IDA PAPER P-2494

STRATEGY AND MECHANISMS FOR ENCOURAGING REUSE IN THE ACQUISITION OF STRATEGIC DEFENSE INITIATIVE SOFTWARE

James Baldo
Craig A. Will

Dennis W. Fife, *Task Leader*

June 1990

Prepared for
Strategic Defense Initiative Organization

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

IDA Log No. HQ 90-035639

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts (a) to record substantive work done in quick reaction studies, (b) to record the proceedings of conferences and meetings, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract MDA 903 89 C 0003 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

This Paper has been reviewed by IDA to assure that it meets high standards of thoroughness, objectivity, and appropriate analytical methodology and that the results, conclusions and recommendations are properly supported by the material presented.

Approved for public release, unlimited distribution: 12 July 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1990	Final

4. TITLE AND SUBTITLE
Strategy and Mechanisms for Encouraging Reuse in the Acquisition of Strategic Defense Initiative Software

5. FUNDING NUMBERS
MDA 903 89 C 0003

Task T-R2-597.2

6. AUTHOR(S)
James Baldo, Craig A. Will

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses (IDA)
1801 N. Beauregard St.
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Paper P-2494

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Strategic Defense Initiative Organization (SDIO)
Room 1E149, The Pentagon
Washington, D.C. 20301

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, unlimited distribution: 12 July 1991.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

This paper addresses nontechnical barriers inhibiting the insertion of software reuse technology for the Strategic Defense Initiative (SDI). The paper provides a brief overview and definition of software reuse and its benefits, and an assessment is presented of the current state of practice of software reuse. Based on this foundation, factors currently inhibiting software reuse are identified and a strategy for encouraging reuse within the SDI program is presented. Nontechnical barriers that are addressed include the lack of financial incentives, organizational difficulties, and the lack of specific contractural mechanisms. Mechanisms for implementing the software reuse strategy are realized in the form of additions and changes to the Request for Proposal (RFP) and contract language. Incentives for software reuse and guidelines for evaluating contractor proposals with respect to reuse are also discussed.

14. SUBJECT TERMS
Software Reuse; Contracts; Strategic Defense Initiative (SDI); Federal Acquisition Regulation (FAR); Defense Federal Acquisition Regulation Supplement (DFARS)

15. NUMBER OF PAGES
74

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

IDA PAPER P-2494

STRATEGY AND MECHANISMS FOR ENCOURAGING REUSE IN THE ACQUISITION OF STRATEGIC DEFENSE INITIATIVE SOFTWARE

James Baldo
Craig A. Will

Dennis W. Fife, *Task Leader*

June 1990

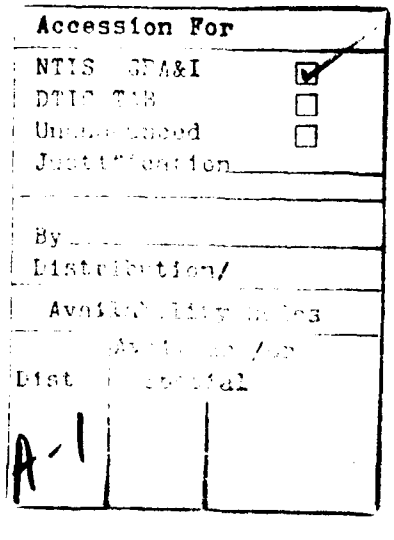
INSTITUTE FOR DEFENSE ANALYSES
Contract MDA 903 89 C 0003
Task T-R2-597.2

PREFACE

The practice of reusing computer software, in which software developed for one system is reused in another software system, has the potential to significantly reduce the cost of software and increase its reliability. There are, however, significant barriers to reuse resulting from nontechnical factors such as lack of financial incentives, organizational difficulties, and lack of specific contractual mechanisms. This paper is intended to provide the Strategic Defense Initiative (SDI) community with appropriate background to understand these issues and recommendations that, when implemented, can be applied to help encourage reuse.

This paper was prepared for the Strategic Defense Initiative Organization (SDIO) by the Institute for Defense Analyses (IDA). It responds to the requirement of Task Order 597.2, SDIO Software Technology Plan, in "recommending changes to the acquisition of SDIO reusable software and describing modifications to the Request for Proposal (RFP) process, contact language, and evaluation of proposals."

The paper was reviewed by the following members of IDA: Mr. Herbert R. Brown, Mr. Michael S. Nash, Dr. Marko Slusarczuk, and Dr. Richard L. Wexelblat.



SUMMARY

INTRODUCTION

This paper describes and defines software reuse and its benefits, discusses factors currently inhibiting software reuse, and presents a strategy for encouraging reuse within the Strategic Defense Initiative (SDI) program.

There is little agreement on what constitutes software reuse. Some definitions include any way in which information developed for one system is reapplied to another. Others are more restrictive, allowing, for example, only the reuse of software that was specifically designed to be reused. Software reuse is defined for this paper as the process of creating and using representations at any level (specifications, design, or code) that are designed in such a way as to be general enough to be effectively reused in developing a software system.

Software reuse has many potential benefits for the development of software for SDI. The principal benefits are (1) increased productivity of developers and (2) increased quality of design, code, and documentation. These principal benefits can lead to others such as cost savings, reduced development time, and increased reliability.

Although many of the benefits of software reuse can be achieved now, its full potential will only be realized when certain technical barriers have been breached. Work is particularly needed in the areas of domain analysis, reuse libraries, software componentization, and others.

Nontechnical problems inhibiting software reuse are the primary focus of this paper. They are poor organizational structures, financial disincentives, and the lack of specific contractual mechanisms.

Poor organizational structures. Software reuse can be practiced in four ways: (1) Reuse at the project level within a particular company; (2) Reuse across projects within a particular company; (3) Reuse across different companies working on a team project; and (4) Reuse of standard components as is the case with hardware. At the present time, nearly all reuse is done within a particular company and is relatively unplanned. While it is expected that large-scale reuse and its accompanying benefits will only occur when reuse occurs across different companies and is more carefully planned, this will require increased management commitment and agreement on approaches to and standards for reuse.

Financial disincentives. A widespread perception among contractors is that there are significant financial disincentives to software reuse. For example, contractors have as their principal asset expertise in an application area and the ability to apply this expertise by developing software

for a particular system. If contractors are required to deliver their expertise to the government in the form of reusable software components, they can lose their competitive advantage. To avoid such disincentives, mechanisms need to be developed that can provide net positive incentives for reuse.

Lack of specific contractual mechanisms. Specific contractual mechanisms that encourage reuse do not presently exist within SDI and the services, and some practices, such as the tendency to demand excessive data rights, actively discourage reuse. In the short term, approaches are needed that can apply existing mechanisms, such as award fees, to provide incentives for reuse. In the long term, it may be necessary to modify federal acquisition regulations, for example, to allow contractors to retain more data rights to reusable components.

RECOMMENDATIONS

Strategy for Reuse in SDI. The software reuse strategy that we propose for SDI includes the following recommendations:

- Initiate pilot programs to demonstrate the feasibility of potential reuse practices and to gather empirical data on SDI reuse opportunities.
- Organize a Strategic Defense Initiative Organization (SDIO) software reuse office to establish and advocate a software reuse policy across all programs.
- Develop a Strategic Defense System (SDS) software reuse operational plan based on active participation of the SDIO office, Service, and element program offices and contractors.
- Develop economic incentives for software reuse in contracts.
- Make use of software reuse technologies and management approaches from other DoD programs, and develop joint efforts with these programs.
- Adopt a software reuse cost model.
- Investigate the proper role of domain analysis.

Implementation Mechanisms. The following implementation mechanisms can help implement the proposed reuse strategy:

- Require offerors bidding on a contract to address software reuse in the description of their technical expertise, approach, and management in their proposal.
- Require contractors to have a Software Reuse Plan as part of the Software Development Plan required by DOD-STD-2167A.

- Include software reuse considerations in the evaluation criteria for a request for proposal.
- Use a performance-based award fee to reward successful reuse or the adoption of reuse methodologies.
- Allow contractors to gain certain rights to software developed with government funds as a means of rewarding successful reuse.

TABLE OF CONTENTS

1. INTRODUCTION

The reuse of software in constructing the large-scale software systems that will be required for the Strategic Defense System (SDS) offers the potential of increasing productivity in building parts of the system and increasing the quality of the system. These increases in productivity and quality can result in cost savings, reduced development time, higher system reliability, and other benefits. The concept of software reuse includes the use of existing software in building new systems as well as the construction of new software that is designed to be reusable.

The goal of Institute for Defense Analyses (IDA) Paper P-2494, Strategies and Mechanisms for Encouraging Reuse in the Acquisition of SDI Software, is to provide the necessary information to allow contracts to be written that encourage the reuse of software for SDS. The paper consists of an introduction to the technology of reuse, its current state of practice, strategy for encouraging reuse, specific implementation language for Requests for Proposals (RFPs) and contracts, and guidance for proposal evaluation.

Although the authors of this paper acknowledge that major technical problems must be solved before the full potential of software reuse can be realized, we feel that some benefits can be obtained now. However, software reuse is difficult because of such nontechnical factors as organizational structures, financial disincentives, and lack of specific contractual mechanisms that allow and encourage reusable software. This paper attempts to address these problems and to provide potential solutions that will allow software reuse to benefit Strategic Defense Initiative (SDI) software development.

2. SCOPE

This paper is intended to provide software managers and contracts personnel in the SDI program with information to help them expand the use of software reuse in procuring software. The paper describes and defines software reuse and its benefits, discusses factors currently inhibiting software reuse, and presents a strategy for encouraging reuse within the SDI program. It presents specific ways of implementing software reuse, including additions and changes to RFPs and contracts, and guidelines for evaluating contractor proposals with respect to reuse. The paper was developed from a review of the unclassified technical and management literature on software reuse, an analysis of current RFPs, contracts, relevant Federal Acquisition and Defense Regulations, the SDI Software Policy, relevant standards (e.g., DOD STD-2167A, Defense System Software Development), and interviews with contracts personnel in the Department of Defense (DoD) and in the Federal Government.

3. SOFTWARE REUSE BACKGROUND

3.1 POTENTIAL BENEFITS OF SOFTWARE REUSE

A wide variety of benefits have been claimed by the software reuse community :

a. Increased productivity

b. Increased reliability

c. Higher quality design and code

d. Decreased development time

e. Reduced need for testing

f. Cost savings

g. Reduced maintenance

h. Lower risk

i. An ability to build larger, more complex systems

These claims can be difficult to sort out because they are interrelated. In fact, most of these claimed benefits result from two principal factors: (1) increased productivity (of designers and implementors) and (2) increased quality (of design, code, and documentation). Increased productivity of personnel is perhaps the most common claim, and one that, if true, can legitimately result in cost savings, decreased development time, and an increased ability to build larger, more complex systems. Higher quality of code, design, and documentation generally can result in a

reduced need for testing, reduced maintenance, higher reliability, lower risk, and an increased ability to build larger, more complex systems. A reduced need for testing and reduced maintenance can also result in cost savings.

These claims should be viewed with skepticism, as we will discuss in more detail in Section 3.4 (Reuse in Practice). There is little clear evidence that any or all of these benefits can be predictably achieved for any given software project. Much of the available data is either for software developed under conditions that may not be easily replicated elsewhere, for specific applications that may not be similar to SDI problems, or for small-scale tests that may not be applicable to major industry software efforts. Data from many industry efforts is not available or is incomplete for competitive reasons. For any given software project, attempts to reuse code in an inappropriate way may well have a negative effect.

3.2 WHAT IS SOFTWARE REUSE?

Software reuse is not a new idea. In one form or another, it has been practiced almost since the beginning of computing, by using subroutine libraries.

The modern move to reuse has frequently been attributed to a challenge posed in an address by M. D. McIlroy at a 1968 NATO Software Engineering meeting in which he pointed to the success of the hardware components industry (which was beginning to produce standard components in the form of digital electronic chips) (McIlroy, 1969). McIlroy asked why an analogous industry could not be developed around software components. The extent to which this analogy is appropriate has been the subject of considerable debate, as has been the reason why, more than two decades later, such an industry has not developed.

One of the difficulties in understanding reuse is the very different ways that software reuse has been defined, with experts failing to agree on any one definition. These definitions typically reflect differences in what contexts are considered legitimate and in what level of representation (from abstract knowledge to specific program code) are appropriate for reuse.

For example, the following definition takes a broad interpretation of the allowable context for reuse:

"Software reuse is the reapplication of a variety of kinds of knowledge about one system to another similar system in order to reduce the effort of development and maintenance of that other system." (Biggerstaff, 1989, p. xv)

Another definition, however, strictly limits the context: "Software reuse, to me, is the process of reusing software that was designed to be reused. Software reuse is distinct from software salvaging, that is reusing software that was not designed to be reused. Furthermore, software reuse is distinct from carrying over code, that is reusing code from one version of an application to another." (Tracz, 1989, p. 36-46)

The appendix contains a list of definitions of reuse that indicate the diversity of ways that different experts view the concept of reuse.

From these definitions, the following conclusions can be drawn:

a. The present interest in and emphasis on reuse is a result not of any specific technical achievement, but instead simply reflects a way of thinking that accepts reuse as potentially important and as feasible. This view is moving toward considering all entities involved in developing a product—whether they are specifications, requirements, source code, object code, documentation, etc.— as potentially reusable *objects*. Most experts assume that, in most cases, successful reuse of a component will require modification of that component.

b. Experts have been concerned with whether truly reusable software must be specifically designed for reuse; some disparage the practice of using any unplanned reuse, terming it *software salvaging*. (Tracz, 1989, p. 36)

Because there is no generally accepted definition for software reuse, we felt it necessary to provide the reader with such a definition designed with the SDS situation in mind. In our definition of reuse, we take a broad interpretation of the allowable level of reuse—it can be design,

code, specifications, etc.—but a narrow interpretation of the allowable context: Software reuse is the process of creating and using representations at any level (specifications, design, or code) that are designed in such a way as to be general enough to be effectively reused in developing a software system.

3.3 SOFTWARE REUSE IN THE DEVELOPMENT CYCLE

Until recently, software reuse has occurred in a relatively unplanned, idiosyncratic fashion. Despite the debate over just what practices might be considered legitimate "reuse," it is fair to say that the greater the extent to which reuse is planned, rather than accidental, the greater the benefits that will be achieved and the lower the risk.

Typically, a more sophisticated attempt to apply software reuse involves four activities: domain analysis, component and tool construction, system construction, and development of a reuse library. The order that these activities are carried out can vary considerably depending upon the situation, as discussed in the following paragraphs.

Domain analysis involves the detailed study of a particular application problem and how solutions to it can be developed for implementation with a computer. Most domain analysis today is done relatively informally by consultation with experts in narrow and relatively well understood domains. However, it is widely agreed that the broader and more detailed the domain analysis, the easier it will be to find commonalities among different problems that allow the construction of more general software components that can be reused. Thus, it is usually recommended that a broad, extensive domain analysis be done at the beginning of a software development effort. The usual barrier to this is that an extensive domain analysis is very costly, an initial investment that is often hard to justify.

Component and tool construction involves the design and implementation of software components and whatever supporting tools are required to aid in their development. The term *component* is frequently interpreted broadly and can refer to high-level software designs and test

plans as well as source code. There is some overlap between domain analysis and component implementation, in that implementation issues must be considered in performing an effective domain analysis.

System construction involves the development of a system, using components that are newly designed with a system in mind (but also intended for reuse elsewhere) as well as components that may have been used elsewhere and that may well need modification.

The *development of a reuse library* involves the creation of a central source of information about what software components are available, their capabilities, restrictions on their use, and how they can be obtained. Such a library, also known as a repository, has some way of aiding search for a desired component. This is generally done with relatively simple techniques, such as keyword matching. More sophisticated systems might make use of a taxonomy or categorization scheme developed for a particular broad application area as the result of a domain analysis. A library may contain the actual software components and their associated documentation for retrieval by the user, or it may simply provide information about where they can be obtained.

How these different activities relate to each other depends on the nature of the system being built, the extent to which reuse is planned, and whether reuse is done primarily in a small group of developers or is attempted across a large project involving many different contractors. A small effort carried out within a single group can effectively reuse software even with a relatively informal domain analysis. Obtaining effective reuse in a large, complex, multi-organization project, however, will require an increased formality of domain analysis, a more organized way of planning component, tool, and system construction, and a reuse library.

3.4 REUSE IN PRACTICE

We have observed that software reuse is practiced in four ways:

a. Reuse at the project level within a company. Project team members share source code, designs, test data, etc. Reuse objects are usually maintained at the level of the individual or project. Usually, the reuse is not planned or explicitly noted in customer's contract, since the reuse occurs after the contract award. The software reuse objects have generally not been developed specifically for reuse and may require modification.

b. Reuse across projects within a company. Project team members working on different projects share source code, designs, test data, etc. If existing software was identified for reuse before proposal submission, the contract may identify the reuse.

c. Multi-contractor reuse on a project. Contractors teamed to build a system may reuse software from different team members. This type of reuse more than likely would be planned, which would be explicitly noted in the contract.

d. Reuse component industry. Contractor purchases components (usually source code) from a vendor for use on a project. The cost of the component and any necessary modification would be explicitly noted in the contract.

Of the four types of reuse listed, *a* and *b* are practiced the most. Reuse at the project level within a company is generally not planned, done on a large scale, or based on any formal methodologies. Although the benefits in productivity and reliability can have noticeable effects, it will not be able to provide the type of savings and improvements that future DoD systems will require. For example, the need for domain analysis in applying software reuse is critical for maximizing reuse. This type of reuse also tends to be narrow. It is generally restricted to team members and depends on their level of expertise and experience. Management generally does not actively participate in the reuse activities.

Reuse across projects within a company is usually applied when commonalities on different projects are easily identified. For example, two projects might require communication subsystems that are identical. However, more subtle reuse. such as specialized sorting routines, are more difficult to identify and make available.

On a project and across projects within a company reuse generally occurs without much management effort or corporate investment. Planned reuse, however, requires a firm corporate commitment that consists of a financial investment and change in organizational structure to incorporate software reuse. Reuse applied on a multi-contractor project will require strong management involvement and agreed upon technical approach(s) for software reuse. In the past, contractors have been resistant about sharing valuable software without strong data rights protection. Contractors are also hesitant about using software that was developed by another contractor. These issues are strong inhibitors for practicing reuse of this type in the current contract and corporate environments.

3.5 Research Issues and Activities

Although many benefits of software reuse are achievable now, its full potential will only be realized when the technology for reuse is developed more fully by research.

We see seven significant technical areas that are being (or should be) investigated, with progress in any of the areas likely to result in enhanced reuse capabilities:

a. Improved methods for domain analysis.

b. Improved indexing and retrieval systems for reuse libraries.

c. Improved conceptual understanding and representations for reuse.

d. Methods for raising the assurance that software performs as expected.

e. Reuse methods that take into account the fact that software not only carries out a functional task but does so with certain resource utilization characteristics.

f. Techniques for managing the increased number of parameters that are required for large components.

g. Improved software tools for reuse.

Each area is discussed in further detail in the following sections.

3.5.1 Domain Analysis

Methods that can help identify commonalities in related application problems that can lead to reusable components are badly needed. These methods include better theoretical approaches to understanding problem-solving, and software tools that aid in domain analysis. It is likely that research in expert systems and artificial intelligence in the areas of knowledge acquisition and knowledge elicitation, in automated knowledge acquisition tools, and in cognitive science approaches to understanding the mechanisms of human problem-solving can help in those areas where expert human problem-solving is to be encoded into software.

There are several efforts under way aimed at improving domain analysis. Biggerstaff has been investigating what information is best captured during a domain analysis and what methods are best at capturing it. He has built several tools to prototype different approaches to domain analysis (Biggerstaff, 1989). In addition, the Software Engineering Institute has used the Common Ada Missile Packages (CAMP) products to investigate domain analysis (McNicholl, 1988). SofTech is also including domain analysis capabilities into their Reusable Ada Packages for Information Systems Development (RAPID) Ada component library.

3.5.2 Indexing and Retrieval Systems

In large-scale software reuse, especially across different organizations, the ability to find components that might be useful is critical. Current systems make use of approaches borrowed from the indexing and retrieval of textual documents, including keyword search strategies and classification taxonomies.

Software classification schemes provide a basis for a library catalog that will allow users to locate information they need. In fact, several classification schemes providing cross-indexes to available reusable objects may be essential to find and evaluate reusable software components. Proposed schemes include functional classifications, code taxonomies, and type hierarchies. None of these schemes, however, has been tested in prototype catalogs for full-scale libraries.

Two types of automated search techniques are immediately available for libraries: database query systems and literature search systems. Database query systems respond to specific questions that have concrete answers derived from stored data. Literature search systems, often based only on keywords in abstracts, provide approximate searches that may not find all references on a topic and sometimes return unrelated references. Approximate searches are useful when there is not enough information to formulate precise queries. Natural language query systems have been demonstrated for narrowly defined application areas and should be considered for future repository access.

Other approaches that are becoming popular include hypertext browsing and searching systems (Latour, 1988), and object-oriented hierarchies. Proponents of object-oriented hierarchies tend to support a different philosophy in which reuse occurs as a natural result of dynamic interaction with an object-oriented system, rather than as a result of searching a separate library system for a specific desired component.

3.5.3 Conceptual Understanding and Representation

The problem of breaking a problem down into components and how components are best represented is important to developing reuse technology.

There is controversy about what entities or levels of representation are the most appropriate for definition as a component. Biggerstaff and Richter, for example, have argued that designs rather than implementations are the best level of abstraction for reuse (Biggerstaff, 1987). This is because implementations involve details that are specific to the application problem or target machine, limit the generality of the component, and inhibit its reuse. However, Booch (1987) has argued that to make reuse of components truly attractive, implementation issues must be addressed. In addition, Edwards and Baldo (1990) have argued that the implementation level must be considered because the major savings resulting from reuse occur during maintenance and reuse of designs alone will not provide the desired savings during the maintenance phase.

One approach to understanding how software can be divided into components has been to develop a formal model for software reuse. This approach was taken at the Reuse in Practice workshop (Baldo, 1990), where the beginnings of such a model, the 3C Model, were produced. This model is an attempt to describe reusable software components based on three aspects: (1) the concept, (2) the content, and (3) the context. The concept refers to a description, at an abstract level, of what the component actually does. The content refers to the details of how the abstract concept is actually implemented. There can be more than one content for a concept, since generally there are many algorithms for implementing a given task, usually with different performance and resource characteristics. The context of the component refers to those aspects of the software environment that can give additional meaning to either the concept or the content. In the case of the concept, the context of a particular data type for an operation defined by the concept can, for example, provide additional meaning to that concept. The content also has a context which can alter the meaning of its implementation. For a more detailed description of the 3C model, see Edwards & Baldo (1990).

3.5.4 Methods for Assuring that Software Performs as Expected

One frequently stated inhibitor of reuse is that programmers do not trust software that they did not write. Unless a programmer has confidence in a component from a library, he or she will not use it. Methods that can help assure that a component performs as expected can help address this problem (Hooper, 1990, p. 86).

3.5.5 Resource Utilization Characteristics

Currently, reusable software components are both designed and classified based on a *functional* model, i.e., *what* the component does. Component selection, on the other hand, is driven not only by the function a component performs, but also its resource utilization characteristics. Resource utilization is a term that refers to the fact that each component requires various abstract resources from its environment in order to run (for example, memory space, processor execution time, file handles, access to a terminal, etc.). Furthermore, the types and amounts of resources are different for each implementation of each component.

Unfortunately, the resource utilization of a component is currently treated in an random manner. For effective reuse, it is important not only to have a model of the functional behavior of a component, but also a well-defined model (or set of models) of the resource behavior as well. However, no such model(s) are now in use. Instead, components are currently stored in a library along with simple performance information based on benchmarks measured on a specific target, and which only address memory or cpu utilization.

Without an effective and comprehensive means of capturing and presenting resource utilization information to a potential component user, it will be very difficult for resource trade-offs to be made during component selection. This will also make reusable components less viable for real-time or embedded systems, where resource performance is a critical concern.

3.5.6 Management of Parameters

It is frequently stated that the most savings from reuse results from reapplying relatively large components (Biggerstaff, 1987). As components become large, however, the size and complexity of the parameters that are required to drive them can result in a secondary requirement for parameter configuration assistance. The principal problem is the empirically observed limitation in the ability of humans to deal effectively with long lists of relatively unstructured parameters. Another more temporary problem is the inability of some compilers, particularly Ada compilers, to handle components with large numbers of parameters.

The tendency of large components to have many parameters results from the fact that large components are typically constructed from smaller components, which may each be constructed from still smaller components, forming a hierarchy. While many of these parameters can be fixed, large components may still include many of the parameters of each smaller component that it uses. Components at the highest level can thus develop very large sets of parameters.

Approaches need to be developed to help humans deal with such large sets of parameters, and there has been work toward solving this problem, such as operating system configuration tools, window system defaults editors, shells for selection and composition of components, and shells for statistically well-designed experimental parameter search

3.5.7 Software Tools for Reuse

Once techniques in each of the previous six areas have been uncovered by research, production-quality software tools must be developed. Such tools are the key to moving these techniques from the research domain into practice. Without automated support, reuse techniques will only be slowly adopted and will also be slow to show savings.

4. STRATEGY FOR SOFTWARE REUSE

This section discusses the development of a strategy for software reuse for SDI. We first discuss the context of the problem by briefly mentioning some existing programs that involve reuse, including the Software Technology for Adaptable Reliable Systems (STARS) program, CAMP, and Ada software repository. We then discuss in some detail the Joint Integrated Avionics Working Group (JIAWG) program. The JIAWG effort has developed a strategy for encouraging software reuse across certain Air Force, Army, and Navy programs, and, since we agree with much of the JIAWG approach, we have tried to adopt a similar strategy for SDI.

4.1 STARS/CAMP PROGRAMS

Several DoD programs can be expected to develop technologies, standards, methodologies, and contractual and management strategies for software reuse. At the present time, however, only the JIAWG program has developed contractual strategies. Other programs have been primarily technology oriented.

The STARS program was established to improve software in DoD programs. Key goals are improving quality, reliability, and productivity, with software reuse a means to this end. STARS has funded the prototyping (by Boeing) of a software reuse library and other reuse technologies are expected to developed. Although STARS has not been very involved in contractual and management issues, the program has started to consider legal and data rights issues.

The CAMP program has developed methodologies, tools, and a library for reuse of software for missile applications. In this program, a domain analysis was performed on software for ten different missiles, with over 250 reusable software components being identified. A software

tool was developed that assisted a software designer in locating relevant existing components that could be used in a new missile, in analyzing candidate components, and in creating new components. Although the program has not addressed contractual issues in any detail, a report by Computer Sciences Corporation (1986) has suggested that contractual incentives must be provided for reuse.

4.1.1 JIAWG

The JIAWG was established by the Services in response to a mandate by Congress to identify common aspects of avionics equipment in three aircraft programs—the Navy Tactical Aircraft (ATA, currently known as the A-12), the Air Force Advanced Tactical Fighter (ATF), and the Army Lightweight Helicopter (LH, previously known as the LHX). As part of the JIAWG effort, a software reuse subcommittee has been developing plans and standards to support and encourage software reuse within each of these programs as well as across the programs, including:

a. A plan for the development of reuse standards and a reuse library.

b. Reuse standards which address specific activities that occur in different stages in the software life cycle.

c. A plan for providing incentives to contractors for the reuse software products.

d. A plan for maintaining the integrity of reusable software products.

The JIAWG subcommittee views the technologies needed to create a library and use the components in the library as mature enough to be used. They recommend populating a library with reusable components that are written as part of a demonstration/validation (DEM/VAL) program and then use these components in a full-scale development (FSD) project. They also expect to reuse components in a later pre-planned product improvement (P^3I) phase and in a post-deployment software support (PDSS) phase.

The overall JIAWG software reuse plan is divided into four phases: analysis, implementation, operational insertion, and transition and transfer.

In the analysis phase, the subgroup performed a functional domain analysis and cost analysis (JIAWG, 1989, p. 3), which concluded that "an investment of $20 million over the next few years could save DoD $50 million."

In the implementation phase, the contractor would be required to include a description of the software reuse approach in the Software Development Plan (SDP). This approach must address the following issues:

a. A proposed approach for analyzing the system to identify, utilize, and implement reusable software life cycle objects based on the JIAWG-approved domain analysis methodology.

b. Implementation of a contractor library based on a JIAWG standard for reuse libraries.

c. Incorporation of the JIAWG standards for reusable software into the contractor's software development process.

In the operational insertion phase, a library will be populated with reusable objects that met the JIAWG standard, and the components cataloged.

In the transition and transfer phase, the library will continue to be populated. However, projects will begin to extract reusable objects for use on FSD instead of DEM/VAL efforts. JIAWG also expects that during this phase, the need to furnish contractor incentives for reuse will diminish.

4.2 RECOMMENDED SDI SOFTWARE REUSE STRATEGY

At present, the SDI Software Policy advocates software reuse, however, there is no programwide description on how software reuse will be realized. Since the impact of software reuse on SDS is unknown, a strategy is needed that will assess the program to identify potential

applications of reuse that will provide reasonable cost reductions and a plan to achieve this objective. This section provides a set of recommendations that can be used by SDIO in planning for the insertion of software reuse into the SDS program.

Initiate pilot programs to demonstrate the feasibility of potential reuse practices and to gather empirical data on SDI reuse opportunities.

Most of the existing evidence demonstrating the feasibility of reuse is in areas such as financial applications and not embedded systems. Evidence indicating where reuse might be effective in SDI is badly needed. To implement such a plan, software reuse pilot programs can be initiated on DEM/VAL activities amongst the elements. The pilot programs can be either shadow projects or part of the actual effort when reuse technology being used is well understood and mature.

To do so, a partial and highly abbreviated domain analysis is necessary across a few SDS Elements. Such an analysis would identify software components or computer software configuration items (CSCIs)/computer software units (CSUs) that may be sufficiently similar in different Elements to serve as prospective reuse opportunities.

Since there has been no direct or formal analysis of component identification for software reuse for SDS, it is recommended that the System Engineering Integration Contractor (SEIC) Functional Analysis document and interviews with Element Offices be used to identify commonality across components for potential software reuse. It is recommended that the SDI System Engineer identify common software functions that have potential for software reuse across a set of SDS elements.

After identifying candidate components, an Element office should be selected to build a component based on a set of software reuse methods and standards (i.e., if they exist) that is agreeable to all users. Support for the component will be the responsibility of the Element Office selected for development. All users shall be invited to all component design, coding, and testing reviews.

The monitoring of the pilot programs should be well defined with respect to metric data to be collected and questions to be answered. The pilot program should at a minimum collect statistics on the following[1]:

a. Software architectural design metrics. The amount of resources used to identify components that have potential to be reused across a set of elements. For the pilot project, the SEIC Functional Allocation Document and interviews with Element office design engineers will be used as a means to identify components that can be reused across elements. The number of man-months required for this activity, methods, experience of personnel performing the analysis, classification of personnel interviewed at Element Offices, number of site visits, and tools used should be identified.

b. Component unit testing. Type of testing methods used, time to run tests, time to correct errors found in test, and machines and other resources required for tests.

c. Component integration testing. Time required and tests used for component integration.

d. Component modifications. Any modifications required for component to execute in Element environment.

e. Integration of component. Time required to integrate component into target system (i.e., including modifications in item *d.*).

Organize an SDIO software reuse office to establish and advocate a software reuse policy across all programs.

This office should provide software reuse guidance on software reuse methodologies, software reuse standards, and software reuse tools to the Element Offices. It should maintain a set of software reuse statistics for SDS (e.g., number of software designs reused, percentage of

1. The reader should note that the metrics being considered for the reuse pilot program are for the design, implementation, and testing phases of the software cycle.

software test data reused on a specific element) and lists of current and future applications of software reuse within SDS. Finally, the office should provide input to the SDI software policy as needed for software reuse.

A software reuse manager should be appointed and staff selected in accordance to the reuse activity ongoing within SDS. The SDIO Computer Resource Working Group subcommittee on software reuse should be utilized for input as requested by the SDIO/EN Software Manager.

Develop a SDS software reuse operational plan based on active participation of the SDIO office, Service, and element program offices and contractors.

The plan would describe the structure of the SDS software reuse office, operations of the SDS software library, selection of reuse standards, etc. Software reuse in SDI involves many technical, contractual, economic, legal, and management issues that interact in a complex manner. Successful reuse will not occur on a large scale unless all agree with the approach and are motivated to cause reuse to take place.

Develop economic incentives for software reuse in contracts.

Such incentives should encourage contractors to reuse existing software when appropriate, rather than developing new software. When developing new software, design for reuse should be encouraged but only when appropriate. Incentives can include performance fees based on successful demonstration of reuse or reuse practices, and allowing contractors to retain data rights to reusable software components to allow economic benefit when software components are reused.

Make use of software reuse technologies and management approaches from other DoD programs, and develop joint efforts with these programs.

As new DoD or government programs are established and become active, SDS should share information (as it is now doing with STARS and JIAWG) and work with program offices to develop joint efforts. Joint efforts are needed in funding software reuse research, in developing empirical data on reuse feasibility, and on developing contract incentives (such as consistent data rights policies).

Adopt a software reuse cost model.

The costs of reuse in the SDI context must be better understood. We recommend that SDI adopt a cost model that can predict the costs of software development with and without reuse. This can be done by selecting an existing cost model, modifying an existing model, or creating a new one.

Investigate the proper role of domain analysis.

Software reuse can be done most effectively when a relatively extensive, large-scale domain analysis is an early step in the development process. Unfortunately, a full domain analysis is difficult, expensive, and may require long periods of time to achieve. Efforts to define the SDI problem should be reviewed for their relevance to domain analysis. The relative benefits and costs of a full, formal domain analysis across SDI need to be compared with less formal approaches.

5. IMPLEMENTATION MECHANISMS TO ENCOURAGE SOFTWARE REUSE

This section discusses specific implementation mechanisms and language for contracts and RFPs that can help implement the strategy described in the previous section.

Five specific implementation changes in the contracting process are suggested:

a. Require offerors bidding on a contract to address software reuse in the description of their technical expertise, approach, and management in their proposal.

b. Require contractors to have a Software Reuse Plan as part of the Software Development Plan required by DOD-STD-2167A.

c. Include software reuse considerations in the evaluation criteria for an RFP.

d. Use a performance-based award fee to reward successful reuse or the adoption of reuse methodologies.

e. Allow contractors to gain certain rights to software developed with government funds as a means of rewarding successful reuse.

Sections 5.1 through 5.5 discuss how these items can be implemented.

5.1 DESCRIPTION OF SOFTWARE REUSE TECHNICAL EXPERTISE, TECHNICAL APPROACH, AND MANAGEMENT

In the RFP, contractors can be directed to specify their technical expertise and approach for software reuse in the proposal. Similarly, efforts by the contractor to develop organizational and management structures that ~~can~~ help reuse can also be considered.

The following text is an example of what should appear in the Technical Proposal section of an RFP:

> Software Reuse
>
> As part of the description of the offeror's expertise and technical approach to software development in the Technical Proposal, the offeror shall describe its expertise and technical approach to software reuse.
>
> Software reuse is the process of creating and using representations at any level (specifications, design, or code) that are designed in such a way as to be general enough to be effectively reused in developing a software system.
>
> The practice of software reuse is either the reuse of existing software rather than developing new software, or the development of new software in such a way that its components can be reused in some other part of the SDS than it was designed for.
>
> The description of the offeror's expertise in software reuse shall include the following: 1) examples of projects in which the contractor has developed software using reusable components and/or developed reusable components, and achieved high levels of productivity and quality as a result, 2) examples of software reuse research and development performed by the contractor, 3) training programs in reuse for contractor personnel, and 4) examples of projects in which the contractor has planned reuse efforts through the use of domain analysis.
>
> The description of the offeror's technical approach to software reuse shall include the following: 1) a preliminary analysis of the problem domain that is sufficient to indicate the most likely alternative methods for making use of existing software, 2) a discussion of potential benefits and risks of reusing existing software, developing reusable software, and using conventional approaches, 3) a description of any specific technical methodology for reuse that the contractor intends to use.
>
> [The description of the technical approach referred to above constitutes an initial version of the Software Reuse Plan discussed in Section 5.2.]
>
> The following reusable software is available in the SDS Software Library or commercially and must/may *[select one]* be evaluated for use in the software effort. A preliminary evaluation shall be made and included in the proposal, with a more in-depth evaluation, where appropriate, carried out as part of the contract work. The usefulness of software shall be evaluated with regard to the anticipated costs and benefits of utilizing or not utilizing each component.
>
> [SDIO should provide a short list of software that must be evaluated, either in the RFP or in an appendix to the RFP. Libraries can also be listed here that can be searched by the contractor

for potentially reusable software. SDIO should assist contractors in gaining assess to libraries.]

The following text should be inserted in the RFP in the section on the "Management Proposal."

> The offerors shall describe what organizational structures and management initiatives they expect to use that help encourage software reuse. These initiatives may include 1) organization of software development teams along functional rather than product lines, including use of a matrix organization, 2) relationships with other contractors that tend to encourage a single organization or group to design and implement software that is functionally similar across different subsystems of a platform or across different platforms, 3) relationships with other contractors that encourage reuse of software from one organization entity or group to another, and 4) incentive programs that reward employees for successful reuse initiatives.

5.2 SOFTWARE REUSE PLAN

Contractors should be required to develop and maintain a Software Reuse Plan, stating how reuse is incorporated into the software development effort. Such a plan can be considered a part of the Software Development Plan required byDOD-STD- 2167A. An outline of the Software Reuse Plan should be provided as part of the proposal.

There are two possible ways of implementing this requirement. *One is to provide* instructions in the RFP for inserting information in the Software Development Plan itself. Another way is to use a separate Contract Data Requirements List (CDRL), and define what is required for the Software Reuse Plan in the Statement of Work. Each approach is discussed in more detail in Sections 5.2.1 and 5.2.2.

5.2.1 Inclusion in the Software Development Plan

To make the Reuse Plan part of the Software Development Plan, the contractor should follow the instructions provided in an RFP that refers specifically to DOD-STD-2167A, DI-MCCR-80030A.[2]

10.2.5.1.1 Contractor Facilities

In this subparagraph identify, describe, and highlight the location of those items in the software engineering environment (SEE) that will be used to implement software reuse. This should include any libraries or tools that support the software reuse activities outlined in subparagraph 10.2.5.2.1 (Software Development Plan subparagraph 3.2.1).

10.2.5.1.2 Government-furnished Equipment, Software, and Services

In this subparagraph summarize the expected percentage reuse of any government-furnished software, and whether it is taken as information only, whether it is to be used with modification, or whether it is to be used as is.

10.2.5.1.3 Organizational Structure

If a Contractor Reuse Manager and/or Support Personnel are planned for the organization, this subparagraph shall detail the duties, responsibility, authority and/or role of the individual(s).

10.2.5.1.4 Personnel

If a structure has been defined to support reuse, indicate total number of personnel involved in this activity.

10.2.5.2.1 Activities

Any reuse activities should be identified and described in this subparagraph, and any impacts on the schedule should be addressed. Reuse activities include but are not limited to domain analysis, developing reusable software objects (RSOs), the populating and use of a reuse library, the reuse of existing RSOs, the evaluation and testing of RSOs, and reuse training.

---

2. These instructions have been developed by the JIAWG Reuse Subcommittee.

In this context, the term "domain analysis" refers to an identification of areas for potential reuse within the relevant technology or product domains, and to the determination of feasibility from the perspective of cost benefit and schedule impact to produce RSOs in the areas identified.

The term "reusable software object" refers to life-cycle objects that are created during the software development process, that are needed to operate, maintain, and upgrade the deliverable during its lifetime, and that have the potential for reuse. The objects may include (but are not limited to) requirements specifications, design documents (both top level and detailed), source and object code, test specifications, test code, test support data, user manuals, programmer notes, and algorithms. These objects may be textual, graphical, or both; and they are usually stored on electronic media.

10.2.5.2.3 Source Identification

This subparagraph should identify and describe the source for all pre-existing RSOs, and should provide a plan for obtaining the RSOs, indicate the need date and availability of each, and indicate any preparations required to make the RSOs usable.

10.2.5.3 Risk Management

Procedures for managing risk associated with the reuse activities specified in subparagraph 10.2.5.2.1 (Software Development Plan subparagraph 3.2.1) should be identified in this subparagraph.

10.2.5.5 Interface with Associate Contractors

This subparagraph should include a description of the interface with all associate contractor reuse activities.

10.2.5.7 Subcontractor Management

This subparagraph should include a description of the management of all subcontractor reuse activities.

10.2.5.9 Software Development Library

This subparagraph should contain a description of access and control procedures for RSOs, including any RSOs that were obtained under a license agreement.

10.2.6.1.1 Organizational Structure – Software Engineering

If reuse activities are planned, this subparagraph should describe the organizations responsible for those activities.

10.2.6.1.2 Personnel – Software Engineering

If reuse activities are planned, this subparagraph should describe the number and skill levels of the personnel involved.

10.2.6.1.3.1 Software Items

In the list of software items identify and describe any reuse libraries or tools, and any other items used or software reuse activities within the software engineering environment.

10.2.6.2.1 Software Development Techniques and Methodologies

For all reuse activities detailed in subparagraph 10.2.5.2.1 (Software Development Plan subparagraph 3.2.1), this subparagraph should explain how the techniques and methodologies adopted support those activities.

10.2.6.2.3 Design Standards

This subparagraph should specify design standards that support both general software engineering goals and specific software reuse goals. As appropriate, the following areas and issues should be addressed:

a. The design of the system software to a specific code implementation, such as Ada
b. The approach to Ada compilation unit dependencies
c. The approach to performance monitoring
d. How the design standards promote information hiding
e. How existing RSOs will be incorporated into the software designs
f. The approach to the design of new RSOs
g. The use of design objects from previous program phases
h. How the design will be adaptable for changing requirements within the system lifecycle, and will support growth options
i. How the design will be adaptable in other systems of a similar sort
j. How the packaging (tasks, packages, generics, and subprograms) of the objects and functions in the design supports the adaptability of the systems software
k. How the design will isolate project-specific requirements, such as processor-specific objects, project hardware specific objects, project-specific algorithms, and objects that require a security classification
l. What will justify the selection of project-specific design objects as opposed to more generic and adaptable design objects

10.2.6.2.4 Coding Standards

Coding standards, which conform to DOD-STD-2167A, Appendix B, should address specific items that will improve the reusability of the code modules.

10.2.8.2.1 Procedures

This subparagraph should identify and describe the procedures for evaluating and qualifying a product as a RSO.

10.2.9.1.1 Organizational Structure – Configuration Management

In this subparagraph describe the organization(s) responsible for the configuration management of reuse libraries.

10.2.9.1.2 Personnel - Configuration Management

In this subparagraph describe the number and skill levels of the personnel responsible for the configuration management of reuse libraries.

10.2.9.3.1 Flow of Configuration Control

In this subparagraph describe the process for dealing with problems and changes to RSOs within a reuse library.

10.2.9.3.3 Review Procedures

In this subparagraph describe the purpose of and procedures to be followed by the review board associated with the configuration control of a reuse library.

5.2.2 The Reuse Plan as a Separate CDRL

Another alternative is to require a reuse plan separate from the Software Development Plan. For this alternative, add Section 5.2.2.1 to the Statement of Work and add a new CDRL to the contract as described in Section 5.2.2.2.

5.2.2.1 Language for Statement of Work

Add the following language to the Statement of Work, followed by the text from Section 5.2.1 (if desired, the items can be renumbered):

Software Reuse Plan

The contractor shall prepare, keep current, and deliver a Software Reuse plan that contains the following:

[include text here from Section 5.2.1]

5.2.2.2 CDRL for Software Reuse Plan

A separate CDRL shall list the Software Reuse Plan as a deliverable, with the statement, "Contractor Format Acceptable" in item 4 (Data item report number).

5.3 EVALUATION CRITERIA FOR SOFTWARE REUSE IN THE RFP

In evaluating each contractor's proposal for a given contract, the software part of the proposal is assigned a weight that reflects how heavily software should be considered in assessing the entire proposal. The practice of software reuse is in turn assigned a weight that reflects the extent to which software reuse considerations should play a role in assessing the software part of the proposal.

In writing proposals, contractors can reasonably be expected to emphasize requirements of the RFP in proportion to their assigned weight. The assignment of a reasonably significant weight to reuse indicates to contractors that SDI is serious about utilizing software reuse. It is also necessary that SDI select a weight that reasonably reflects the value of reuse in terms of its expected benefits.

We recommend that software reuse account for about 15% of the weight assigned to software. We feel that at the present time there are sufficient unknowns about the risk and benefits of utilizing reuse that a weight higher than this is not justified. Contractors should receive a favorable

evaluation for considering software reuse in a prudent and knowledgable way. Contractors who respond with a correct analysis of reuse possibilities but who conclude that it is not feasible in the given situation, can be judged to be highly responsive.

The criteria used to calculate a total software reuse score (i.e., maximum value of 15% with respect to the total software proposal weight) will be based on the following evaluation criteria:

a. The description of the software reuse expertise of the contractor

b. The description of the technical approach proposed for software reuse

c. The description of management and organization for software reuse

d. The software development plan

Each criteria will be discussed in the next four sections (5.3.1, 5.3.2, 5.3.3, and 5.3.4) and assigned a weight to be used in calculating the total software reuse score.

5.3.1 Evaluation Criteria for Software Reuse Expertise

In evaluating a contractor's expertise in reuse, three main factors should be considered:

a. The contractor's experience with software reuse on past projects and contracts

b. Research and development efforts in reuse

c. Training programs in reuse

The following information should be given regarding prior projects:

a. What percentage of the project cost was allocated for reuse? (This includes direct costs such as library construction as well as indirect costs such as the increased cost to produce a module if it is reusable.)

b. Did the contractor support a reuse library?

c. Did the contractor make use of reusable objects from a reuse library?

d. Did the contractor develop reusable components? If so, were the reusable objects delivered to the library?

e. What methodologies and standards were used for reuse?

f. What research activities, such as those with internal research and development (IR&D) funding, were undertaken? (Efforts should be rated much more highly if there is a plan to make use of research efforts in actual development and in training of software developers.)

Since planned large-scale software reuse has been a recent innovation in DoD systems, most software engineers and managers will likely have limited experience in reuse. Training activities such as internal courses, university courses, and seminars, as well as efforts to integrate these new skills into existing software development projects, should be considered in the evaluation.

It is recommended that this criteria be weighted as 20% of the total software reuse score.

5.3.2 Evaluation Criteria for Software Reuse Technical Approach

So far, examples of software reuse that have achieved a high degree of success have been in narrow, well-defined domains. For example, many system facilities in operating systems are designed and implemented as objects that can be accessed through the run-time library to build application or system programs. Although domain analysis has yet to become a standard requirements in DoD contracts, an informal form of domain analysis is usually involved in software development. Evaluators should examine the contractor's technical approach for evidence of expertise in the particular domains that will be needed for the system as well as experience in informal and formal activities that are similar to domain analysis.

The "Technical Approach" section of the proposal should contain an explicit description of the relative costs and benefits of applying reuse to different aspects of the problem, identify the risks involved, and provide a justification for their analysis. Evaluation should not emphasize the percentage of code reused or developed, but the general pragmatic approach and prudent choices based on proper assessments of benefits and risks.

The evaluator should note all reuse methodologies, practices, and tools contained in the technical approach, assess each with respect to the maturity of that technology, the appropriateness of the technology, and the contractor's experience with the technology.

It is recommended that this criteria be weighted as 40% of the total software reuse score.

5.3.3 Evaluation Criteria for Description of Software Reuse Management and Organization

The evaluator should assess the contractor for its ability to encourage software reuse in the following areas:

a. Does the contractor generally support reuse on projects on its own initiative?

b. Does the contractor support reuse across projects?

c. Does the contractor support reuse in prime/subcontractor arrangements?

d. Does the contractor reward and encourage its software engineers to apply software reuse?

e. Does the contractor provide training and participate in software reuse research and development activities?

f. Does the contractor have a reuse library or other organized means of identifying reusable components?

It is recommended that this criteria be weighted as 20% of the total software reuse score.

5.3.4 Evaluation Criteria For Software Development Plan

Section 5.2.1 lists sections of the Software Development Plan in which contractors will provide information pertaining to their approach for software reuse. The present set of instructions for the Software Development Plan supplied in the RFP is based on recommendations from the JIAWG. Unfortunately, the JIAWG ATF RFP, which will contain these instructions, will not be let until the fall of 1990. Since this part of the proposal has the potential to be large and difficult to assess, we recommend that SDIO and JIAWG perform a joint study to determine a set of evaluation criteria for the Software Development Plan and produce a report that documents this effort.

It is recommended that this criteria be weighted as 20% of the total software reuse score.

5.4 CONTRACT INCENTIVES

To provide economic incentives for software reuse, two approaches are discussed along with suggested contract language. The first approach is the procurement of reusable objects and the second approach is the use of reusable objects in SDS.

In order to procure a set of potential reusable components for SDS, the program will initially have to provide contract incentives to stock a library with a set of baseline components. These reusable objects will be designed, implemented, and tested for reuse, which will add additional cost to their development and maintenance. Since it is not feasible from a cost perspective to require all SDS software to be developed for reuse, only objects which have a high potential for reuse should be considered.

Also, SDS will rely upon the reuse expertise of the contractor to identify reusable objects. This provides the contractor with the ability to make their own decision on what reusable objects of the system have the potential of keeping the lowest risk and with most cost efficient approach.

The recommended approach to procure such objects is to have the contract identify reusable objects in their proposal, with an award fee to be determined by the SDS program office after the reusable object has been delivered by the contractor and evaluated by the government. The following contract language can be inserted[3] in the solicitation, Part 1 - The Schedule, Section B - Supplies or Services and Price/Costs:

> Award Fee Provision
>
> An award fee provision shall be included in the contract which rewards the contractor for outstanding performance in developing reusable objects, and for effectively reusing such software objects.
>
> Award Fee Criteria
>
> An evaluation shall be conducted on any reusable object identified in the contractors proposal and accepted by the government for inclusion in a SDS software reuse library. The period for evaluation of award fee should be at least [TBD], and no longer than every [TBD]. It is anticipated that the award fee funds will be allocated over the award fee periods with percentages applied that relate to development activity milestones. Award fee procedures should require the contractor to provide a self-assessment to the Government prior to each award fee review. The SDS Reuse Evaluation Board shall include SDIO, Services, and element representatives. The contractor's efforts shall be evaluated to determine an award fee to be paid based on the Award Fee Criteria stated below:
>
> The contractor will also be awarded for the use of reusable objects in system development.

The criteria[4] will be provided in the solicitation, Part 1 - The Schedule, Section B - Supplies or Services and Price/Costs:

> Award Fee Criteria for evaluating the use of reusable objects:
>
> Degree that software objects are reused that exceed what was proposed. This includes both reusable objects developed under this contract and from other sources.
>
> Cost savings resulting from reuse based on the actual cost of software compared with proposed cost.

---

3. The contract language used in this document was modified from the *JIAWG Contract Elements for Software Reuse*, J89-S8, 15 March, 1990.
4. The contract language used for this criteria has been modified from the *JIAWG Contract Elements for Software Reuse*, J89-S8, 15 March 1990.

Improving upon proposed software schedule due to reuse without negatively impacting the overall contract cost or schedule.

Demonstrated productivity improvement.

5.5 DATA RIGHTS

The Federal Government procurement regulations for the acquisition of computer software are based on the Federal Acquisition Regulation (FAR) for civilian agencies and DoD FAR Supplement (DFARS) for the DoD. The DFARS require that unlimited data rights for any software developed in whole or in part by Government funds be assigned to the Government. The rationale for unlimited rights is stated in DFARS 227.472-1(a):

> For defense purposes, millions of separate equipment and supply items ... Technical data resulting from research and development and production contracts must be obtained, organized and disseminated to many different users. Finally, the Government must make technical data widely available in the form of contract specifications in the interest of increasing competition, lowering costs, and providing for mobilization by developing and locating alternate sources of supply and manufacture.

Industry views this problem in a different light. Industry representatives of the Data Technical Working Group [Probert, 1984, p. 11] concluded the following issues about unlimited data rights:

> Industry is reluctant to invest in new technology for the government because [of] sweeping data rights demands by the government, and apprehensiveness about the loss of proprietary information. ... This creates a climate unfavorable to the transfer of such technology [to the Government].

> ... the Government is failing to obtain the most innovative and creative computer software technology from its software suppliers. Thus, the government has been unable to take full advantage of the significant American lead in the software technology for the upgrading of its mission-critical computer resources.

A major objective of software reuse is to utilize *reusable software artifacts*[5] throughout the entire software development process. To provide software developers with the capabilities to access and use reusable software artifacts requires a legal and contractual framework that enables the appropriate reuse technologies to be used and provides incentives for industry to utilize reuse technology as it matures and becomes available.

The area of software reuse and its associated legal issues is an extremely complex and controversial topic. Intellectual property rights mechanisms, such as copyrights, patents, and trade secrets, often pose substantial barriers to reuse. These mechanisms need to be studied so that ways of eliminating or reducing these barriers can be found, and the mechanisms harnessed appropriately so as to help provide economic incentives for reuse.

The liability of software developers resulting from the malfunction of software is another significant issue that can inhibit reuse, because of questions about who is responsible when software that is originally developed by one organization and reused by another malfunctions.

Future work by the authors will consider these questions within the context of SDS and federal acquisition regulations.

---

5. Any product of the software development process (Peterson, 1989).

APPENDIX - REUSE DEFINITIONS

"...any way in which previously written software can be used for a new purpose or to avoid writing more software." (Kerighan, 1984).

"The terms 'software reuse' and 'software reusability' are applied to many techniques, methods, and processes. This spectrum includes portability in [the] classical sense, code-sharing in successive releases, common subsystems, common routines in applicaation families, [and] repeated exploitation of algorithms..." (Lenz, Schmid, & Wolf, 1987).

"... *reusable code* [is] defined as any technique that increases productivity by eliminating redundant practices in program generation." (Cavaliere, 1989, p. 132).

"Reusability is a general engineering principle whose importance derives from the desire to avoid duplication and to capture commonality in undertaking classes of inherently similar tasks." (Wegner, 1984).

"Reuse is employing knowledge that has been compiled through previous experience." (Agresti & McGarry, [March 1988]).

"***Reusability.*** The extent to which a **module** can be used in multiple applications [bold in original]. (*IEEE Standard Glossary of Software Engineering Terminology*, p. 30).

Software is defined to be *any machine processable representation of systems, subsystems, or system elements, and associated documentation and data.* This definition of software includes all computer programs and data bases, whether embedded in SDS components or development for models, simulations, analyses, automatic test equipment, or other uses pertaining to SDS and, therefore, covers both formally delivered and non-deliverable software products." (SDISP, 1989).

REFERENCES

Agresti W. & F. McGarry, March 1988. *The Minnonbrook Workshop on Software Reuse: A Summary Report.* NASA/GSFC, Greenbelt, MD, Computer Sciences Corporation, Beltsville, MD.

Baldo, J., Jr. (Ed.) (1990). *Reuse in practice workshop summary.* Alexandria, VA: Institute for Defense Analyses. IDA Document D-754.

Biggerstaff, T. J. & Perlis, A. J. (1989). *Software reusability. Volume 1: Concepts and models.* New York: ACM Press.

Biggerstaff, T. J. & Richter, C. (1987). Reusability framework, assessment, and directions. *IEEE Software, 4,* 41-50.

Booch, G. (1987). *Software components with Ada: Structures, tools, and subsystems.* Menlo Park, CA: Benjamin/Cummings.

Cavaliere, M. J. (1989). Reusable code at the Hartford Insurance Group. In T. J. Biggerstaff & A. J. Perlis (Eds.), *Software reusability. Volume 2: Applications and experience* (pp. 131-141).

Computer Sciences Corporation. (1986). *Ada reusability study.* (Tech. Rep. No. SP-IRD9). Morristown, NJ: Computer Sciences Corporation, August, 1986.

Edwards, S. H. & Baldo, J. (1990). *An approach for constructing reusable software components in Ada.* Alexandria, VA: Institute for Defense Analyses. IDA Paper P-2378.

Hooper J. & R. Chester. (1990). *Software Reuse Guidelines and Methods.* Plenum Publishing Corp.

IEEE. (1983). *IEEE standard glossary of software engineering terminology.* New York: IEEE Computer Society, Software Engineering Technical Committee.

JIAWG. (1989). *Software reuse rationale.* Working paper, Joint Integrated Avionics Working Group, May 26, 1989.

Kerighan, B. (1984). The Unix system and software reusability. *IEEE Transactions on Software Engineering, SE-10,* 513-518.

Latour, L. & Johnson, E. (1988). Seer: A graphical retrieval system for reusable Ada software modules. In *Proceedings of the third international IEEE conference on Ada applications and environments.* Manchester, NH. Los Alamitos, CA: IEEE Computer Society Press.

Lenz, M., Schmid, H. A., & Wolf, P. F. (1987). Software reuse through building blocks. *IEEE Software, 4,* 34-42.

McIlroy, M. D. (1969). Mass produced software components. In P. Naur & B. Randell (Eds.), *Software Engineering* (pp. 138-155). Garmisch, Germany: NATO Science Committee, January, 1969.

McNicholl, D. G. et al. (1988). *Common Ada missile packages—Phase 2 (CAMP-2). Volume I: CAMP parts and parts composition system.* (Final Rep. No. AFAL-TR-88-62, Volume I). St. Louis, MO: McDonnell Douglas Astronautics Company, November, 1988.

Peterson, A. S. (1989). Coming to terms with terminology for software reuse. In J. Baldo (Ed.), *Reuse in practice workshop summary* (pp. 88-92). Alexandria, VA: Institute for Defense Analyses. IDA Document D-754.

Probert, T. (1984). *Report of the rights in Data Technical Working Group (RDTWG). Volume 1: Executive summary.* Alexandria, VA: Institute for Defense Analyses. IDA Record Document D-52

Strategic Defense Initiative Software Policy. (1989). *Strategic Defense Initiative Software Policy.* Washington, DC: Strategic Defense Initiative Organization, Deputy for Engineering.

Tracz, W. (1989). Where does reuse start? In J. Baldo (Ed.), *Reuse in practice workshop summary* (pp. 36-46). Alexandria, VA: Institute for Defense Analyses. IDA Document D-754.

Wegner, P. (1984). Capital-intensive software technology. *IEEE Software, 1,* (3) 7-45.

ACRONYMS

3C Model	Concept, Content, and Context Model
ATA	(Navy) Tactical Aircraft
ATF	(Air Force) Advanced Tactical Fighter
CAMP	Common Ada Missile Package
CDRL	Contract Data Requirements List
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
DEM/VAL	Demonstration/Validation
DFARS	Defense Federal Acquisition Regulation Supplement
DoD	Department of Defense
EN	Engineering
FAR	Federal Acquisition Regulation
FSD	Full-scale Development
IDA	Institute for Defense Analyses
IRD	Internal Research and Development
JIAWG	Joint Integrated Avionics Working Group
LH	(Army) Lightweight Helicopter
NATO	North Atlantic Treaty Organization
P3I	Pre-planned Product Improvement
PDSS	Post-deployment Software Support
RAPID	Reusable Ada Packages for Information System Development
RFP	Request for Proposal
RSO	Reusable Software Object
SDI	Strategic Defense Initiative
SDIO	Strategic Defense Initiative Organization
SDP	Software Development Plan
SDS	Strategic Defense System
SEIC	System Engineering Integration Contractor
STARS	Software Technology for Adaptable, Reliable Systems

Distribution List for IDA Paper P-2494

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Lt Col James Sweeder Chief, Computer Resources Engineering Division SDIO/ENA The Pentagon, Room 1E149 Washington. D.C. 20301-7100	2
Other	
Mr. John Ablard USN/NRL Contracts Division Code 3200 4555 Overlook S.W. Washington, D.C. 20375	1
Mr. Dennis Ahern Westinghouse Electronic Systems Group Aerospace Software Engineering, MS-432 P.O. Box 746 Baltimore, MD 21203-0746	1
Mr. Axel H. Ahlberg GE Aerospace National Test Facility, MS-N9020 Falcon AFB, CO 80912-5000	1
Mr. Ted Allen Teledyne Brown Engineering Cummings Research Park 300 Sparkman Drive MS-174 Huntsville, AL 35807-7007	1
Mr. Don Alley Software Productivity Consortium SPC Building 2214 Rock Hill Road Herndon, VA 22070	1

NAME AND ADDRESS	NUMBER OF COPIES
Dr. Dan Alpert, Director Program in Science, Technology & Society University of Illinois Room 201 912-1/2 West Illinois Street Urbana, Illinois, 61801	1
Capt. Emily Andrew NTBJPO Falcon AFB, CO 80912	1
Mr. Robert Bowes DSD Labs 75 Union Avenue Sudbury, Mass 01776	1
Mr. Richard Bremner G.E. Aerospace Suite 800 5933 W. Century Blvd Los Angeles, CA 90045	1
Ms. Gina Burt WRDC/AL-A Wright Patterson AFB Dayton, OH 45433 513-255-9614	1
Mr. Richard Cheston United States General Accounting Office (GAO) Room 4073 441 G. Street, NW Washington, DC 20548	1
Dr. Sholom Cohen Software Engineering Institute Carnegie Mellon University 4500 Fifth Avenue Pittsburgh, PA 15213-2691	1
Mr. Jay Crawford Naval Weapons Center Code 31C China Lake, CA 93555	1

NAME AND ADDRESS	NUMBER OF COPIES
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Mr. Dennis S. Deutsch Gallo Gefiner and Fenster 235 Main Street Hackensack, NJ 07601	1
Capt Tony Dominice Air Force ATF Liaison Army Aviation Systems Command AMCPEO-LHX-TM 4300 Goodfellow Blvd St. Louis, MO 63120-1798	1
Mr. Tom Durek TRW MS FVA6/2050C 2701 Prosperity Avenue Fairfax, VA 22031	1
Mr. Dan Edwards Boeing Military Airplanes P.O. Box 7730, K80-13 Wichita, KS 67277-7730	1
Mr. Bill Farrell Robert Bowes DSD Labs 75 Union Avenue Sudbury, MASS 01776	1
Ms. Marielena Finn SAIC 1710 Goodridge Drive MS T2-8-2 McLean, VA 22102	1
Cpt Tim Fisk HQ SDIC Division/CN1 P.O. Box 92960 Los Angeles, CA 90009-2960	1

NAME AND ADDRESS	NUMBER OF COPIES
Ms. Diane Foucher Naval Weapons Center Code 251 China Lake, CA 93555	1
Dr. Joe Fox Software Architecture and Engineering, Inc. 1600 Wilson Blvd., Suite 500 Arlington, VA 22209	1
Mr. Bill Frakes Software Productivity Consortium SPC Building 2214 Rock Hill Road Herndon, VA 22070	1
Mr. John Gaffney Software Productivity Consortium SPC Bldg 2214 Rock Hill Road Herndon, VA 22070	1
Mr. Harold Gann UIE 1500 Perimeter Parkway Suite 123 Huntsville, AL 35806	1
Mr. Terry Gill CMRI/CEC Martin Marietta NTB MS 8700 Falcon AFB, CO 80912-5000	1
Mr. Todd Goodermuth GE Aerospace P.O. Box 1000 Blue Bell, PA 19422	1
Mr. Jack Gorman NASA Johnson Space Center NASA Road 1 Houston, TX 77058	1

NAME AND ADDRESS	NUMBER OF COPIES
Dr. Ron Green Deputy Commander USA/SDC ATTN SFAE-SD-GST-D 106 Wynn Drive Huntsville, AL 35807	1
Mr. Harley Ham NAC-825 Naval Avionics Center 6000 East 21st Street Indianapolis, IN 46219-2189	1
Mr. Ron Halbgewacks POET, Suite 300 1225 Jefferson Davis Hwy Arlington, VA 22202	1
MAJ Gary Hausken Intellectual Property Law Division Office of the Judge Advocate General ATTN USA/JALS-IP 5611 Columbia Pike Falls Church, VA 22041-5013	1
Mr. Jim Hill Martin Marietta Falson AFB, CO 80912	1
Mr. Bill Hodges Boeing Aerospace and Electronics P.O. Box 3099 Mail Stop 9Y38 Seattle, WA 98124-2499	1
Mr. Robert Holibaugh Software Engineering Institute Carnegie-Mellon Pittsburgh, PA 15213	1
Mr. Danny Holtzman Vanguard 10306 Eaton Place, Suite 450 Fairfax, VA 22030	1

NAME AND ADDRESS	NUMBER OF COPIES
Prof. James Hooper UAH Computer Science Building Room 111 Huntsville, AL 35899	1
Mr. Richard Iliff SDIO ENA Room 1E149, The Pentagon Washington, D.C. 20301 703-693-1826	1
Mr. William Jarosz 14800 Aviation Avenue L.A. Air Force Base Hawthorne, CA 90009 213-363-8699	1
Mr. Raj Kant Honeywell Systems and Research Center 3660 Technology Drive Minneapolis, MN 55418	1
Ms. Kerry Kent MITRE Falcon AFB, CO 80912-5000	1
Mr. Jack Kleinert SAIC 1710 Goodridge Drive P.O. Box 1303 McLean, VA 22102	1
Dr. John F. Kramer STARS Technology Center 3701 North Fairfax Drive Arlington, VA 22204-1714	1
Ms. Virginia P. Kobler Chief, Technology Branch Battle Management Division US Army Strategic Defense Command P.O. Box 1500 Huntsville, AL 35807	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Ed Koss Advanced Technologies Inc. 14 Cove Rd Melbourne Beach, FL 32951	1
Dr. Larry Latour Department of Computer Science University of Maine Neville Hall Orono, ME 04469-0122	1
Mr. Doug Lea Visiting Researcher Syracuse University CASE Center 2-173 Science & Technology Center Syracuse, NY 13244	1
Mr. Ronald Liedel SDC ATTN CSSD-SA-EE P.O. Box 1500 Huntsville, AL 35807 205-955-3972	1
Dr. Charles Lillie SAIC 1710 Goodridge Drive P.O. Box 1303 McLean, VA 22102	1
Mr. Gary Mayes SDC/GSTS PO U.S. Army ATTN SFAE-SD-TST 106 Wynn Drive Huntsville, AL 35807-3801	1
Mr. Ron McCain IBM Corporation 3700 Bay Area Blvd Houston, TX 77058-1199	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Charles McNally Westinghouse Electronics Systems Group Contracts P.O. Box 746, MS-1112 Baltimore, MD 21203-0746	1
Mr. Stan McQueen MITRE 1259 Lake Plaza Drive Colorado Springs, CO 80906 719-380-3325	1
Mr. Mike Mitrione Dynamics Research Corp. 1755 Jefferson Davis Hwy, Suite 802 Arlington, VA 22202	1
Mr. Robert Nelson Information Systems Management NASA Space Station Program Office 10701 Parkridge Blvd. Reston, VA 22091	1
Ms. Terri Payton UNISYS 12010 Sunrise Valley Reston, VA 22091	1
Ms. Joanne Piper CIVPERS Stop H-3 Ft. Belvoir, VA 22060-5456	1
Mr. Frank Poslajko SDC MS CSSD-SP 206 Wynn Drive Huntsville, AL 35807	1
Dr. Ruben Prieto-Diaz SPC 2214 Rock Hill Road Clarendon, VA 22070	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Don Reifer Reifer Consultants, Inc. 2550 Hawthorne Blvd. Suite 208 Torrance, CA 90505	1
Mr. Ernie Roberts McDonnell Douglas Corp. P.O. Box 516 D309/B66/L2N/Room 210 St. Louis, MO 63166	1
Mr. Jim Robinette DCA/Z4S/SMBA 3701 N. Fairfax Drive Arlington, VA 22203	1
Mr. Jack Rothrock SofTech, Inc. 2000 North Beauregard Street Alexandria, VA 22311	1
Mr. Rich Saik Teledyne Brown Engineering Cummings Research Park 300 Sparkman Drive MS-174 Huntsville, AL 35807-7007	1
Mr. Robert Saisi DSD Labs 75 Union Avenue Sudbury, MASS 01776	1
Ms. Pamela Samuelson School of Law University of Pittsburgh 3900 Forbes Avenue Pittsburgh, PA 15260	1
Dr. John Salasin GTE Govt. Systems Corp. 1700 Research Blvd Rockville, MD 20850	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Robert Schafrit TA&T, Inc. 133 Defense Hwy Suite 212 Annapolis, MD 21401 301-261-8373	1
SDIO Technical Information Center DRC 1755 Jeff Davis Highway Suite 802 Crystal Square 5 Arlington, VA 22202	1
Mr. Robert Seltzer Meta Software Corporation Cambridge, MA 02140	1
Dr. John Solomond AJPO OUSDRE/R&AT The Pentagon, Room 3E114 Washington, D.C. 20301-3081	1
Mr. Barry Sookman McCarthy and Tetrault P.O. Box 48 Toronto Dominion Center Toronto, Canada M5K1E6	1
Mr. Terry Starr GE P.O. Box 1000 Blue Bell, PA 19422	1
Ms. Geree Streun General Dynamics/Ft. Worth Div. P.O. Box 748, MZ 4050 Ft. Worth, TX 76101	1
Dr. Ted Tenny General Dynamics/Ft. Worth Div. P.O. Box 748 MZ 1761 Ft. Worth, TX 76101	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Will Tracz IBM Corporation Mail Drop 0210 Route 17C Owego, NY 13827-1298	1
Mr. Otis Vaughn TBE Cummings Research Park 300 Sparkman Drive MS 198 Huntsville, AL 35807-7007	1
Dr. Anthony Wasserman, President Interactive Development Environments, Inc. 150 Fourth Street, Suite 210 San Franciso, CA 94103	1
Prof. Bruce W. Weide Dept. of Computer and Information Science The Ohio State University 2036 Neil Ave. Mall Columbus, OH 43210-1277	1
Mr. Nelson Weiderman Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213-3890	1
Mr. Paul Wilbur Teledyne Brown Engineering Cummings Research Park 300 Sparkman Drive MS-174 Huntsville, AL 35807-7007	1
Mr. Joan Winston Office of Technology Assessment (JTET) Project Director Room 309 U.S. Congress 600 Pennsylvania S.E. Washington, DC 20510-8025	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Neal Winters TBE Cummings Research Park 300 Sparkman Drive MS 174 Huntsville, AL 35807-7007	1
Mr. Joel Zimmerman ESD/Contracts Division (PK) Hanscom AFB, MA 01731-5000	1

IDA

General Larry D. Welch, HQ	1
Mr. Philip L. Major, HQ	1
Dr. Robert E. Roberts, HQ	1
Ms. Ruth L. Greenstein, HQ	1
Mr. James Baldo, CSED	2
Dr. David J. Carney, CSED	1
Ms. Anne Douville, CSED	1
Mr. Stephen Edwards	1
Dr. Dennis W. Fife, CSED	2
Dr. Richard J. Ivanetich, CSED	1
Mr. Terry Mayfield, CSED	1
Mr. Mike Nash, CSED	1
Ms. Katydean Price, CSED	2
Ms. Mary Elizabeth Springsteen, CSED	1
Dr. Richard L. Wexelblat, CSED	1
Mr. David A. Wheeler, CSED	1
Dr. Craig A. Will, CSED	2
IDA Control & Distribution Vault	3