1, pp. 32-41, 1991.

[10] T. P. Bowen, G. B. Wigle, and J. T. Tsal, “Specification of Software Quality Attributes - Software Quality Specification Guidebook, Vol. I-III.,” Rome Air Development Center, Griffiss AFB, NY, February 1985.

[11] E. Dean, “Parametric Cost Analysis,” <http://akao.larc.nasa.gov/dfc/pca.html>, April 1995.

[12] S. Dekleva and N. Zvegintzov, “Real maintenance statistics,” in *Software Maintenance News*, vol. 9, no. 2, 1991, pp. 6-9.

[13] D. Ferens, “Review of Software Cost Estimation,” in *Software Methodology Handbook*, G. Novak-Ley and S. Stukes, Eds.: Space Systems Cost Analysis Group, Software Subgroup, 1995, pp. 2-1 - 2-54.

[14] J. L. Floyd and P. C. Gould, “Software Volatility Analysis - A Historical Approach to Future Software Maintenance,” presented at The Fifth Annual Software Technology Conference, Salt Lake City, 1993.

[15] J. Goguen, “Parameterized Programming,” Center for the Study of Language and Information, Stanford, CA CSLI-84-10, Aug. 1984.

[16] J. A. Hager, “Developing maintainable systems: A full life-cycle approach,” presented at Conference on Software Maintenance, 1989.

[17] J. A. Hager, “Software Cost Reduction Methods in Practice: A Post-Mortem Analysis.,” *The Journal of Systems and Software*, vol. 14, no. 2, pp. 67-79, 1991.

[18] B. Holchin, “Software Maintenance Survey,” in *Software Methodology Handbook*, G. Novak-Ley and S. Stukes, Eds.: Space Systems Cost Analysis Group, Software Subgroup, 1995, pp. 5-1 - 5-13.

[19] J. M. Hops and J. S. Sherif, “Development and Application of Composite Complexity Models and a Relative Complexity Metric in a Software Maintenance Environment,” *The Journal of Systems and Software*, vol. 31, no. 2, p. 157, 1995.

[20] B. M. Horowitz, “The Importance of Software Architecture,” The MITRE Corporation, Bedford, MA June 1991.

[21] F. Land, “Adapting to Changing User Requirements,” *Information & Management*, no. 5, pp. 59-75, 1982.

[22] B. Lientz, E. Swanson, and G. Tompkins, “Characteristics of Application Software Maintenance,” *Communications of the ACM*, vol. 21, no. 6, pp. 466-471, 1978.

[23] J. Martin and C. McClure, *Software Maintenance: The Problem And Its Solutions*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

[24] D. Parnas, “Designing for Ease of Extension and Contraction,” *IEEE Transactions on Software Engineering*, vol. SE-5, no. 2, pp. 129-137, 1979.

[25] Paul the Apostle, “First Letter to the Corinthians,” 13:1-8.

[26] D. E. Peercy, “Software Logistics National Workshop,” Society of Logistics Engineers, McLean, VA 15-16 Aug. 1989.

[27] D. N. Podger, “High Level Languages - A Basis for Participative Systems Design,” in *Design and Implementation of Computer-Based Information Systems*, N. Szyperski and E. Groschla, Eds.: Sijthoff & Noordhoff, 1979.

[28] J. Ruhl, “Why a computer system is not like a bathtub,” *Software Maintenance News*, vol. 6, no. 12, p. 12, 1988.

[29] E. Swanson, “The Dimensions of Maintenance,” presented at 2nd International Conference on Software Engineering, San Francisco, 1976.

[30] E. C. Van Horn, “Software Must Evolve,” in *Software Engineering*, vol. 1, H. Freeman and P. M. Lewis, Eds.: Academic Press, 1980, pp. 209-226.