EGG-CATT--10183

DE92 017982

Independent Verification and Validation of Large Software Requirement Specification Databases

A Thesis Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Science with a Major in Computer Science
in the College of Graduate Studies, University of Idaho

Kevin E. Twitchell

Published April 1992

Idaho National Engineering Laboratory
EG&G Idaho, Inc.
Idaho Falls, Idaho 83415

Prepared for the
U.S. Department of Energy
Under DOE Idaho Field Office
Contract DE-AC07-76ID01570

ABSTRACT

To enhance quality, an independent verification and validation (IV&V) review is conducted as software requirements are defined. Requirements are inspected for consistency and completeness. IV&V strives to detect defects early in the software development life cycle and to prevent problems before they occur. The IV&V review process of a massive software requirements specification, the Reserve Component Automation System (RCAS) Functional Description (FD), is explored. Analysis of the RCAS FD error history determined that there are no predictors of errors. The size of the FD mandates electronic analysis of the databases. Software which successfully performs automated consistency and completeness checks is discussed. The process of verifying the quality of analysis software is described.

The use of intuitive ad hoc techniques, in addition to the automatic analysis of the databases, is required because of the varying content of the requirements databases. The ad hoc investigation process is discussed. Case studies are provided to illustrate how the process works.

This thesis demonstrates that it is possible to perform an IV&V review on a massive software requirements specification. Automatic analysis enables inspecting for completeness and consistency. The work with the RCAS FD clearly indicates that the IV&V review process is not static; it must continually grow, adapt, and change as conditions warrant. The ad hoc investigation process provides this required flexibility. This process also analyzes errors discovered by manual review and automatic processing. The analysis results in the development of new algorithms and the addition of new programs to the automatic inspection software.

The results of the work from this thesis are also relevant to quality reviews of smaller software requirements specifications.

CONTENTS

FIGURES

TABLES

Chapter 1
INTRODUCTION

1.1 Overview

An early phase of software development is requirements definition. Requirements for a system include anticipated inputs, mandatory outputs, and the requisite processing to transform the input data stream into the output data stream. One method of defining requirements is to use data flow diagrams supplemented with descriptive text. Data flow diagrams graphically depict the inputs, processing (or transformation) of the inputs, and the resulting outputs. Supplementary text provides additional detail, such as narrative describing the transformation.

The requirements specification must accurately document the needs of the user. In addition, it must possess quality attributes such as completeness and consistency. A requirements specification is complete to the extent that all of its components are present and fully developed. It is consistent to the extent that none of its provisions conflict with other provisions within the specification or with other related specifications. If a requirements specification is not complete and consistent, subsequent development activities will be faulty. To minimize rework costs, a verification and validation (V&V) review is conducted as the software requirements are defined. This review inspects the requirements specification to ensure that quality attributes are present. If the nature of the system requires increased confidence in its operation (e.g., for safety reasons), independent verification and validation (IV&V) is appropriate. IV&V is V&V performed by an agency independent from the development organization.

IV&V, as with development, increases in difficulty as the size and complexity of the development effort increases. Requirements IV&V of a small system is relatively simple.

IV&V of the requirements for a large system is more complex but still manageable. However, some systems are massive in scale and cost billions of dollars. One such project, the Reserve Component Automation System (RCAS), has a 35,000 page Functional Description (FD), or software requirements specification. The FD consists of data flow diagrams, process descriptions, and a data element dictionary. IV&V of a requirements specification of this size is necessarily difficult.

The question is, then, how does anyone perform an IV&V review of such a massive functional description? This thesis reviews the concept of V&V in general and requirements V&V in particular. An overview of the RCAS project and FD are provided. This thesis also investigates the IV&V review process of an immense software requirements specification, the RCAS FD. Because of the size of the FD, much of the IV&V review is performed electronically; a complete manual review is simply not possible. The scope of the IV&V review is limited to completeness and consistency for the RCAS FD.

Work to find predictors of areas in the FD which require closer scrutiny is reported. A suite of software providing automatic consistency and completeness checks is discussed. Automatic checks are not sufficient to complete an IV&V review of the FD. A single collection of predetermined automated checks does not locate all of the errors. The process of conducting ad hoc, or intuitive, investigations are detailed.

1.2 Independent Verification and Validation

1.2.1 Verification

Verification is an activity which occurs throughout the software development process. The generally accepted definition of verification is provided in the IEEE Standard Glossary of Software Engineering Terminology [IEEE83]. "Verification is the process of determining whether or not the products of a given phase of the software development cycle

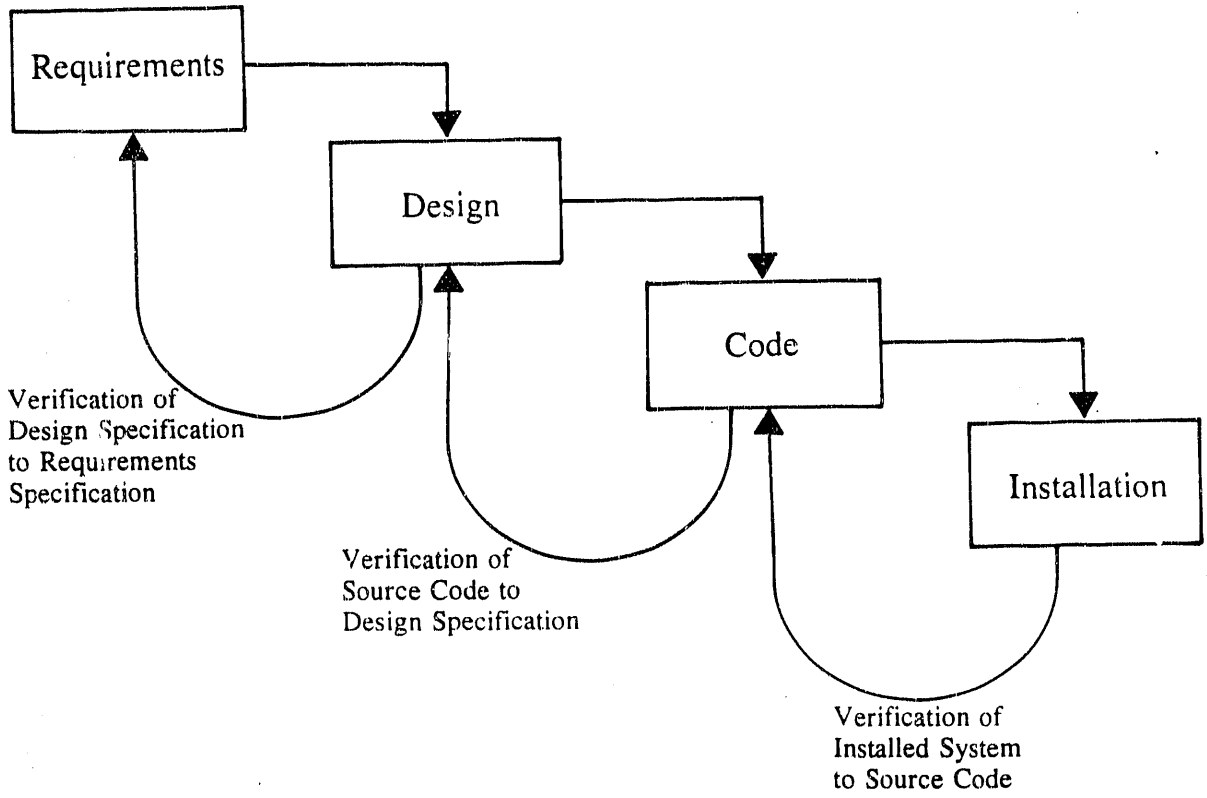

Figure 1. Verification in the Software Development Process.

fulfill the requirements established during the previous phase." Thus, the design specification is verified against requirements specification, the source code is verified against the design specification, etc. (see Figure 1). In addition, verification "determines that each phase and subphase product is correct, complete, and consistent with itself and with its predecessor product" [Andr86].

The Department of Defense Standard DOD-STD-2168 combines these two aspects of verification in its definition. "Verification is the process of evaluating the products of a given phase of the software development cycle to ensure correctness and consistency with respect to the products and standards provided as input to that phase" [Schu87]. Perhaps the simplest and best definition for verification is given by Boehm: "Am I building the product right?" [Boeh84].

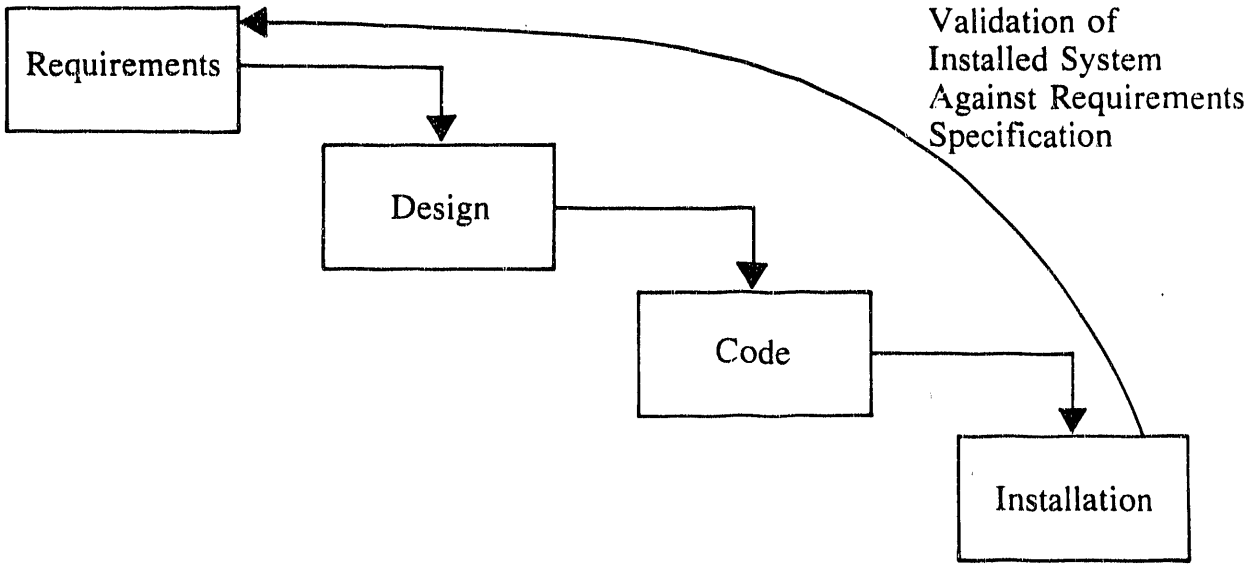

Figure 2. Validation in the Software Development Process.

1.2.2 Validation

Validation is traditionally viewed as determining, at the end of the software development process, if the completed system fulfills the original requirements (see Figure 2). The IEEE Standard Glossary of Software Engineering Terminology [IEEE83] defines validation as "the process of evaluating software at the end of the software development process to ensure compliance with software requirements."

However, determining that a completed system does not meet requirements results in expensive modifications and costly delays. Some authors suggest that validation should take place at each phase of the software development life cycle (see Figure 3) [Deut88, Evan87, and Andr86]. As with verification, Boehm succinctly summarizes validation: "Am I building the right product?" [Boeh84].

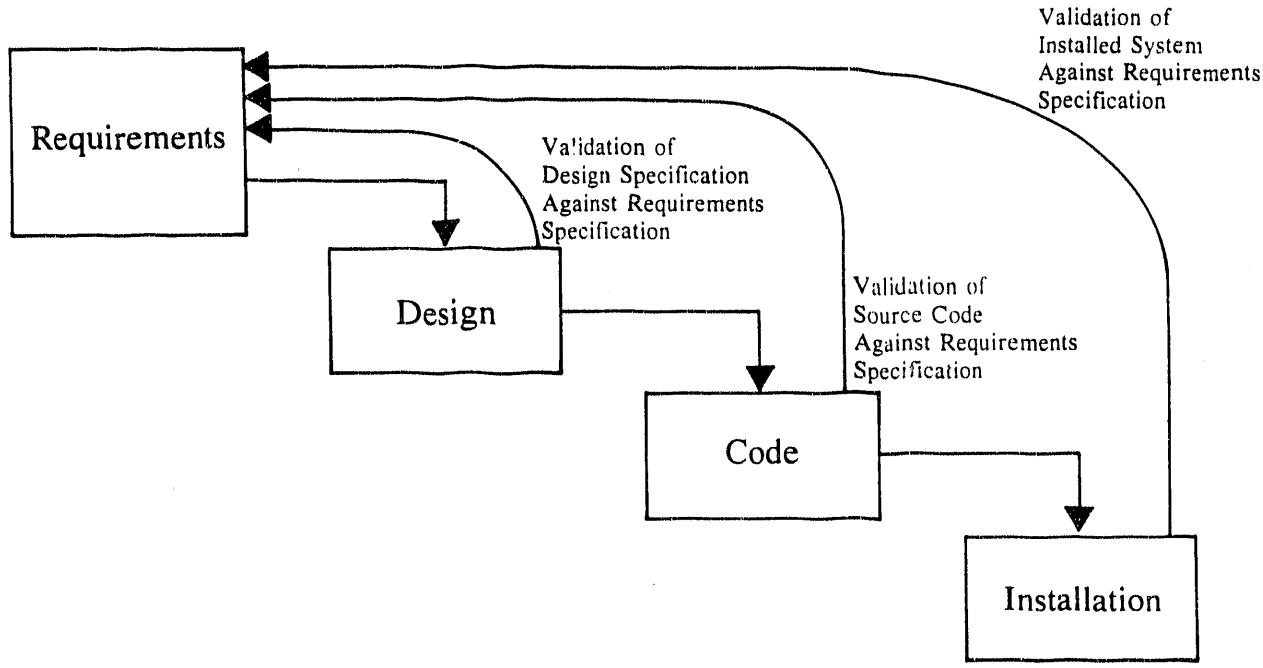

Figure 3. Verification in the Software Development Process (revised).

1.2.3 Verification and Validation

Verification and validation (V&V) is more than just the simple combination of verification and validation activities (see Figure 4). It is a process of continual review [Andr86]. V&V determines if the developing software will perform its intended functions and helps ensure the quality of the resulting software [Deut88 and Schu87]. It provides feedback to management regarding the achievement of quality goals as the product is being developed [Deut88]. "V&V comprehensively analyzes and tests software to determine that it performs its intended functions correctly, to ensure that it performs no unintended functions, and to measure its quality and reliability" [Wall89].

As a quality function, V&V strives to detect defects early in the software development life cycle and to prevent

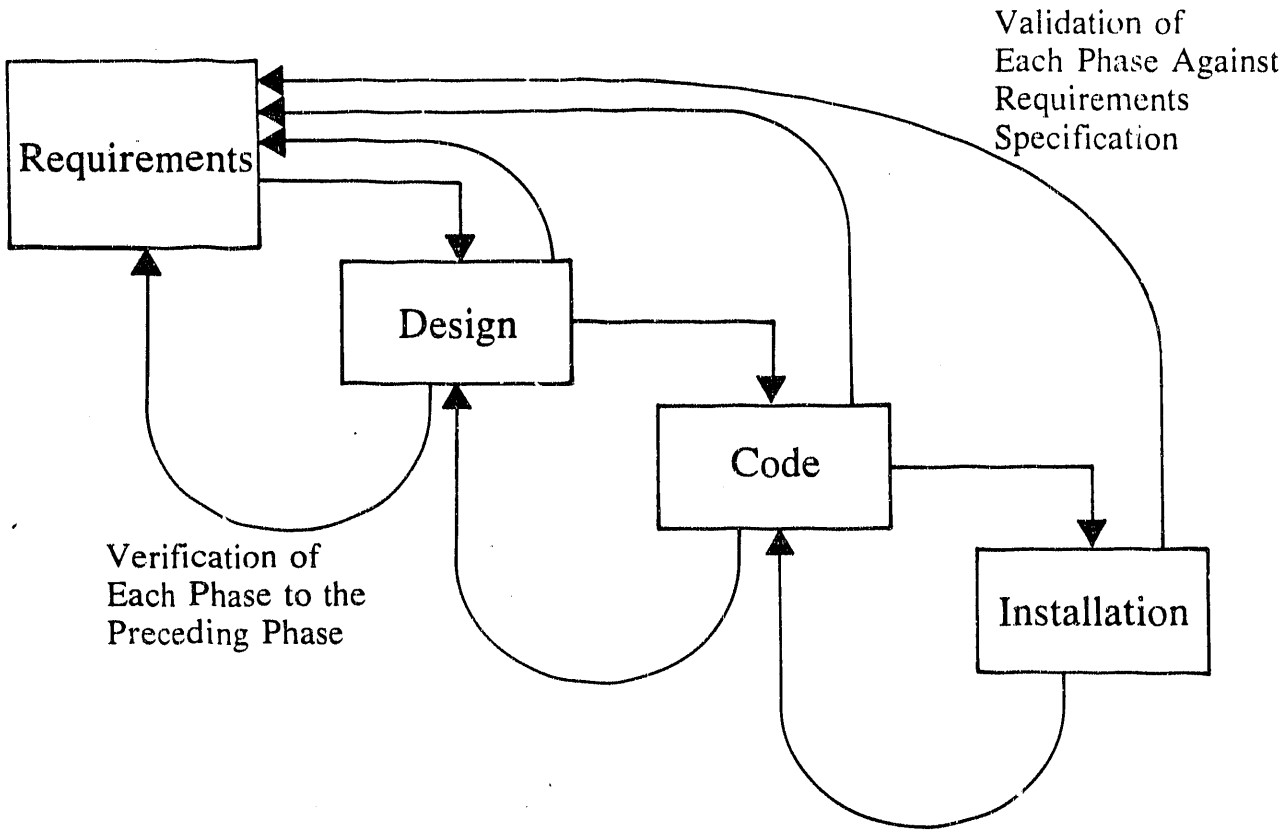

Figure 4. Verification and Validation in the Software Development Process.

problems before they occur [Deut88].

The benefits of verification and validation are [Wall89]:

- improvement in quality
- more stable requirements
- more rigorous development planning
- errors caught earlier
- better schedule compliance and progress monitoring
- project management more aware of interim quality and progress
- better criteria and results for decision making at formal reviews and audits

There are also disadvantages to verification and validation [Wall89]:

- adds 10 to 30 percent to the development cost
- requires additional interfaces between project groups

- lowers developer productivity if programmers and engineers spend time explaining the system to V&V analysts
- adds to the documentation requirements if the V&V group is receiving incremental program and documentation releases
- requires the sharing of computing facilities and classified data with the V&V group
- increases the paperwork to provide written responses to the V&V group's error reports and other V&V data requirements

Wall suggests "... the number of discrepancies detected in software and the improvement in documentation quality resulting from error correction suggest that V&V costs are offset by the resulting more reliable and maintainable software." [Wall89]

1.2.4 Independent Verification and Validation

Quality is the responsibility of the producer of a product. Verification and validation, as a quality function, is no exception. V&V "is the responsibility of all elements on the software project to perform it" [Deut88].

However, there may be a problem in giving the sole responsibility of V&V to the software developer. Given the impact of V&V functions on cost and schedule, the manager of development could put pressure on V&V personnel to be less stringent in their reviews. Consequently, many hold that V&V activities should be conducted by a group with at least nominal independence from the development group [Schu87]. As summarized by Boehm, "Verification and validation activities produce their best results when performed by a V&V agent who operates independently of the developer or specification agent" [Boeh84].

The level of independence of those performing V&V work is a function of the importance of the software being developed

[Schu87]. "Independent verification and validation (IV&V) is employed when added insurance is required for the functioning and performance of the system" [Deut88]. IV&V is performed by "an organization that is technically and managerially separate from the organization responsible for developing the product or performing the activity being evaluated" [DOD-STD-2168 in Schu87]. The reason for the independence is to ensure the "disinterest in the outcome of the evaluation, i.e., the agent performing the validation should have no stake in the results and should be free of influence from those who do" [Deut88].

IV&V should not preclude V&V activities in the development organization. Rather, IV&V is a supplement to software quality assurance activities in the development organization. "The objective of IV&V is not necessarily to avoid all errors, but rather to eliminate, to a high degree of certainty, those errors that can result in catastrophic results" [Deut88].

1.3 Requirements Verification and Validation

1.3.1 Need for Requirements V&V

Requirements verification and validation benefits the schedule and budget of a development project [Deut88]. Errors detected early in the life cycle require fewer resources, i.e., time and money, to effect a repair [Deut88].

"The basic objectives in verification and validation of software requirements and design specifications are to identify and resolve software problems and high-risk issues early in the software life cycle. ...for large projects, the savings of up to 100:1 are possible by finding and fixing problems early in the life cycle. For smaller projects, the savings are more on the order of 4-6:1, but this still provides a great deal of leverage for early investment in V&V activities. Besides the major cost savings, there are also significant payoffs in improved reliability, maintainability,

and human engineering of the resulting software product" [Boeh84].

1.3.2 Requirements V&V Criteria and Attributes

Boehm maintains that the four basic requirements verification and validation criteria are completeness, consistency, feasibility, and testability. The following is a summary of his classic article on requirements verification and validation [Boeh84].

Completeness. A requirements specification is complete to the extent that all of its components are present and fully developed. A specification is complete if:

- there are no to-be-determined's (TBDs)
- all processes, data flows, etc. which are referenced actually exist and are defined
- all items required by the standard format of the specification are present
- all necessary functions and products, e.g., recovery functions and test tool products, are defined or called for

The first two items above can be verified automatically; the remaining two items require human intuition.

Consistency. A requirements specification is consistent to the extent that none of its provisions conflict with other provisions within the specification (internal consistency) or with other specifications (external consistency). If items in the specification clearly indicate the items from which they were derived, the specification is said to have traceability.

Feasibility. A requirements specification is feasible to the extent that the system's benefits are expected to exceed its costs over its lifetime. Verification and validation of feasibility requires examination of:

- human engineering issues -- how the system will function with users

- resource engineering -- how the system fulfills the requirements
- program engineering -- cost-effectiveness of the system over its life

Feasibility also requires identifying and resolving any high-risk issues before committing resources to development. The four major sources of risk are technical, cost/schedule, environment, and interaction effects.

Testability. A requirements specification is testable to the extent that one can develop a reasonable means for determining whether the finished system fulfills the requirements. This requires that the specification be specific, unambiguous, and quantitative wherever possible.

Evans and Marciniak assert that the attributes of a requirements specification which are critical to the quality of the system are nonambiguity, traceability, testability, implementability, completeness, reality, acceptability, user responsiveness, and design free [Evan87]. A requirements V&V should check for the presence of these attributes.

Nonambiguity. Each requirement is clear and the meaning is obvious.

Traceability. Each requirement is uniquely specified and is traceable throughout the development project.

Testability. Each requirement can be verified by later testing, measurement, observation, or analysis.

Implementability. All requirements can be completed given the ability of the development staff. Each requirement is technically achievable.

Completeness. All aspects of the system are described.

Reality. The requirements are within the technical scope, limitations, and constraints of the project.

Acceptability. A system based on the specified requirements is acceptable to the user.

User responsiveness. The requirements are responsive to the needs of the user and the environment in which the system will be used.

Design free. The requirements specify what the system should do (requirements) rather than how the system will do it (design).

Zave states that "the things we do with requirements specifications are 1) use them as vehicles for communication, 2) change them, 3) use them to constrain target systems, and 4) use them to accept or reject final products" [Zave82]. For the first two items, the requirements must be understandable and modifiable. For the third item, they must be precise, unambiguous, internally consistent, complete, and minimal. For the final item, requirements should be formally manipulable (for verification) and testable (for acceptance testing).

1.3.3 Requirements V&V Activities

There are two basic requirements V&V activities [Deut88]. One is conducted in parallel with the generation of the requirements specification. This activity helps create desirable characteristics such as completeness, consistency, feasibility, and testability. It involves systematic methods, such as structured analysis, that emphasize human reasoning. The other activity is conducted after the requirements specification is generated (e.g., in draft form). This after-the-fact activity is characterized by reviews and inspections.

The two activities are iterative and continue until all known deficiencies are corrected and no new deficiencies are discovered.

1.3.4 Requirements V&V Techniques

Boehm suggests several techniques for performing a requirements verification and validation [Boeh84]. These

include simple manual, simple automated, detailed manual, and detailed automated techniques.

Simple manual techniques include reading, manual cross-referencing, interviews, checklists, manual models, and uncomplicated scenarios. Simple automated techniques consist of automated cross-referencing and basic automated models. Detailed manual techniques entail extensive scenarios and mathematical proofs. Detailed automated techniques include complex models and prototypes.

For large systems, the most effective V&V technique for completeness and consistency is automated cross-referencing. Also very effective are detailed automated models and prototypes. Manual cross-referencing, interviews, and detailed scenarios are also helpful. Checklists are also beneficial for completeness V&V.

Boehm recommends the following for large specification V&V [Boeh84]:

- use automated cross-referencing
- use simple-to-detailed manual and automated models for critical performance analyses
- use simple-to-detailed scenarios for critical user interfaces
- prototype high-risk items that cannot be adequately verified and validated by other techniques

1.3.5 Requirements V&V Tools and Techniques

Most of the tools and techniques which could be used for requirements V&V were developed during the mid-1970's [Deut88]. Indeed, with the exception of the proliferation of Computer-Aided Software Engineering (CASE) tools, there have been few innovations since then. Tools and techniques developed in the mid-1970's include Information Systems Design and Optimization Systems (ISDOS), Requirements Engineering and Validation System (REVS), and Structured Analysis and Design Technique (SADT). "All provide a disciplined framework for

expressing requirements and thus aid in the checking of consistency and completeness. Although these tools provide only rudimentary validation procedures, this capability is greatly needed and is the subject of current research." [Andr86]

The following is a summary of tools and techniques which are used for requirements specification. The main characteristics of each tool and technique are provided. Most of the following tools and techniques do not provide specific capabilities for performing requirements V&V. However, the techniques do supply a structure for the requirements specification which can be checked in a requirements V&V review. For example, a transform (process) in a structured analysis data flow diagram is not complete if it does not have both inputs and outputs. The automated tools are used to extract data for V&V analysis and review. Thus a tool could be used to generate a list of data flows without defining data elements, a deficiency in completeness.

Information Systems Design and Optimization Systems (ISDOS). Problem Statement Language / Problem Statement Analyzer (PSL/PSA) was created as a result of the ISDOS project. It is one of the first CASE tools and was developed in the mid-1970's. PSL is the formal language used to specify the software system. PSA processes the PSL statements and stores the corresponding information in a database. PSA also provides a user interface facilitating retrievals of information from the database. PSL/PSA is the CASE tool for the RCAS project and is discussed in detail in Section 2.1.2, RCAS Computer-Aided Software Engineering Tools.

Structured Analysis and Design Technique (SADT) [RoSc77 and Ross77]. SADT is SofTech's proprietary methodology for handling complex system problems. It is based on Structured Analysis (see Section 2.1.2, RCAS Computer-Aided Software Engineering Tools) and the maxim, "Everything worth saying about anything worth saying something about must be expressed

in six or fewer pieces." It consists of techniques for performing system analysis (e.g., structured decomposition) and provides a process for applying these techniques in requirements definition. It uses graphics and text to communicate the elements of the requirements specification. An automated tool such as PSL/PSA is compatible with the SADT methodology.

Computer-Aided Design and Specification Analysis Tool (CADSAT) [Deut88]. CADSAT is also based on structured analysis and is a tool which provides automated checking of the forms and structures in a requirements specification. It is a direct descendant of PSL/PSA. Requirements are defined in a formal language (graphic and textual) with a precise syntactical structure. CADSAT automatically checks the relationships between data and functions for consistency. It generates reports that assist the analyst in better understanding system flow, system structure, data structure, and data derivation.

Process-oriented, Applicative, Interpretable Specification Language (PAISLey) [Zave82]. PAISLey is process-oriented, or operational, in that it emphasizes constructing an operating model of the system functioning in the system's environment. PAISLey is intended to specify the requirements for embedded systems (such as process control applications). Embedded systems require the specification of asynchronous parallel activities and crucial performance requirements. PAISLey is based on applicative, or functional, language theory and is interpretable (executable). As such, it is a language with a precisely defined grammar and syntax.

SOFSPEC [Nyar83]. SOFSPEC is a system for the automatic verification of commercial application specifications. It is part of the SOFTING Software Engineering System [Snee83]. The SOFSPEC specification schema is based on the entity-relationship model, including entities, relationships, and attributes. SOFSPEC checks for completeness by determining if

all declared entities and relationships have been fully defined. It does not check for completeness by determining if all essential entities and relationships have been included; this requires human involvement. SOFSPEC provides 16 consistency checks including checking objects against their relationships, analyzing data and function trees for internal consistency, and checking the consistency between function and data definitions and their structural characteristics. A set of reports are automatically generated to aid the user in determining that the specification meets the actual requirements.

System Verification Diagram (SVD) [Deut88]. An SVD is a manual technique based on the concept of threads. A thread is a path through a system that connects an external event or stimulus to an output event or response. An SVD consists of graphical representations of functional requirements from the requirements specification. The SVD exhibits a sequence of stimulus-response elements. A stimulus is an input event and a response is an output event. The SVD procedure focuses additional attention on the content of the requirements. Because of the graphical nature of an SVD and because requirements are viewed in a parallel manner, the analyst's attention is directed to detect possible inconsistencies, redundancies, and omissions.

Requirements Engineering and Validation System (REVS) [Alfo77, Bell77, Davi77, and Boeh84]. REVS is part of the Software Requirements Engineering Methodology (SREM). REVS accepts Requirements Statement Language (RSL) as input. RSL is a formal language used to document requirements using elements, relationships, attributes, and structures. REVS checks the requirements for completeness and consistency by automatically searching for errors such as undefined, incomplete, and unused elements. It maintains traceability to originating requirements and simulations. REVS generates simulations to aid validation of the correctness of

requirements. REVS contains a reporting system and provides data extraction and analysis capabilities. REVS software includes an interactive graphics package to aid in the specification of flow paths.

Chapter 2
THE RESERVE COMPONENT AUTOMATION SYSTEM

2.1 Project Environment

2.1.1 Reserve Component Automation System

The Reserve Component Automation System (RCAS) is a 1.6 billion dollar information system intended to modernize the operations of the Army National Guard and the U.S. Army Reserve. It provides hardware and software for reserve units in thousands of locations throughout the United States and the world. RCAS supports daily administrative tasks as well as the mobilization of forces. It encompasses office automation, telecommunications, data processing, and exchange of data with related Army information systems.

RCAS work is directed by the RCAS Program Management Office (PMO). The PMO is responsible for the development, acquisition, implementation, and fielding of RCAS. Assisting the PMO are several contractors, tasked with individual responsibilities. One contractor is tasked with the preparation of the Functional Description (FD). Another contractor is tasked with performing independent verification and validation reviews of the FD.

2.1.2 Computer-Aided Software Engineering Tools

Final work on the FD is being completed using two Computer-Aided Software Engineering (CASE) tools. One of the CASE tools executes on a personal computer (PC) and primarily serves an input function. The other CASE tool executes on a Digital Equipment Corporation (DEC) VAX. The full information of the FD appendices resides in the databases of this VAX CASE tool. Both CASE tools are products of Meta Systems, Ltd.

The PC CASE tool is Structured Architect. Initially, Structured Architect (SA) 1.2 was used to prepare the FD. Meta Systems replaced SA 1.2 with Structured Architect Workbench (SAW), which is currently used to prepare the FD.

Structured Architect is a CASE tool which is based on structured analysis [Stru86 and Intr89]. Structured analysis is an approach to requirements specification developed and popularized in the 1970s [DeMa78]. It combines graphic illustrations (data flow diagrams) and textual descriptions (structured English) to specify the processing of data within the system being defined. It emphasizes a hierarchical, or top-down, approach to decomposition and partitioning.

The DEC VAX tool is Problem Statement Language / Problem Statement Analyzer (PSL/PSA). Unlike SA 1.2 and SAW, it does not enforce any particular methodology [Teic77]. PSL/PSA was developed by the Information Systems Design and Optimization Systems (ISDOS) Project at the University of Michigan [Intr87]. It consists of two parts, PSL and PSA. PSL is the formal language used to describe components of the system. Since the language has a limited vocabulary and a controlled syntax, it can be processed by the analyzer, PSA (see Figure 5). PSA also processes user requests for information and generates reports.

PSL/PSA is based on entity-relationship modeling in which a system is defined using entities, relationships between and among entities, and attributes of the entities [Intr87 and Mapp87]. In simple terms, entities are "things," relationships are "connections between things," and attributes are "characteristics of things" [Mapp87]. In the RCAS FD, an entity is a process, data flow, data store, or data element. Relationships connect data flows to processes and data stores. Relationships also link data flows and data elements (a data flow CONSISTS OF data elements). Attributes are defined for processes (e.g., process number and description) and data elements (description, long name, data type, and length).

PSL/PSA provides 19 object types, 102 relation types, and general provisions for attribute definitions [Intr87]. Fortunately, when SA 1.2 and SAW generate PSL, they use a defined subset of these objects, relations, and attributes.

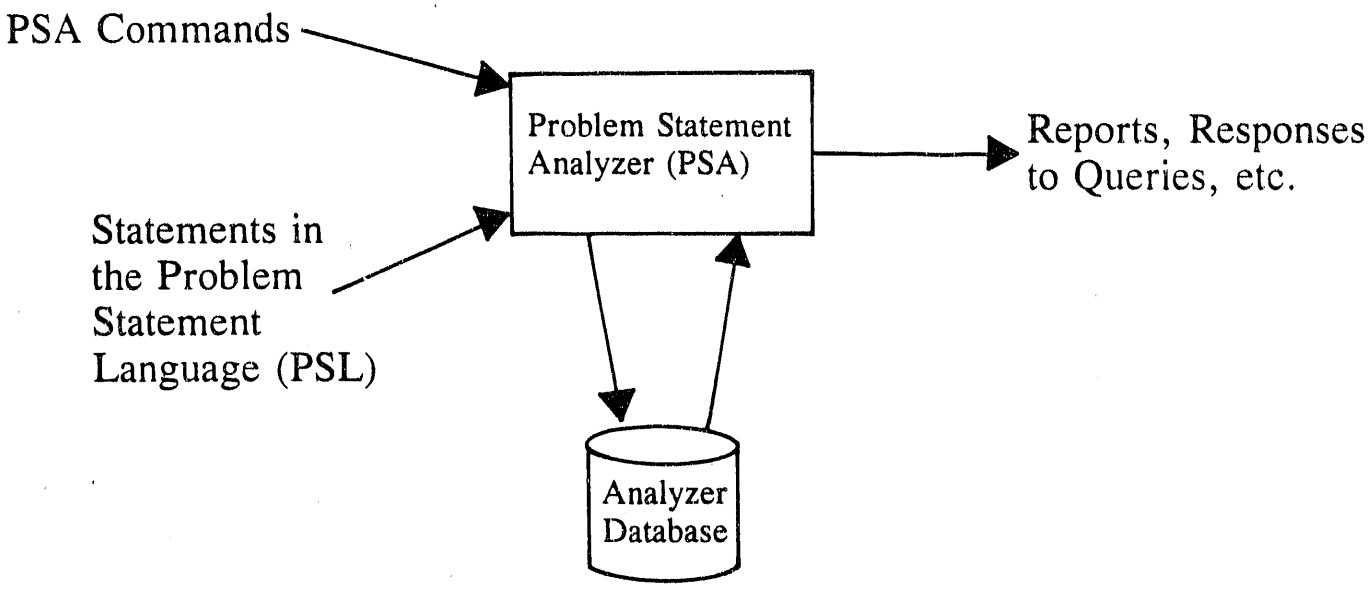

Figure 5. Problem Statement Language / Problem Statement Analyzer (PSL/PSA).

This simplifies the analysis of the PSL/PSA databases.

To aid understanding of later discussion, the following definitions are offered:

Objects. A functional primitive is the lowest-level decomposition of a process. A functional primitive has the PSL/PSA object type of PROCESS. A data flow represents the data transferred from a source to a destination. Sources and destinations of data flows can be processes, data stores, or non-existent (e.g., to a process not on the data flow diagram). A data flow has the PSL/PSA object type of ELEMENT in the main database and GROUP, ENTITY, INPUT, or OUTPUT in the data flow dictionary. An intermediate entity is an optional link between data flows and data elements; it is an additional grouping of data elements. A data flow may consist of other data flows, intermediate entities, and data elements. An intermediate entity may consist of the same items. The difference between the two is that the data flow appears on the data flow diagram while the intermediate entity is part of the textual description. An intermediate entity has the

PSL/PSA object type of GROUP, ENTITY, INPUT, or OUTPUT. A data element is the smallest stand-alone component of a data flow and is represented in PSL/PSA as an ELEMENT in both the data flow and data element dictionaries. A data store is a place to temporarily or permanently store data. In the RCAS FD, a data store is also used to represent some external interfaces such as the user. The data store has the PSL/PSA object type of SET.

Relations. Data flows are identified by their relationships with processes and data stores. If a data flow is an input to a process, the relationships used are RECEIVES, EMPLOYS, REFERENCES, and OBTAINS. The relationships used for output data flows are GENERATES, DERIVES, MODIFIES, UPDATES, and SENDS. A data flow FLOWS from one entity (process or data store) to another entity. CONSISTS OF is the relationship used to link data flows, intermediate entities, and data elements. For example, a data flow CONSISTS of data elements. SATISFIES is the relationship used to associate echelon, classification, frequency, and sensitivity data to functional primitive processes and external interface information to data flows.

Attributes. Processes have attributes of process number and textual description. Data elements have attributes of long name, description, data type, length, proponent, and information class. Data flows, intermediate entities, and data stores do not possess attributes.

PSA is capable of generating 37 pre-formatted reports [PSAR87]. Two of these reports, Name Selection and Selected Formatted Statements, are significant in the analysis of the PSL/PSA databases.

The Name Selection report extracts a list of names typically based on a PSL/PSA object type such as PROCESS. These names provide an entry into the PSL/PSA databases for analysis.

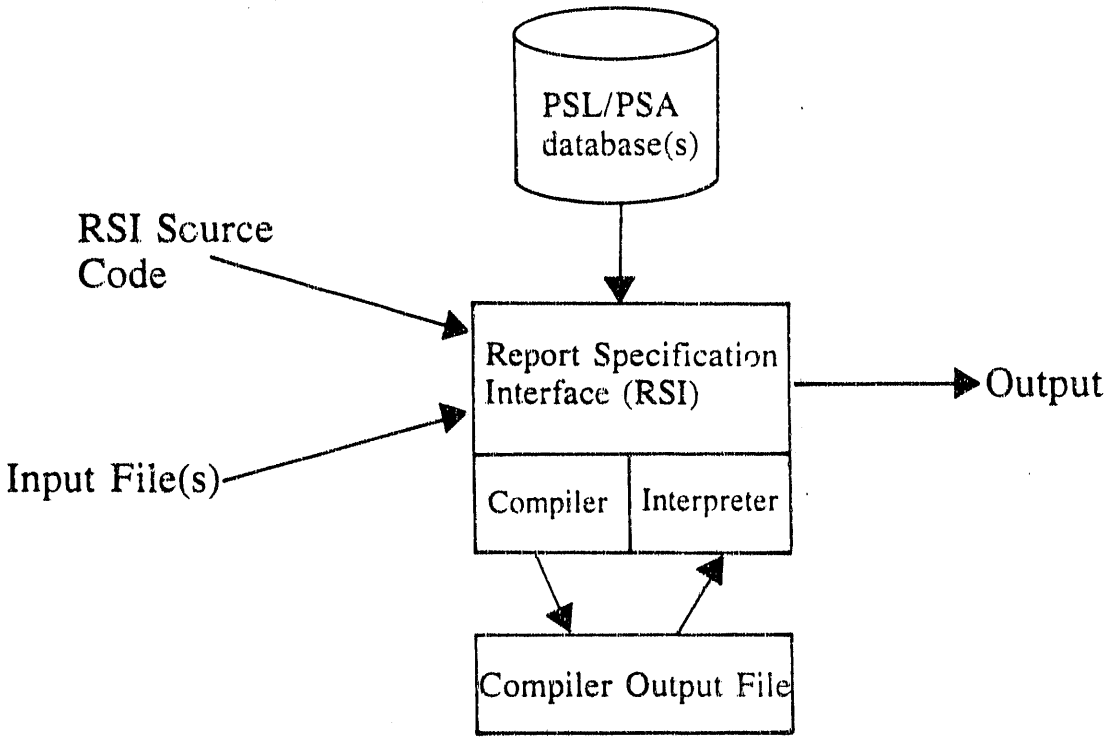

Figure 6. Report Specification Interface (RSI).

The Selected Formatted Statements report generates the PSL from a database. This is useful in comparing the contents of databases and in extracting general information from a database.

Another component of the PSL/PSA CASE tool is the Report Statement Interface (RSI) Language used to generate customized reports [Refe90]. RSI is an executable language and provides interface to PSL/PSA databases (see Figure 6). RSI is the primary tool used to automatically analyze the PSL/PSA databases.

2.2 RCAS Functional Description

2.2.1 Overview

The RCAS FD conforms to DOD-STD-7935A, DOD Automated Information Systems (AIS) Documentation Standards [Mili88]. It consists of 26 volumes (see Table 1). The FD proper,

because of where it resides, is commonly referred to as Volume 1. Appendix E contains the definition of External Interfaces of the RCAS with other Army information systems. Appendix T specifies the echelon, classification, frequency, and sensitivity associated with each FD functional primitive. Appendices F through I, K through S, and W through AB contain process descriptions and data flow diagrams. The RCAS Data Element Dictionary is a PSL/PSA database.

A subset of the FD functional primitives have been defined as "critical elements." These processes are to be

Volume	Element	Title
1	FD	Functional Description
	Appendix A	References
	Appendix B	Glossary
2	Appendix C	Mobilization Stations
	Appendix D	The Developmental Army Readiness and Mobilization System
3	Appendix E	Interface Systems
4	Appendix F	Mobilization Planning Management
5	Appendix G	Mobilization Execution Management
6	Appendix H	ARNG Force Authorization Management
7	Appendix I	ARNG Human Resources Management
8	Appendix J	deleted; merged into Appendix P
9	Appendix K	ARNG Financial Management
10	Appendix L	ARNG Logistics Management
11	Appendix M	ARNG Installations Management
12	Appendix N	USAR Force Authorization Management
13	Appendix O	USAR Human Resources Management
14	Appendix P	ARNG/USAR Training Management
15	Appendix Q	USAR Resource Management
16	Appendix R	USAR Logistics Management
17	Appendix S	USAR Engineering Management
18	Appendix T	Index of Functions
19	Appendix U	Workload Analysis
20	Appendix V	Site Inventory
21	Appendix W	ARNG Aviation Management
22	Appendix X	ARNG Safety Management
23	Appendix Y	USAR Safety Management
24	Appendix Z	USAR Aviation Management
25	Appendix AA	ARNG Information Management
26	Appendix AB	ARNG Internal Review Management
n/a	DED	RCAS Data Element Dictionary

Table 1. RCAS Functional Description Organization [RCAS91].

implemented by the development contractor first. Critical elements are defined in Volume 1 of the FD. Actual implementation of the FD has been broken into three phases designated as Blocks 1, 2, and 3.

The first phase, Block 1, includes approximately half of the critical elements. Block 1 provides functionality for force authorization and human resources. The FD definition of Block 1 processes was released to the development contractor on 01 Oct 1991. The second phase, Block 2, consists of all critical elements not included in Block 1. Areas of Block 2 functionality include training, logistics, additional human resources and force authorization, mobilization, finance, aviation, and engineering. The third phase, Block 3, consists of all remaining FD functional primitive processes.

The FD is composed of three parts, designated as Part A, Part B, and Part C (see Figure 7). Part A consists of textual description and data flow diagrams. It constitutes the hard copy portion of the FD and is the standard against which parts B and C are compared for accuracy. To facilitate maintenance, Part A is being replaced by reports generated from the Part C PSL/PSA databases.

Part B is developed using a CASE tool resident on an IBM-compatible personal computer. Up to and including the 04 Feb 1991 release, Structured Architect (SA) 1.2 was used. At that time, the SA 1.2 databases were converted to Structured Architect Workbench (SAW).

Part C of the FD is generated from Part B. PSL is generated from the Structured Architect databases and loaded into PSL/PSA databases. In addition, synonyms of process names, consisting of process numbers, are added in a "post-processing" step.

The historical study of the FD covered the 11 electronic releases of the VAX-based PSL/PSA databases (see Table 2). The first three releases were prepared by a contractor other than the one currently preparing the FD.

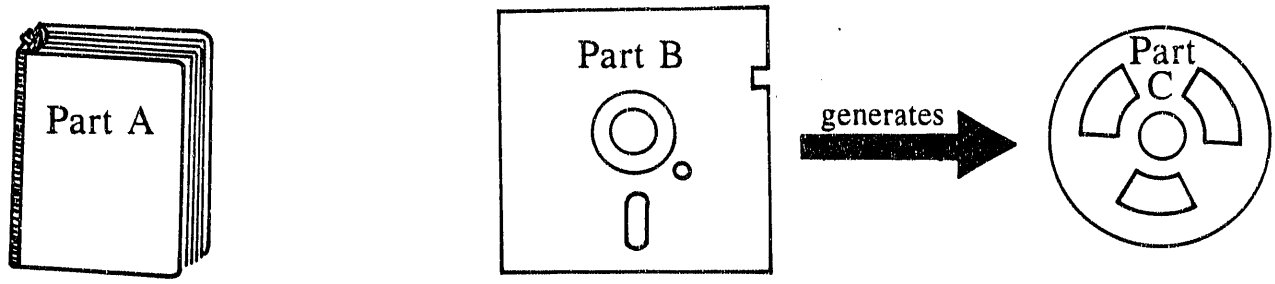

Figure 7. Parts of the Functional Description.

Until the 07 May 1991 release, all Part C PSL/PSA databases were generated from SA 1.2. Part C PSL/PSA databases were generated from SAW starting with the 07 May 1991.

2.2.2 PSL/PSA Database Structure

Each release of Part C (PSL/PSA databases) includes as a minimum at least one PSL/PSA database file. This database, or the main database, contains information regarding processes, data flows, and data stores (see Figure 8). Data flows are further defined in the data flow dictionary (DFD). The DFD defines data flows with respect to data elements. Data elements are defined in the Data Element Dictionary (DED).

In addition, the releases of the Part C databases include any combination of the following databases:

The main database information has been included in:

- a single database containing information for only one appendix
- a single database containing information for multiple appendices
- multiple databases containing information for only one appendix

Release	Comments
02 Nov 1989	first FD prepared using PSL/PSA on the VAX
29 Nov 1989	identical to 02 Nov 1989 for Block 1
26 Feb 1990	last release by the previous contractor
06 Apr 1990	first release by the current contractor
14 May 1990	intermediate release for IV&V review
19 Nov 1990	identical to 14 May 1990 for Block 1
22 Dec 1990	intermediate release for IV&V review prior to issuing the 04 Feb 1991 baseline
04 Feb 1991	hard copy and electronic baseline; identical to 22 Dec 1990 for Block 1; last Part C generated from SA 1.2
07 May 1991	first pass of refinement effort for the entire FD; first Part C generated from SAW
03 Jul 1991 24 Jul 1991 16 Aug 1991	Block 1 only; continuation of refinement effort; issued to proponents for review
01 Oct 1991	Block 1 only; released to development contractor and is therefore correct by definition

Table 2. Functional Description Part C Releases.

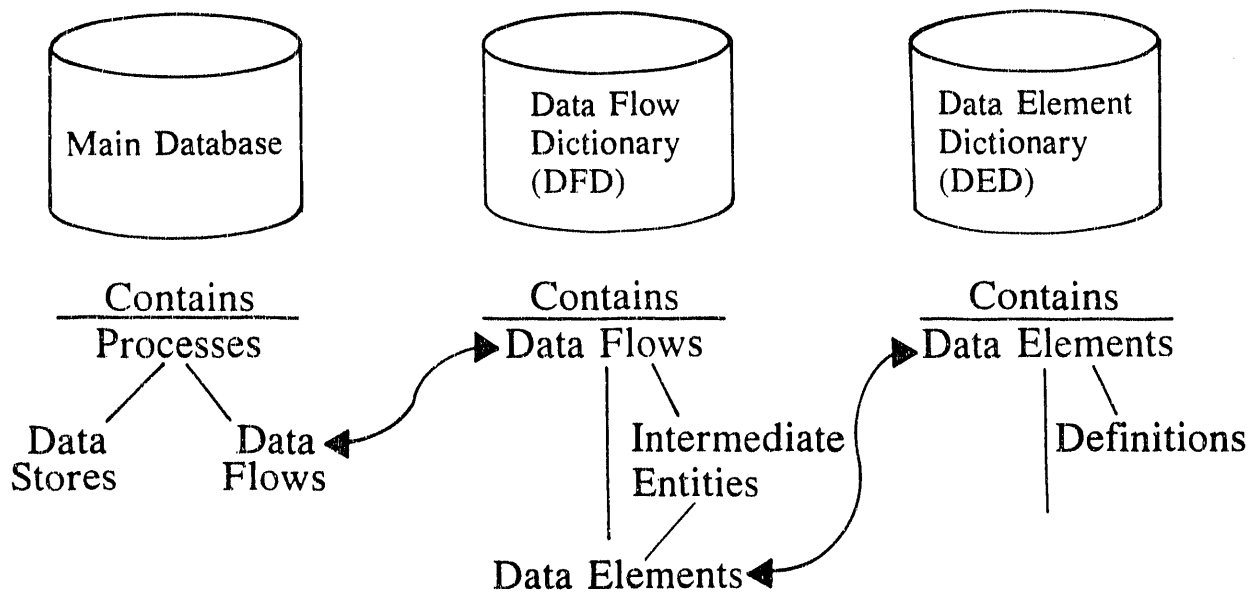

Figure 8. RCAS PSL/PSA Databases.

The DFD (data flow dictionary) information has been included in:

- the main database
- a separate file dedicated to a single appendix
- a separate file common to multiple appendices

The DED (data element dictionary) information has been included in:

- the main database
- a separate file dedicated to a single appendix
- a separate file common to multiple appendices

2.2.3 Search for Error Predictors

Data from the various releases of the RCAS FD were analyzed to determine if error predictors exist in the FD data. A predictor would indicate where errors are more likely to occur in the FD and would facilitate improving the way the IV&V review of the FD is conducted. With a predictor, those portions of the FD which are more likely to contain errors would undergo a more intense review than those portions which are not as likely to contain errors.

The first step to locating error predictors is to gather data. The content of the releases of the FD is recorded. Historical FD data are reviewed for error content. The types and quantities of errors are noted. In addition, the differences between a given release and the final release are also counted. These differences represent the evolution of the FD over time. In addition to enumerating errors and differences, changes from one release to the next are also cataloged.

The next step is to analyze the data. Normalization of the data is necessary to facilitate comparisons of data from the various FD Appendices and releases. If patterns or correlations are found in the data, then an error predictor exists. The absence of a pattern or correlation may signal

the presence of some other factor which prevents the existence of an error predictor.

2.2.3.1 Functional Description Data

The historical study of the releases of the FD Part C PSL/PSA databases is restricted to Block 1 functional primitive processes. Three of the 11 releases are excluded because the content, for Block 1 functional primitive processes, is identical to that in the previous release (see Table 2).

The study includes Block 1 functional primitive processes from Appendices N, H, I, and O. In addition, the four appendices are considered as a whole (Block 1). Consideration of Block 1 allows for the sharing of data flows, data stores, and data elements among the four appendices. However, processes are kept distinct even though process names may be shared.

Appendix A of this thesis summarizes the content of the releases databases according to the number of databases in the release including the main database, the DFD, and the DED. In addition the number of functional primitive processes, data flows, intermediate entities, data elements, and data stores are listed.

Appendix B of this thesis summarizes changes in a given release from the previous release with regard to:

- lists of functional primitive process, data flow, intermediate entity, data element, and data store names.
- process descriptions for each and every process. Comparisons are made on a line by line basis. Thus changes include formatting changes as well as textual changes.
- data element definitions for each and every data element. A change is noted if any portion of the data element definition is modified.

Appendix C of this thesis enumerates errors in each of the releases. Appendix D of this thesis summarizes the differences from the final (01 Oct 1991) release over time. These differences represent one of two things. One is that the differences are the result of a gradual evolution of the FD over time. Changes are made to the FD as more is learned about the RCAS or as the users change the definition of how the RCAS should function. The other is that the differences are actually errors in the FD which are corrected over time. Differences include not only the existence (or non-existence) of objects but the names of objects.

2.2.3.2 Data Normalization

Block 1 Appendix N consists of one functional primitive while Block 1 Appendix O consists of 933 functional primitives. In order to compare errors and changes over time, it is necessary to normalize the data for each appendix and for Block 1 as a whole. Data are normalized using three values: simple object sum, comprehensive object sum, and modified Bang metric.

2.2.3.2.1 Simple Sum of Objects

The first value used to normalize the data is the sum of the number of unique names of functional primitive processes, data flows, data elements, and data stores. These are the elements which appear on the FD Part B data flow diagrams. The simple object sum excludes multiple use of data flows, intermediate entities, data stores, and data elements which are defined only once in the PSL/PSA database. Any subsequent changes or correction of errors are made in only one place in the database.

2.2.3.2.2 Comprehensive Sum of Objects

The simple sum of objects has deficiencies. It does not include intermediate entities, the building block of data

flows. It also excludes data flows which are used more than once but have the same name. This occurs when a given process outputs the same information to multiple destinations or when the same information is processed by more than one process. A data flow should be counted each time it is used since it is the connecting relationship between functional primitives and data stores or other functional primitives. The comprehensive sum of objects, then, is the sum of functional primitive processes, each occurrence of each data flow, intermediate entities, data stores, and data elements.

2.2.3.2.3 Modified Bang Metric

DeMarco formulated a metric, called Bang, to forecast the cost and duration of a software project using the requirements specification as input [DeMa82]. The metric is intended to measure system functionality and is not specifically intended to be used to normalize data. However, it is a useful tool for comparing software projects and as such, should be practical for normalizing data.

DeMarco defines two types of systems. One is function-strong. This is a system which can be thought of almost entirely in terms of the operations performed upon data. The other type of system is data-strong. This system is one that must be thought of in terms of the data it acts upon rather than the operations upon the data. RCAS, as defined in the FD, is a function-strong system.

DeMarco maintains that the principal indicator for most function-strong systems is the functional primitive (FP). DeMarco's functional primitive is the lowest level of decomposition of function and is equivalent to an RCAS PSL/PSA functional primitive. However, DeMarco recognizes that not all functional primitives are equal. Some functional primitives require more effort to implement than others. Normalization of the RCAS FD data using a simple FP count is not realistic. A significant number of errors and changes in

the FD come from data flows and data elements, not just from functional primitives. Accordingly, the FP count requires modification.

DeMarco recommends first correcting for variations in functional primitive size. This is accomplished by using the data token count (TC) for each functional primitive as an approximation of size. DeMarco's data token is equivalent to an RCAS PSL/PSA data flow. The token count is the sum of input and output data flows. Thus, if a functional primitive has three input data flows and two output data flows, the TC is five.

The Corrected FP Increment (CFPI) for a given functional primitive is calculated using the formula

$$CFPI = 0.5 * TC * \log_2(TC)$$

The sum of all CFPIs is the Corrected FP (CFP).

The CFP can be used to normalize data. Since the number of data flows for any given functional primitive in the PSL/PSA databases varies from one release to the next, adjusting the FP count using data flows provides a better basis for comparing errors and changes in the various appendices.

DeMarco next recommends correcting for variations in complexity. This is achieved by classifying each functional primitive and then by multiplying the CFPI for that functional primitive by a weighting factor appropriate for the class assigned to the functional primitive.

Complexity is a significant factor in software development and thus is an important part of the Bang metric. Correction for complexity, however, has little value for normalizing the RCAS data. There is virtually no variation in number of functional primitives in an appendix from one release to the next. There is very little variation in function of functional primitives from one release to the next. Consequently, complexity essentially remains a

constant. No correction is made for complexity and a modified Bang metric is used.

To verify the assumption that complexity is not a major factor in normalizing the RCAS data, the functional primitives for Appendices N and H were classified. CFPIs were calculated and modified for complexity. A Bang metric was determined for the two appendices combined. The combined data for the two appendices were normalized using several techniques. Table 3 summarizes the normalization of the data using the simple object sum, comprehensive sum, FP count only, modified Bang (CFP without modification for complexity), and Bang (CFP with modification for complexity). To facilitate comparison of the different normalization methods for the various releases, the data have been scaled for each method using the maximum value for the set of releases as the divisor. Three observations are made from Table 3. First, the scaled normalized values for simple, comprehensive, and FP only are very similar. Second, the scaled normalized values for modified Bang and Bang are also very similar. Finally, the values for the first group (simple, comprehensive, and FP only) are approximately twice the value of those of the second group (modified Bang and Bang). This is especially evident in the ratios of scaled normalized changes to scaled normalized errors. Compensating for complexity does not appear to make a significant contribution to normalizing the data.

2.2.3.2.4 The Normalized Data

Appendix E contains the results of data normalization using the simple sum of objects, the comprehensive sum of objects, and the modified Bang metric. Change counts, error counts, and difference counts are normalized. The normalized data are used as input to determine if predictors of errors in the functional description can be found.

Appendices N and H	02 Nov 89	26 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
total changes	n/a	92	228	968	29	66	1056	1064
total errors	934	948	1034	66	141	110	2	0
normalized changes								
simple	0.0	0.1	0.2	0.9	0.0	0.1	1.0	1.0
comprehensive	0.0	0.1	0.2	0.9	0.0	0.1	1.0	1.0
FP only	0.0	5.8	14.3	60.5	1.8	4.1	66.0	56.0
modified Bang	0.0	1.3	3.3	14.2	0.4	0.9	15.8	32.0
Bang	0.0	2.0	5.1	21.7	0.6	1.5	25.9	47.3
normalized errors								
simple	1.0	1.0	0.9	0.1	0.1	0.1	0.0	0.0
comprehensive	1.0	1.0	0.9	0.1	0.1	0.1	0.0	0.0
FP only	58.4	59.3	64.6	4.1	8.8	6.9	0.1	0.0
modified Bang	14.3	13.6	15.2	1.0	2.1	1.6	0.0	0.0
Bang	21.0	20.9	23.1	1.5	3.2	2.5	0.0	0.0
normalized changes scaled to range from 0.0 to 1.0								
simple	0.0	0.1	0.2	0.8	0.0	0.1	1.0	1.0
comprehensive	0.0	0.1	0.2	0.9	0.0	0.1	1.0	1.0
FP only	0.0	0.1	0.2	0.9	0.0	0.1	1.0	0.8
modified Bang	0.0	0.0	0.1	0.4	0.0	0.0	0.5	1.0
Bang	0.0	0.0	0.1	0.5	0.0	0.0	0.5	1.0
normalized errors scaled to range from 0.0 to 1.0								
simple	1.0	1.0	1.0	0.1	0.1	0.1	0.0	0.0
comprehensive	1.0	1.0	1.0	0.1	0.1	0.1	0.0	0.0
FP only	0.9	0.9	1.0	0.1	0.1	0.1	0.0	0.0
modified Bang	0.9	0.9	1.0	0.1	0.1	0.1	0.0	0.0
Bang	0.9	0.9	1.0	0.1	0.1	0.1	0.0	0.0
ratio of scaled normalized changes to scaled normalized errors								
simple	0.0	0.1	0.2	13.7	0.2	0.6	492.1	n/a
comprehensive	0.0	0.1	0.2	14.7	0.2	0.6	528.1	n/a
FP only	0.0	0.1	0.2	14.4	0.2	0.6	517.0	n/a
modified Bang	0.0	0.0	0.1	6.9	0.1	0.3	250.2	n/a
Bang	0.0	0.0	0.1	7.2	0.1	0.3	258.2	n/a

Table 3. Comparison of Data Normalizations.

2.2.3.3 Analysis of the Normalized Data

The primary reason for normalizing the data is to facilitate analysis of errors and changes of the various appendices of Block 1. Changes in the FD are relatively easily detected using the CASE tools and custom written software on the host computers. One would normally expect that as the number of changes increases, the number of errors also increases. Figures 7 through 10 show the relationship between the number of normalized changes and the number of normalized errors. If there is a correlation between the number of changes and the number of errors, one should see a distinct pattern to the data points on the figures. Unfortunately, the data points appear to be randomly placed on the graph, suggesting there is no correlation between the number of errors and the number of changes in the Block 1 data.

One would also expect that as the number of processes, data flows, data stores, and data elements (i.e., the size of the appendix) increases, the number of errors per object also increases. This is based on the assumption that complexity increases with size and that complex tasks are more error prone than simple tasks. However, there appears to be no correlation between the size of the appendix and the number of errors, as shown in Appendix E (Normalized Errors). Appendices N and H, the smallest Block 1 appendices, generally show the highest error rate.

Thus there appear to be no predictors of errors. Historical information cannot be used to select areas in the FD for closer scrutiny.

2.2.3.4 FD Preparation Process Stability

All processes vary. A process is stable if the variation is random and displays no identifiable pattern of change. Random variation is usually the result of internal factors rather than external factors which can be manipulated.

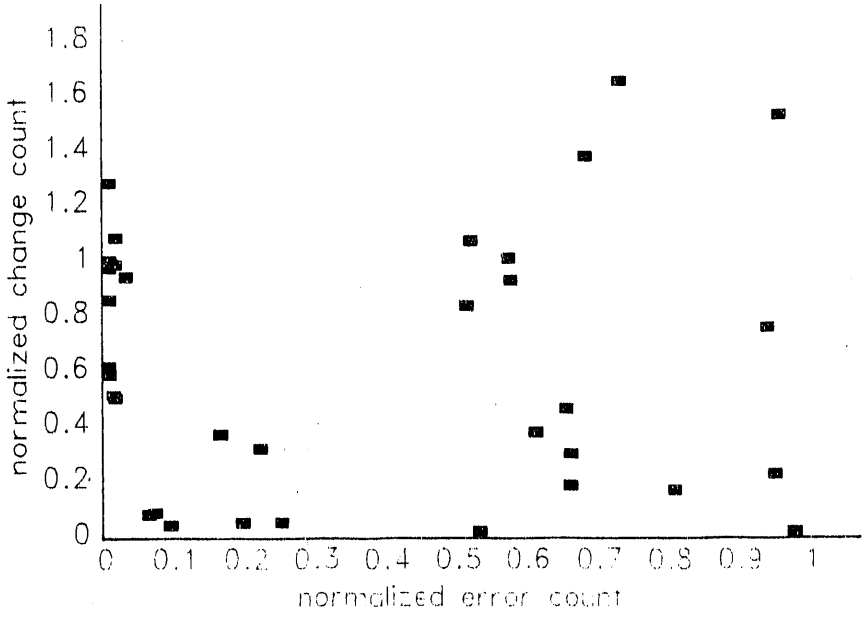

Figure 9. Errors and Changes Normalized by Simple Object Sum.

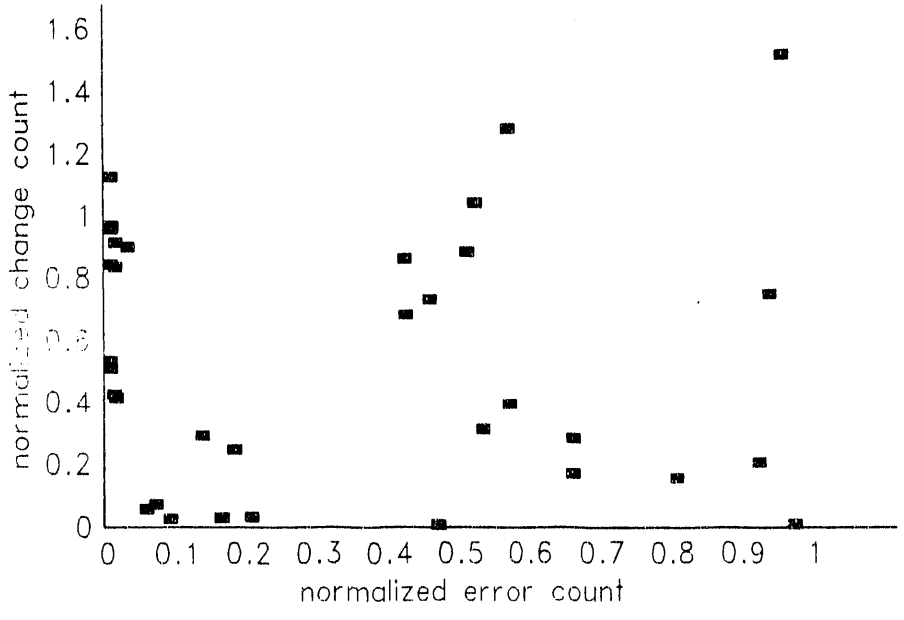

Figure 10. Errors and Changes Normalized by Comprehensive Object Sum.

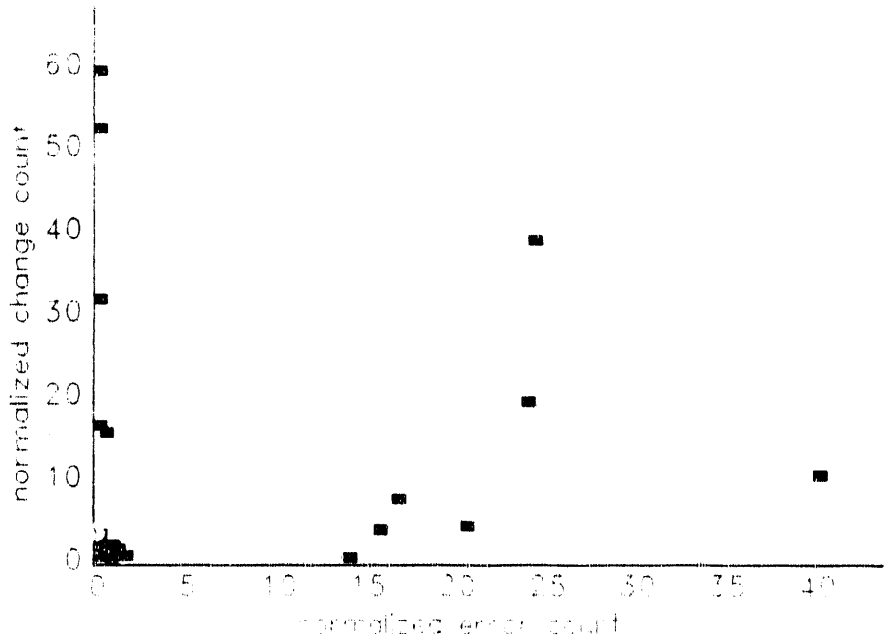

Figure 11. Errors and Changes Normalized by Modified Bang Metric.

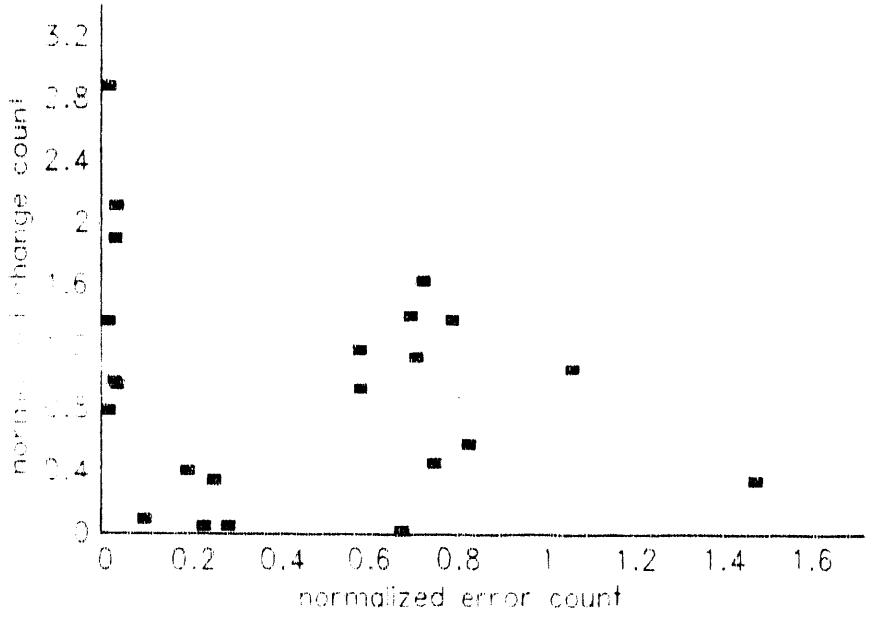

Figure 12. Errors and Changes Normalized by Modified Bang Metric (enlarged).

A stable process is required to see a correlation in the data generated from that process. Analysis of historical FD data reveals that the FD preparation process lacks stability. Statistical analysis is not used because of the small number of data points (there are only eight releases). Instead, the data is displayed graphically to illustrate trends. The lack of process stability is most clearly seen in Figures 11 through 13 which show normalized difference counts over time. One would normally expect that the number of differences from the final release would continually decrease over time. As seen in the figures, this is not the case. Lack of stability is also illustrated in Figures 14 through 16. One would normally expect that the number of changes would decrease over time as the product neared its final form. The figures illustrate that this is also not true.

One cause of instability is the change of the contractor tasked with preparing the FD. Releases prior to 06 Apr 90 were prepared by one contractor. The 06 Apr 90 and subsequent releases were prepared by the current contractor.

Another cause of instability is the change in PC-based CASE tools, from SA 1.2 to SAW. Releases prior to the 07 May 91 release were prepared using SA 1.2. The 07 May 91 and subsequent releases were prepared using SAW. The conversion from SA 1.2 to SAW was neither painless nor without error. Many errors were introduced into the FD as a result of the conversion (see Figures 17 through 19). SAW uses an entirely different structure than SA 1.2. The magnitude of the differences in the PSL/PSA database data flow relationships generated from SA 1.2 and SAW is illustrated in Table 5.

As seen in Figures 11 through 13, the 22 Dec 1990 (SA 1.2) release is more similar to the final release than is the subsequent 07 May 1991 (SAW) release. Error counts also increased from the 22 Dec 1990 release to the 07 May 1991 release (see Figures 17 through 19). Conversion from SA 1.2 to SAW is the primary cause of instability during this period.

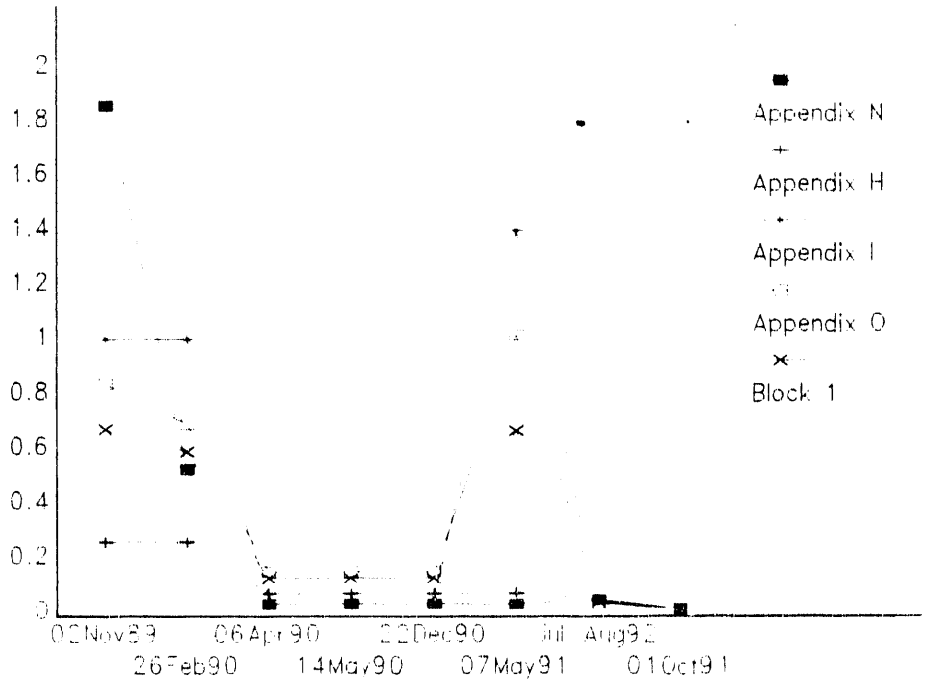

Figure 13. Differences Normalized by Simple Object Sum.

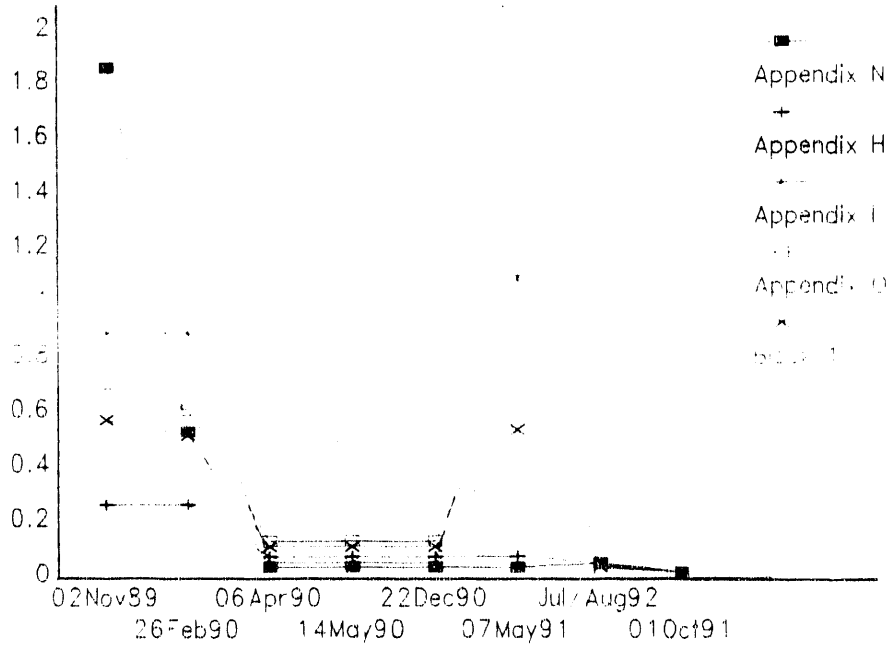

Figure 14. Differences Normalized by Comprehensive Object Sum.

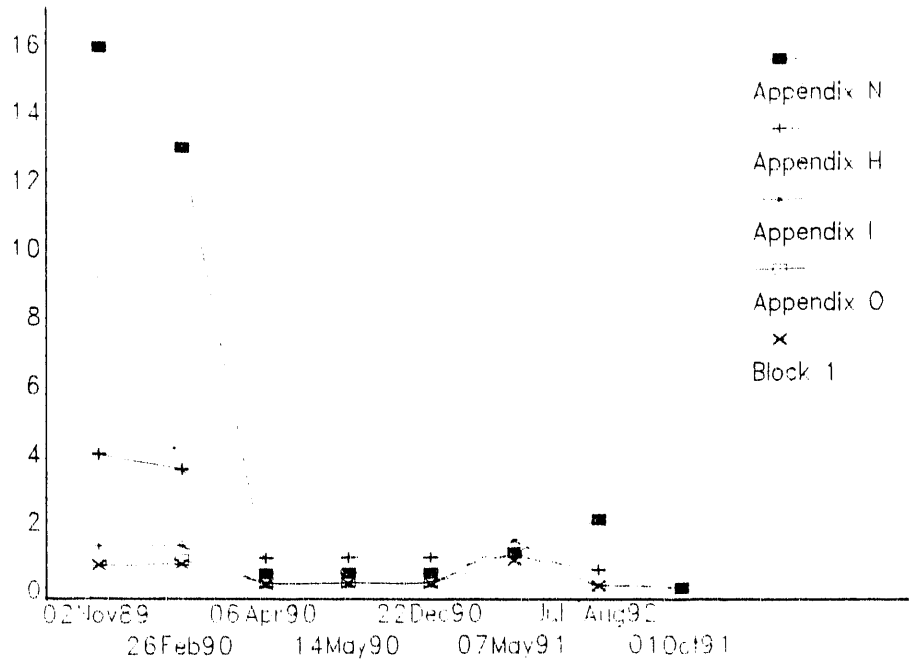

Figure 15. Differences Normalized by Modified Bang Metric.

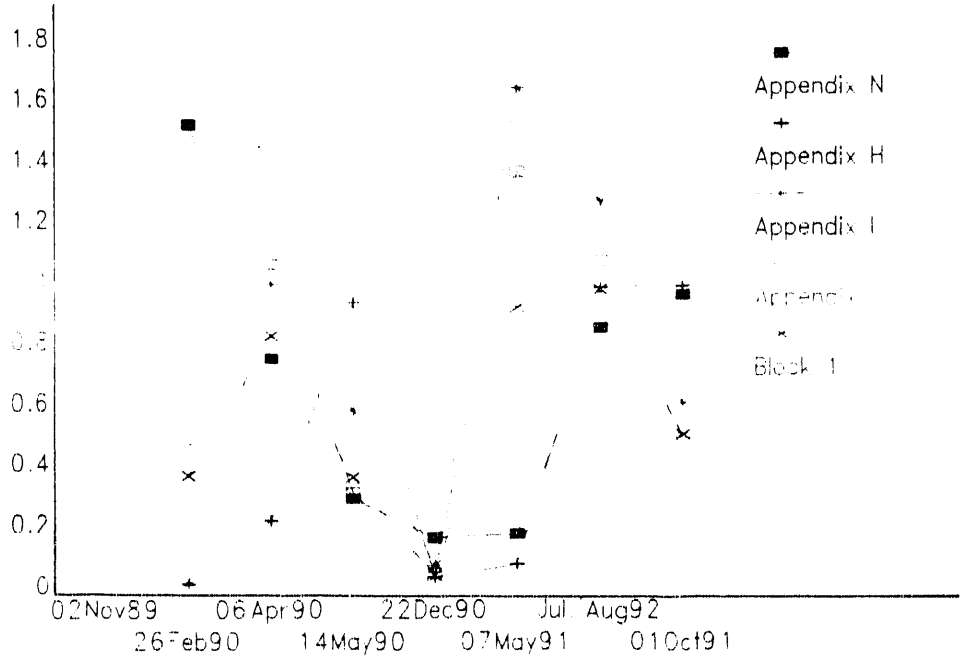

Figure 16. Changes Normalized by Simple Object Sum.

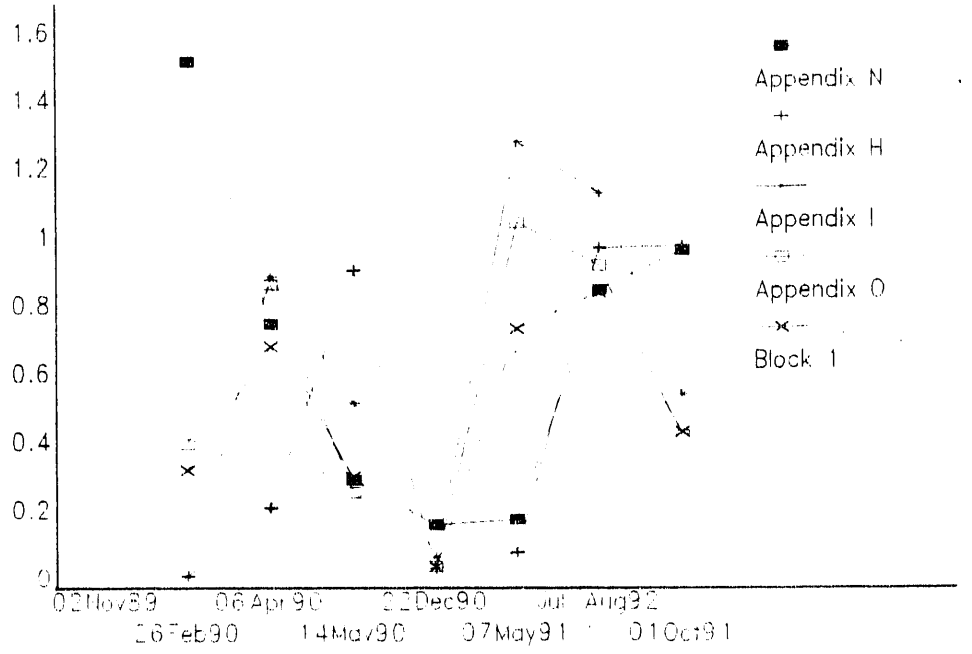

Figure 17. Changes Normalized by Comprehensive Object Sum.

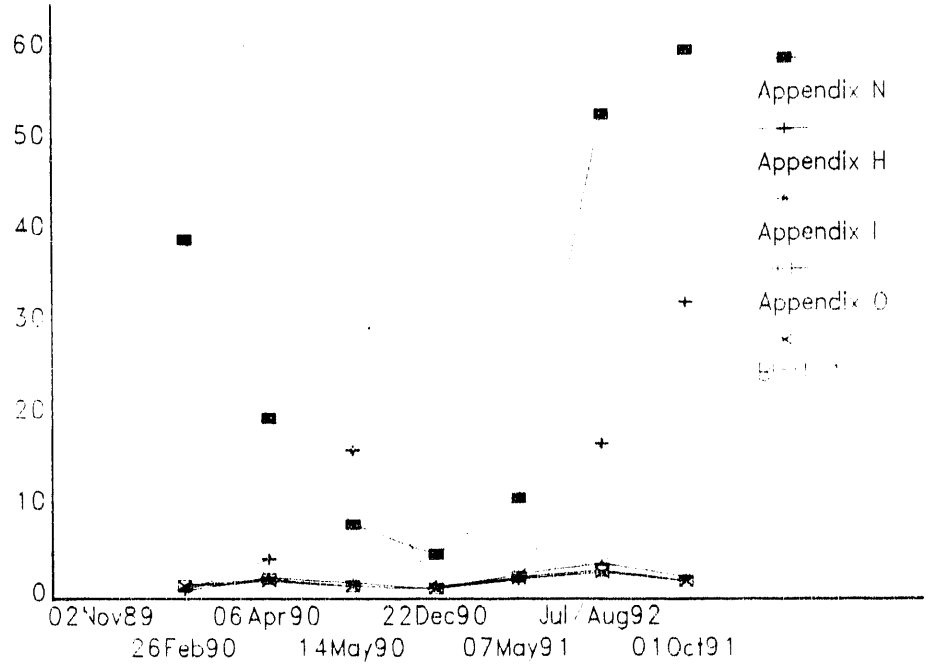

Figure 18. Changes Normalized by Modified Bang Metric.

SA 1.2	SAW
no process-to-process communication; all data pass through an intermediate data store	direct process-to-process communication
receives	(not generated)
employs	employs
generates	(not generated)
derives	derives
modifies	(not generated)
updates	(not generated)
references	(not generated)
(not generated)	sends / flows / obtains

Table 4. Comparison of SA 1.2 and SAW Generated Data Flow Relationships.

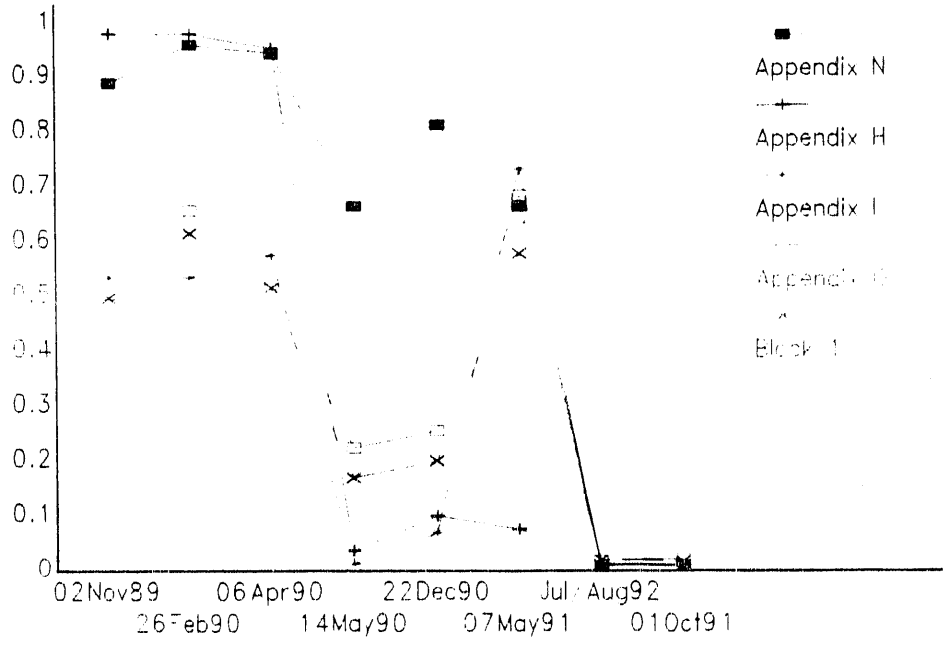

Figure 19. Errors Normalized by Simple Object Sum.

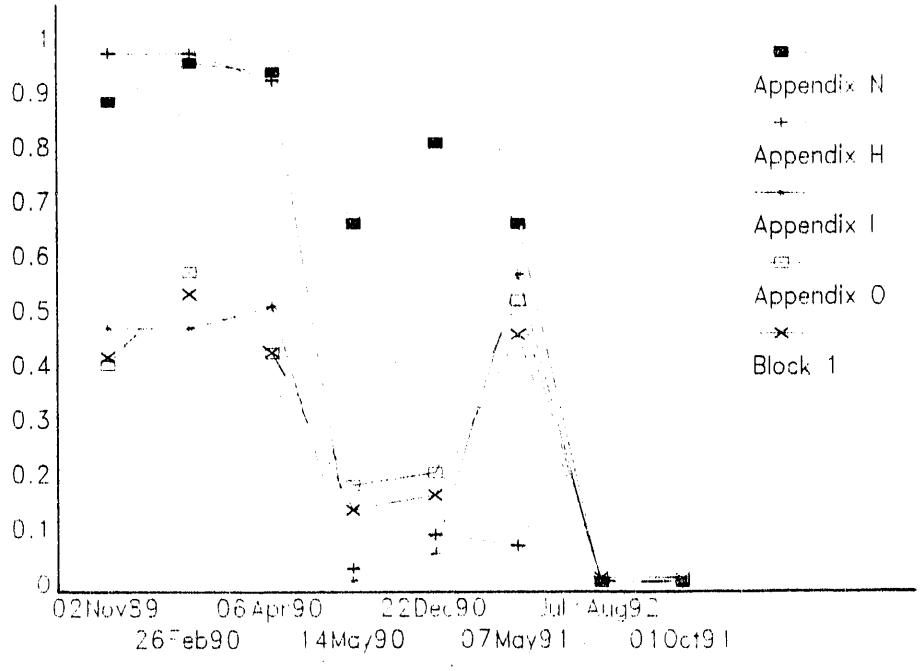

Figure 20. Errors Normalized by Comprehensive Object Sum.

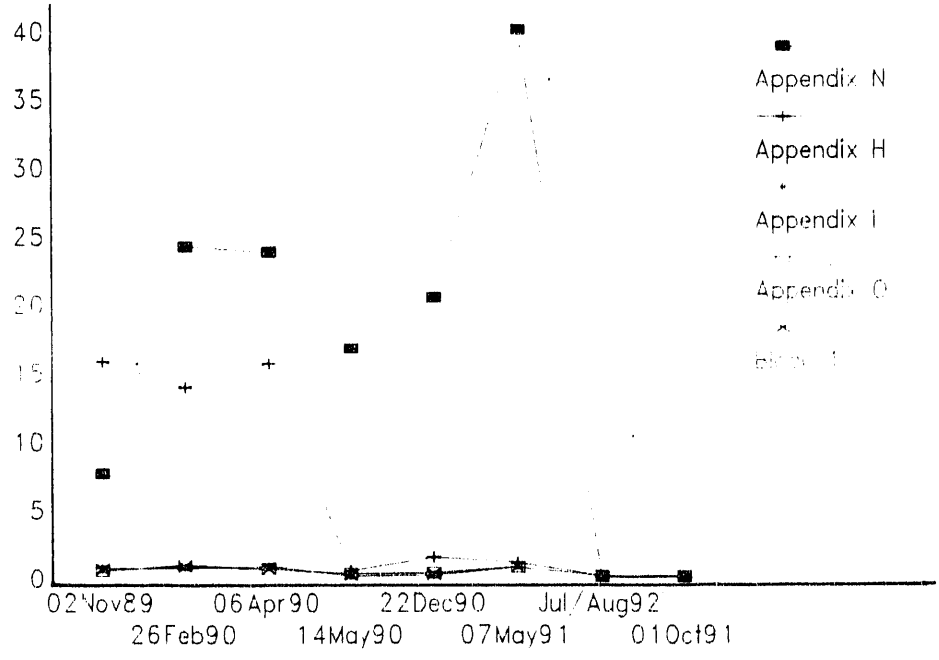

Figure 21. Errors Normalized by Modified Bang Metric.

Chapter 3
THE IV&V REVIEW PROCESS

3.1 IV&V Review Process Overview

The RCAS Program Management Office (PMO) provides the following items for IV&V review:

- Part B SAW database(s) containing processes, process descriptions, data flows, data stores, external interface information (associated with data flows), and information regarding echelon, classification, frequency, and sensitivity (associated with functional primitive processes)
- Part C PSL/PSA databases containing the same information as the SAW database(s) as well as data flow definitions (to the data element level) and data element definitions
- hard copy reports generated from the PSL/PSA databases by the preparer of the FD
- error reports based on data in the PSL/PSA databases generated by the preparer of the FD
- control reports documenting changes made to the SAW and PSL/PSA databases generated by the preparer of the FD

As shown in Appendix A, the RCAS PMO can deliver multiple databases for any given appendix. To simplify matters, the following discussion assumes that only one SAW database and only three PSL/PSA databases are delivered. The three PSL/PSA databases are:

- main database -- the contents of the SAW database
- DFD -- the data flow dictionary; data flows are defined to the data element level
- DED -- the data element dictionary; data elements are defined with a description, a long name, a data type, a length, a proponent, and an information class

3.1.1 Processing of the FD Part B SAW Database

The SAW database received from the RCAS PMO is loaded onto the host machine, a personal computer. Data flow diagrams are printed for later manual comparison to the Part A hard copy of the baseline FD.

Using a capability of SAW, PSL is generated from the Part B SAW database. The PSL is uploaded to the VAX and into a PSL/PSA database. This PSL/PSA database is created to facilitate comparison between the supplied Part B SAW database and the supplied Part C PSL/PSA main database. The comparison of the Part B and Part C PSL/PSA databases should yield no significant differences.

Selected Formatted Statements (SFS) are generated from the Part B PSL/PSA database for later comparison with the delivered Part C PSL/PSA database. Lines containing date and time information are stripped from the Part B SFS file. The date and time of processing are automatically inserted into the PSL/PSA database upon loading of the PSL. Since the Part C PSL/PSA was created at a different time, the date and time information interferes with direct comparison of information from the two databases.

3.1.2 Processing of the FD Part C PSL/PSA Databases

The PSL/PSA databases received from the RCAS PMO are loaded onto the host machine, a DEC VAX. Selected Formatted Statements (SFS) are generated from the Part C PSL/PSA databases. SFS from the main PSL/PSA database are used for comparison with the generated Part B PSL/PSA database. Lines containing date and time information are stripped from this SFS file. In addition, process synonym information is also removed from the file. Process synonyms, consisting of process numbers, are added as a post-processing step by the preparers of the FD. These synonyms do not exist in the SAW database and as such, hinder direct comparison of the information in the Part B and Part C databases.

The Part B and Part C databases are compared to verify that their content is essentially identical (see Section 3.1.3). Automatic analysis of the Part C PSL/PSA databases is performed. This entails the application of the suite of automatic checks discussed below in Section 3.4. Finally, ad hoc investigations are conducted as discussed in Section 3.5.

3.1.3 Comparison of Part B and Part C

The stripped Part B and Part C SFS files are electronically compared using the VAX/VMS difference utility. There should be no significant differences between the two since Part C should have been generated from Part B. Any differences are the result of uncontrolled editing of either Part B, Part C, or both.

Detected differences are evaluated for significance. Insignificant differences are items such as changes in textual descriptions which do not alter the meaning. For example, the deletion of blank lines in process descriptions may indicate that independent editing of Part C has occurred, but the results are not consequential. Another example of an insignificant difference is the occurrence of identical statements but in a different order. PSA retains the order in which objects and relationships are entered. If the order is different between the two databases, there was at least one independent modification to a database.

Significant differences include the presence or absence of new objects (e.g., data flows) and the modification of PSL/PSA relationships between objects. Significant differences are analyzed as errors as discussed below.

3.2 Preliminary Investigation of PSL/PSA Database(s)

3.2.1 High Level Review of Databases

The primary assumption regarding newly delivered databases is that they are significantly different from previously delivered databases. Previous analysis techniques,

both manual and automatic, may fail or, worse yet, provide invalid results. Consequently, a high level review of the recently delivered databases is necessary before any analysis can take place.

The first activity is to execute a PSA function to generate database summaries of each of the PSL/PSA databases. The database summary lists each PSL/PSA object type which occurs in the database. "New" object types, those which have not been encountered in previous releases, may require modifications to the automatic check software prior to proceeding with the automatic analysis. The new object types may also indicate the presence of errors.

Examining the database summary may quickly indicate the presence of major errors in the database. This activity may also indicate the need to modify subsequent analysis activities. For example, the 07 May 1991 release of Appendix M contained 1190 undefined objects. This was the result of an attempt to "populate" the data flows with data elements following the uploading of the Part B SAW database into the Part C PSL/PSA database. The data flows contained what were intended to be data elements, but the data elements had not been properly defined in the PSL. It became necessary to make special provision to extract the undefined objects and use them as data elements.

The next activity is to determine the set of PSL/PSA relationships for the database. "New" relationships probably require modifications to the automatic check software prior to automatic analysis. A prime example of this is found in the transition from SA 1.2 to SAW. As shown in Table 4, SA 1.2 generates a different set of PSL relationships than does SAW. Software developed to analyze PSL/PSA databases generated from SA 1.2 is not effective in analyzing databases generated from SAW.

As with new objects, new relationships may indicate errors. For example, if SAW encounters a data flow with no

origin and a data store as the destination, it generates the INCLUDE relationship in PSL. Consequently, the INCLUDE relationship is not one of the standard set of relationships one would expect to see in an RCAS FD PSL/PSA database. Its presence is an indication of an error (an unconnected data flow).

Following the high-level review of the databases, required modifications, if any, are made to the existing software used in the suite of automatic checks.

3.2.2 Simple Arithmetic and Number Comparisons

Many errors can be discovered using simple arithmetic and number comparisons. For example, the number of data flows in the main database should equal the number of data flows in the DFD file. Similarly, the number of data elements in the DFD file should equal the number of data elements in the DED file. If either of these pairs of numbers is not equal, there is a consistency error.

One example of this type of error is in the 24 Jul 1991 release of Appendix I. There are 901 data flows in the main database, but only 900 data flows in the DFD file. The missing data flow is actually defined as a data element in the DFD and the DED files. The missing data flow had been omitted from the previous release but was added to the 24 Jul 1991 release. However, the name assigned to it was identical to the name of a non-Block 1 data element (which was not included in the 24 Jul 1991 release). When the preparer of the FD extracted information from the databases of the previous release to prepare the 24 Jul 1991 release, information was extracted by name rather than by function. As discussed in Section 3.3.3.1, PSA requires inputs of name lists. Consequently, the data element information was extracted for the data flow.

Another example of an error detected using simple arithmetic and number comparisons is in the 22 Mar 1991

release of Appendix N. There are 1081 items of object type ELEMENT in the PSL/PSA database created from the Part B SAW database. There are 1049 items of object type ELEMENT in the delivered Part C PSL/PSA database. The cause of this problem was that all of the 30-character names were truncated to 29 characters. Many of these names were identical except for the thirtieth character. The difference in the counts is the result of the loss of uniqueness of these names.

A package detailing the Block 2 functional primitive processes was provided to facilitate the preparation for review of the Block 2 FD. A total of 1135 functional primitives was specified. The package included a list of functions with the number of processes for each function. This totaled to 1155 functional primitives. However, there were 1136 processes specifically enumerated in the list, two of which were erroneously duplicated, leaving a total of 1134 processes. This left the following questions to be resolved: Was the specific list of processes correct (except for the obvious error)? Was the total of 1135 correct meaning that there was only one process excluded from the list? Had 21 processes been excluded from the list and was the correct count 1155?

Simple arithmetic and comparing counts of PSL/PSA objects is an inexpensive yet powerful method in detecting errors.

3.3 Obstacles to Automatic Analysis of the Databases

The FD does not lend itself easily to automatic analysis. As explained earlier, the process used to generate the FD is not stable. Indeed, the major assumption in analyzing any given release of the FD is that there are significant differences between it and previous releases. Even appendices within a given release may be significantly different. The input to the analysis process is never the same. In addition, there may be a single database to process or there may be multiple databases (as shown in Appendix A). Several of the

obstacles to automatic analysis are discussed. These include unknown database contents, multiple databases for a given appendix, and the means of accessing data in the databases.

3.3.1 Unknown Database Contents

The contents and structure of each newly received PSL/PSA database is unknown. A given appendix may be different from other appendices. The current appendix may be different from previous releases of the same appendix. These differences may cause the software used to automatically analyze the databases to fail. For example, software created for Appendices N and H may not execute correctly against Appendices I and O because there are new PSL/PSA object types, INPUT and OUTPUT, for data flows. The Block 2 Appendix P databases also use the PSL/PSA object types INPUT and OUTPUT for data flows. In addition, whereas all other appendices use the object type ENTITY for at least some of the data flows, Appendix P does not use ENTITY at all. Instead, it uses another new object type, GROUP. Consequently, most of the software used to evaluate the databases has been modified to avoid dependence on object type.

Other problems in the databases can cause established and tested software to fail. For example, the software which creates a list of data elements associated with data flows abnormally ended. The problem was that one of the intermediate entities containing data elements also contained itself. The software, which used a recursive procedure to track through intermediate entities to data elements, became locked in an infinite loop. The program terminated after a stack overflow.

Unfortunately, because the content of the current databases are unknown and because they may be significantly different from other databases, experience from previous database investigations may not be applicable for the current databases. Many of the programs used to automatically analyze

the databases must be used to generate data for manual analysis and ad hoc investigations (discussed in Section 3.5).

3.3.2 Multiple Databases for a Given Appendix

There may be only one database or there may be multiple databases for a single appendix in a given release. In the event of multiple databases for a single appendix, it must be determined which of the databases contain data relevant to the current analysis. For example, in the 07 May 1991 release only two of the three Appendix H, 16 of the 19 Appendix I, and four of the nine Appendix O databases contain Block 1 information. Thus the software used for automatic analysis of the databases must not only make accommodations for the multiple databases, it must also make allowance for some of the databases not having any pertinent data.

The software uses lists of names (e.g., process names) as input. With multiple databases for a single appendix, the software must use either a single list of names which cover all databases or it must use multiple lists of names, one for each database. There are disadvantages to each approach.

With the single list of names approach, the software must determine if each name is in the database prior to processing the name. While this is good programming practice, the software must also <u>not</u> generate an error message for each name which is not in the database. This can result in faulty analysis, for it assumes that all of the names are present and accounted for in at least one of the databases.

With the multiple lists of names approach, provision must be made for the software to associate the correct list with the current database being analyzed. While error messages may be generated if the name is not in the database, extra work must be performed to ensure that all of the names (e.g., data flow names) are present for the entire appendix. Regardless of the approach, additional processing is required.

Special handling is also necessary if the same information is contained in multiple databases. Additional study is necessary to make sure that overlapping objects are consistently defined. For example, a parent process in one database may be a functional primitive in another database. This happens when the process is decomposed to functional primitives in one database but not in the other. Since analysis is conducted on functional primitives, care must be taken to make sure this parent process is not included in the list of functional primitives for either database.

Another problem with multiple databases is that analysis of the databases results in multiple output files, one for each database. The information from these output files must be merged into a single output file for input into other programs or to facilitate manual analysis and error reporting. Normally, this merged information must be sorted (e.g., by process number) to simplify further processing. This can cause problems if more than one line is used for each piece of output information.

Special handling is necessary when the same output information is generated from multiple databases. The output must be checked to ensure that the overlapping information is consistent.

Having multiple databases for a single appendix in one release increases the difficulty in comparing current data with data from a previous release which lacks an identical organization. This may be due to a different number of databases or to a different allocation of data to the multiple databases. The comparison of current data to previous data is necessary to determine the changes from the previous release to the current release. In this situation, one must determine where (i.e., in which database) the data item to be compared is located for each release. This adds significant complexity to processing, whether it be automatic or manual.

3.3.3 Accessing Data in the PSL/PSA Databases

3.3.3.1 Process Numbers and Process Names

Analysis of information in a database begins with processes. Functional primitives are a subset of all processes in the database. Data flows and data stores are determined by their relationships with the functional primitives. However, PSL/PSA requires process names to access related data. Block 1 (and Block 2) functional primitives are defined using process numbers and not process names. Indeed, process names change over time and as a result, one cannot search the database using the same name used in a previous release. Process numbers, on the other hand, remain fairly stable.

In the PSL/PSA database, the process number is an attribute of the process name. Given a process name, the process number can be determined. Unfortunately, the inverse is not true. Given a process number, one cannot directly access the process name or any related information.

Somehow, one must be able to cross the bridge from process numbers to process names in order to analyze the information in the database. The contractor creating the databases has attempted to do this by defining PSL/PSA synonyms. Synonyms are commonly used in the databases to associate process names and process numbers. One can directly access the process using a process number synonym. The synonym is added after loading PSL generated from Structured Architect. However, synonyms are not always used in the databases and there is no consistency as to when or where the synonyms are included.

Another complication is that process numbers may, in rare cases, change over time. In other situations, further decomposition may result in additional processes. Functional primitives may be consolidated resulting in fewer processes. Thus when given a new database, searches using process numbers may not be without error.

In order to correctly and automatically analyze the information in a newly delivered database, it is critical that the process numbers be correctly converted to PSL/PSA process names for that database.

3.3.3.2 Full Appendix or Subset

During the concentrated refinement effort of Block 1, the releases contained only Block 1 processes and related data flows, etc. In creating the suite of software for automatic analysis, one could thus make assumptions about the content of the databases which simplified the software and other aspects of processing. However, the Block 2 databases delivered for review contain information for the entire appendix and not just for Block 2 processes. The IV&V review process is restricted to Block 2 processes and their associated data flows, intermediate entities, data stores, and data elements. Assumptions about the nature of the Block 1 databases are not valid for Block 2 databases.

The unknown content of the databases makes fully automatic processing of the information in those databases impossible.

3.4 Automatic Analysis of the PSL/PSA Part C Databases

The following discussion details the processing of newly received PSL/PSA databases. To simplify discussion, the following assumptions are made. Since typically only one appendix is processed at a time, the discussion assumes that the new databases are for one appendix only. The discussion also assumes that there is only one main database for the appendix. If there are multiple main databases, the software uses separate lists of names, each list being restricted to a single database. The software also generates an output file for each database. These output files are automatically merged to create one output file for the appendix.

3.4.1 Preparation for Automatic Analysis

The following steps are necessary before executing the suite of software which performs the automatic analysis of the databases. Some of these activities have been automated and may also be considered part of the suite of software. However, their prime function is to generate information as input to additional processing.

A list of the databases for the release is created. This list contains the VAX directory name where the database is located and the name of the database. One list contains information on the main database, the database which contains processes, data flows, and data stores. A second list contains information on the DFD, data flow dictionary, database. The third list contains information on the DED, data element dictionary, database. These lists are used by all downstream processing of the databases to ensure that each and every database is processed. Using these lists accommodates those situations when there are multiple databases for a single appendix or when it is desired to process multiple appendices (such as the same appendix but from different releases).

PSA is used to create a list of all process names in the main database. A program uses this list as input and generates a list of process numbers and names. This list is manually checked to verify that all processes have associated process numbers. This list is also used to manually verify hard copy reports and for general reference.

The next program which is executed determines if an input process is a functional primitive process of interest (e.g., Block 1). A list of process numbers and names for these functional primitives is generated. A manual check is made here to verify that all functional primitives are present and accounted for.

Using the functional primitives as input, a program finds all associated data flows and outputs their names. Another

program uses the same input and finds and outputs all associated data stores. A third program uses the data flow names as input, finds all associated data elements, and outputs a list of names.

These five lists of names of processes, specific functional primitives, data flows, data stores, and data elements, are the primary inputs for automatic analysis of the contents of the databases.

3.4.2 Standard Suite of Automatic Checks for Errors

3.4.2.1 Completeness

Completeness checks consist primarily of determining if items are missing from the requirements specification. Specific completeness criteria for the RCAS FD include

- no missing functional primitive processes
- no incomplete process descriptions
- no processes lacking inputs or outputs
- no data flows without data elements
- no undefined data elements
- no missing echelon, classification, frequency, and sensitivity information

Missing functional primitive processes. The first check for completeness is to verify that all processes meeting a specified criteria, e.g. Block 1 functional primitive processes, are present in the database. A list of the process numbers of processes meeting the required criteria is generated. This is compared with a list of process numbers of processes which should be in the database and which has been previously verified for accuracy. Processes on the verified list but not in the database are considered missing. Processes in the database which meet the criteria but are not on the verified list are considered extra. Both missing and extra processes are investigated further using the ad hoc investigation process (see Section 3.5).

Some "missing" processes may actually be in the database. Extraction of processes from the database is by process number. If a Block 1 process has been defined without the PROCESS-NUMBER attribute, it is listed as a missing process even though it is in the database. Thus if a process is missing, additional checks are necessary to be sure that the process truly is missing.

One might think that selecting processes by name rather than by process number resolves the above problem. However, process names are not as stable as process numbers. Selecting processes using names rather than numbers increases the number of "misses" and thus makes the selection process more error prone. In either case, the list of missing processes must be verified for accuracy.

Incomplete process descriptions. The next check for completeness concerns process descriptions. Each process should have a description contained in a PSL/PSA DESCRIPTION text field. The first check for the completeness of the process description occurs when the PSL from the Part B SAW database is loaded into a PSL/PSA database. PSA uses semi-colons as delimiters for the description text field. If the process description contains an embedded semi-colon (which is not intended to be a delimiter), PSA truncates the description at that semi-colon. The following text will usually be meaningless to the PSA parser and error messages will be generated. Thus the error log indicates the presence of a truncated process description.

The absence of the description field indicates that the process lacks a description. However, there are occasions where the field is present, but it is empty or contains nothing but new line characters. Consequently, the contents of the field must be checked to verify that there is some text present. The meaningfulness of that text is not checked.

In addition, some appendices include additional process description text in the PSL/PSA PROCEDURE text field. A list

of these processes is generated. The information for these processes in the supplied hard copy reports is manually verified to ensure that the PROCEDURE information is included with the process description.

Processes lacking inputs or outputs. Another completeness check is to determine that each functional primitive process has inputs and outputs. Data must flow into the process, be transformed, and flow out of the process. The directional data flow relationships are examined to determine if a process has both inputs and outputs. For example, a process EMPLOYS an input and DERIVES an output. A process SENDS (outputs) data to another process which OBTAINS (inputs) the data. The data FLOWS from the first process to the second process. A process without either input or output relationships is in error.

Data flows without data elements. Data flows are checked to verify that they are "populated" with data elements. That is, they must have contents, or structure. This is determined using the PSL/PSA CONSISTS OF relationship; a defined data flow consists of data elements. A data flow is without data elements if the CONSISTS OF relationship is absent or if the data flow is not defined in the data flow dictionary.

However, a data flow may have a CONSISTS OF relationship and still not contain any data elements. A data flow may consist of one or more levels of intermediate entities which ultimately are devoid of data elements. Thus each data flow must be checked past the intermediate entities to the data element level of decomposition to determine if any data elements exist. A data flow which ultimately does not have at least one data element is in error.

Undefined data elements. The next completeness check is for data elements without complete definitions. A complete definition consists of a long name, a description, a data type, and a length. A data element lacking any one of these minimally required definitions is considered to be undefined.

The long name and the description are textual fields, just as the process description discussed above. The completeness check includes determining that the textual fields exist and that each one contains at least some text. As with the process description, there is no check for the meaningfulness of the textual content.

The data type and length are attributes of the data element, just as the process number is an attribute of the process. Unlike the long name and description textual fields, either these attributes are present or they are not. Their absence constitutes an error. In addition, the data type is expected to consist of an established code, such as AN for alpha-numeric. A data element with a TYPE attribute value which does not match any of the established codes is output as undefined. There are no specified limits for the length value.

Missing echelon, classification, frequency, and sensitivity information. Each functional primitive process must have echelon, classification, frequency, and sensitivity information. Echelon is the organizational level at which the information is processed. Classification refers to the security classification, e.g. secret, of the data being processed. Frequency indicates how often the data are received for processing. Sensitivity indicates if the data are protected by the Privacy Act. This information is encoded using the PSL/PSA REQUIREMENTS object type with the SATISFIES relationship to the process. The absence of a SATISFIES relationship indicates that the process is missing the required echelon, classification, frequency, and sensitivity information. In addition, the contents of these REQUIREMENTS fields are extracted and sorted for manual review of correctness.

Isolated (unused) objects. Typically, a full set of completeness checks would include a search for unused objects in the PSL/PSA database. These isolated objects indicate the

presence of a possible error in that the object was included in the database but erroneously not connected to any other object. However, the IV&V review of the PSL/PSA databases is limited to those processes and related objects which satisfy a specified criteria, such as being a Block 1 functional primitive process. Thus, an isolated object is by definition excluded from the review since it does not belong to a process meeting the specified criteria.

The absence of a specific check for unused objects does not negatively impact the IV&V review process. Unused objects which should be associated with a process under investigation and are not connected to that process are detected by the other completeness and consistency checks.

3.4.2.2 Consistency

There are two aspects to consistency: internal and external. Internal consistency exists when none of the information within any of the databases for a newly received appendix contradicts other data for that appendix in those databases. External consistency exists when none of the information within any of the databases for the newly received appendix contradicts the data contained in the databases for previously released appendices.

Internal consistency checks consist of external interface flow in the correct direction, data flows not in the data flow dictionary, data elements not in the data element dictionary, and uniqueness of object names. The external consistency check is verification of the external interface information in the supplied appendix as compared with the information in Appendix E of the FD.

External interface flow in the correct direction. For Block 1 processes, external interfaces are indicated in the PSL/PSA main database using the REQUIREMENTS object type with the SATISFIES relationship to the data flow. This is the same mechanism used to indicate echelon, classification, frequency,

and sensitivity information for functional primitive processes. The first character of the REQUIREMENT SATISFIES value is an I if the data flow is an input and an O if the data flow is an output. This is followed by a hyphen and the abbreviated name of the external interface. A check is made to verify that if the data flow is from an external interface, it has an I if it is an input to a process and an O if it is an output from a process. Outputs with an I indicator and inputs with an O indicator are incorrectly labeled. In addition, the names of the external interfaces are extracted and sorted for manual review of correctness.

Data flows not in the data flow dictionary. All data flows in the main database should also appear in the data flow dictionary. Data flows which do not appear in the DFD are detected during the completeness check of searching for data flows without data elements.

Data elements not in the data element dictionary. All data elements in the data flow dictionary should also appear in the data element dictionary. Data elements which do not appear in the DED are detected during the completeness check of searching for undefined data elements.

Uniqueness of object names. Object names within the appendix must be unique. For example, a data element cannot have the same name as a data store. The difficulties in this check are that the same item, e.g. a data flow, legitimately appears in more than one database and that each object name must be classified as to what it represents, e.g. a data store.

Processes and data stores should only be found in the main database. Data flows are found in both the main and DFD databases. Data elements are found in both the DFD and DED databases. Consequently, simple lists of names from each database cannot be used to check uniqueness of names.

Determining what an object represents is simple for the main database file. In the main database, an object of type

PROCESS is a process, of type ELEMENT is a data flow, and of type SET is a data store. This is even more simple for the data element dictionary. In the DED, an object of type ELEMENT is a data element. It is noted that object types are not uniquely used; object type ELEMENT is used for both data flows and data elements. The problem in classifying the objects in the databases is in the data flow dictionary. An object of type ELEMENT is usually a data element but it also might be a data flow without data elements. A data flow in the DFD may have an object type ENTITY, GROUP, INPUT, or OUTPUT. But not all objects with these object types are data flows. Some may be intermediate entities.

PSA provides the capability to extract objects by object type from a database. However, because objects of the same type are used to represent different components of a data flow diagram, comparisons of list of objects are not effective in determining uniqueness of names.

Another problem with the two possible approaches to verifying uniqueness of names discussed above is that the IV&V review is limited to Block 1 or Block 2 functional primitive processes and related data flows, data stores, and data elements. Both of the above approaches use all of the data in the databases and not the required limited portion.

The automatic check for uniqueness of names then, must take data generated during the preparatory phase of data analysis (see Section 3.4.1). In this phase, lists of functional primitive processes, data stores, data flows, intermediate entities, and data elements are created. Each of these lists are compared with the remaining lists to verify that all names are unique.

External interface information consistent with Appendix E of the FD. As discussed earlier, external interface information is associated with the appropriate data flows using the REQUIREMENT object and the SATISFIES relationship. Information is extracted from the main database and manually

compared to Appendix E. This information includes the associated process number, the direction of the data flow, and the name of the data flow. The comparison consists of two activities. The first activity verifies that all information in the PSL/PSA databases is also in Appendix E. The second activity verifies that all relevant information in Appendix E is included in the databases for the appendix being reviewed.

3.5 Ad Hoc Investigations

3.5.1 Overview of Ad Hoc Investigations

Predictors of errors in the databases could not be found (see Section 2.2.5). One reason for the lack of correlation between changes and errors is that the development of the FD is not a stable process. This lack of stability requires the use of ad hoc techniques in addition to the automatic analysis of the databases.

As the name suggests, ad hoc investigations are not directed at the appendix as a whole, at least not initially. Ad hoc techniques are used when something doesn't "look" right. That particular item is investigated until it is determined that either there is no aberration or until the probable cause of the aberration is discovered. If a cause is discovered, the ad hoc investigation may result in additional software being added to the automatic analysis of the databases.

The ad hoc investigation is essentially an intuitive process. The success of ad hoc techniques are highly dependent on the skill and experience of the investigator. A pictorial representation of the ad hoc investigation process is presented in Figure 22.

3.5.2 Ad Hoc Investigation Process

Figure 22 illustrates the ad hoc investigation process. The process typically starts with the discovery of errors, either manually or automatically. However, these are not the

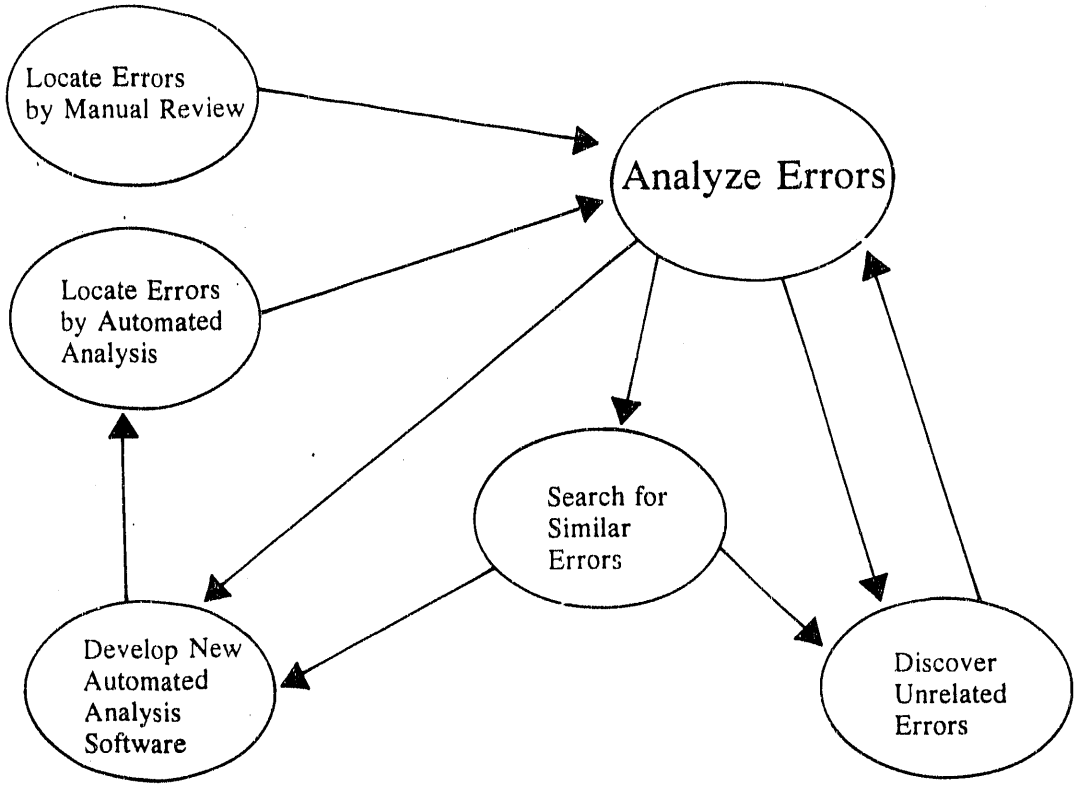

Figure 22. Ad Hoc Investigation Process.

only starting points for the process. Errors are usually located by manual review of hard copy portions of the FD and of automatically generated reports. The software used to perform automatic analysis also detects and reports errors.

Regardless of how the error was discovered, each error (or class of errors) is investigated to determine the correctness of the error reporting and analyzed to determine a probable cause of the error. Verification of error reporting correctness is necessary because of the constantly changing nature of the databases. There is no guarantee that software which was effective on the last set of databases is effective on the current set of databases. Determination of probable causes of errors is useful in gaining a greater understanding of the databases, of the process used to prepare the FD, and in discovering additional errors.

Part of the error analysis process is to search for similar errors. The techniques used in searching for like errors are the basis of algorithms for new software for automated analysis of the databases. These new analysis tools are added to the software suite. During testing and execution of the new software, additional errors may be discovered which are in turn analyzed.

Error analysis may result in the incidental discovery of new, unrelated errors. These new errors are analyzed as described above.

3.5.3 Examples of the Ad Hoc Investigation Process

The ad hoc investigation process can best be understood from examples. Four examples are provided to illustrate how the process works.

3.5.3.1 Undefined Data Element

Locate errors by automated analysis. In Block 2 Appendix P, the preparers of the FD reported 434 undefined data elements in the Data Element Definition Report. The automatic analysis of the databases discovered 435 undefined data elements. The additional undefined data element, CONUSA-ID, was a data element specified in the DFD but not included in the DED.

Analyze errors. Why did the preparers of the FD miss this data element? Part of the investigation was to check a hard copy report generated by the preparers of the FD. This report, the Data Flow Contents Report, lists all data flows and their contents to the data element level. In addition, it indicates which data elements lack complete definitions and thus serves as a double check to the Data Element Definition Report. The PSL/PSA databases were checked to determine which data flows are associated with CONUSA-ID. These data flows were then checked in the report. CONUSA-ID was not listed as a data element for any of the data flows. However, a CONUSA

was listed for each of the data flows. CONUSA is not in the DFD. The report indicated that CONUSA was a fully defined data element, which fact was confirmed by searching the DED database. It appears that CONUSA-ID and CONUSA represented the same data element.

Why then did the hard copy reports indicate CONUSA and the DFD database contain CONUSA-ID? A clue was found when the full entry for CONUSA-ID in the DFD was examined. The date of last change for the data element was 17 Dec 1991. The Data Flow Contents Report was generated on 11 Dec 1991. Thus it is probable that the name of the data element was changed after the report was generated. And why was the name of the data element changed? The name change was probably the result of the discovery of a conflict with a data store in the main database which was also named CONUSA (Block 1 Appendix O also has a data store named CONUSA). The data store was probably named CONUSA because of a change in the approach to indicating some of the external interfaces. External interfaces not explicitly defined in Appendix E of the FD are represented by a data store with the name of the external interface. For example, a process with an interactive dialogue with a user would have data flows to and from a data store named USER. CONUSA is one such non-Appendix E external interface.

Search for similar errors. Now that a probable cause has been established, similar errors are searched for. In this situation, the error is best found by searching the dates of last change. Changes made after the reports are generated may be the source of errors. In addition, one of the IV&V review tasks is to verify that the reports were generated from the supplied databases. Obviously, a report generated after the creation of the source database fails this test.

Develop new automated analysis software. Software is created to extract the date of last change for each object in each database. The dates are sorted and duplicates are removed. In addition, the frequency of each date is

determined. Frequency helps determine the extent of modifications to the database. The software is run against all of the delivered PSL/PSA databases. No new errors were discovered, but a new check has been added to the suite of automated analysis software.

3.5.3.2 Empty Descriptions

Locate errors by manual review. While looking through the hard copy of Selected Formatted Statements reports, it was noted that many processes had a DESCRIPTION keyword with which the process description should be associated. However, there was no text even though there were several blank lines. This problem had not been detected by the automatic analysis.

Analyze errors. A detailed analysis revealed that the description field contained new line characters. The description field was present and technically it was not empty. However, it contained no meaningful text.

Search for similar errors. Other occurrences of descriptions containing nothing but new line characters were found. During this process, it was discovered that the length of each of these empty text lines was one.

Develop new automated analysis software. Existing software was modified to not only search for the DESCRIPTION field but to also test it to verify that it contained something other than new line characters. This modification became part of the automated analysis software suite.

3.5.3.3 Empty Descriptions Continued

Analyze errors. While solving the above problem, it was reasoned that if process descriptions could contain only new line characters, so could two of the four minimally required definition fields of data elements. The long name and description are both textual fields used to define data elements.

Develop new automated analysis software. The same test for new line characters in process descriptions was added to existing software used to determine if data elements are fully defined.

3.5.3.4 Incomplete Process Descriptions

Discover unrelated errors. Part of the process of investigating a problem with a SAW data flow diagram is to print out an object summary of one of the processes. It was noticed that the definition of the process included a section titled "Structured English." This information has not been seen in any other appendix.

Analyze errors. Other processes in this appendix were checked and they too contained the Structured English section. The text in this section supplemented the process description by providing specific information regarding the processing of data. This information did not appear in the process description and was not included in any of the supplied hard copy reports. The question remained if exclusion of this text constituted an error.

Search for similar errors. During the manual comparison of the SAW data flow diagrams with the 04 Feb 1991 baseline data flow diagrams, it was noted that the baseline diagrams indicated that the destination of two data flows were to processes in another appendix. The SAW diagrams did not contain this information. This information was not contained in any of the hard copy reports supplied with the appendix. The information had been completely lost except for the textual description in the Structured English section. Thus omission of the Structured English text was an error because information had been lost as a result.

Develop new automated analysis software. The check for the existence of the Structured English text was added to the automated analysis software. A list of processes containing the Structured English text is generated. The information for

these reports is manually verified to ensure that this information is included with the process description.

3.6 Verifying Software Used to Analyze Databases

There is justifiable concern regarding the quality of the analysis software. The software is developed and tested on unknown inputs (the PSL/PSA databases, see Section 3.3.1). Consequently, the correct outputs are also unknown. There are three steps taken to verify the quality of the analysis software. The first is a check for the reasonableness of the output. The second is a check for consistency with previously reported data. The third is a check for consistency with data generated by other IV&V reviews.

3.6.1 Reasonableness of Output

The first check is to compare the volume of the output with what one would intuitively expect. Is the number of errors discovered reasonable? For example, if the software is to search for undefined data elements and the output contains all of the data elements in the DED, one may quickly assume that there is a major problem in the software. The same assumption is valid if no undefined data elements are reported. The problem occurs when half the data elements are reported as undefined.

3.6.2 Consistency with Previously Reported Data

The preparers of the FD generate reports specifying "known" errors to the RCAS PMO and to the proponents reviewing the FD. The purpose of these reports is to alert knowledgeable personnel who can resolve these errors. Normally, the number of reported errors is close to the number of actual errors. If the number of errors uncovered by the analysis software significantly differs from the number of reported errors, further investigation is necessary.

In addition, the results from the analysis software are compared with the reports of known errors. A reported error which is also detected by the analysis software is probably a valid error and no further verification is needed. On the other hand, each reported error not discovered by the analysis software and each unreported error discovered by the analysis software requires specific investigation to determine if the error is valid. The process of ad hoc investigation of errors (see Section 3.5.2) is followed during this error investigation.

3.6.3 Consistency with Other IV&V Reviews

The final test of the validity of the output, and hence the quality of the software generating that output, is the comparison of the output with results generated by another member of the IV&V review team. The other person searches for the same errors but uses different tools and techniques. The outputs from the two different approaches to the same problem are compared. As above, errors detected by both approaches are considered to be valid and additional verification is not necessary. Errors which are found by only one of the approaches are individually investigated. The process of ad hoc investigation of errors is followed during this investigation.

Chapter 4
CONCLUSIONS

4.1 Summary and Conclusions

This thesis examines the process of performing independent verification and validation (IV&V) reviews of databases containing a massive software requirements specification, the Reserve Component Automation System (RCAS) Functional Description (FD). The FD is the equivalent of 35,000 printed pages consisting of data flow diagrams, process descriptions, and a data element dictionary. The size of the FD mandates electronic analysis of the databases.

Analysis of the history of RCAS FD errors determined that there are no predictors of errors. A predictor would indicate where errors are more likely to occur in the FD. Those areas would be subjected to more intense scrutiny during a review. However, because the process of preparing the FD is not stable, predictors of errors within the FD do not exist.

A stable process is one which displays no identifiable pattern of change; variation is normally limited to statistically defined bounds. The FD preparation process shows significant variations as a result of changes in contractors preparing the FD and in one of the CASE tools.

The research for this thesis demonstrates that it is possible to perform an IV&V review on a massive software requirements specification for aspects of completeness and consistency. A suite of software is used to automatically inspect the contents of the FD databases.

Each new release of the FD is different from the previous release. Consequently, an ad hoc investigation process is used to supplement the automatic analysis of the FD databases. The ad hoc investigative process analyzes errors discovered by manual review and automatic processing. This analysis results in the development of new algorithms and the addition of new programs to the automatic inspection software.

The automatic analysis software enables inspecting for completeness and consistency. The work with the RCAS FD clearly indicates that the IV&V review process is not static; it must continually grow, adapt, and change as conditions warrant. The ad hoc investigation process provides the required flexibility in performing IV&V reviews.

4.2 Problems and Future Research

One problem with the ad hoc investigation process is that, as a result of the process, new software is continually added to the suite of automatic checks. There is no assessment of the value of the additional check compared with the cost to fully execute that check. Additional research is necessary to determine if there is a means of applying a cost/benefit analysis to a new automatic check.

The work of this thesis has been limited to the requirements defined in the RCAS FD. The approach taken in this thesis should be corroborated by applying the aspects of the automated analysis software and the ad hoc investigation process to work involving other large software requirement specifications. In addition, research to determine if this same approach would be effective in conducting IV&V reviews in other phases of the software development process (e.g., design) would be beneficial.

The work with the RCAS FD uses the PSL/PSA CASE tool, which was developed in the mid-seventies. Many of the newer CASE tools have built-in completeness and consistency checks. The approach used in this thesis should be applied to work using newer CASE tools to determine how best to supplement the inherent quality checks of these tools.

More research into the ad hoc investigation process would be beneficial. Understanding the process would help analysts gain skill and experience more quickly (and perhaps without as much pain). Additional research could result in improved techniques being used in the process.

REFERENCES AND BIBLIOGRAPHY

[Alfo77] Alford, M. W., "A Requirements Engineering Methodology for Real-Time Processing Requirements," IEEE Transactions on Software Engineering, volume SE-3, number 1, January 1977, pages 60-68.

[Andr86] Andriole, S. J., editor, Software Validation, Verification, Testing, and Documentation, Petrocelli Books, Princeton, New Jersey, 1986.

[Bell77] Bell, T. A., Bixler, D. C., and Dyer, M. E., "An Extendable Approach to Computer-Aided Software Requirements Engineering," IEEE Transactions on Software Engineering, volume SE-3, number 1, January 1977, pages 49-60.

[Boeh84] Boehm, B. W., "Verifying and Validating Software Requirements and Design Specifications," IEEE Software, volume 1, number 1, January 1984, pages 75-88.

[Davi77] Davis, G. G. and Vick, C. R., "The Software Development System," IEEE Transactions on Software Engineering, volume SE-3, number 1, January 1977, pages 69-84.

[DeMa78] DeMarco, T., Structured Analysis and System Specification, Yourdon, Inc., New York, New York, 1978.

[DeMa82] DeMarco, T., Controlling Software Projects, Yourdon Press (Prentice-Hall, Inc.), Englewood Cliffs, New Jersey, 1982.

[Deut88] Deutsch, M. S. and Willis, R. R., Software Quality Engineering: a Total Technical and Management Approach, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1988.

[Evan87] Evans, M. W. and Marciniak, J. J., Software Quality Assurance and Management, John Wiley & Sons, New York, New York, 1987.

[IEEE83] Software Engineering Technical Committee of the IEEE Computer Society, IEEE Standard Glossary of Software Engineering Terminology, IEEE-STD-729-1983, IEEE, New York, New York, 1983.

[Intr87] Introduction to PSL/PSA, Meta Systems, Ltd., Ann Arbor, Michigan, 1987.

[Intr89] Introduction to Structured Analysis Using Structured Architect Workbench, Meta Systems, Ltd., Ann Arbor, Michigan, 1989.

[Mapp87] Mapping and Conversion Techniques for PSL/PSA 6.0, Meta Systems, Ltd., Ann Arbor, Michigan, 1987.

[Mili88] Military Standard, DOD Automated Information Systems (AIS) Documentation Standards, DOD-STD-7935A, 31 Oct 1988.

[Nyar83] Nyari, E. and Sneed, H., "SOFSPEC: A Pragmatic Approach to Automated Specification Verification," The Journal of Systems and Software, volume 3, number 3, September 1983, pages 193-200.

[PSAR87] PSA Reports Overview, Meta Systems, Ltd., Ann Arbor, Michigan, 1987.

[PSLL87] PSL Language Structure, Meta Systems, Ltd., Ann Arbor, Michigan, 1987.

[RCAS91] Reserve Component Automation System (RCAS) Functional Description, ADSM 18-J04-HVR-XXX-FD, volume 1, Reserve Component Automation System Program Management Office, Newington, Virginia, 04 Feb 1991.

[Refe90] "Reference Guide," Report Specification Interface User Reference, Meta Systems, Ltd., Ann Arbor, Michigan, 1990.

[RoSc77] Ross, D. T. and Schoman, K. E. Jr., "Structured Analysis for Requirements Definition," IEEE Transactions on Software Engineering, volume SE-3, number 1, January 1977, pages 6-15.

[Ross77] Ross, D. T., "Structured Analysis (SA): A Language for Communicating Ideas," IEEE Transactions on Software Engineering, volume SE-3, number 1, January 1977, pages 16-34.

[Schu87] Schulmeyer, G. G. and McManus, J. I., editors, Handbook of Software Quality Assurance, Van Nostrand Reinhold Company, New York, New York, 1987.

[Snee83] Sneed, H., "Softing Software Engineering System," The Journal of Systems and Software, volume 3, number 1, March 1983, pages 63-76.

[Stru86] Structured Architect User's Guide (Version 1.2), Meta Systems, Ltd., Ann Arbor, Michigan, September 1986.

[Stru89] Structured Architect Workbench General User's Guide, Meta Systems, Ltd., Ann Arbor, Michigan, 1989.

[Teic77] Teichroew, D. and Hershey, E., "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Transactions on Software Engineering, volume SE-3, number 1, January 1977, pages 41-48.

[Wall89] Wallace, D. and Fujii, R., "Software Verification and Validation: An Overview," IEEE Software, May 1989, pages 10-17.

[YueK89] Yue, K., "Validating System Requirements by Functional Decomposition and Dynamic Analysis," Proceedings of the Eleventh International Conference on Software Engineering, May 15-18, 1989, Pittsburgh, Pennsylvania, pages 188-196.

[Zave82] Zave, P., "An Operational Approach to Requirements Specification for Embedded Systems," IEEE Transactions on Software Engineering, volume SE-8, number 3, May 1982, pages 250-269.

APPENDIX A

Database Content

Appendix N	02 Nov 89	26 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
databases in release	1	1	1	2	2	2	3	3
processes	1	1	1	1	1	1	1	1
data flows	2	2	2	2	2	2	2	2
intermediate entities	0	0	0	0	0	0	0	0
data elements	44	57	57	57	57	57	57	57
data stores	3	1	1	1	1	1	2	2
simple sum of objects	50	61	61	61	61	61	62	62
comprehensive sum of objects	50	61	61	61	61	61	62	62
modified Bang metric	5.8	2.4	2.4	2.4	2.4	1.0	1.0	1.0

Appendix H	02 Nov 89	26 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
databases in release	1	1	1	2	2	2	3	3
processes	15	15	15	15	15	15	15	18
data flows	39	39	40	40	40	56	54	44
intermediate entities	0	0	29	29	29	16	12	17
data elements	860	860	977	977	977	977	960	960
data stores	2	2	2	2	2	2	8	9
simple sum of objects	916	916	1034	1034	1034	1050	1037	1031
comprehensive sum of objects	918	918	1063	1063	1063	1066	1049	1049
modified Bang metric	56.5	64.1	62.7	62.7	62.7	66.7	62.9	32.2

Appendix I	02 Nov 89	26 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
databases in release	1	1	1	2	2	16	3	3
processes	333	333	333	333	333	333	333	333
data flows	884	884	882	882	882	883	883	883
intermediate entities	2	2	0	0	0	0	0	0
data elements	1354	1354	1596	1596	1596	0	1596	1597
data stores	5	5	8	8	8	8	15	15
simple sum of objects	2576	2576	2819	2819	2819	1224	2827	2828
comprehensive sum of objects	2919	2919	3164	3164	3164	1572	3177	3178
modified Bang metric	2037.3	2037.3	2051.0	2051.0	2051.0	1240.1	1240.8	1240.8

Appendix O	02 Nov 89	26 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
databases in release	1	1	1	2	2	5	3	3
processes	930	929	933	933	933	933	933	933
data flows	2128	2199	1844	1844	1844	2221	2174	2188
intermediate entities	5	10	0	0	0	0	0	0
data elements	2060	3375	2926	2926	2924	0	2919	2920
data stores	145	145	41	41	41	158	187	195
simple sum of objects	5263	6648	5744	5744	5742	3312	6213	6236
comprehensive sum of objects	6329	7617	7051	7051	7049	4347	7262	7286
modified Bang metric	5387.2	5283.4	5181.8	5181.8	5181.8	3262.9	3147.6	3176.9

Block 1 Appendices N, H, I, and O	02 Nov 89	26 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
databases in release	2	3	3	4	4	24	10	6
processes	1279	1278	1282	1282	1282	1282	1282	1285
data flows	3012	3082	2735	2735	2735	3120	3071	3115
intermediate entities	7	11	29	29	29	16	12	17
data elements	3286	4471	4151	4151	4149	1031	4127	4129
data stores	154	152	49	49	49	167	209	218
simple sum of objects	7731	8983	8217	8217	8215	5600	8689	8747
comprehensive sum of objects	9183	10338	9931	9931	9929	7041	10142	10167
modified Bang metric	7486.8	7387.2	7297.9	7297.9	7297.9	4570.7	4452.3	4450.9

APPENDIX B

Changes in Databases

Appendix N	26 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process names							
data flow names	4						
intermediate entity names							
data element names	75	30					
data store names	4					1	2
process descriptions	1	1			1		
data element definitions	8	14	17	9	9	51	57
total changes	92	45	17	9	10	52	59

Appendix H	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process names		4			2	2	3
data flow names		17			18	4	20
intermediate entity names		29			13	4	5
data element names		147				17	
data store names						6	7
process descriptions		15			15	15	15
data element definitions			951	20	21	960	960
total changes	0	212	951	20	69	1008	1010

Appendix I	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process names						28	
data flow names		20			1		82
intermediate entity names		2					
data element names		2418			1596	1596	1
data store names		3				7	
process descriptions		332			333	332	2
data element definitions			1595	163	65	1596	1597
total changes	0	2775	1595	163	1995	3559	1682

Appendix O	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process names	211	12				116	4
data flow names	255	449			409	159	106
intermediate entity names	5	10					
data element names	1479	3959		2	2924	2919	1
data store names	50	110			125	99	10
process descriptions	908	929			933	393	1
data element definitions	27	554	1707	172	97	2915	2867
total changes	2935	6023	1707	174	4488	6601	2989

Block 1 Appendices N, H, I, and O	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process names	209	16			2	141	7
data flow names	258	477			419	163	168
intermediate entity names	4	40			13	4	5
data element names	1383	4453		2	3115	3127	2
data store names	54	111			124	111	17
process descriptions	1241	945			1282	740	18
data element definitions	34	641	2864	218	123	4114	4073
total changes	3183	6683	2864	220	5078	8400	4290

APPENDIX C

Errors in Databases

Appendix N	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process numbers								
processes w/o descriptions		1						
processes w/o inputs & outputs								
processes w/o inputs only								
processes w/o outputs only								
data flows not in the DFD								
data flows w/o data elements								
data elements not in the DED				40				
undefined data elements	44	57	57		49	40		
total errors	44	58	57	40	49	40	0	0

Appendix H	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process numbers								
processes w/o descriptions								
processes w/o inputs & outputs								
processes w/o inputs only								
processes w/o outputs only								
data flows not in the DFD								
data flows w/o data elements	30	30						
data elements not in the DED				26	71			
undefined data elements	860	860	977		21	70	2	
total errors	890	890	977	26	92	70	2	0

Appendix I	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process numbers	1	1	1	1	1	1		
processes w/o descriptions	1	1	1	1	1			
processes w/o inputs & outputs								
processes w/o inputs only								
processes w/o outputs only	1	1	1	1	1	1		
data flows not in the DFD								1
data flows w/o data elements						883	1	
data elements not in the DED				1				
undefined data elements	1354	1354	1596		164	?		
total errors	1357	1357	1599	4	167	885	1	1

Appendix O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process numbers								
processes w/o descriptions	18	929	18	18	18	18	1	1
processes w/o inputs & outputs	3	3						
processes w/o inputs only	1	1						
processes w/o outputs only	2	4						
data flows not in the DFD								
data flows w/o data elements	425					2221	1	6
data elements not in the DED				1219	4			
undefined data elements	2060	3375	2926		1389	?	57	57
total errors	2509	4312	2944	1237	1411	2239	59	64

Block 1 Appendices N, H, I, and O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
process numbers	1	1	1	1	1	1		
processes w/o descriptions	19	931	19	19	19	18	1	1
processes w/o inputs & outputs	3	3						
processes w/o inputs only	1	1						
processes w/o outputs only	3	5	1	1	1	1		
data flows not in the DFD							15	
data flows w/o data elements	468	43				3062	2	6
data elements not in the DED				1284	75			
undefined data elements	3284	4469	4148		1471	110	59	57
total errors	3779	5453	4169	1305	1567	3192	77	64

APPENDIX D

Differences from Final Release

Appendix N	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
process names							
missing processes							
extra processes							
missing data flows	2						
extra data flows	2						
missing intermediate entities							
extra intermediate entities							
missing data elements	48	15					
extra data elements	35	15					
missing data stores	2	1	1	1	1	1	1
extra data stores	3						1
total differences	92	31	1	1	1	1	2

Appendix H	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
process names	3	3	2	2	2	1	
missing processes	3	3	3	3	3	3	
extra processes							
missing data flows	17	17	9	9	9	5	5
extra data flows	12	12	5	5	5	17	15
missing intermediate entities	17	17				5	5
extra intermediate entities			12	12	12	4	
missing data elements	132	132					
extra data elements	32	32	17	17	17	17	
missing data stores	7	7	7	7	7	7	4
extra data stores							3
total differences	223	223	55	55	55	59	32

Appendix I	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
process names	13	13	13	13	13	13	
missing processes							
extra processes							
missing data flows	50	50	41	41	41	41	41
extra data flows	51	51	40	40	40	41	41
missing intermediate entities							
extra intermediate entities	2	2					
missing data elements	1330	1330	2	2	2	1597	1
extra data elements	1087	1087	1	1	1		
missing data stores	10	10	7	7	7		
extra data stores							
total differences	2543	2543	104	104	104	1692	83

Appendix O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
process names	147	58	56	56	56	56	2
missing processes	7	8	4	4	4	4	
extra processes	4	4	4	4	4	4	
missing data flows	278	145	429	429	429	96	60
extra data flows	218	156	85	85	85	129	46
missing intermediate entities							
extra intermediate entities	5	10					
missing data elements	2201	1753	1	1	1	2920	1
extra data elements	1341	2208	7	7	5		
missing data stores	92	95	167	167	167	72	9
extra data stores	45	45	13	13	13	35	1
total differences	4338	4482	766	766	764	3316	119

Block 1 Appendices N, H, I, and O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
process names	163	74	71	71	71	70	2
missing processes	10	11	7	7	7	7	3
extra processes	4	4	4	4	4	4	
missing data flows	347	212	479	479	479	142	106
extra data flows	244	179	99	99	99	147	62
missing intermediate entities	17	17				5	5
extra intermediate entities	7	11	12	12	12	4	
missing data elements	2483	2065	2	2	2	3112	2
extra data elements	1641	2408	24	24	22	17	
missing data stores	108	110	180	180	180	85	13
extra data stores	45	45	12	12	12	35	4
total differences	5069	5136	890	890	888	3628	197

APPENDIX E

Results of Data Normalization

Normalized Changes

Appendix N	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	1.51	0.74	0.28	0.15	0.16	0.84	0.95
comprehensive sum of objects	1.51	0.74	0.28	0.15	0.16	0.84	0.95
modified Bang metric	38.33	18.75	7.08	3.75	10.00	52.00	59.00

Appendix H	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.00	0.21	0.92	0.02	0.07	0.97	0.98
comprehensive sum of objects	0.00	0.20	0.89	0.02	0.06	0.96	0.96
modified Bang metric	0.00	3.38	15.17	0.32	1.03	16.03	31.37

Appendix I	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.00	0.98	0.57	0.06	1.63	1.26	0.59
comprehensive sum of objects	0.00	0.88	0.50	0.05	1.27	1.12	0.53
modified Bang metric	0.00	1.35	0.78	0.08	1.61	2.87	1.36

Appendix O	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.44	1.05	0.30	0.03	1.36	1.06	0.48
comprehensive sum of objects	0.39	0.85	0.24	0.02	1.03	0.91	0.41
modified Bang metric	0.56	1.16	0.33	0.03	1.38	2.10	0.94

Block 1 Appendices N, H, I, and O	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.35	0.81	0.35	0.03	0.91	0.97	0.49
comprehensive sum of objects	0.31	0.67	0.29	0.02	0.72	0.83	0.42
modified Bang metric	0.43	0.92	0.39	0.03	1.11	1.89	0.96

Normalized Errors

Appendix N	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.88	0.95	0.93	0.66	0.80	0.66	0.00	0.00
comprehensive sum of objects	0.88	0.95	0.93	0.66	0.80	0.66	0.00	0.00
modified Bang metric	7.59	24.17	23.75	16.67	20.42	40.00	0.00	0.00

Appendix H	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.97	0.97	0.94	0.03	0.09	0.07	0.00	0.00
comprehensive sum of objects	0.97	0.97	0.92	0.02	0.09	0.07	0.00	0.00
modified Bang metric	15.75	13.88	15.58	0.41	1.47	1.05	0.03	0.00

Appendix I	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.53	0.53	0.57	0.00	0.06	0.72	0.00	0.00
comprehensive sum of objects	0.46	0.46	0.51	0.00	0.05	0.56	0.00	0.00
modified Bang metric	0.67	0.67	0.78	0.00	0.08	0.71	0.00	0.00

Appendix O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.48	0.65	0.51	0.22	0.25	0.68	0.01	0.01
comprehensive sum of objects	0.40	0.57	0.42	0.18	0.20	0.52	0.01	0.01
modified Bang metric	0.47	0.82	0.57	0.24	0.27	0.69	0.02	0.02

Block 1 Appendices N, H, I, and O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91	01 Oct 91
simple sum of objects	0.49	0.61	0.51	0.16	0.19	0.57	0.01	0.01
comprehensive sum of objects	0.41	0.53	0.42	0.13	0.16	0.45	0.01	0.01
modified Bang metric	0.50	0.74	0.57	0.18	0.21	0.70	0.02	0.01

Normalized Differences

Appendix N	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
simple sum of objects	1.84	0.51	0.02	0.02	0.02	0.02	0.03
comprehensive sum of objects	1.84	0.51	0.02	0.02	0.02	0.02	0.03
modified Bang metric	15.86	12.92	0.42	0.42	0.42	1.00	2.00

Appendix H	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
simple sum of objects	0.24	0.24	0.05	0.05	0.05	0.06	0.03
comprehensive sum of objects	0.24	0.24	0.05	0.05	0.05	0.06	0.03
modified Bang metric	3.95	3.48	0.88	0.88	0.88	0.88	0.51

Appendix I	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
simple sum of objects	0.99	0.99	0.04	0.04	0.04	1.38	0.03
comprehensive sum of objects	0.87	0.87	0.03	0.03	0.03	1.08	0.03
modified Bang metric	1.25	1.25	0.05	0.05	0.05	1.36	0.07

Appendix O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
simple sum of objects	0.82	0.67	0.13	0.13	0.13	1.00	0.02
comprehensive sum of objects	0.69	0.59	0.11	0.11	0.11	0.76	0.02
modified Bang metric	0.81	0.85	0.15	0.15	0.15	1.02	0.04

Block 1 Appendices N, H, I, and O	02 Nov 89	28 Feb 90	06 Apr 90	14 May 90	22 Dec 90	07 May 91	03 Jul 91
simple sum of objects	0.66	0.57	0.11	0.11	0.11	0.65	0.02
comprehensive sum of objects	0.55	0.50	0.09	0.09	0.09	0.52	0.02
modified Bang metric	0.68	0.70	0.12	0.12	0.12	0.79	0.04

END

DATE FILMED

8/31/92