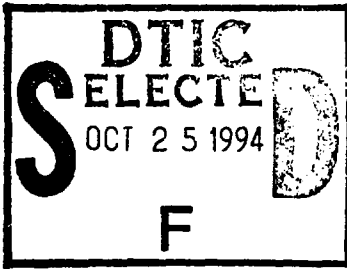


AD-A285 708



2

NAVAL POSTGRADUATE SCHOOL Monterey, California



THESIS

INDUSTRY VERSUS DOD: A COMPARATIVE STUDY OF SOFTWARE REUSE

by

Robert W. Therriault
and
Kristina E. Van Nederveen

September, 1994

Thesis Advisor:

James C. Emery

94-3294S



1038

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 2

94 10 21 03 8

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE Industry versus DoD: A Comparative Study of Software Reuse			5. FUNDING NUMBERS	
6. AUTHOR(S) Robert W. Therriault and Kristina E. Van Nederveen				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Software reuse is a longtime practiced method. The technical issues, such as how to link software repositories and programming for reuse, have been resolved. The problems faced by industry and the Department of Defense are of a non-technical nature and can be categorized into three broad categories: managerial, economic, and legal. This thesis compares industry and DoD reuse efforts highlighting common problems and lessons learned. The comparison is between IBM, Hewlett-Packard, the Air Force's Central Archive for Reusable Defense Software (CARDS), and the Restructured Naval Tactical Data System (RNTDS). Each reuse effort is studied using personal interviews and written descriptions. Problems encountered by private industry and their solutions are analyzed and compared to DoD. Many of industry's problems are found to be prevalent in DoD. Industry recognizes these issues and is taking steps to rectify them. Legal issues are the least understood by industry and DoD, and need further research to overcome these hurdles. Some economic and managerial issues are recognized by DoD and are in process of being resolved. Industry is more advanced than DoD in their programs and understanding of reuse. DoD can alleviate some of its software reuse problems by employing the lessons learned from industry.				
14. SUBJECT TERMS Software Reuse Program, Software Development			15. NUMBER OF PAGES 104	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

Industry versus DoD: A Comparative Study of Software Reuse

by

Robert W. Therriault
Lieutenant, Supply Corps, United States Navy
B.S., Florida State University, 1984

Kristina E. Van Nederveen
Captain, United States Army
B.S., The Ohio State University, 1987

Submitted in partial fulfillment
of the requirements for the degree of

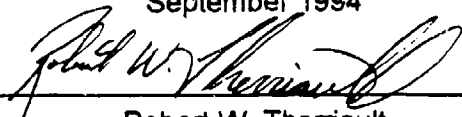
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL

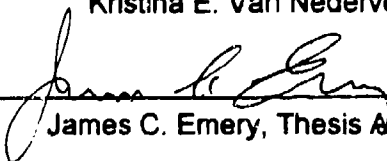
September 1994

Author:

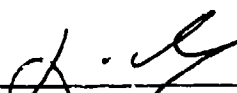

Robert W. Therriault


Kristina E. Van Nederveen

Approved by:


James C. Emery, Thesis Advisor


William Short, Associate Thesis Advisor


David R. Whipple, Chairman
Department of Systems Management

ABSTRACT

Software reuse is a longtime practiced method. The technical issues, such as how to link software repositories and programming for reuse, have been resolved. The problems faced by industry and the Department of Defense are of a non-technical nature and can be categorized into three broad categories: managerial, economic, and legal. This thesis compares industry and DoD reuse efforts highlighting common problems and lessons learned. The comparison is between IBM, Hewlett-Packard, the Air Force's Central Archive for Reusable Defense Software (CARDS), and the Restructured Naval Tactical Data System (RNTDS). Each reuse effort is studied using personal interviews and written descriptions. Problems encountered by private industry and their solutions are analyzed and compared to DoD. Many of industry's problems are found to be prevalent in DoD. Industry recognizes these issues and is taking steps to rectify them. Legal issues are the least understood by industry and DoD, and need further research to overcome these hurdles. Some economic and managerial issues are recognized by DoD and are in process of being resolved. Industry is more advanced than DoD in their programs and understanding of reuse. DoD can alleviate some of its software reuse problems by employing the lessons learned from industry.

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution	
Availability Codes	
A-1	Availability or Special

TABLE OF CONTENTS

I. INTRODUCTION	1
A. BACKGROUND DISCUSSION	1
1. Overview	1
a. <i>What is Software Reuse</i>	1
b. <i>General Software Reuse</i>	2
c. <i>DoD Software Reuse</i>	2
2. <i>Reuse's Shortcomings</i>	4
a. <i>Software Reuse Standardized Methodology</i>	4
b. <i>Incentives</i>	5
c. <i>Legal Considerations</i>	7
B. BRIEF SUMMARY OF THE AREA OF RESEARCH	8
C. DEVELOPMENT OF RESEARCH QUESTION	9
II. LITERATURE SEARCH	11
A. DEFINITIONS	11
B. DISCUSSION OF LITERATURE SEARCH	13
1. <i>Software Reuse</i>	13
a. <i>Prieto-Diaz Facet Classification of Reuse</i>	13
b. <i>Other Reuse Classifications Discussion</i>	16
c. <i>DoD Reuse Standardization Efforts</i>	16

2. Software Reuse Legal Issues	18
3. Economic Factors of Reuse	18
III. INDUSTRY AND DOD SOFTWARE REUSE EVALUATION...	20
A. INDUSTRY REUSE EVALUATION	20
1. IBM Boeblingen	20
2. Hewlett-Packard	26
B. DOD REUSE EVALUATION	32
1. Restructured Naval Tactical Data System (RNTDS)	32
2. Central Archive for Reusable Defense Software (CARDS)	38
IV. FINDINGS	46
A. PURPOSE OF CHAPTER	46
1. Software Reuse Origins	46
B. LEGAL ISSUES	47
1. Legal Issues Found in the Evaluations	47
2. Patents, Copyrights, and Liabilities	50
3. Software Repositories	55
a. Access	56
b. <i>Content of Repositories</i>	57
c. <i>Who is Liable for Faults</i>	58
4. Opening Software Reuse to Outside Users	59
5. Effects of Legal Issues on DoD Reuse Program	60

C. ECONOMIC FACTORS OF SOFTWARE REUSE	61
1. Software Reuse Economics	61
2. Economic Barriers	62
3. Incentives	65
4. Economic Models	68
5. Effects of Economic Issues on DoD Reuse Program	70
D. MANAGERIAL ISSUES OF SOFTWARE REUSE	71
1. Upper-level Management Involvement	73
2. Reorganizing the Organization	73
V. CONCLUSION AND RECOMMENDATIONS	76
A. DETERMINATIONS	76
1. DoD Software Reuse in General	76
2. Positive Effects of Software Reuse in DoD	78
<i>a. Support for Software Reuse</i>	78
<i>b. Software Reuse Horizon</i>	78
3. Negative Effects of the Software Reuse Program in DoD	81
<i>a. DoD Software Reuse Initiative Infrastructure</i>	81
<i>b. Repository Concerns</i>	82
(1) Domain Analysis	82
(2) Certification and Search Tools	83
(3) Legal Concerns	84

c. Acquisition Regulations	84
d. Education and Training	85
B. CONCLUSIONS	86
C. RECOMMENDATIONS FOR FURTHER STUDIES	87
1. Software Development and Reuse Legal Issues	87
2. DoD Acquisition Regulations	88
LIST OF REFERENCES	89
INITIAL DISTRIBUTION LIST	94

I. INTRODUCTION

A. BACKGROUND DISCUSSION

1. Overview

a. What is Software Reuse?

Software reuse is accomplished by creating programs from previously developed software modules. Many different aspects of software can be reused, such as source code, documentation, design, and algorithms (Krueger, 1992; Prieto-Diaz, 1993). This should provide system developers with greater productivity by reducing the work required to produce modules that commonly occur in other systems. Reuse is expected to lead to reduced system development time and maintenance while increasing reliability by using existing working modules. Most important are the potential monetary savings that would be realized by using such a system. In a climate of shrinking assets, reducing costs is welcomed by both the Department of Defense (DoD) and the military services as well as industry.

Software is the Achilles' heel of implementing any portion of an information system. Software drives the hardware and is quite often the breaking point for either successful implementation or failure of that system. Software reuse is intended to simplify this development process and abate high

maintenance costs. The idea is to use some type of software reuse to build an application quicker and more efficiently than starting from scratch.

b. General Software Reuse

The idea of reuse has been around for a long time. Instead of reinventing solutions, it is easier to extract the applicable information from previously developed solutions, and use it in the creation of a new one. Mathematics is an example where the reuse concept has been formalized for a long time. "Formal generic mathematical models are good examples of successful reuse because they can be applied to solve specific problems across several engineering fields and domain" (Prieto-Diaz, 1993). This idea of reuse can also be applied to the various aspects of related software products. Why reinvent the wheel when somebody else has already created it?

c. DoD Software Reuse

Software reuse is being touted as a primary way to reduce systems development expenditures. The Department of Defense (DoD) and the military services perceive that it will greatly reduce the cost of systems development. System development currently consumes about 10 percent of DoD's annual budget (Endoso, 1992). To accomplish these savings, systems will need to "reuse" previously designed and implemented software modules. These modules would be maintained at centralized software repository centers for reuse by authorized vendors and the military services.

The gains that would be experienced by investing in a software reuse system are attractive at first glance. There are, however, some major obstacles to be hurdled to achieve success. To facilitate a venture of this immensity one central authority will be required to be the focal point of all efforts. The central authority would be responsible for:

- Standardization of requirements for the DoD system;
- Acting as the quality filter for the reusable modules being sponsored for acceptance;
- Providing incentives to encourage sponsorship and use of the reusable modules;
- Maintaining a central repository library system and reuse system interface;
- Providing mechanisms for distribution.

DoD has taken bold steps to achieve this with the Corporate Information Management (CIM) initiative. This initiative seeks to improve DoD's management of information resources. CIM calls for functional interoperability between systems, standards compliance, and efficiency in software development. The efficiency would result from dependence on reusable software components (RSC), commercial off-the-shelf (COTS) products, and computer-aided systems engineering (CASE) tools. CIM additionally promotes the establishment of a software reuse repository. The purpose of the repository is to develop a central DoD-wide RSC clearinghouse, establish a data dictionary for DoD, and build an integrated repository for C3I software (Bui, et al., 1993).

This will require a reuse library system capable of multiple user access, interconnectivity across different operating systems, search and retrieval, cataloging, classification, and user/library documentation

At the same time that DoD is grappling with these issues and trying to introduce a global software reuse system, private sector firms have dealt successfully with some of these issues and are pursuing software reuse as a way to cut development costs and production time. By no means have they perfected reuse, but some corporations are starting to reap the benefits of their reuse program.

2. REUSE'S SHORTCOMINGS

Software reuse is hindered by issues such as training, costs, legal problems, poor incentives, technical difficulty, and cultural resistance (Garry, 1992). Reusable code is not a cure-all for programmers and does not always provide significant benefits. Quite often maintaining old programs or developing shell scripts for reuse of old code is overlooked (Plauger, 1993). A brief discussion of the important issues follows.

a. Software Reuse Standardized Methodology

There currently exists no clear-cut directive or documentation from higher authority detailing how reuse should be accomplished in DoD. The services are pursuing reuse, but are doing so with their own individual guidance instead of a unified DoD basis. This is going to produce four to five sets of different service-unique requirements and methodologies.

Paul A. Strassman, former director of the DoD's CIM program, defined two of the challenges facing reuse as: 1) developing identifiers that are easy to understand for the code in the repository and 2) providing search capability (Perry, 1992). Each military service has developed identifiers and search capabilities for their programs, but each one is service-unique. This has led to no standard identifier format or search mechanism shared by the military services. Without some form of repository and reuse standardization, users will ultimately be forced to using an interconnected system and having to learn the various idiosyncrasies of each. This will not encourage use of the system, but rather discourage it. Users will not use a system that requires a lot of effort to learn. The system will either be viewed by the user as not being user-friendly or it may be perceived by them to be more frustrating than useful.

b. Incentives

For a repository to be effective, the RSCs it accepts and maintains should be delivered at a high level of quality. The present method utilized to certify candidate RSCs has been adopted from the Army's Reusable Ada Product for Information System Development (RAPID). This certification determines the attributes of the RSC, its reuse potential, and the level of re-engineering needed. Following this, one of four levels is assigned to the RSC. The levels range from no formal testing and documentation (first level) to fully tested and documented that meets all standards (fourth level). There is a

fifth level that is not currently being used, for designation in the future (Bui, et al., 1993).

Sponsors of software for reuse have no incentive to provide anything but level one or two RSCs. This is because there are no monetary incentives to provide a RSC with the necessary documentation and testing for level three or four certification. Nor are there any directives stipulating this must occur for every RSC submitted to the repository(ies). RSC's at level one and two are in a less developed state and will result in software development personnel having to analyze and tailor them more than levels three and four. Though this has to be done for all levels, the process of fitting levels one and two to the project needs will prove more costly than building it from scratch due to their less developed state. Any consideration of reusing components created by others must include analysis of cost and benefits, quality, achievable reuse goals, domain analysis, staff experience, development, and recognition of the effort involved (IEEE Software, 1993).

Another problem is that vendors have no incentive to use RSCs if they are not directed to or do not get paid a bonus for using them. Even if these methods are implemented, a vendor may still not use software reuse unless it can be demonstrated to save them time and money. Internal and external incentives have to be determined to make the software reuse system pay for itself. Also, the DoD must offer vendors a motivation for reuse, and the

companies must provide the same things to their employees to encourage their reuse (Endoso, 1992).

c. Legal Considerations

Problems exist with the current DoD acquisition regulations that hinder giving incentives for reuse. Many questions about legal rights and government ownership of the reusable components have also surfaced. Software reuse has not been addressed very well at the acquisition management level. Acquisition personnel who have to implement reuse do not have the proper guidelines in place. In fact, the regulations do not address reuse at all (Endoso, 1992). Additionally, these personnel are ill-trained concerning many of the other aspects pertaining to information systems, which only compounds the problems experienced in acquisition. Acquisition is the basic tool used in instituting the development of new software systems and is where concerns regarding the legal issues are the most relevant.

The General Accounting Office (GAO) recently released a report on the DoD software reuse plan. The report stated that technical, organizational, and legal barriers must be overcome by the DoD to realize the benefits of software reuse. The report, entitled "Software Reuse: Major Issues Need to be Resolved Before Benefits Can be Achieved," dated January 1993, points out many of the previously discussed shortfalls of reuse in this chapter. The report found the following:

- Methodologies for implementing reuse have not been fully developed

- Tools supporting a reuse process are lacking, and
- Standards for guiding critical software reuse activities have not been established (GAO, 1993).

The GAO report presented some of the barriers to software reuse, such as higher initial costs to develop reusable software and the possible legal battles that may evolve among software suppliers, repositories, and users (Endoso, March 1993). This report did not, however, address the various legal issues surrounding software reuse repositories. Software reuse, in theory, is an attractive idea, but the GAO report highlighted various pitfalls and, as a result, has caused more scrutiny of software reuse efforts.

Industry has been studying software reuse for many years and one of the first case studies, at the Hartford Fire Insurance Group, was published in 1983 (Biggerstaff and Perlis, 1989). Many information technology managers in industry have implemented some type of software reuse program, with some success. These case studies reveal the experiences and growing pains of implementing a successful software reuse program. These experiences should provide valuable insight into the reuse methodologies, legal issues, and economic factors that industry has had to tackle. These experiences can be utilized to assist DoD in its similar goal of implementing software reuse.

B. BRIEF SUMMARY OF THE AREA OF RESEARCH

In order to reduce the staggering expense for software development and quell the critics, software reuse has to prove that it is a viable cost cutting

methodology for future software development. Currently, each of the services conducts software reuse using various methodologies, techniques, and procedures. DoD has mandated software reuse, but has not developed a detailed game plan to achieve this directive. This thesis will explore issues associated with the implementation of software reuse, such as software reuse methodologies, incentives for software programmers, ownership of software modules, compensation for software developers, and procurement changes.

All of these issues need to be addressed before DoD can institute an effective software reuse program. Using examples from both industry and DoD, these issues will be explored, drawing on both successes and failures in developing reuse, to determine the best practices of software reuse. Comparing and contrasting both adequate and inadequate practices in industry to what is being achieved in government may give the Armed Forces new insight on how to implement software reuse.

C. DEVELOPMENT OF RESEARCH QUESTION

The objective of this research is to compare and evaluate how private firms have implemented software reuse and how private firms' methods compare to DoD's effort. The thesis will focus on issues concerning ownership of modules, incentives to use software reuse modules, and how intellectual property rights impact software reuse initiatives. How has industry structured reuse? Is it a global concept or an internal concept? Did actual software reuse exist prior to being implemented as part of the software development cycle? What came first,

ad-hoc reuse or structured reuse? Are there lessons that can be learned from successful industry implementations that can be applied to DoD?

The research will be achieved using case studies and personal interviews with programmers, executives, and others involved with the establishment and use of software reuse in industry and DoD. Literature searches will be conducted on the relevant issues addressed previously in this chapter. Data will be collected from these various sources and then analyzed to attempt to determine similar trends. Hopefully, this comparison will assist in conceiving a process of how to best conduct software reuse

II. LITERATURE SEARCH

The literature search was conducted at the Knox Library, the Naval Postgraduate School, the Computer Center Library in Ingersoll, personal interviews, CD ROM, and via the Internet. Many of our contacts were made by subscribing to newsgroups on the Internet and receiving mailings from news sources. In our search we came across certain authors many times, such as Ruben Prieto-Diaz, Ted Biggerstaff, Will Tracz, Joyce Endoso, and Barry Boehm.

A. DEFINITIONS

While conducting the literature search on the topics to be covered in this thesis, certain terms were found to be defined in various manners. To clarify any potential misinterpretation of terms and provide continuity throughout, it is necessary to define these terms in the context in which they will be used in this thesis. These definitions will apply to various aspects of software reuse, legal issues, and economic topics. The definitions are as follows:

Software reuse - The use of existing software components to construct new systems. Software products such as source code, designs, specifications, objects, text, architectures, processes, domain analysis, megaprogramming, etc. can be reused.

Software reuse repository - Resembles a conventional library system. Used to store the various software components to be used in the reuse effort. These components are the same as those described in the software reuse definition. Various classification schemes are used to file software components in the repository.

Internal reuse - Reuse of software produced internal to the organization that is developing the program. Usually conducted by the same programmer who wrote the original software product.

External reuse - Reuse of software produced external to the organization that is developing the program. Author of the software product is not part of that particular organization. For the purposes of this thesis, an organization is defined as a DoD service or a corporation in industry.

Domain-specific reuse - Reusing components in a specific domain to build an instance of an application in that domain (Direction-Level Handbook, 1994).

Patent - A government grant of monopoly on an invention for a limited amount of time. Usually in the United States, these patents last for seventeen years. An invention is a new device, or a composition of matter, or a newly created technical method (Henderson, 1993).

Copyright - A grant of an exclusive right to produce or sell a book, motion picture, work of art, musical composition, or similar product during a specified period of time (Hirsch, 1988). It protects the expression of an idea, not the idea itself. A copyright is automatic, in the sense that it is obtained by simply displaying prominently within the document the word 'copyrighted' followed by the year of creation and the author's name. This copyright lasts for three years.

Liability - Addresses the legal responsibilities of the software developer and software repository administrator for any misuse of the software products contained in the reuse repository.

B. DISCUSSION OF LITERATURE SEARCH

1. Software Reuse

a. Prieto-Diaz Facet Classification of Reuse

Industry appears to be as committed to achieving a viable software reuse program as is DoD. Software factories in Japan, such as those at Hitachi, NEC, and Toshiba, as well as programs in the United States at AT&T, GTE, IBM, and Hewlett-Packard, demonstrate this resolve. In conducting the literature search for this thesis, we found many different types of reuse to be used by industry and DoD. Ruben Prieto-Diaz, who has published many essays on classifying software products for reuse, has stated that all reuse can be viewed as falling into six facets: by-substance, by-scope, by-mode, by-technique,

by-intention, and by-product (Prieto-Diaz, 1993). His facets best illustrate the many approaches that can be used to achieve software reuse and they are discussed below.

By-substance: defines the essence of the items to be reused.

- Idea: Involves the reuse of formal concepts to a particular class of problems.
- Artifacts: Components or parts reuse involve using portions of programs. Parts reuse focuses on quality, reliability and certification of reusable components.
- Procedures: Procedures reuse encompasses reuse of process programming and environments focusing on trying to formalize and encapsulate software development procedures.

By-scope: defines the form and extent of reuse.

- Vertical: Reuse within the same domain or application area. Its goal is to derive generic models for families of systems to be used as templates for new systems.
- Horizontal: The use of generic parts in different applications that do not necessarily perform in the same manner as the original application.

By-mode: defines how reuse is conducted.

- Planned: Systematic and formal practice of reuse where guidelines and procedures for reuse have been defined.

Metrics are collected and used to assess the performance of the reuse effort.

- Ad-hoc: Informal practice in which components are selected, by a programmer, from previous projects or a generalized library of reusable products.

By-technique: defines what approach is used to implement reuse.

- Compositional: The use of well defined, existing components as building blocks for new systems.
- Generative: Reuse propagated by use of application or code generators at the specification level.

By-intention: defines how elements will be reused.

- Black-box: The reuse of software components without any modification or the necessary documentation and specifications to comprehend how the component functions.
- White-box: The reuse of components by modification and adaptation for use in the development of a new system.

By-product: defines what work products are reused.

- Source code: The reuse of actual programming source code.
- Design: The reuse of designs of existing software matched to specifications for the software system being developed.
- Specifications: The reuse of specifications with its respective implementations at the design and source code levels.

- Objects: The reuse of objects using object oriented tools and methods to cover all phases of development.
- Text: The reuse of text by integrating reusable text with all other work products.
- Architectures: Analysis of application domains to find generic designs that are then used as templates for integration of reusable parts or development of code generators.

b. Other Reuse Classifications Discussion

The breakdown presented by Ruben Prieto-Diaz is the most comprehensive classification system for software reuse that we found. Several other ways to categorize software reuse were encountered, but they were not as descriptive. For instance, Tracz places software reuse into two categories: Vertical reuse and horizontal reuse. Vertical reuse can occur when the majority of the applications built by software developers are representative of a single kind of data processing activity, and many software objects that are employed by one can be shared among the others. Horizontal reuse, by contrast, occurs across a broad range of application areas (Tracz, 1987).

c. DoD Reuse Standardization Efforts

In pursuit of developing a formal reuse policy, DoD has established how it envisions software reuse in the DoD Software Reuse Vision and Strategy Initiative document. The vision portrayed in the DoD initiative is "To drive the DoD software community from its current re-invent the software cycle to a

process-driven, domain-specific, architecture, library-based way of constructing software" (DoD, 1992). The strategy details that the purpose of the vision is to incorporate reuse into the software development cycle of each program. The strategy consists of ten elements that basically address the following issues:

- Targeting domains that have the greatest potential for reuse.
- Solving the various legal issues surrounding reuse.
- Establishing a metrics program to measure the payoff and guidance for developers in the selection process.
- Exploiting near-term products and services that can facilitate movement to a reuse paradigm.
- Integrating reuse as an integral aspect of the software acquisition process.

In support of this initiative, DoD established the Defense Software Repository Service (DSRS) to eventually encompass a distributed operation with four remote centers supporting the different armed services. DSRS is intended to support reuse efforts at the centers performing domain analysis and plan for effective software reuse. In order to achieve a centralized DoD clearinghouse the services have been developing a telecommunication infrastructure with a standard network interface. This is intended to eventually allow users to access software code at the four reuse centers that now use separate systems. This will permit access to each of the DoD reuse repositories without having to access them separately.

Five centers will eventually become involved as integrated test sites for the system. The sites are (Bui, et al., 1993):

- DSRS, Washington, D.C.
- Standard Systems Center, Gunter Air Force Base, Ala.
- Army's Software Development Center, Falls Church, Va.
- Navy Computer and Telecommunications Station, Washington, D. C.;
- Marines Corps Central Design Activity, Quantico, Va.

The ability to access the system will be controlled by the services and library security will be maintained by the services (Endoso, 1992).

2. Software Reuse Legal Issues

Something that many software developers tend not to think about too often are the legal issues involved with software development. Software reuse has added some unforeseen complications to these issues. The legal issues involved with software reuse center on the software repositories and copyrighted or patented software modules. For many programmers, copyrights, patents, and other legal issues are too abstract and unrelated to their process of software development and reuse. Yet the fear of patent infringements, copyright violations, and the associated liability, make software development personnel increasingly hesitant to use repositories or administer them. Software reuse will not be a viable solution to reducing software development costs if these legal issues are not resolved. These legal issues will be discussed further in Chapter IV.

3. Economic Factors of Software Reuse

Economic factors play an important role in software development. Project managers have to keep the product on time and within budget. If software reuse is combined with this process, then economic factors will include the cost of the reuse program. These costs include incentive programs, education of all personnel, and changes to the organization. Any organization that embarks on a software reuse program has to be willing to accept the high initial costs associated with such a program. Reuse will require a significant departure from the normal operations of the software development team. Costs associated with finding reusable software, verification and validation of software, and adaptation of software must be considered before embarking on a software reuse program (Emery and Zweig, 1993).

Software reuse, in the short term, may seem to be an expensive method to reduce software development costs. This could cause management to become unwilling to consider software reuse because it may be viewed as a drain of critical resources. This impression is further complicated by the lack of a "standard" economic model for software reuse to measure any potential cost savings. The cost of reusing software must be less than the cost of developing new software, but much of the reuse efforts today do not consider the cost factor (Conn, 1993). The economic factors of software reuse will be discussed in greater detail in chapter IV.

III. INDUSTRY AND DOD SOFTWARE REUSE EVALUATION

A. INDUSTRY REUSE EVALUATION

Industry has been toying in the reuse arena for several years. Some companies have recognized the benefit of software reuse, or more generally reuse that includes the design phase through the maintenance and implementation phase of software development. The benefits are realized in quicker product releases and better quality products. In this section we will examine two examples from industry where reuse has been successfully instituted, IBM and Hewlett-Packard.

1. IBM Boeblingen

IBM is one of the giants in computer software and hardware. Recently IBM has been forced to reorganize its corporate structure to stay competitive in the industry. IBM uses Market-Driven Quality (MDQ) to identify ways of attaining greater productivity and quality goals. MDQ is IBM's version of Total Quality Management (TQM), a new management style that emphasizes quality of the product and meeting the customer's needs and desires.

As a result of MDQ, there was a greatly increased demand for reusable software, designs, and documentation (Wasmund, 1993). Through software reuse, the corporation can develop products quicker and more efficiently. Reuse started as an ad-hoc idea and has expanded throughout the corporation. It was

centered around programmers using existing components from previous projects. In the early days of software reuse at IBM it was done by-substance (artifacts) and by-mode (ad-hoc) (Prieto-Diaz, 1993).

As in many new projects, the inhibitors to the new project had to be identified and addressed. In the case of software reuse, four major inhibitors were found. These were the lack of reusable components, lack of component compatibility, lack of appropriate development environment, and lack of appropriate database retrieval mechanisms (Wasmund, 1994). Although IBM used incentive programs to entice programmers to create and use the reusable components, it eventually had to mandate the reuse program in order to make any progress. To make reuse a more accepted practice throughout IBM, the project required support from management and a structured method for implementing software reuse. In order to achieve this, IBM has focused on establishing reusable parts centers.

The first reusable parts center was established in Boeblingen, Germany. IBM calls software modules "parts," since they are not limited to software code, but can also include documentation, designs, and specifications. The parts center was created as a pilot program, but has proven to be a successful way to proliferate reuse through the corporation. Its objective is the production and maintenance of highly generic and reusable software components for worldwide use within IBM. The concept is that for reuse to occur there has to be a "trading infrastructure" or "reuse marketplace" to get customers and suppliers together.

For reuse to be successful, there has to be a forum or place where the developers of software modules can interact with the customer to create the parts they require for a project. Management support also plays an important, if not crucial part in the implementation of software reuse. Without the support, the trading of parts would be restricted. Management can oversee the process and straighten out any problems.

A repository is used to store parts and their descriptions. The parts in IBM's repository are assigned a particular certification level, which indicates the level of completeness and the defect rate level of the part. The more a part is used the better it becomes. Each time a part is used it is refined and hidden defects are discovered. The certification level alerts a programmer that the component has either been newly created, and therefore has a low certification level and the potential for defects, or it has been used numerous times and has proven its worth by having most of the bugs corrected, deserving a higher certification level. Managing the repository requires continuous maintenance in order to keep the components up-to-date and usable for the programmers. Maintenance of these parts is very important to both the user and the quality of the part. This is consistent with IBM's MDQ philosophy.

An accounting system is also needed to trace and account for exchanges and costs in order to determine savings. Existing accounting systems had to be modified to reflect the high initial cost of the reuse program with the cost benefits coming later once the program was fully operational.

Michael Wasmund, who is the site coordinator at IBM Beoblingen, has been the driving force of the parts centers for reuse. He established a reuse methodology using the Critical Success Factors (CSF) method (Wasmund, 1993). The Critical Success Factors method takes a problem or tasking and lets one build a project out of that problem by identifying critical tasks that need to be accomplished in order to achieve the goals set out by the project. The basic steps of the CSF method are as follows (Wasmund, 1993).

- 1. Define a goal
- 2. Decompose the goal into a set of factors.
- 3. Define activities.
- 4. Build and validate the CSF matrix.
- 5. Execute the activities.

While defining the goal, it is of utmost importance to be as specific and detailed as possible about purpose, scope, and time constraints. It has to include the exact definition of the quantities desired by the goal.

The next step takes the goal and breaks it down into factors. The factors should describe entities and not activities which are mission essential. The factors state things or entities that must be obtained to reach the goal. An example of a factor is availability of a specific software engineering tool. These factors have to be as independent of each other as possible. In a reuse environment these factors are best derived from brainstorming and intense discussions (Wasmund, 1993).

In step 3, activities are defined and include verbs that describe the action required to achieve the goal. These activities describe the actual work that needs to be accomplished. In this step it is essential to consult everyone who is involved in the environment to be changed or who might be affected by the changes otherwise the execution of the defined activities will meet with low acceptance (Hooper and Chester, 1993).

Next, in step 4, a matrix is built using factors and supporting activities as the matrix outliners. This matrix, Figure 1, helps identify unsupported factors, redundant activities, and priorities. From the matrix, unsupported factors can be recognized, redundant activities eliminated, and activities can be tracked and readjusted for better project management.

	Critical Success Factors	A	B	C	D	E	F
Activity							
1		X	X			X	X
2			X	X			X
3		X			X		X
4					X	X	
5				X		X	
6			X	X		X	
7			X	X		X	

Figure 1: CSF Matrix

The last step, step 5, is executing the activities. Execution has to be geared to the local work culture and experience level, otherwise problems can occur. "Every failure started with a great plan. If the plan cannot be translated into action, then nothing will come from it" (Wasmund, 1993).

The following are some lessons learned by IBM Boeblingen when they implemented the CSF method for establishing software reuse:

- The greater the effort spent in Steps 1 and 2 of the CSF to obtain accurate and agreeable statements, the less rework was required to achieve the overall goal. The goal was to shorten the development cycle, increase the reliability of marketed products, and attain high reuse maturity within IBM (Wasmund, 1993).
- IBM explored the issue of providing tools for reuse compared to establishing methodologies for reuse. IBM decided to use both simultaneously to allow teams to start the reuse effort. There still is a problem with tool integration because the repository has to be compatible with the configuration libraries required for the development of the product. The requirement for data transfer standards has not been fully developed yet. There is a disconnect because the standards have not been developed due to the various development environments. Tools and methodologies have to be consistent throughout the company to allow interaction between different projects.
- IBM found that although incentives worked well for the cost associated with them, overall, their effect was limited. The purpose of the incentive program

was achieved, because programmers created items, i.e. parts for reuse and these items were actually reused. Yet the incentive program did not bring about cultural change within the organization to shift to reuse. The incentive program was tied to other already existing award programs. The incentives were not based on just the creation of parts for the repository, but on having parts used in products and the creation of parts for others to use. Rewards were awarded only if the part did not require modification. Incentives are still being used, but are not seen as the main catalyst for implementing reuse throughout the organization.

- IBM used education and consultation to spread the concept of reuse through the organization. Education heightened the awareness of IBM's software reuse concept. Consultation, on the other hand, was not as useful. This was due to the consultation being viewed as interference from the outside.

Through the CSF method, IBM Boeblingen is making parts reuse an inherent tool in developing high-quality products faster. The reuse concept is cutting costs and is recognized as a fundamental tool in the creation of new products in the future. IBM is actively pursuing software reuse by incorporating it into developing new products, educating management, and continuing research in new technologies.

2. Hewlett-Packard

Hewlett-Packard has extensively researched software reuse and has applied it to its organization. Serious work on reuse started in the early 1980s.

Early work involved the development of instrument libraries in BASIC, the construction and use of databases to store and distribute software components, and more recently the use of Objective-C or C++ to develop class libraries (Griss, 1993). Early reuse was centered around software code only. Most of these software modules were reused in-house, but some were actually provided to outside developers. Hewlett-Packard used incentives to elicit software from contributors. The problem with this was that the repositories were filled with software modules that did not work well together (there were no standards or methodologies to make modules compatible). This problem was exacerbated by the fact that incentives were given to contributors but not to users of the library.

The mission statement of the software reuse program at Hewlett-Packard is stated as follows: "We have initiated a multifaceted corporate reuse program to help introduce the best practices of systematic reuse into the company, complemented by multidisciplinary research to investigate and develop better methods for domain-specific, reuse-based software engineering" (Griss, 1993). The aim of the reuse program is "to make software reuse a more significant and systematic part of the software process" (Griss, 1993). At this point in time Hewlett-Packard has not focused on just one way to do software reuse. Instead they are exploring various methods and studying new business management philosophies and engineering concepts.

In order to support the corporate reuse program at Hewlett-Packard, a general model was developed. The general model used at Hewlett-Packard has four steps:

- Step 1 Business Analysis. This requires that the business be modeled to understand how reuse may be applied and what key issues are involved.
- Step 2 Technical Analysis. In this step, software reuse process elements are identified and matched to the entire product development process.
- Step 3 Social Analysis. This step involves the definition of the new organization based on the experience of knowledge work organizations, reuse organizations, and the software processes used by other software entities.
- Step 4 Tools and Environment. Create a software factory through kits of software factory support components to provide a technical infrastructure.

The development of this four-step model was based on research on how to improve reuse at Hewlett-Packard. From the research done at Hewlett-Packard, it was found that most problems associated with software reuse were not based on technical problems; instead they were attributed to non-technical problems that were managerial or economic in nature. The research also focused on inhibitors to reuse and divided them into three categories: people, process, and technology. Inhibitors to software reuse were attributed to one of these categories and alleviators to these inhibitors were developed.

First and foremost among these inhibitors was management support of software reuse. Long-term support and up-front investments are needed to get a software reuse program up and working. The organizational culture needs to be changed. Employees need to have confidence in the reuse program. This can be achieved through incentives, training, and management backing. The organization also needs to change with respect to financial policies, contracting models, and legal policies.

Technical aspects may be expressed in guidelines and standards for building, testing, and documenting reusable work. Hewlett-Packard has found that the best way to introduce software reuse to an organization is to start with a pilot program. Starting off with oversight on a small project can demonstrate the benefits derivable from software reuse with only a small investment up front. A small project is easier to manage, making it possible to convert the nonbelievers and lead to a change in culture.

The researchers also found that to achieve efficient and effective software reuse, there had to be a shift from the library model to a model that emphasizes software engineering. "The library metaphor and model, used for many years to guide work in reuse, needs to be replaced by a software engineering model based on kits, factories, manufacturing, and engineering. Software engineers and managers need to change their view of software reuse from that of simply accessing parts in a software library, to that of systematically developing and

using well-designed parts following a careful process within a reuse-based software factory" (Griss, 1993).

Hewlett-Packard's work on kits has centered around the LEGO concept for kits. The fundamental idea of LEGO is to use blocks that fit together with other blocks in the kit to create a product. In the LEGO for kids, kits are designed to let them build castles, space stations, and pirate ships. These kits can be combined to build more complex projects. "A 'kit' should contain well-designed and packaged compatible reusable work products, tools, and processes to assist in providing more 'complete' solutions for application developers" (Griss, 1993).

In the beginning Hewlett-Packard used by-substance (artifacts), by-mode (first ad-hoc, then planned), and by-product (source code) reuse (Prieto-Diaz, 1993). More recently Hewlett-Packard has pursued another area of research that led them away from software repositories to what they term software factories. A software factory is a way to combine factory and manufacturing concepts to develop and produce software in a flexible manner. Flexible, in this case, means applying a less rigid and concrete software engineering method that is able to be easily manipulated. This carries the concept of kits one step further. The software factories use the same concepts found in industry in order to build a production line of software products. It brings back the engineering side to software product development. The factory is designed to use the components in the kits to create the product, but also includes the engineering

and rework associated with different products. The goal is to be able to cut out redundancy in the redesign and reengineering of different products.

The lessons learned are based on earlier reuse efforts by Hewlett-Packard that brought about the framework for software reuse discussed above.

- Incentives lacked the desired effect at Hewlett-Packard because it populated the repositories with modules but did not increase the instances of reuse.
- Since the repositories did not bring about dramatic increases in software reuse, Hewlett-Packard looked for a more systematic approach to reuse that combined such aspects as fourth-generation languages (4GL), object-oriented technology, computer-aided software engineering (CASE) tools, and formal methods for specifications.

The bottom line, based on Hewlett-Packard's experience, is that for an effective reuse program to be established it has to start small, be well-supported from the start, and gain experience through pilot projects. Research has established three major stages of reuse adoption: introduce the commitment to try reuse, institutionalize the commitment to change and expand the pilot program, and sustain the commitment to improve (Griss and Latour, 1992). Hewlett-Packard has made a commitment to reuse by establishing a corporate reuse program. There are numerous pilot programs being studied by researchers to determine which way to expand the programs. Hewlett-Packard

does not want to narrow its options by limiting its reuse program to just one facet, but instead is exploring various aspects.

B. DOD REUSE EVALUATION

DoD has employed many different approaches in its aspiration to achieve software reuse. As discussed in Chapter II, these approaches are similar to those pursued by industry. DoD, however, seems to be attempting to narrow its scope from an ad-hoc, informal and unstructured strategy to a structured, planned, compositional methodological approach to software reuse. This is evident with the Air Force's Central Archive for Reusable Defense Software (CARDS), the Navy's Restructured Naval Tactical Data System (RNTDS), and the Army's Reusable Ada Product for Information System Development (RAPID) program. The Defense Software Reuse System (DSRS) has chosen to emulate the RAPID program.

The common threads among all of these DoD systems is the combining of domain-specific, compositional, systematic and vertical methodologies of software reuse to derive a more efficient program. The RNTDS and CARDS programs were selected to exemplify a couple of the efforts that have been pursued in DoD.

1. Restructured Naval Tactical Data System (RNTDS)

As far back as 1976, the Naval Sea Systems Command's software support activities wanted to develop an architecture to take advantage of the commonality present in the Combat Direction System (CDS) domain. The CDS

domain consists of numerous command and control functions - such as navigation, fire control, control of air assets, identification, tracking, and operator display - that present the current tactical situation and allow the execution of the operators' orders. Achieving this requires that the CDS have over a dozen asynchronous interfaces with the ship's combat system. The CDS can be viewed as the integrating system of each ship class. This is what the RNTDS architecture was designed to exploit, the commonality between these systems across various ship classes. The commonality would lead to reuse of the modules that were found in each of the different CDS's (Stevens, 1991).

Early studies conducted found that there was a high degree of commonality among the CDS program functions. The study concluded that only 20 to 40 percent of the requirements were unique to one ship class. To take advantage of this functional commonality, software reuse was considered to be the best methodology to exploit the advantage. The original basic goals of the RNTDS were (Stevens, 1991):

- Automate the construction of multiple CDS programs with varying requirements from a single repository of small, reusable components.
- Reduce the life cycle maintenance costs through the abstraction of common processing requirements.
- Reduce the impact of software change resulting from corrections, improvements, or hardware changes.

- Provide the ability to deliver CDS program improvements across all ship classes through a single implementation.
- Create a common operator interface across all ship classes to minimize training requirements.

The following goals evolved subsequent to the original ones and were adopted for inclusion into the RNTDS goals (Goodall, 1992):

- Ensure transportability of applications.
- Improve program reliability.

The CDS software development in the RNTDS architecture followed a structured methodology that consisted of the following phases: specification development, program performance specification development, preliminary design analysis, pre-code analysis, function testing, performance acceptance testing, combat systems integration testing, and fleet introduction support and testing (Stevens, 1991). Software reuse was incorporated in the program performance specification development phase. The RNTDS development can be classified as using a combination of the facets described by Prieto-Diaz. They are by-substance (artifacts), by-scope (vertical), by-mode (planned), and by-product (architecture) (Prieto-Diaz, 1993).

The performance requirements of each CDS program are specified in a standard syntax adhering to a strict format to promote clarity and to allow for consistency and completeness checks. The domain model used by the RNTDS was the Program Performance Specification, which is a stand-alone document.

In this document the CDS processing requirements are expressed in small discrete numbered paragraphs. Each paragraph contains pointers to the next one that continues the processing.

This is done to be able to allow for stimulus-response threads to be traced through the entire specification. Maintaining one specification to serve all applications permits an automated matrix to be used to relate which paragraphs are needed for each program. A change made to one requirements paragraph results in the change being made to all related programs that include that paragraph (Stevens, 1991). The key point to this philosophy is that software reuse begins by reusing specifications and forcing development to reuse specifications and not "reinvent the wheel." The programmers knew they could reuse code, but they wanted to focus the reuse effort on the program performance specifications instead (Goodall, 1992).

The domain model became the point of departure for all of the programs that followed after the original development of the first program. When new requirements were provided they were analyzed against the domain model (Program Performance Specification) to ascertain the similarities. This analysis identified which requirements paragraphs could be reused or modified and which ones would have to be generated. A new specification document is not generated for each project; instead, a column is added to the matrix after the new and modified paragraphs are added to the specification.

The artifacts from the first project were used to populate the reuse repository, which is referred to as the Common Reusable Library (CRL). As a project is completed, the new and modified artifacts are included in the CRL. The CRL is used in conjunction with the analysis in determining which requirements paragraphs can be reused or modified or need to be built from scratch (Goodall, 1992). The RNTDS program has used support tools that were developed concurrently with the application program. The tools were also managed under the same configuration as the application software.

Through the development of the RNTDS program there were many valuable lessons learned by the project members. The first lesson, related to the project, assumed that all artifacts are reusable if they were designed from their inception to be reused. The artifacts that were used included performance specifications, design specifications, source code, test procedures, and user manuals (Stevens, April 1994).

The second lesson learned by the project team was related to the size of the code components. They found that with the smaller code components reuse was enhanced, but complicated program construction. What they learned from this lesson was that it was necessary to automate construction with stringent, automated source update procedures supported by automated tools to assure adherence to architecture standards (Stevens, April 1994). Additionally, it was found necessary to use configuration management to assist in simplifying maintenance.

The third lesson learned dealt with the development and evolution of the support tools. They first built a working methodology and then worked to optimize it by improving performance, including transitioning to a unified relational database and to commercial hardware. They learned that support tools are not static and should not be expected to remain static. They determined that evolution of the tools to a more robust, dynamic set of tools should be built into the project's acquisition plan (Goodall, 1992). They also observed that the tools and application software need to be co-located to be effective.

Lastly, the project team found that it was necessary to maintain an independent repository engineering group. The group was made responsible for coordinating the interaction of application program development teams with the repository (CRL) (Stevens, April 1994).

The RNTDS project also used published standards set forth by the Software Productivity Consortium (SPC) and Software Engineering Institute (SEI), as well as applicable DoD standards and instructions. The SPC economic models applied will be discussed later in this chapter. The SEI affiliation and use of their Software Capability Maturity Model was instrumental in providing measurement throughout the project development. They also applied available standard government progress and planning practices and found them to be adequate (Stevens, June 1994).

The future enterprise of the RNTDS is to continuously improve the present processes and continue to deliver CDS programs with high degrees of software reuse. This will be accomplished by applying the RNTDS reuse techniques to other CDS's, reengineering existing programs to make use of the functional commonality, applying architectural lessons learned, and migrating software reuse techniques to workstation-based technologies to improve its portability. Pursuit of these goals will serve to make the RNTDS program a more flexible system capable of maintaining a high degree of reuse for the future.

2. Central Archive for Reusable Defense Software (CARDS)

The Central Archive for Reusable Defense Software (CARDS) was initially designed for the Air Force command center domain. Since the origin of the program, CARDS has evolved to become structured around the elements found in the DoD's Software Reuse Vision and Strategy document discussed in Chapter II. The main purpose of this emphasis is to remove the redundancies found in software development by using a process-driven, domain-specific, architecture-centric, library-based way of building software. CARDS has evolved to a global concept so that it could be used by other DoD organizations in implementing their software reuse programs. The four main goals of the CARDS program are (Technical Document, 1993):

- Program, document and propagate techniques and processes to enable domain-specific reuse throughout DoD and to support contractors across various domains.

- Develop a franchise plan that provides a blueprint for institutionalizing domain-specific, library-assisted reuse techniques to be used throughout the DoD.
- Implement the franchise plan and provide a tailored set of services to support reuse.
- Develop and operate model-based, domain-specific library systems and necessary tools in support of the franchisees.

To assist in identifying and removing technical and business related obstacles to domain specific reuse, the CARDS program is developing a "Knowledge Blueprint" for reuse. It will include the necessary materials to support the transition of the "Blueprint" into the software community. The blueprint is communicated by a "Franchise Plan" that is supported by library operation and maintenance related documentation, reuse adoption handbooks, and training and education materials. The Franchise Plan provides descriptions of reuse processes and the appropriate instructions for tailored implementation of domain-specific reuse processes. The Plan also describes in precise steps a scenario for implementing a domain-specific library (Direction-Level Handbook, 1994). The Plan is embraced as a universal document that provides the outline of the processes needed to implement domain-specific reuse for any DoD organization or DoD contractor.

CARDS was designed to incorporate a domain-specific, model-based approach to software reuse. A model-based approach to reuse emphasizes the

relationships between components, as well as the actual components themselves; in contrast, the component-based approach emphasizes just the component. A component is usually easily described and follows well-partitioned functional lines. Examples of these include documents, models, subroutines, and applications. The CARDS reuse effort can be classified as using a combination of the facets described by Prieto-Diaz: by-substance (artifacts), by-scope (horizontal), by-mode (planned), by-technique (compositional), and by-product (all forms) (Prieto-Diaz, 1993).

The domain-specific software architecture lends itself to being the organizing principle of the library model. This supports traceability of requirements to particular components and the subsystems implemented by aggregations of components, which is also captured in the library model. The architecture is also used for the process of qualifying the reusable components for the library. The architecture appraises the software "form, fit, and function" against the requirements mandated by the software architecture. Domain-specific component qualification goes further than qualitative assessments of conventional reuse libraries to show how a component is used within an application domain (Technical Document, 1993).

The CARDS model-based libraries differ in many ways from conventional reuse libraries, in the following ways:

- In CARDS, components are not easily defined and include concepts such as requirements, generic architectures, other conceptual models and their interrelationships.
- CARDS does not use component search and retrieval, with its singular application emphasis supporting interactive search, but rather a collection of applications tailored to the domain of interest and a prescribed user. These can include a graphical browser, system composer, and component qualification.
- CARDS relies on the use of modeling formalisms to describe, manage, and use complex sets of relationships that are characteristic of an application domain and its software architecture and components. CARDS modeling formalisms are better than lower-level data modeling formalisms used in conventional library schemes. This sophistication is due to the fact that CARDS applies knowledge representation technology to derive its modeling formalisms.

To support this modeling, CARDS has engaged the Reusable Library Framework (RLF), which is an open system knowledge representation framework designed for use as a foundation for domain-specific reuse repositories/libraries. RLF is comprised of a semantic network and a rule-based inferencing system, similar to an expert system, that captures and maintains the constraints of actual problem domains. RLF was found to be the best repository

candidate, used in the right environment, from 12 government and commercial programs because of its unique abilities (Reuse Tool Assessment, 1993).

The CARDS system provides a group of library applications adapted to the domain of interest for the individual user. The CARDS model-based library is designed to be used for more than component storage; it also provides a collection of reuse tools. These tools are implemented on the open systems programming specification provided by the RLF. Individual tools provide specialized reuse services adapted to certain application domains. This is made possible due to the tools using the RLF to manipulate domain models and the ability of the RLF to support domain models for different application domains. The domain model, in turn, contains significant information concerning the application domains, including requirements, architectures, and rationale. Ultimately, the domain model refers to reusable software components kept in a component store for use by the user (Technical Document, 1993).

While the lessons learned by the RNTDS program were mainly technical in nature, the ones that evolved from the CARDS program were managerial in context. Based on experiences of the program, it became evident that to implement software reuse, more than just the technical side of the program needed to be emphasized. In response to this, four handbooks were developed to aid the adoption of software reuse and were targeted at specific audiences. These handbooks were based upon past CARDS experiences in constructing

and operating domain-specific reuse repositories/libraries. The four handbooks are:

- Direction-Level Handbook is targeted at top-level acquisition managers to assist in their understanding of implementing software reuse. It provides a framework to aid them in establishing plans to manage reuse across their systems and obtain the goals outlined in the DoD Software Reuse Vision and Strategy document (Direction-Level Handbook, 1994).
- Acquisition Handbook is directed at program managers and contracting and legal professionals involved in the acquisition/software development cycle. It is designed to assist them in incorporating software reuse into all phases of the acquisition life cycle (Acquisition Handbook, 1994).
- Engineer's Handbook is geared towards software engineers and other technical personnel in discussing the integration of software reuse development methods and techniques into their own command/contractor software engineering processes.
- Component and Tool Developer's Handbook provides a technical basis for the creation of components and tools for domain specific reuse libraries (Technical Document, 1993).

Another lesson was derived from the initial efforts that were centered around the singular domain of command centers. As the efforts proceeded, there came the understanding that this singular CARDS focus could evolve to include support for other domain libraries as necessary and that it could serve

as the test bed for emerging technologies. This evolution provided the groundwork for testing and developing more fully the concepts and processes. Both the testing and development underlie both the creation and the use of model-based, domain specific reuse libraries.

The CARDS program also realized that to achieve software reuse, education and training of all personnel involved in the software development cycle was necessary. From this lesson originated a training plan that was written for DoD and industry personnel, as well as undergraduate and graduate computer science and software engineering students. The plan and courses were developed to support the integration of library-assisted, domain-specific reuse into the development cycle. The purpose of this task is to educate software professionals and to assist in the elimination of cultural barriers (Technical Document, 1993).

The CARDS blueprint was designed with the consideration of future transition to other DoD organizations. The blueprint will serve as the overall plan for developing and supporting library-assisted, domain-specific infrastructures throughout the DoD. The future vision of the CARDS program is that of a virtual component library consisting of locally kept software components with a seamless, high-speed interoperation with other libraries. This would increase the number of assets available to the user and, in theory, reduce software development costs. This has been prototyped by CARDS, in

partnership with the STARS/ASSET and DISA/DSRS component libraries (Schwartz, 1993).

The CARDS program has taken well coordinated steps to assure that software reuse becomes an integral part of its software development cycle. The fact that the program is flexible enough to be ported to other DoD activities displays the thoroughness of the concept design and implementation. This forward looking approach allows the continual enhancement necessary for the program to remain viable in the future.

IV. FINDINGS

A. PURPOSE OF CHAPTER

The four examples described in Chapter III are a small sample of what is being accomplished in both industry and in DoD. Each of the examples have some commonalities, but each is distinctly different in its approach to software reuse. In this chapter we will draw out the similarities and differences of the examples and discuss these findings. From the examples, readings, and interviews with personnel in DoD and industry, there were three major areas of inhibitors to successful reuse that we encountered and propose to discuss in this chapter. These inhibitors can be grouped into legal, economic, and managerial issues. Each group of these issues, in its own right, can be enough to undermine the best software reuse initiative.

1. Software Reuse Origins

Each reuse effort in the examples from Chapter III tended to start with software reuse being conducted in small pockets internal to the organization. This sometimes occurred when a programmer used a part of a software program that he or she borrowed from another programmer and used it in a new project. More often it was the programmer reusing his or her own previously written software in another program (Emery and Zweig, 1993). Reuse was born out of these pockets of closet reusers and its value was recognized by management.

Management, swamped by the need to produce high quality products in a short period of time and at reduced costs, endorsed the idea of reuse.

The small groups of programmers who were doing reuse did so in an ad-hoc, unstructured basis. As reuse was consolidated at each of the four organizations, it became more formal and structured. This can be clearly seen in the DoD examples. The fundamental reason that CARDS and RNTDS were established was to provide a systematic approach and a structured framework to conduct software reuse. Out of the spontaneous impulse of a programmer to use a previously created piece of software evolved a sometimes mundane and bureaucratic formal structure. Each of the organizations realized this and sought to refine their reuse structure. IBM pursued a parts center geared to alleviating the search and retrieval frustration of programmers and project managers. Hewlett-Packard drifted away from the software repository to its factory concept, while RNTDS narrowed its focus on CDS commonalities and CARDS focused on using all forms of reuse.

B. LEGAL ISSUES

1. Legal Issues Found in the Evaluations

Reuse in industry has remained mainly an internal process because of the legal issues involved in porting reuse to outside users. The reason IBM gives for keeping software reuse internal is because of the legal issues related to copyrights and patents (Wasmund, 1994). Neither IBM nor Hewlett-Packard have sold their parts library or software kits to their customers. One can argue

that they feel that the technology is still in its infancy and not ready to be released to customers. "New methods and technologies can yield highly reusable domain-specific kits, comprising reusable components, frameworks, and glue languages. New processes and organizations can produce flexible and effective software factories. These approaches offer great promise for further gains, and a more systematic attack on the software problem" (Griss, 1993).

Hewlett-Packard and IBM have demonstrated so far that software reuse is a viable option in the development of software. Their research has been geared towards making internal software reuse - i.e., for use within the organization - a practical solution. "Before a product can be shipped, the origin of all its internal code must be clear. This means exclusion of public domain code in almost all cases" (Wasmund, 1994). At IBM, software is limited to only IBM-owned software. They have not explored the possibilities of porting software reuse to their customers, or at least have not disclosed such a possibility if they have. Both RNTDS and CARDS were originally designed to reside internal to the organization, but with their maturity are now being used as portions of each service's model for global implementation (Legal Workshop, 1993; Huber, 1994; GCN, 1994).

Another argument is that software reuse is an internal tool to develop and produce a better product and keep the companies away from costly lawsuits. By selling the parts repository or software kits, the companies would lose their competitive edge. "Increased and formalized reuse of software and related

assets is one essential method to allow faster delivery of high-quality products to the market" (Wasmund, 1993). Letting other customers produce their own software with the assistance of the software repository would limit the growth potential of the original owner of the software repository. When IBM uses outsourcing, the corporation tries to develop cross-license agreements under which both parties license their respective patent portfolios to each other (EDGE, 1993). In essence, IBM buys up the licenses it needs to keep its market dominance. For competitive reasons, IBM does not use software reuse when working with outside contracts unless all the rights are owned by IBM.

The most compelling reason for keeping software reuse internal is the multitude of legal issues related to it. Organizations are hesitant to open reuse to outsiders, because the law is not clear on how to deal with software reuse legal issues. "Generally, as the law is based on facts, legal assessment should only be made on detailed facts. However, it must be remembered that specific case law on computer software technology, in general, is lacking and for reusable software there is none" (Baxter, 1994). In the last few years there has been an increase in lawsuits brought against software and computer companies for patent and copyright infringements. There are two main lessons learned from these lawsuits. One is that the patent process is hopelessly outdated, which makes patent searches difficult to conduct. The other is that the judicial system of the United States is not familiar enough with computers, algorithms, and software to be able to make fair and just judgments.

There has been no case in the courts dealing with the legal complexities of software reuse (Legal Workshop, 1993). This may be contributed to commercial enterprises staying away from permitting outsiders to use their software reuse program. By keeping the reuse internal, the company can keep intellectual property rights to any module. Each programmer and analyst, when joining a firm, usually signs a statement giving the rights of any software designed by them to the company. Anything that is developed inside the company belongs to the company and is usually copyrighted by them.

Through our interviews and readings, we presume that the reason that IBM and Hewlett-Packard have emphasized internal software reuse is due to the legal issues involved in incorporating third-party software. DoD, on the other hand, has to resolve these legal issues due to the type of software (mostly contractor developed) it maintains in its software repositories.

2. Patents, Copyrights, and Liabilities

One of the problems with software repositories and software reuse initiatives is the content of these repositories represents the work of some person. The general term for this is intellectual property. Intellectual property can be protected by patents (used mostly for inventions), copyrights (applies to the expression of an idea), and trade secrets (Huber, 1994).

The difference between a copyright and a patent is that a copyright applies only to the specific expression of an idea, while a patent protects the idea itself (Bielefield and Cheeseman, 1993). Both the patent and the copyright

are used in guarding against the illegal use of software. Technically, any time a program is written, it is an expression of an idea and therefore copyrighted. The patent protects against the new idea being stolen and used by somebody else, even if expressed in a different manner (in another computer language, for example).

Why is there a need for patents and copyrights in industry? Some see copyrights and patents as ways of gaining monopolies. "The economic philosophy behind the clause empowering Congress to grant copyrights is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare through the talents of authors ... in science and useful arts!" (Bielefield and Cheeseman, 1993). These monopolies are intended to improve the quality of the product or promote standardization. Others believe that patents and copyrights protect small software development companies from being taken over by the industry giants. Without the patent process, small software companies cannot compete with the larger companies.

Another prevalent view is that patents and copyrights are seen as a hindrance to the software development process. Paul Eggert, in the misc.legal.comp. newsgroup on the Internet, summarizes it as follows: "Early experience suggests that patents are not a cost-effective way to promote progress in general-purpose computer software, and this is a principal reason why their use is so controversial. It is reminiscent to the controversy associated with the discredited earlier practice of granting patents on non-inventions."

Others build upon this argument. "Every single piece of software builds on the work of previous software developers" (Miller, 1994). Many in industry deem that copyrights can sufficiently protect software. Copyright laws seem to be easier to enforce, because the copyright lasts only three years. A monopoly is granted for the expression of an idea, not the idea itself. Instead, only intellectual property is guarded and software development can continue to be innovative and unrestrained.

The question is how to protect those rights. "If systems are built incorporating proprietary reusable components, how is the proprietary software protected from being plagiarized in delivered systems?" (Tracz, 1987). This is a question that has been asked in industry, but no clear-cut answer has been found. It appears that many companies pursuing software reuse avoid the issue by focusing on only internally created software. The company owns the patents and copyrights of software developed by the programmers who work for the company.

DoD, in contrast, is soliciting and placing software modules in its repositories that were created by contractors for use by other contractors. DoD realizes that there are problems with this and is working on agreements to limit copyright and patent liabilities for use of the software repositories (Huber, 1993). The users generally have to agree to pay royalty fees for copyrighted and patented software modules and release the repository of any responsibility in collecting these royalties.

"US copyright laws give the copyright to the author of a work, but the 'Work Made for Hire' doctrine enables the ownership of the copyright to be transferred to the employer when stipulated in writing" (Bennun, 1994). In recent years there has been a noticeable increase in the number of patents requested and received by corporations. "IBM received 1,085 patents in 1993 and displaced a Japanese company from first place in the US Patent Office's list of top 10 patent winners" (Chartrand, 1994). Not all of the patents were for software, but industry has realized that to protect their products, they have to patent them so that other corporations cannot reverse engineer them and develop their own products from this process. Patents also provide a means of keeping ideas developed at one corporation even if the person who developed the concept leaves and moves on to another company. "Most of the IBM patents are for multimedia, computer networks, storage devices and software and were developed by managers no longer with IBM" (Chartrand, 1994). It is detrimental to IBM's welfare for its former employees to take vital information with them when they leave. IBM, and other companies, protect themselves with employer-employee contracts when the employee begins work to prevent this from happening.

Liability applies to the process of how reuse modules are used. There are three bases of liability to be considered (Legal Workshop, 1993). First there is contractual liability, which arises from oral or written agreements. The risk with this form of liability arises when a term or condition of the contract is breached.

"For example, a subscriber agreement to a software repository provides the terms and conditions. If the subscriber fails to meet one of these terms and conditions, then there is a breach of agreement unless there is some defense which the subscriber can raise" (Legal Workshop, 1993).

The second basis of liability is statutory. This arises from copyrights, patents, and international trade and commerce. No contract is required for statutory liability. For example, statutory liability occurs when a federal copyright law is violated, and the infringer is not even aware of it (Legal Workshop, 1993).

The last basis of liability is tort, which pertains to a legal duty that must be extended to participants (Huber, 1994). "For example, a library of reusable software components which holds itself out to a community of subscribers to be qualified to conduct testing for conformance of components of standards has a duty to apply standard of care which pertains to a person assuming the role. That is, the library, having undertaken to do the testing and certifying of components for conformance based on the results of that testing, has a duty to adhere to that standard of care appropriate for that role" (Legal Workshop, 1993).

Liability comes into play after a software reuse program has been established. How to deal with these liabilities has to be discussed and decided prior to the program being used. CARDS has made assertive efforts to resolve liability issues, present in DoD, by conducting workshops and briefs regarding these issues. Through these efforts, the CARDS program has found that

developers, users, and maintainers of DoD repositories, as well as legal counsel, are not knowledgeable regarding software reuse legal issues (Huber,1993).

Since copyrights and patents for software belong to the corporations, the only issue in industry is to ensure that the new product developed using software reuse does not infringe on any other corporation's patent or copyright. In contrast, DoD does not own any of the copyrights or patents for the software developed by its contractors. Instead, DoD is able to use the software throughout the services on an unlimited basis if they have unlimited distribution rights (Huber, 1994). The contractor who developed the software application is the holder of the patents and copyrights associated with the application.

3. Software Repositories

Each of the four organizations in Chapter III required a means of storing reusable software for use in the software reuse program. Most were similarly based on the software library theory of reuse - that the keys to reuse are a large library of objects within the application domain, and a catalog to help locate them as needed (Banker, et al., 1993). From this the software repositories were established and populated. Each of the four examples started their reuse program with a software repository as its centerpiece. Criteria for how to place software modules into the repositories were developed and the software modules were evaluated to determine their maturity level. Procedures for storing, cataloging, and retrieving the modules were designed by each

organization. Hewlett-Packard is the only company of the four that is moving away from the software repositories and the library metaphor (Griss, 1993).

The software repository has some obligations and responsibilities when it comes to storing and disseminating the modules contained within it. First of all, the library has to obtain the appropriate rights from the supplier. Second, the library has to ensure that the component is marked properly - i.e., with disclaimers, copyright information, or patent numbers. And most importantly, the library has to ensure that the user is made aware of the status of the original owner's rights (Huber, 1994). If the software repository is internal to an organization, then the obligations and responsibilities are not required to be written in legal jargon. At IBM, the user of the repository is more concerned with the quality of the part than the royalty. Since the company owns the rights, it would be paying the royalties to itself. Management can mediate disputes between two development teams and resolve the issues without going to court.

a. Access

In order to regulate the access to the software repository and decrease risk, it is best to limit access to only certified and registered users. In the case of DoD repositories, this may be difficult. There is a question as to whether the information stored in the DoD repositories are subject to the provisions of the Freedom of Information Act (FOIA) which allows public access to unclassified information. To date there has been no court case challenging the public's access to software repositories (Baxter, 1994).

The concern of access to the repository is to allow only authorized users to access the software modules. If there is no mechanism in place to control access, then the library cannot guarantee its content, because any person can alter the module, or delete or add faulty software modules. The public's right to access the information in the repositories, through FOIA, can lead to many legal complications.

Additionally, a mechanism is needed that lets the repository interact with the user. This enables the repository to obtain the necessary information from suppliers, subscribers, and other libraries, and manage the repository better (Huber, 1994). The software module information contained in the repositories is important to the user as a measure of quality and legitimacy. Repository access is also important to internal software reuse. The difference is that the user is more concerned with the quality of the module than with the copyright of the module. Yet the same mechanism as discussed above is needed to guarantee the quality of the module. Whether the repository is internal to an organization or has external users, access to it has to be regulated to ensure the quality and completeness of the modules. Users should not have unlimited reign to manipulate modules.

b. Content of Repositories

Another aspect of the repository that can cause legal headaches is the content of the repository. One way to reduce the risk of legal action is to have the supplier attest in writing to the module's accuracy and completeness. The supplier should also certify that they have the authority to submit the module. A

standard procedure is required for the certification process. "If there are no standards to control what is entered into the component library, then time and money must be spent setting and maintaining the standards" (Chandersekaran and Perriens, 1983). Yet even the best certification process will miss an error on occasion. Dealing with third-party software modules complicates the issue even more. If those modules are copyrighted or patented, then the library has to track the use of them and develop a method for ensuring that the software developer is properly compensated for the use of the module. RNTDS and CARDS have their own certification processes, but this is not the case for all DoD repositories. A recent market study, conducted by the CARDS program, indicated that less than 50 percent of the personnel surveyed used a minimum set of criteria to evaluate reusable components prior to placing the modules into the repositories (Market Study, 1994).

c. Who is Liable for Faults

"If a defect appears in a program developed using reused components, who is legally responsible for damages?" (Tracz, 1987). This is another question asked often in industry, but it has not been answered because of the legal issues involved. Again it is easily solved if the software repository is located within the same organization. The quality of the software repository relies on the quality of each software module.

The question arises, if a faulty component is in the library and an application developer uses it in his product and the product fails, who is at fault?

What if that developer, due to that error, defaults on his government contract because of a module he received from a government software repository? "Courts have rejected the theory that persons in the computer industry are subject to professional malpractice standards" (Baxter, 1994). Since there is no established legal precedence for software reuse, each case will have to be examined individually.

Another issue to be considered when starting a repository is whether it will be operated by the government or a contractor. If it is contractor-operated, a method has to be developed for the contractor to evaluate other contractors' software modules in an unbiased, objective manner. If the method is applied incorrectly, there may be a potential conflict of interest (Legal Workshop, 1993).

These are some of the issues facing CARDS as its repositories are activated. RNTDS did not experience similar problems because, like IBM and Hewlett-Packard, their work was internal to their organization (Stevens, June 1994). CARDS is the only repository of the four examples that has expressed concerns about the legal issues involving third-party software modules.

4. Opening Software Reuse to Outside Users

As long as software reuse is accomplished in-house under the umbrella of one company, the legal issues are manageable. Opening up software modules to outside users raises new legal issues. The same holds true for bringing in outside software modules to be reused in-house. "Incorrect contracting mechanisms actively discourage reuse. The lack of contracting mechanisms

makes it hard to create agreements that can be trusted or enforced. Increasing the use of third-party software increases the importance of this issue. The solution may be to develop new contracts, maintenance agreements, and royalty systems" (Griss, 1993).

Hewlett-Packard realizes that for software reuse to be ported outside of the company, legal and business practices have to change. Internally, Hewlett-Packard is working on agreements between different divisions to facilitate software reuse. This is to protect both the supplier and user. DoD has come to the realization of this also and has put in place disclaimers, subscription agreements, and supplier agreements (Huber, 1994). Since this is virtually new legal territory, it is best to take extra precautions and relax them as precedence is established.

5. Effects of Legal Issues on DoD Reuse Program

Since DoD is currently pursuing a software reuse program that lets outside users browse the modules, it has to deal with many legal issues not faced by industry (Legal Workshop, 1993). One example is that of a contractor being told by the contracting office that reusable components are available in a software repository. The contractor spends money to reuse the module, only to find out that it is defective. The library can be held liable depending on the mitigating circumstances of the scenario. To minimize these issues, the government has to reduce the risks involved in running a software repository. For instance, library risk depends upon the library activities. In order to avoid lawsuits, the repository

needs to have clear and definite operating procedures that are flexible enough to cover even the most obscure situation.

The case of the DoD software repositories is made more complicated because many of them are government-owned and contractor-operated. The components or modules in the software repository are gained from government agencies, commercial organizations, academia, and individuals. Depending on the supplier, copyrights and patents have to be enforced and royalties paid. The user is either a contractor working for the government, or a government employee, or a casual browser not related to the government at all. Depending on the user, certain rules and agreements have to be followed. The library has to maintain oversight of the suppliers and users and ensure that the operating procedures are being followed. Once again, the most foreboding fact is that there has been no legal case involving software reuse that has been heard in court (Legal Workshop, 1993). This means programs like CARDS and RNTDS will provide the lessons learned for future software reuse development programs.

C. ECONOMIC FACTORS OF SOFTWARE REUSE

1. Software Reuse Economics

The economic payoff of software reuse in DoD, as in industry, comes from the promise of increased productivity and quality. Systems that are developed with reusable software components should cost less and contain fewer defects (Tracz, 1987). The promise software reuse holds for potential economic benefits

has led to its pursuit by DoD and industry. This was largely in response to conventional development practices, in DoD and industry, having the following problems (Abdel-Hamid and Madnick, 1991):

- Exceeded budget and schedule.
- Maintained legacy systems that absorbed available appropriations.
- Consisted of inflexible applications unable to adapt to changing needs.
- Were abandoned after millions of dollars were already expended on them.

There have been many Government Accounting Office (GAO) studies and various Congressional reports that have provided a chronicle of the ineffectiveness of software development practices in DoD. The above factors, coupled with the dramatic increases in computing power due to technological advances in hardware and the proliferation of personal computers, have propelled these organizations to search for better development methods.

The search has led to software reuse with its enticement of producing software that would cost less and be of higher quality (Tracz, 1987). Software reuse has become an economic focal point for all four of the organizations included in the study. In both the DoD and industry examples it was understood that for software reuse to be successful the process must produce economic benefits while overcoming some of the early barriers to its adoption.

2. Economic Barriers

One potentially harmful barrier to software reuse is the high initial costs associated with such a program. Software reuse requires that the software

development team change to realign themselves with the new development approach. Effecting these changes will mean assuring that members receive education and training in software reuse methods. The Hewlett-Packard and IBM programs realized that the conversion and education of personnel concerning software reuse required an up-front investment. Most software reuse projects require several years of investment before savings are realized. This causes managers to be reluctant to make long-term investments without some assurance of success (Griss, 1993). DoD has made initial investments in software reuse and has recently come to the realization that acceptance of reuse begins with education and training (DoD, 1992). The CARDS program is rectifying this oversight and is placing great emphasis on providing education and training for all development personnel at all levels (Technical Document, 1993).

We found that software reuse, as a long-term investment, has to be adequately articulated to management. Congress and DoD upper management, who expect savings to appear in the near term, will find the higher initial costs an impediment if the process is not thoroughly communicated to them. They need to understand that the cost of producing reusable components and implementing the reuse process must be written off over the long-term. Hewlett-Packard and IBM have achieved this and are in the process of streamlining their programs (Griss, 1993; Wasmund, 1993). Management of the RNTDS and CARDS programs understand the need to communicate the benefits (and costs) of

software reuse. CARDS has taken innovative steps to produce handbooks and establish training courses to educate personnel on the software reuse process and its long-term benefits (Technical Document, 1993). The RNTDS process system is being used as the foundation for the Navy's Software Reuse Implementation Plan and Software Reuse Guide (GCN, 1994).

Repositories, in most instances, are populated with reusable components that must be acquired, stored, validated, catalogued, indexed, documented, transmitted, and maintained. These services are costly and will need to be paid for either by cost recovery fees or subsidy by a central agency (Emery and Zweig, 1993). Populating the repository requires that a search be conducted to find reusable candidates. This is costly because there exists no standard conventions for naming, indexing, documenting and designing reusable candidates (Emery and Zweig, 1993). In fact, both the RNTDS (Stevens, 1991) and CARDS (Technical Document, 1993) programs use disparate methodologies for achieving software reuse that further complicates the usability of the repositories for a programmer in a global operation. With Hewlett-Packard and IBM, the cost of a software reuse program is recouped by passing the costs to the project. They found that the longer the program is in place the less the cost and the number of defects. This is due to more and more reusable parts being used and less original work having to be accomplished (Griss, 1993; Wasmund, 1993; Poulin, et al., 1993).

Maintenance of software repositories and their components is a continuous effort that is expensive to sustain. In the future, maintaining repositories may not be exempted from the Defense Budgeting Operating Fund (DBOF), which requires activities to be fully reimbursed for all costs related to their services. The current practice for the RNTDS and CARDS sponsored reuse systems is for each to subsidize their own efforts with no fee charged to the user. Hewlett-Packard and IBM, which are profit-based, have no delusions in regard to who will pay for the use of repositories. IBM uses their own accounting methods to charge the other users within the company (Wasmund, 1993) and Hewlett-Packard's software reuse division did not charge others originally, but is actively pursuing their own research in accounting for repository operations (Malan, 1993; Griss, 1993).

3. Incentives

Many forms of incentives were used by Hewlett-Packard and IBM that met with various levels of success. Incentives in industry have taken both monetary and non-monetary forms. Both have used incentives to populate software repositories with modules for reuse. Hewlett-Packard's experience with this type of program led to the population of a repository that collected large quantities of software components regardless of their quality. They found that while the size of the library increased, the amount of software reuse did not. Programmers were willing to contribute components, but not to search the repository and use them in their own projects (Griss, 1993). Hewlett-Packard is in the process of

reworking their incentive programs. Malan (1993), of Hewlett-Packard, provides a detailed discussion of the aligning of reuse incentives with reuse goals and the organization's objectives.

IBM's experience in the incentive arena began with what they called a Non-point based program. The program granted monetary or other awards to selected individuals or teams to reward successful cases of software reuse. The criteria were qualitative in nature, emphasizing innovation over more quantitative measurements such as cost savings or quality gains. Over the course of a year, there were three reuse awards submitted and only one was granted. IBM found that schedules did not encourage making software reusable even if a programmer spent his or her own time producing it. The incentive to contribute a component was reduced because the component had to be reused prior to receiving an award, adding both uncertainty and a time lag. They found the impact of this method of determining incentives led to only minor success and did not significantly change people's adherence to traditional development methods (Wasmund , 1993).

Realizing little progress was effected by the Non-point based program, IBM mandated software reuse by establishing a formal reuse target for the organization and included the targets as part of individuals' performance plans (Wasmund , 1993). IBM established a program, named the Point based system, that rewarded an individual practicing reuse with monetary awards after the accumulation of a required point total. Practicing reuse meant either using

available components or producing reusable components for deposition in the repository. This led to an immediate increase in the number of requests for reusable components. This program forced the professional to depart from the traditional development process to support the reuse target and management team (Wasmund , 1993). However, there were also some side effects to this program:

- Some organizational units overdid the setting of targets by setting contribution targets regardless of the work individuals were conducting.
- Unprepared application of complex reusable components caused integration problems because of the zeal of some groups to implement reuse without appropriate training.
- No differentiation in the application of the target across the various groups caused a perceived inequity because the individual groups' actual ability to use reuse was not considered.
- The method of assigning a target caused great confusion about what the target meant.
- Individuals tried to enhance their personal reuse by integration of extra large components whose entire function set was not needed by the program (Wasmund , 1993).

Notwithstanding these side effects, there was a marked difference between the two programs. It was clear to IBM that introducing a new technology through the establishment of a sensible quantitative target was

successful for their particular organization, but that it could result differently for other organizations.

Incentives have been more successful in industry than they have been in government. In government, the use of incentives has come face-to-face with some inflexible barriers. Foremost of these barriers are the restrictive acquisition laws that DoD has to obey. Most software development for the government is contracted out to vendors, and current acquisition regulations hinder giving incentives for reuse (Endoso, 1992). Even when software is developed internal to DoD, incentives face many obstacles based on the existing civilian and military personnel regulations. There are many award programs that are detailed in appropriate personnel directives, but these are generic and do not consider software development or reuse separately. The RNTDS and CARDS programs do not have, nor can they maintain, the same flexibility that industry has to reward people for contributing to the reuse effort. When DoD does resolve the incentive problem, a system similar to IBM's Point-based program should be considered because it only rewards personnel who practice reuse.

4. Economic Models

The development of an economic model to accurately measure software reuse has been an elusive goal for both DoD and industry. The lack of economic metrics to measure software reuse has resulted in one of the major inhibitors of an integrated software reuse program (Poulin, et al. 1993).

Economic models are required for a program manager or software development organization to effect decisions pertaining to software reuse. Measurement usually accounts for the software development process and does not consider software reuse. Software reuse measurements have generally been measured as a ratio of reused code to the total code present in a particular system (Banker, et al., 1993). Though many models have been published in the literature, they tend to focus on limited aspects of the reuse process, usually after completion of the project. Our research found that none of the organizations we studied was found to be using a common economic model to evaluate its software reuse program. Each organization uses varying methods of measuring its economic success and the measurements of one organization cannot effectively be compared to another in most cases.

There does not seem to be a consensus, in industry or DoD, on what economic model of software reuse is suitable for capturing the cost saving associated with reuse. Organizations use economic models of software reuse to assist them in making managerial decisions throughout the development process. The measurement tools used by the organizations vary with some using internally-developed models and others using previously published models. IBM uses an economic model designed internal to its Critical Success Factors approach (Wasmund, 1993), Hewlett-Packard has its own model (Griss, 1993), RNTDS uses Gaffney/Durek and SEI's Software Capability Maturity Model (Stevens, June 1994), and CARDS uses both of the models used by

RNTDS, as well as their own cost model used for the preliminary stages of reuse (Direction-Level Handbook, 1994).

5. Effects of Economic Issues on DoD Reuse Program

Industry and DoD differ greatly in the manner in which software reuse is being handled. In industry there is a more dynamic interpretation of software reuse. Even though Hewlett-Packard and IBM have established reuse programs, they are continuously researching new and emerging technologies and how they can effect software reuse. Through this type of research, Hewlett-Packard has moved away from the use of software repositories. DoD, in most cases, has a more static approach to software reuse. To date, DoD has emphasized only one way to do reuse, and that is to use software repositories. CARDS, however, provides some encouragement with its emphasis on using the program as a testbed for emerging technologies (Direction-Level Handbook, 1994).

The cost savings that the software reuse program has promised have not yet been realized. This is due to initial investments into a software reuse program eating away the early cost savings. This initial investment also does not guarantee that the program will be a success. Compounding this is a lack of a corporate infrastructure that encourages and rewards reuse and the fear that higher degrees of reuse may lead to reduced staffs (Kim and Stohr, 1992). These factors make the acceptance of software reuse difficult to accomplish in any organization, not just DoD.

D. MANAGERIAL ISSUES OF SOFTWARE REUSE

Software reuse has evolved to become a corporate- or department-wide concept in industry and DoD. At IBM and Hewlett-Packard, for reuse to work, there has been support from the highest levels of management. Each of the organizations wants to eventually achieve software reuse in a global context. Prior to going global and pursuing a world-wide reuse program, industry recognized the need to accomplish organizational changes and changes in the work culture (Tracz, 1987). Without making the necessary changes the effort is attempted in a halfhearted manner and can fail to be completed.

DoD has not fully recognized these problems and has not embraced software reuse as wholeheartedly as industry. "For example, at a jointly sponsored workshop by the Software Productivity Consortium, the Microelectronics and Computer Technology Corporation, the Software Engineering Institute, and the Rocky Mountain Institute on Software Engineering, attendees unanimously agreed that management generally has a shortsighted view on software development and is often not willing to commit resources to acquire needed tools and training in software reuse technology" (GAO, 1993). DoD's reuse program is more erratic, with top management requiring software reuse without fully understanding the issues, especially the non-technical issues. Although the vision of how reuse should be conducted has been articulated in a vision strategy (DoD, 1992), no specific guidelines have been established for the reuse effort.

Industry and DoD, the latter only recently, have realized that for software reuse to become an effective tool, all inhibitors have to be addressed (Endoso, 1994). There are many problems that can crop up when implementing a reuse effort. They can be of a technical, managerial, or socioeconomic nature. Neglecting any one of these inhibitors can undermine the best reuse effort. As DeMarco and Lister have noted, we too often focus on the technical rather than the human side of work, not because it is more crucial, but because it is easier (DeMarco and Lister, 1987). Within each of the organizations these inhibitors are being addressed, and though the solutions may differ, DoD has recognized them and is now seeking to eliminate them.

Education is another important factor in the successful implementation of a software reuse program. Each of the organizations emphasizes the importance of not only educating the technical staff, but also the managerial personnel. Education needs to be geared to the respective audience so that it helps institutionalize the reuse concept. The CARDS program provides the best example of the effort needed to educate and train personnel involved in the software development life cycle. CARDS has emphasized training of technical, managerial, and acquisition personnel. CARDS has even gone as far as to develop graduate and undergraduate courses in software engineering to assist in educating people concerning software reuse (Technical Document, 1993). This is noteworthy, because a recent survey found that very few respondents

had learned about reuse through their own education and training process (Frakes and Fox, 1993).

1. Upper-Level Management Involvement

Upper-level management needs a planned approach to support software reuse in a perceptible manner for it to be creditable to the organization. This approach should follow the classic phases of technology transfer: increasing awareness, cultivating interest, and persuading someone to try the technology, followed by prototyping and then complete adoption (Basili and Musa, 1991). When personnel see that upper-level management is supportive of the program, they will be much more likely to discard the "not invented here" syndrome. Additionally, upper-level management must comprehend the complexities of software reuse and the associated technical and non-technical problems. Management can be supportive of software reuse, but if there exists no baseline understanding among managers then it will be hard for them to institute reuse and guide implementation.

2. Reorganizing the Organization

In order to achieve the promised productivity and cost savings of making software reuse a standard organizational process, the organization needs to establish a support structure as part of the software development process (GAO, 1993). One way to acquire this infrastructure is to create a central support staff organization. It has been observed in case studies that the attainment of a high level of reuse requires the entire organization to be oriented to the goal (Banker,

et al., 1993). Organizations committed to the idea of software reuse must not only change their way of developing software, but must make fundamental changes in the manner in which they are structured.

In order to change the organization, the way management assigns projects has to change. "Survival from a managerial perspective has meant handling multiple, concurrent, and conflicting goals" (Hyman, 1993). Management has to reduce red tape and overhead and encourage rapid prototyping and responsiveness, while maintaining the necessary supervisory controls for reliability and quality (Hyman, 1993). Management has to juggle the assets and determine the best way to apply each asset. Software reuse adds another ball to the juggling act. Currently, industry is grappling with these issues. Hewlett-Packard and IBM developed and implemented, respectively, software kits based on the LEGO building block concept and Critical Success Factors to change management styles so as to incorporate software reuse. In DoD, "project managers and software developers must be willing to make fundamental changes in the way they develop software" (GAO, 1993). In other words, all management levels have to be included to effect the necessary changes.

IBM and Hewlett-Packard have been committed to making the necessary adaptations to their development processes as they have discovered the need (Griss, 1993; Wasmund, 1993). RNTDS has been so successful that the Navy is developing its Software Reuse Implementation Plan and a Software Reuse Guide based on the RNTDS system (GCN, 1994). CARDS has also blazed new

paths in understanding organizational challenges and designed a program that can be acquired and implemented by any organization in DoD (Technical Document, 1993). All four of the organizations researched have recognized the compelling need to restructure their software development infrastructures and has modified them as each circumstance has dictated.

V. CONCLUSION AND RECOMMENDATIONS

A. DETERMINATIONS

1. DoD Software Reuse in General

In January of 1993, GAO presented a report to the Chairman, Subcommittee on Defense, Committee on Appropriations, House of Representatives, on the DoD Software Reuse Initiative. The report highlighted the discrepancy between what the initiative promises and what is actually achieved. There are many points made in the report that illustrate the contradictions that were found.

As discussed earlier in this thesis, the report found that there are no standard methods for domain analysis or classifying software for repositories nor consistent software metrics, yet all of these are considered important aspects of the initiative (GAO, 1993). The GAO report went on to further discuss non-technical barriers to software reuse, such as higher initial costs to develop reusable software, lack of management support and commitment, and various potential legal issues which could encumber software suppliers, repositories, and users.

All software reuse goes through growing pains. IBM and Hewlett-Packard did not achieve a global software reuse program overnight. The same issues that are being discussed in the GAO report were also issues in IBM and Hewlett-Packard. The difference is that these two corporations have recognized

these issues and have fixed them while pursuing better ways to conduct software reuse. We conclude that each organization moves through these stages when implementing a software reuse program.

Though there exists a software reuse initiative, the initiative does not provide clear direction or documentation from higher authority detailing how reuse should be accomplished in DoD. Each of the services is energetically pursuing reuse, but is accomplishing it with their own independent, versus integrated, guidance. This has caused a duplication of effort, expenditure of valuable shrinking assets, confusion among programmers due to the various methodologies, lack of cooperation in sharing solutions to technical and non-technical issues among the services, and the proliferation of systems dissimilar in their operations.

For software reuse to be successful, it must be understood by all levels involved in software development. Hewlett-Packard supports this concept. "A fairly broad, well-coordinated software reuse program involving management, process, and technology was needed to make significant progress (Griss, 1993). The news is not all bad because there are some good systems in the DoD inventory that should be emulated and distributed to DoD sites. The two DoD programs that were discussed in this thesis are exemplary models of the progress that has been made on the individual program scale.

2. Positive Effects of Software Reuse in DoD

a. Support for Software Reuse

The efforts in DoD to incorporate software reuse as part of the software development process are not lacking. Each of the services has its own software reuse programs as part of DoD's remote reuse centers (Bui, et al, 1993). Many of the different service personnel engaged in developing software were recently surveyed and foresee software reuse as necessary due to budgetary and resource constraints. Many of these same participants believe that there is support for software reuse at all levels of DoD management (Market Study, 1994).

By virtue of the different DoD programs there is not a lack of assets being appropriated for the pursuit of software reuse. Even with the obstacles presented by widespread software reuse, DoD officials feel that the potential savings are worth investing the time and money to solve the problems (Endoso, 1993). The Honorable John Murtha (D-Penn), Chairman of the House Appropriations Subcommittee on Defense, recently offered to increase funding for software reuse efforts (Endoso, 1994). It is apparent that DoD is willing to expend the funds to make software reuse a functional part of the software development process.

b. Software Reuse Horizon

The services have been conducting software reuse and implementation efforts since the introduction of the software reuse initiative in 1991. These efforts have been individual, and they have experienced similar troubles

associated with obtaining an employable software reuse program as in industry. Hewlett-Packard's first step towards the inception of a software reuse program was to establish a software repository, but none of the programmers used it (Griss, 1993). IBM tried a volunteer program of software reuse, but found that few programmers were willing to use it even with the offer of incentives (Wasmund, 1993). Similar troubles in DoD have not escaped the scrutiny of the House Appropriations Committee, which last year, as part of the 1993 Defense appropriations bill, commented, "The department with its decentralized approach, runs the risk of permitting the many organizations participating in reuse initiatives to misdirect or duplicate reuse efforts" (Endoso, 1994).

These disjointed, independent efforts have been recognized by DoD as being counterproductive in nature. In a recent Pentagon report to the House Appropriations Committee concerning the initiative, DoD discarded the original voluntary software reuse initiative for one that will assign roles and responsibilities to all service components. The plan, dubbed the Software Reuse Initiative, or SRI, calls for developing an infrastructure, bringing reuse technology into the mainstream, and encouraging, rewarding and institutionalizing effective software reuse. The new SRI plan is projected to be completed by September 1994 according to Pentagon officials. These objectives will be the responsibility of a new software reuse program office within the Defense Information Systems Agency (Endoso, 1994). This is a step in the right direction, because focusing the attention of the individual services will allow

them to concentrate on one aspect of software reuse. This should alleviate many of the duplicate efforts and allow for better dissemination of information.

Another possible contribution to software reuse is an economic model developed and researched, with the assistance of students, by Dr. Tarek Abdel-Hamid of the Naval Postgraduate School. The Dynamica Reuse Model is a computerized program that simulates a software development organization practicing organization-wide software reuse (Abdel-Hamid, 1993). The model has three significant characteristics that differentiate it from other published economic models. The model integrates the complex technical and managerial functions required for organization-wide software reuse; it provides feedback using the principles of system dynamics to better comprehend software reuse organizational complexities; and it uses computer simulation to handle over 200 different equations integrating hundreds of variables relating to technical and managerial issues in organization-wide software reuse (Gallup, 1994; Abdel-Hamid, 1993).

The major benefit of this model is that it actively integrates software reuse into an economic model for software development. It not only brings software reuse into the mainstream of software engineering, but also standardizes the economic benefits available through software reuse. This model has not reached full maturity yet, but is more robust than any other published model that integrates software reuse and holds promise for the future. Further research is being conducted to expand the model from a single

organization simulation to multiple organizations engaged in the process of software reuse as a group (Gallup, 1994).

3. Negative Effects of the Software Reuse Program in DoD

a. DoD Software Reuse Initiative Infrastructure

The DoD Software Reuse Initiative has been a voluntary program that did not require the services to pursue software reuse as part of their software development process. Software reuse was envisioned to occur whether DoD took an active role or not (DoD, 1992). As documented by GAO (GAO, 1993) and other published literature (Banker, et al, 1993), software reuse will not flourish if there is no support from management. At IBM the software reuse program flourished once everyone in the organization from top management to the programmers understood the potential benefits (Wasmund, 1994). IBM and Hewlett-Packard, as well as other researchers, found that lack of a clear software reuse strategy has been a major factor that inhibits the institution of the reuse process (Griss, 1993; Poulin, et al., 1993; Biggerstaff and Richter, 1987). Moreover, management needs to recognize that software reuse is more than a method to reduce costs; is part of the greater software development process. Software reuse should not be viewed as a cost-cutting method. Instead it should be included from the start of a software project and used throughout to achieve future benefit.

Prior to introducing software reuse in a global context, a software reuse infrastructure that reduces the costs of implementing and operating a software reuse program should be developed. DoD's software reuse program has not

taken this approach. In fact, DSRS populated a repository before reuse was incorporated as part of the development process. Research has found that organizations successful in producing high levels of software reuse are relatively small and located in one geographic area (Tracz, 1987; Kang and Levy, 1989). Software reuse, in other words, was established and matured in one location before it was implemented throughout the organization.

At both IBM and Hewlett-Packard, the software reuse programs were started as pilot projects. Once the pilot projects had proven themselves, the corporations expanded the pilot projects into the global strategies for software reuse. Most thriving and effective software reuse programs begin small, are funded from the start, and have acquired their experience through pilot projects (Griss, 1993). The point here is that DoD should concentrate on a pilot project and work out the many legal, economic, and managerial issues before they implement a global reuse program.

b. Repository Concerns

(1) Domain Analysis. The maturity of repositories is another area where DoD has experienced many growing pains. Domain analysis is one of the key elements of the SRI strategy, yet there exists no DoD standard method on how to process and represent information about a domain (GAO, 1993). The domain analysis process itself is more of an art than a science, and only with time can applicable design decisions occur that optimize the design for the purpose of reuse (Prieto-Diaz, 1990). IBM and Hewlett-Packard have realized the need for a structured standard method and have established their own

formal methods to conduct domain analysis at their companies (Poulin, et al., 1993; Griss, 1993). Lack of a standard DoD method for domain analysis has produced several sets of methodologies and program implementation philosophies that are unique to each service.

(2) Certification and Search Tools. Similar to the domain analysis problem are the certification and search processes for reusable software components. Certification requires certain characteristics to be present in a reusable component prior to inclusion into the repository. The DSRS/RAPID programs use varying "levels of confidence" dependent on the components quality (Bui, et al, 1993). CARDS and RNTDS use other means to certify their reusable software components. In DoD's global concept each certification scheme would require user training. Users do not want to spend the effort to learn the various aspects of each scheme. Multiple certification schemes and dissimilar search tools will only discourage use of the system.

Hewlett-Packard and IBM both found that users will be hesitant to use a system that they have no confidence in because the certification process is ambiguous - i.e., no universal standards are followed (Poulin, et al., 1993; Griss, 1993). Well documented, tested, verified, and classified reusable components need to be developed for programmers to have confidence to use them (Tracz, 1987). Unless a standard certification and search tool is used at each of DoD's repositories, a user will be frustrated by new jargon and unfamiliar methods. The user will be reluctant to accept the risk of using a reusable

software component without first spending considerable effort verifying that it is safe to use (Emery and Zweig, 1993) or forego even using reusable software and build it themselves (Banker, et al., 1993).

(3) Legal Concerns. Legal questions regarding establishment and operation of a repository have surfaced and have not been adequately addressed. The CARDS program sponsored a workshop for government lawyers to discuss the legal aspects of operating a software reuse library to attempt to resolve these various issues (Huber, 1993). This workshop displayed a lack of knowledge, on counsel's part, regarding software reuse. In general, however, there is a lack of legal cases dealing with software issues. In Chapter 2, some of these legal issues were discussed. Clear, concise legal counsel is necessary to assist in solving software reuse legal issues, but this will prove difficult until precedents are set through individual cases (Legal Workshop, 1993).

c. Acquisition Regulations

Many legal problems find their foundations in the DoD acquisition process. There have been two studies concerning the DoD acquisition process and the legal ramifications regarding software development and reuse. Both found that changes to the Federal Acquisition Regulation and the DoD Federal Acquisition Regulation Supplement were necessary to effect software reuse (CSRO, 1993; STARS, 1991). Changes to the regulations are not easy to make because they can become political points of contention.

DoD acquisition regulations also fail to provide incentives for contractors engaged in the software reuse process (Endoso, 1992). IBM found that its software reuse program gained more use once a fair and equitable incentive program was established (Wasmund, 1993). Incentives are necessary to motivate contractors to reuse software, who must then reward their own personnel. Incentives are also hard to award to government personnel due to the legal restrictiveness of personnel regulations and guidelines. Awards cannot be established for one specific program and are up to the approving authority's whim. Additionally, there are no provisions in any of the appropriate governing regulations measuring the amount of award.

d. Education and Training

Software reuse implementation should start with education and training for all individuals involved in the software development process. Both IBM and Hewlett-Packard recognized that to bridge their non-technical inhibitors they had to develop an educational program. Everyone related to the software development process, from the programmer to top management, was educated in the software reuse process (Griss, 1993; Poulin, et al., 1993). Training needs to be addressed as early as possible in the life cycle so that software reuse is understood. No concept can be truly successful unless there is someone who can properly implement it (Direction-Level Handbook, 1994). Experiments conducted with software development personnel have led to the conclusion that personnel untrained in software reuse cannot truly assess the quality of a reusable component. The experiments also found that the participants were

influenced by minor features and not by the important features of reusable software (Woodfield, et al., 1987).

Some progress has been made in educational efforts by some DoD affiliated programs. SEI has been involved with some colleges and universities in establishing software engineering programs at the undergraduate and graduate levels (Direction-Level Handbook, 1994). The CARDS program has developed education and training courses for DoD and contractor personnel (Technical Document, 1993), but like much of the other training programs, these courses do not receive the visibility they should. Training programs are available, but need to be advertised more aggressively in order for more people to become aware of them.

B. CONCLUSIONS

This thesis has attempted to address the software reuse issues in DoD by analyzing reuse efforts at two companies in industry and two programs within DoD. Software reuse is being accomplished in industry and in DoD, but as acknowledged by Rear Admiral John Hekman, Commander of the Naval Information Systems Management Center, "Software reuse is not even close to being standard operational procedure" (Endoso, 1994). As learned from the examples of IBM and Hewlett-Packard, DoD is experiencing the same growing pains as in industry.

IBM and Hewlett-Packard have evolved their software reuse program into a more robust and mature software development tool. DoD is still faced with many

economic and managerial issues such as a definite implementation plan, standardization of domain analysis, certification of reusable software, search tools, education, and training programs. Industry and government must still untangle the myriad of legal issues presented by software and software reuse. When it comes to legal issues, DoD has to resolve more complex situations due to the composition of the software repositories.

Industry - specifically IBM and Hewlett-Packard - started with small reuse projects located in one geographical location. Once the worth of the reuse project was demonstrated, it was expanded as a global concept for use by the entire corporation. DoD, in contrast, jumped straight into trying to implement a global reuse program without full consideration of other established programs and the lessons they taught. Widespread DoD software reuse was attempted before personnel at all levels understood what it was and how to achieve it. This approach has proved difficult because issues have tended to become magnified at this level of enforcement. DoD needs to learn from industry's lesson learned and apply the recommended changes of this thesis to its own reuse effort. Only then will DoD be able to achieve local acceptance of software reuse, then expand it to a global concept as it matures DoD-wide.

C. RECOMMENDATIONS FOR FURTHER STUDIES

1. Software Development and Reuse Legal Issues

Though this thesis covered many different legal issues pertaining to software reuse, it was by no means an exhaustive review. There still exist many

unanswered legal questions relating to the software development process, of which software reuse is just a portion. Some good ground work has been achieved by DoD (Legal Workshop, 1993; Baxter, 1994; Huber, 1993 and 1994) in the arena of software reuse, but this needs to be expanded to include software development.

2. DoD Acquisition Regulations

The software development life cycle is governed by DoD acquisition regulations, which do not adequately address software reuse. There is a need to change the regulations, and some preliminary work (CSRO, 1993; STARS, 1991) has been conducted. Other research has resulted in the development of handbooks to assist acquisition personnel with the inclusion of software reuse into the development life cycle (Technical Document, 1993). An interesting next step would be to see how the regulations need to be modified to incorporate incentives and solve the legal issues surrounding software and software reuse.

LIST OF REFERENCES

"Current FAR and Budget/Finance Environments", STARS-AC-03501/001/00, technical report, (STARS) March 1991.

"Lawsuit: IBM Files Patent Suit Against Conner Peripherals," *EDGE: Work-Group Computing Report*, p. 29, 16 August 1993.

"Legal/Acquisition Issues: A Technical Report", 1222-04-210/49.1, technical report, DoD ,Center for Software Reuse Operations (CSRO), February 1993.

"Managing Reuse: Exposing The Hidden Agenda," *IEEE Software*, January 1993.

"Proceedings: Software Reuse Legal Issues Workshop (Legal Workshop) (22-24 Mar 94)", Central Archive for Reusable Defense Software (CARDS), Informal Technical Data, STARS-AC-04117/001/00, 30 April 1993.

"Six Offices Hook Up to Navy Net," *Government Computer News (GCN)*, March 21, 1994.

Abdel-Hamid, Tarek K., "Modeling the Dynamics of Software Reuse: An Integrated SystemDynamics Perspective," paper presented at the Sixth Annual Workshop on Software Reuse (WISR '93), Owego, New York, 2-4 November 1993.

Abdel-Hamid, Tarek K. and Stuart E. Madnick, *Software Project Management*, Prentice-Hall, 1991.

Banker, Rajiv D., Robert J. Kauffman and Dani Zweig, "Repository Evaluation of Software Reuse," *IEEE Transactions of Software Engineering*, v. 19, n. 4, pp. 379-389, April 1993.

Basili, Victor and John Musa, "The Future Engineering of Software: A Management Perspective", *Computer*, v. 24, n. 9, September 1991.

Baxter, Murry B., Major, "Legal Issues for Reuse Libraries," *The Sixth Annual Technology Conference on Predictable Software: Order Out of Chaos*, Salt Lake City, UT, 10-15 April 1994.

Bennun, Irene, "A 7-step Guide To Consulting Agreements," *Data Based Advisor*, p. 86, June 1994.

Bielefield, Arlene and Lawrence Cheeseman, *Libraries and Copyright Law*, pp. 1-33, Neal-Schuman Publishers, Inc., 1993.

Biggerstaff, Ted and Alan Perlis, *Software Reusability, Volume 2: Applications and Experience*, Addison-Wesley, 1989.

Biggerstaff, Ted and C. Richter, "Reusability Framework, Assessment, and Directions", *IEEE Software*, v. 4, n. 2, March 1987.

Bollinger, Terry B. and Shari Lawrence Pfleeger, "The Economics of Reuse: Issues and Alternatives," *Proceedings of the Eighth Annual National Conference on Ada Technology*, pp. 477-499, 5-8 March 1990.

Bui, T., James Emery, G. Harms, T. Van Hook, and M. Suh, "A Clearing House for Software Reuse: Lessons Learned from the RAPID/DSRS Initiatives," Naval Postgraduate School, October 1992.

Chandersekaran, C.S., and M.P. Perriens, "Towards and Assessment of Software Reusability," *Proceedings of IT Workshop on Reusability in Programming*, 7-9 September 1983.

Chartrand, Sabra, "U.S. Gains on Japan in Patents, I.B.M. was Leading Recipient Last Year," *The New York Times*, 14 March 1994.

Conn, Richard, "Impediments to the Software Reuse Industries", paper presented at the Sixth Annual Workshop on Software Reuse (WISR '93), Owego, New York, 2-4 November 1993.

Demarco, T. and Lister T., *Peopleware*, Dorset House, 1987.

Direction-Level Handbook Update, Central Archive for Reusable Defense Software (CARDS), STARS-VC-B012/001/01, 25 March 1994.

Emery, James and Dani Zweig, "The Use of Ada for the Implementation of Automated Information Systems within the Department of Defense," Naval Postgraduate School, 16 December 1993.

Endoso, Joyce, "Business issues impede software reuse," *Government Computer News*, 9 November 1992.

Endoso, Joyce, "Audit says DOD's software reuse plan faces many barriers," *Government Computer News*, v. 12, no. 6, p. 46, 15 March 1993.

Endoso, Joyce, "House Puts Teeth into DoD Systems Unification," *Government Computer News*, v. 12, n. 22, p. 8, 11 October 1993.

Endoso, Joyce, "Paige moves to put teeth in software reuse program", *Government Computer News*, v. 13, n. 9, p. 1, 2 May 1994.

Frakes, William and C. Fox, "Software reuse survey report", technical report, *Software Engineering Guild*, Sterling, Virginia, 1993

Gallup, Pamela, "A System Dynamics Based Study of Software Reuse Economics", Masters Thesis, Naval Postgraduate School, June 1994.

GAO/IMTEC-93-16 Report (U.S. General Accounting Office, Information Management and Technology Division Report), *Software Reuse-Major Issues Need to be Resolved Before Benefits can be Achieved*, pp. 1-21, 28 January 1993.

Goodall, Thomas, "Restructured Naval Tactical Data System (RNTDS): An Example of Applying Megaprogramming Concepts", Presentation to the Defense Advanced Research Project Agency, 8 December 1992.

Griss, Martin, "Software Reuse: From Library to Factory", *IBM Systems Journal*, p.595, December 1993.

Griss, Martin and L. Latour, "A Working Group on Management and Technology Transfer", *Proceedings of the Fifth Annual Workshop on Software Reuse*, Martin Griss and L. Latour, Editors, University of Maine, November 1992.

Henderson, David, "Patents", *The Fortune Encyclopedia of Economics*, edited by David R. Henderson, 1993.

Hirsch, E. D., Jr., "*The Dictionary of Cultural Literacy*", Houghton-Mifflin, 1988.

Hooper, James W. and Rowena O. Chester, "Software Reuse: Managerial and Technical Guidelines," *Proceedings of the Eighth Annual National Conference on Ada Technology*, Fort Monmouth, New Jersey, pp. 424-435, 1990.

Huber, Theresa, "Findings of the CARDS Sponsored Software Reuse Legal Workshop (22-24 Mar 93)", paper presented at the Sixth Annual Workshop on Software Reuse (WISR '93), Owego, New York, 2-4 November 1993.

Huber, Theresa, "Reducing Risks for Government Software Reuse Libraries," *The Sixth Annual Software Technology Conference on Predictable Software: Order Out of Chaos*, Salt Lake City, UT, 10-15 April 1994.

Hyman, Risa B., "Creative Chaos in High-Performance Team: An Experience Report," *Communications of the ACM*, v. 36, no. 10, p. 56, October 1993.

Kang, K. and L. Levy, "Software Methodology in the Harsh Light of Economics", *Information Software Technology (U.K.)*, v. 31, n. 4, June 1989.

Kim, Yongbeom and Edward Stohr, "Software Reuse: Issues and Research Directions," *Proceedings of the Hawaii International Conference on System Sciences*, v. 4, pp. 612-623, January 1992.

Krueger, Charles, "Software reuse (Creating Applications from Existing Elements)," *ACM Computing Surveys*, June 1992.

Lillie, Charles and C. Paul Bond, "Reuse Tool Assessment: Library Mechanism Summary Report", Asset Source for Software Engineering Technology (ASSET) Report presentation, p. 30, 23 April 1993.

Malan, Ruth, "Motivating Software Reuse", paper presented at the Sixth Annual Workshop on Software Reuse (WISR '93), Owego, New York, 2-4 November 1993.

Market Study, Central Archive for Reusable Defense Software (CARDS), STARS-VC-B001/004/00, 25 March 1994.

Miller, Michael, "Software patents must go", *PC Magazine*, 15 March 1994.

Plauger, P., "Reusability Myths. (Programming on Purpose)," *Computer Language*, May 1993.

Perry, William, "For DoD Software Reuse to Succeed, It Must Be Easy," *Government Computer News*, 26 October 1992.

Prieto-Diaz, Ruben, "Domain Analysis: An Introduction", *ACM Software Engineering Notes*, v. 15, n. 2, April 1990.

Prieto-Diaz, Ruben, "Status Report: Software Reusability," *IEEE Software*, v. 10, no. 3, p. 61, May 1993.

Poulin, J., J. Caruso, and D. Hancock, "The business case for software reuse", *IBM Systems Journal*, v. 32, n. 4, December 1993.

Ray, Garry, "Software Reuse Not A Panacea; Some Firms Pursue It As A Development Goal; Others Question Its Viability," *Computerworld*, 21 December 1992.

Schwartz, Karen, "DoD Reuse Libraries Get Linked Today. (Department of Defense project boosts software engineering efficiency)," *Government Computer News*, v. 12, n. 8, p. 1, 12 April 1993.

Stevens, Barry, "A Case Study in Software Reuse: The RNTDS Architecture," Masters Thesis, Old Dominion University, 1991.

Stevens, Barry, "Results from the Navy's RNTDS Architecture", *The Sixth Annual Software Technology Conference on Predictable Software: Order Out of Chaos*, Salt Lake City, UT, 10-15 April 1994.

Stevens, Barry, Interview conducted on 28 June 1994.

Technical Concept Document Update, Central Archive for Reusable Defense Software (CARDS), STARS-AC-04107A/001/00, 9 February 1993.

Tracz, Will, "Software Reuse: Motivations and Inhibitors," *COMPCON*, Spring 1987, IEEE Computer Society Press, Washington, D.C., 1987.

Wasmund, Michael, "Incentives Versus Target - A Practical Experience," paper presented at the Sixth Annual Workshop on Software Reuse (WISR '93), Owego, New York, 2-4 November 1993.

Wasmund, Michael, "Implementing Critical Success Factors In Software Reuse," *IBM Systems Journal*, December 1993.

U.S. Department of Defense, *Software Reuse Initiative Vision and Strategy*, July 1992.

Zachary, G. Pascal, "Software patent given Tribune Co. unit is overturned by U.S. in wake of protest (Compton New Media unit)", *The Wall Street Journal*, 25 March 1994.

INITIAL DISTRIBUTION LIST

	Number of Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 052 Naval Postgraduate School Monterey, California 93943-5002	2
3. George S. Coker Software Reuse Program Management Office Defense Information Systems Agency Center for Information Management 500 N. Washington Street, Suite 200 Falls Church, VA 22046	1
4. Gary Shupe Head, Software Engineering Branch Software and Data Management Division Naval Computer and Telecommunications Command 4401 Massachusetts Avenue, N. W. Washington, D.C. 20394-5000	1
5. Dr. Michael Mestrovich Office of Integration and Interoperability 5201 Leesburg Pike 3 Skyline Place Suite 1501 Falls Church, VA 22041-3201	1
6. Donald J. Reifer Executive Administrator DoD Software Initiatives Defense Information Systems Agency Center for Information Management 1951 Kidwell Dr., Rm. 521 Vienna, VA 22182	1
7. James C. Emery, Code SM/Hg Department of Systems Management Naval Postgraduate School Monterey, California 93943-5002	1

- | | |
|---|---|
| 8. CDR William B. Short, SC, USN Code SM/Sh
Department of Systems Management
Naval Postgraduate School
Monterey, California 93943-5002 | 1 |
| 9. LT Robert W. Therriault, SC, USN
CINCLANTFLT (N413F)
1562 Mitscher Ave, Suite 250
Norfolk, Virginia 23551-2487 | 3 |
| 10. CPT Kristina E. Van Nederveen
P.O. Box 610
Lincoln, California 95628 | 3 |