IDA DOCUMENT D-1004

PROCEEDINGS OF THE WORKSHOP ON LEGAL ISSUES IN SOFTWARE REUSE

Craig A. Will
James Baldo, Jr.

Dennis W. Fife, *Task Leader*

July 1991

Prepared for
Strategic Defense Initiative Organization

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

IDA Log No. HQ 91-038811

DEFINITIONS

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

Reports

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

Group Reports

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

Papers

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

Documents

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

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

Approved for public release, unlimited distribution: 17 September 1991. Unclassified.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1991	Final

4. TITLE AND SUBTITLE
Proceedings of the Workshop on Legal Issues in Software Reuse

5. FUNDING NUMBERS
MDA 903 89 C 0003

Task T-R2-597.2

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

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses (IDA)
1801 N. Beauregard St.
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Document D-1004

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Strategic Defense Initiative Organization (SDIO)
Room 1E149, The Pentagon
Washington, D.C. 20301-7100

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, unlimited distribution: 17 September 1991.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

This document includes technical papers, reports of working groups, annotated transcripts of discussions, and a report on further analysis of some specific issues raised at a workshop held 23-24 October 1990 in Colorado Springs, Colorado, that discussed the legal issues involved in the reuse of computer software. A plan for a software reuse library for the SDI program is presented, together with a discussion of the legal issues raised by that plan and by software reuse generally. These issues include potential copyright and patent infringement, theft of trade secrets and the associated risks, and concerns about data rights acquired by the government. They also include liability for software malfunction, ways that legal risks resulting from software reuse can be reduced, and mechanisms for rewarding software reuse. Participants in the workshop included both technical and legal representatives from industry and government.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Reuse; Intellectual Property; Patents; Copyrights; Reverse Engineering; Data Rights; SDI.	138
	16. PRICE CODE

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

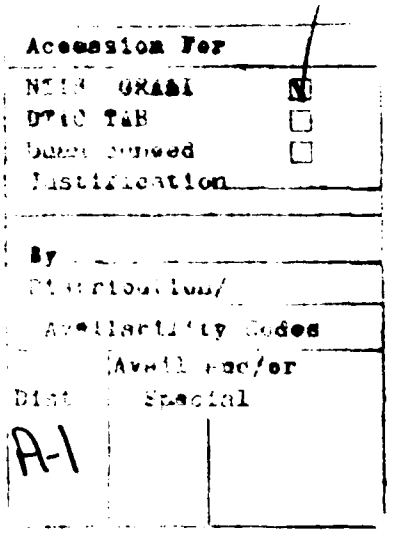
IDA DOCUMENT D-1004

PROCEEDINGS OF THE WORKSHOP ON LEGAL ISSUES IN SOFTWARE REUSE

Craig A. Will
James Baldo, Jr.

Dennis W. Fife, *Task Leader*

July 1991



INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 89 C 0003
Task T-R2-597.2

PREFACE

This document presents the proceedings of a workshop on legal issues in software reuse held 23-24 October 1990 in Colorado Springs, Colorado, and is intended for use by contracting and legal personnel and software managers in the SDI program who are concerned with software reuse. It was written in response to Task Order T-R2-597.2, which requires "a report on the SDI workshop on legal issues in software reuse."

The proceedings includes two formal papers based on lectures presented at the workshop, an edited transcript of a panel discussion held at the workshop, an analysis of specific scenarios discussed at the workshop, and reports of working groups.

Prior to the workshop IDA developed a set of scenarios that suggested likely situations in which attempts to reuse software or construct a reuse library might result in legal difficulties. These scenarios were discussed at the workshop, and IDA also conducted further investigation and analysis of some of these scenarios.

Four working groups met during the workshop and completed reports that are included: (1) Component Licensing/Integration Contract Wording; (2) Legal Obligations, Subscribers, and Developers; (3) Royalties and Incentives; and (4) Potential Approaches to Software Industry Participation.

This document was reviewed internally by Dr. Richard Wexelblat, Mr. Michael Nash, Dr. Reginald Meeson, Dr. Robert Turner, Ms. Audrey Hook, and Ms. Ruth Greenstein.

SUMMARY

Introduction

The principal goals of the *Workshop on Legal Issues in Software Reuse* were: (1) to identify and discuss potential legal problems that could inhibit the insertion of software reuse technology in the Strategic Defense Initiative (SDI) program; and (2) to critique a specific plan for software reuse based on a centralized reuse library proposed by the SDI National Test Bed.

In advance of the workshop, attendees were provided with the National Test Bed Reuse plan and a set of scenarios describing specific situations in which software reuse might raise legal difficulties. Thirty-eight attendees participated in the workshop, including eight attorneys and twenty-one technical representatives.

The workshop began with an introductory lecture by John Morrison of the National Test Bed that discussed his view of software reuse for SDI and the National Test Bed Reuse plan. Barry Sookman, a Canadian trial attorney, then presented a lecture discussing the risks of liability involved in software reuse. Formal papers based on these lectures are included in these proceedings.

The workshop also included a forum in which some of the scenarios were presented and discussed, and a panel discussion. In addition, further investigation and analysis of the scenarios was carried out by IDA. An annotated transcript of the panel discussion and a presentation, discussion and analysis of these scenarios are included in the proceedings.

During the workshop attendees participated in one of four working groups: (1) Component Licensing/Integration Contract Wording; (2) Royalties and Incentives; (3) Legal Obligations, Subscribers, and Developers; and (4) Potential Approaches to Software Industry Participation. Each working group was asked to identify and describe the principal issues in its area and to develop recommendations to address these issues. A written report from each working group is included in the proceedings.

Keynote Speaker

The keynote speaker, Barry Sookman, discussed how to minimize the potential legal liabilities associated with software reuse libraries. There are two principal problems: (1) identifying the risks; and (2) finding ways to allocate responsibility for risks so that developers and users will not be discouraged by excessive liability burdens.

Sookman discussed the different schemes for intellectual property protection for software, and concluded that with all forms of protection—including copyrights, patents, and trade secrets—software reuse raises particular problems of tracking rights associated with each component so that it can be determined that the rights necessary to reuse a particular component have in fact been obtained.

Sookman described the principal risks associated with a reuse library as including the risk of deliberate or inadvertent copyright, patent, or trade secret infringement, liability for inaccurate information concerning software that has been provided to the library, and liability for defective software.

Panel Discussion

A panel of industry representatives discussed a variety of technical and legal issues involving reuse, including the implications of patent and copyright protection, government data rights issues, and the legality of reverse engineering of software.

There was concern about the impact of software patents on the industry, because of the uncertainty of knowing whether a particular software invention is patentable, the possibility of infringement lawsuits after considerable investment has been made in the software, and the high cost of obtaining and enforcing a patent. Concern about copyright protection particularly focused on whether interfaces between components ought to be copyrightable, with panelists tending to be against such protection on the grounds that reuse would be inhibited.

Panelists felt that the complexity and difficulty in understanding government data rights regulations was a particular problem, and that reuse might be improved if industry were allowed to retain more rights than is the case with the present regulations. There was considerable uncertainty and disagreement among the panelists about just what reverse engineering practices were legitimate and legal for software.

Scenarios

A detailed analysis was performed for most of the scenarios after the workshop. Some examples of conclusions from this analysis follow.

New changes to the copyright law that remove the requirement for copyright notices may increase the risk of accidental copyright infringement, but these risks can be minimized for a reuse library if it adopts procedures to track the source of all components and modifications.

Whether interfaces between components are copyrightable has not been resolved by the courts, and while the "look-and-feel" cases now being litigated may have a significant effect on this issue, they may not be fully decided for many years.

A particular risk involving patents might be the case where a component is entered into a reuse library, after which a patent is issued that covers a process in the component. In this case a patent search when components are entered into the library will not necessarily prevent unintended infringement.

Working Group Reports

The Component Licensing/Integration Contract Wording Working Group reviewed the proposed National Test Bed plan. After identifying issues raised by the National Test Bed plan, the group focused its efforts on new regulations and laws (i.e., FAR, DFARS, statutes, etc.), integrating software between systems being built for the three services by major aerospace companies, and license agreements that would be needed for the reuse library as proposed by the National Test Bed.

The Legal Obligations, Subscribers, and Developers Working Group focused on identifying legal issues that could arise in establishing a government run software repository. They developed a software reuse library model based on the RAPID Reuse Library and the proposed National Test Bed plan. The group then analyzed the model for legal issues with respect to contractors, software component suppliers and developers, the library, library subscribers, and consumers of library components.

The Royalties and Incentives Working Group developed recommendations for providing incentives for contractors to develop reusable software components and to reuse components in building software systems. The group identified liability associated with intellectual property, liability for defective software, possible loss of proprietary rights resulting from placing components in the library, appropriate rewards for contractors, ease of use of the library, and form and granularity of components in libraries as significant issues.

The Potential Approaches to Software Industry Participation Working Group identified government actions and activities that could stimulate a software reuse industry. The critical issues facing the insertion of software reuse technology addressed by this group are the establishment of a critical mass of reusable components, demonstration that software reuse works, reuse standards, criteria for evaluating reusable components for quality and reliability, legal issues such as data rights and liability, and efficient mechanisms for inserting software reuse technology into practice.

Conclusions

General conclusions reached as a result of the workshop include:

a. Large-scale reuse will probably require a registry that allows tracking the source of original development and modifications for each component. The establishment of a national registry for reusable software component lineage was recommended.

b. The requirement in the National Test Bed plan in which the government assumes all legal liabilities associated with a reuse library is probably not good policy and may conflict with federal law.

c. There are many uncertainties in the law associated with intellectual property protection for software and liability for software malfunction, and these uncertainties are not likely to be resolved soon. Software reuse raises few, if any, fundamentally new legal issues that have not been confronted before, but large-scale reuse will likely result in these issues being encountered more frequently and in more complex ways.

d. For a large-scale reusable components industry to develop, it is desirable for mechanisms to be developed that can appropriately reward developers who modify and add value to existing components, while still protecting the rights of the developers of the original components.

e. Software patents may have significant potential as a mechanism for encouraging the development of reusable components, however, there are many uncertainties about the validity of software patents and the impact of patents and patent infringement lawsuits on the software industry that need resolution.

f. Licenses for reusable components should be irrevocable, allow liberal rights to produce derivative works, require the developer to fix errors, and contain provisions for determining who is allowed access to components.

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTORY PAPER

NATIONAL TEST BED SOFTWARE REUSE LIBRARY

CONCEPT OF OPERATIONS:

IMPACT OF LEGAL AND CONTRACTUAL ASSUMPTIONS

John S. Morrison, Lt. Col., USAF[1]

Director, System Engineering and Development
National Test Facility
Colorado Springs, Colorado

PURPOSE

The purpose of this discussion is to elaborate on a concept of operations for software reuse at the National Test Facility which could be implemented. The concept is defined in sufficient detail so that the legal and contractual ramifications of the concept may be evaluated and debated by experts schooled in the legal intricacies of intellectual property, technology transfer, and liability.

WHY REUSE SOFTWARE?

SDI is a technology-driven program. This means that the program must consider how to design systems in the face of technological change. Flexibility with respect to new technology will allow the system to adapt to new threats. Potentially one of the most flexible pieces of the system will be software. Reusing software is not inconsistent with the concept of accommodating technological change. Software legacy can be viewed as a bridge between what exists and what can be imagined.

Traditionally, software has been developed one line of code at a time. Unlike the design of hardware systems, the fundamental units of modularity—the building blocks of software systems—haven't really changed in 20 years. However, considering the spectrum of software that must be built to create a Strategic Defense System, there are clear opportunities to aggregate lines of code into larger chunks. Simple functions that are typically used in software programs might have a size on the order of tens of lines of code. Small callable modules with more capability might consist of hundreds of lines of code. Simulations of the SDI system range from 10^3 to 10^4 lines of code. Real time, battle-critical software might be about 10^5 lines of code. The full SDS, to include peacetime functions and test software might range from 10^6 to 10^7 lines of code.

Altering the fundamental unit of software modularity could improve productivity by perhaps one or two orders of magnitude. Small, callable modules—built to be re-engineered and adapted to new contexts, and perhaps measuring a few hundred lines of code in size—might be stored in a central library and used to construct new software projects. They could form reusable building blocks for simulations and other development efforts. Using common parts in this way could still allow us considerable flexibility in how we architect systems. The approach might, in addition, help improve the security and trust of the systems that get developed because security and trust features could be built into the components. Similarly, if reliability and fault tolerance are also

1. Lt. Col. Morrison is now at Technology Transfer International, Colorado Springs, Colorado.

built into the components, then the overall reliability of systems that are constructed with these components might in theory be improved.

Finally, software reuse can be a way to transfer technology quickly between one developer and another. Components can encapsulate knowledge and expertise in a particular problem domain, and bind that knowledge within a package that can be licensed by other developers. Such licensing can provide both incentives for the component developer to maintain the software and a contractual mechanism to control its release.

The effectiveness of software reuse is the extent to which a program can be used in applications other than the one for which it was originally developed. This effectiveness can be measured by the amount of effort that is required to make the software component work in the new application.

There is some limited empirical data supporting the case for reuse. For example, the Lund Institute of Technology in Sweden has applied software reuse in development of a *telecommunications system*. They reused software from existing telephone systems and achieved a fairly high (60-90%) rate of reuse.[2] Of course, there are some caveats attached to this study. It was based on object-oriented methods applied to a telephone system, not an SDI system. Also, it was a relatively small development effort (approximately 2,000 to 3,000 lines of Ada code). There is some question, then, about whether this reuse rate might change as the size of the system scales up. However, the data does suggest a potential cost savings for reuse on at least one real time system and it therefore merits closer attention by the SDI program. The data also suggests there might be a strong correlation between reusable requirements, reusable designs, and reusable code. If you can reuse the requirements, and if from requirements you can derive designs, and from designs you can implement code, then there is potentially a "cascade of efficiencies" associated with reuse at higher levels of abstraction.

It is easy to quantify in dollars and cents how reuse pays off. An example from the Joint Integrated Avionics Working Group (JIAWG) shows how. If you reuse a component, there might be some cost you avoid by not having to build it from scratch. But there is, nevertheless, some cost associated with finding and adapting the component to a new application. The reuser's net savings is just the difference between building the code from scratch and adapting extant code. Over time there are costs to maintain the code, and a commercial vendor would also have to factor in the discount rate for money. The discounted cash flow from reuse could be measured, and the size of the positive cash flow would increase with every additional instance of reuse.

Recognizing the potential benefits to the SDI program which might accrue through reuse, we have put together a concept of operations for a reuse library at the National Test Facility. I will describe the assumptions which underlie the concept, define who the players are, list the contracts that are invoked and the processes which invoke them, describe the objects of the contracts, and the payments and fees which are incurred.

ASSUMPTIONS BEHIND PROPOSED NATIONAL TEST BED CONCEPTS OF OPERATIONS

The NTB software reuse library will be in a secure, government-owned, contractor-operated facility, with daily government management oversight. The library would be operated as part of the NTB integration contract. This contract has provisions and procedures for handling organizational conflicts of interest (OCI). For example, if Martin Marietta operated the reuse library and also became a bidder on one of the systems being developed for SDI, then presumably they would be in a position to gain competitive advantage. Provisions in the NTB integration contract could handle such OCI cases by allowing OCI-sensitive operations to be executed only by a designated subcontractor who

2. *Documentation and Some Metrics on Case Study Design of Service Features in Telecommunications Systems.* Lund Institute of Technology, Sweden. 5 July 1990.

reports directly to the government and who has been excluded from bidding on non-NTB activities within SDI.

Reusable software components will be developed by a variety of organizations, not just the National Test Bed program. The library, then, would include components from the Army, Navy, Air Force, and Department of Energy. The reusable components could be shared among these agencies to build systems with common software parts. For example, the Army might develop a component that the Air Force would use in development of its on-orbit systems.

Another key assumption is that the SDI program would incentivize both development of components and the use of those components on system development contracts. Contractors developing systems might be required to subscribe to the reuse library.

Consumers of components (i.e., the system developers) would pay royalties to the government, and these royalties would pass through to the component developers. Consumer reporting would also be incentivized.

We would contend that software developed under a government contract is owned by the government and title, to include patent and copyright rights, should be explicitly transferred to the government. You will also see this recommendation in the Government Accounting Office (GAO) and Office of Technology Assessment (OTA) reports sent to all attendees prior to this workshop.

Suppose software is not developed under government contract, but is developed instead by an independent commercial vendor like EVB. EVB would continue to maintain ownership rights and the government would have to license such software in order to make it available in a reuse library. The license should provide the government with the ability to sub-license to reuse library subscribers. This may be a controversial point.

Derivative works are another controversial area. We would propose that derivative works should be encouraged for government owned software and that a derivative work would be subject to a pass-through of royalties to previous developers based on a government determination of value added.

For example, Contractor A develops a component and puts it in the library. Contractor B extracts the component from the library, modifies it, and puts it back into the library. The government could determine the value added so that when Contractor C checks out the new (modified) component, there is some pro-rated pass-through of royalties.

What do we have to do to put in place such a capability? We would have to rethink how we presently develop software. We would need acquisition specialists, contract evaluators, and program managers who really understand the issues associated with software reuse. We would also need independent verification and validation (IV&V) specialists who could assess the software, a library staff to operate the reuse library, and what I will call a component assayer, who evaluates the quality of the particular component when it is inserted into the library. Knowledgeable system engineers would be needed to balance the traditional top-down design approach with a bottom-up approach based on available components. Given such a balanced design, system software architects would program primarily in components rather than in FORTRAN, C, or Ada. They would look at what parts are available and then, using a constructor kit, put the system together. At the opposite end of the software development "food chain" would be the component developers (i.e., parts suppliers) who would build software components in a flexible and adaptable manner.

Viewing the overall software component market, there would be a supply region and a demand region. On the supply side, you would have subscribers to the reuse library who primarily contribute components. On the demand side would be the systems developers who are perhaps under contract to the government to build systems. It is the relationship between the supply side and the demand side, and the market place in the middle that would invoke contracts. Subscribers to the library (the market place) would deal in a subscription contract; component suppliers would deal in transfers of title or component licenses; consumers would deal in licenses or sub-licenses; and operators of the library would deal with the library integration contract. This, then, is the contractual landscape. Now a look at the components.

THE REUSABLE SOFTWARE COMPONENT

Components would be library assets consisting of documentation, code, and a variety of other things. The pieces of each component might come from multiple sources.

Component developers would contribute documentation and source code. The documentation would be largely drawn from the 2167A software development file, a version description document and data indicating which compiler, development hardware, and target hardware was used in construction of the component. Supplementary information could include a formal language description of the component, a graphic representation of the design, a simulation of the software, etc.

The reuse library would contribute quality assessment reports (perhaps including a part quality index), access reports (describing who used the component and when), and retrieval keys.

Consumers would contribute consumer reports, to include qualitative consumer assessments of the component, deficiency reports, and perhaps entries in a "Blue Book" that would show the cost of adapting a component to other projects. Ideally, the Blue book would provide statistics from every project which used the component. Blue Book entries could include:

- Re-engineering cost in man-hours (a more stable metric than dollars)
- Project point of contact (to obtain additional data)
- Project information (e.g., type of system and size of system on which the component was used)

From such a listing, prospective consumers of the component could estimate the cost to re-engineer the component.

All elements of the component would be linked to a single software identification number which would be used to track transactions associated with the component. The component, then, would be comprised of the software identification number, the design elements, documentation elements, consumer reports, quality assessment reports and access reports.

There would have to be links between the software identification number, the list of suppliers who contributed to the component, and the consumers who have used the component. These links will be needed to deal with potential problems. For example, if malicious logic is discovered in a component you might want to "recall" it, just as an automaker might want to recall automobiles having defective, unsafe parts. Such a recall involves notification in two directions. You would like to notify the people using the defective component in order to minimize potential damage, and you would like to notify the component producer to correct the problem.

REUSE TRANSACTIONS

The process of software reuse will now be examined in terms of reuse library transactions. These transactions could result in binding contracts, incur costs, or generate income such as royalty payments.

Enrollment in the library invokes a subscription contract. The prospective subscriber would submit a written request to the library which would include the name of the government sponsor. The library would provide the applicant with a standard subscription contract containing a set of terms, conditions and responsibilities. Signing the subscription contract would require the user to abide by a set of rules and procedures for library use, and would establish library usage fees and charges.

The librarian would verify access requirements through the government sponsor. Security access would be approved by the National Test Bed Joint Program Office based on a review of site security accreditation, and a risk analysis of the site.

Based on a successful security review and permission of the government sponsor, the librarian would assign temporary passwords and personal identification numbers to individual members associated with the enrolling organization. In addition, the organization would receive a project authentication number. This authentication number would be

used to electronically "sign" component licenses, incur licensing costs, and receive payments. Control of the authentication number would be left to the using organization, but the library system would assure that only members of the appropriate organization could authenticate with that number. The user account would now be activated.

Once electronically connected to the library and logged on, a user could perform a number of possible activities. He or she could:

- Browse— looking through the inventory of parts and perhaps inspecting individual parts;
- Retrieve— consciously deciding to obtain, license and pay for a component;
- Submit— adding to the library a unique component or a modification of someone else's component;
- Report— adding a consumer evaluation of a component to the library for other consumers to review.

The library would provide customer assistance as required. In addition, library services would automatically track who retrieves what component and when, and provide this data to consumers. Subscribing organizations would be responsible for updating their list of individuals having access to library services.

At the end of the "user life-cycle" would be the de-enrollment process. This process deletes organizations and their members from the list of users of library resources. Any component or royalty that flowed through to the de- enrolled organization would revert to the government if the government had title to the component. If the vendor company dissolves, then the government would retain the right to license the component, with royalties reverting to the government.

Turning now from subscriber-centered to component-centered processes, we will focus on those transactions which are keyed to the software identification number. This number uniquely identifies components and links design elements, documentation elements, quality assessment reports, access reports and consumer reports—those things which constitute the component.

A subscriber logged into the library may want to execute any number of component-centered transactions. The subscriber could:

- Submit a component to the library
- Report on a component—either by submitting a qualitative assessment report or by quantifying "Blue Book" costs associated with adapting a component for a particular project;
- Browse through the catalog of components;
- Retrieve a component.

The library system could also execute component-centered transactions. These include:

- Logging of component transactions;
- Removal of components;
- Adding of components;
- "Recall" of components.

Component submission might work as follows. A subscribing organization would forward a copy of the proposed new component to the librarian, along with the proposed license. In the case of software developed under government contract, the submitting contractor would provide a transfer of title that would give clear ownership, including patent and copyright rights, to the government. In consideration for transfer of title, the government would agree to provide the component developer with royalties on each retrieval of the component from the library. The amount of the royalty could be tied to the quality assessment of the component. The library's quality assessment could also influence consumer demand for the component. The intent would be to incentivize developers to produce higher quality components. In the case of software developed by a commercial vendor, the license would simply

incorporate terms negotiated between the vendor and the government.

I will not go into transactions associated with removal of components from the library catalog except to mention potential legal and contractual implications. Removal of defective components could result in suspension or revocation of component licenses, in the refund of royalty fees, and potential notification of all affected parties.

The browsing process and component retrieval process have important legal and contractual implications. The government would make the component license terms, conditions and costs visible to the browsing consumer. Documentation elements, consumer reports, quality assessment reports and access reports would also be visible. There may be some flat fee associated with browsing, and the terms and conditions of browsing might be included in or referred to by the subscription contract. Source code and designs for a component would be accessed only through retrieval, a process involving a conscious decision on the part of the consumer to invoke a component license using the appropriate authentication number. The transaction would also incur a fee.

A "safety valve" feature may be needed for retrieval. Suppose a developer retrieves a number of components and attempts to use them to construct a system. Later he decides that the component is not as usable as he thought. There should be a way for the consumer to revoke the license and get his money back.

Consumer reporting and library feedback are important features for improving the overall quality of the library. Consumers using the library should be encouraged to report problems with the library approach. Incentives might include library credits to be applied to browsing. Incentives should also be provided for submitting component evaluations, and for entering re-engineering cost data into component Blue Books. Deficiency reports should be incentivized. These will tell component developers what needs to be fixed, will inform unwary consumers of potential problems, and will be visible during browsing until the problem is corrected.

The reuse library must monitor all ongoing transactions. It would log who accesses what components, when, and the transaction costs. This summary data would be made available to subscribers for inspection. The system should also flag any security, integrity, or proprietary exclusion events for immediate action by the library staff. Part of the daily activity would also be to decide what components should be dropped from the library catalog based on lack of interest from consumers.

The library, then, represents a marketplace for components and library services. It makes possible a variety of transactions which result in positive or negative cash flow for the various participants in the market place.

NEEDED CAPABILITIES

Many things need to be worked out in greater detail in order to create an effective software reuse library. Some of these details include:

Software Identification Number (SIN) numbering and classification scheme. The entire SDI program would have to agree on a standard way to uniquely identify and control the software components.

Procedures and tools for quality assessment of software components. The government would need a repertoire of procedures and tools for software evaluation. When subscribers sign up to the library, they would agree to allow the government to evaluate submitted components using these tools and procedures.

Part quality index. Based on quality assessment of submitted components, the government would summarize the quality of the part in terms of a scalar or vector number, which would be included with each component in the library.

Formula for royalty fee determination. For those components developed under government contract, there would be a formula for determining royalty fees, based on the assessed quality of the component and perhaps other factors. This formula would be incorporated by reference into the subscription contract.

Tools and algorithms for assessing derivative works. Such tools would objectively measure "conceptual distance" between a

component and the original work from which it was derived. Alternatively, such tools could be used to evaluate claims of originality or intellectual piracy.

Drop logic for library components. There would have to be some algorithm for determining if or when a component should be dropped from the library, based on lack of subscriber interest in the component.

Library and component monitoring tools. Such tools will be needed not only to provide data on components to consumers, but to implement drop logic described above.

Detailed operating procedures. These have not been developed yet, but must be part of the agreed upon set of rules invoked with the subscription contract.

SUMMARY

One objective of this notional concept of operations was to describe an approach which could accept into a reuse library software components which may not initially be of very high quality. By incentivizing developers to add value to the components, the net quality of the components should increase with time. Consumers, by using and reporting on components, also add value, and would likewise be rewarded with incentives for their time and effort.

It is now up to the lawyers and contracting experts to reflect on these proposed objectives, and to suggest modifications and improvements to the concept of operations which will make the approach legally tractable.

KEYNOTE PAPER

CONFRONTING LEGAL LIABILITY

IN THE REUSE OF COMPUTER SOFTWARE

Barry Sookman

Computer and High Technology Law Group
McCarthy and Tetrault
Toronto, Ontario
Canada

INTRODUCTION

The talk that I was asked to give the workshop was entitled *How shall I sue thee: Let me count the ways.* The sub-topic I was asked to speak about was "Software reuse: How to 'bulletproof' the procedures for reuse to minimize potential liability?" This sub-topic is relevant as the establishment of a reuse library creates potential obligations and liabilities, both to the library itself and to others involved in the process. These others include "subscribers" to the library, component suppliers, component consumers, and users of systems built with components in the library.

Two important legal challenges must be addressed in relation to software reuse libraries. First, the potential risks must be recognized and identified. Second, the risks must be allocated in a sensible way that fosters reuse and does not impose excessive economic burdens on the parties involved in reuse. If these risks are not appropriately addressed owners of software will be discouraged from participating in a reuse library. Among the risks that must be identified are (1) risks in relation to intellectual property, (2) risks involving deficiencies in software and information about software in the reuse library, and (3) enforcement of rights issues.

The plan proposed for the reuse library by Lieutenant Colonel Morrison[1] suggests that four agreements governing the operation of the library would be used: (1) a subscription agreement, (2) a transfer agreement to convey title to the software to the library, (3) a license of the software to the library, and (4) a component license agreement between the library and the component user. Before discussing how to "bullet-proof" these agreements, I will provide an overview of the available intellectual property regimes for protecting software.

INTELLECTUAL PROPERTY PROTECTION REGIMES FOR SOFTWARE

To understand some of the issues involved in establishing a software reuse library and to appreciate the risks to the parties involved, one must step back and consider the different intellectual property regimes that govern protection of software. There are four methods of protecting software: trade secret protection, copyright protection, patent protection, and licensing agreements.

The common law and statutory law governing trade secrets has firmly established that computer software can be protected as a trade secret.[2] The case law has also

1. See John S. Morrison, "National Test Bed Software Reuse Library Concept of Operations", Appendix D, and contracts in Appendix C.

2. University Computing Co. v. Lykes-Youngstown Corp., 504 F.2d 518 (5th Cir. 1974); Com-Share Inc. v. Computer Complex Inc., 338 F. Supp. 1229 (E.D. Mich. 1971), affirmed 458 F.2d 1341 (6th Cir. 1972); Q-Co. Industries Inc. v. Hoffman, 625 F. Supp. 608

established that methodologies,[3] underlying mathematical models, procedures, and statistical assumptions,[4] designs and specifications,[5] knowledge of the trial and error process employed in the creation of computer programs,[6] organization, structure, logical flow,[7] data structures,[8] protocols[9] and computer systems consisting of both hardware and software[10] can also be protected as trade secrets.

Copyright has for some time protected source and object codes of computer programs.[11] In the 1980's protection has also been expanded to protect the structure, sequence, and organization of programs[12] and program user interfaces.[13] Recent case law has also suggested that protocols[14] and computer languages and macros[15] can also be protected.

Patents are becoming one of the most important methods of protecting computer programs. Many of the patents which are now issuing are "pure" software patents that

(S.D.N.Y. 1985); Telex Corp. v. IBM Corp., 179 U.S.P.Q. 777 (N.D. Okla. 1973); Structural Dynamics v. Engineering Mechanics Research Corp., 401 F. Supp. 1102 (E.D. Mich. 1975); Computer Print Systems Inc. v. Lewis, 422 A.2d 148 (Pa. Super. Ct. 1979); Dickerman Associates Inc. v. Tiverton Bottled Gas Co., 594 F. Supp. 30 (D. Mass. 1984); Cybertek Computer Products Inc. v. Whitfield, 203 U.S.P.Q. 1020 (Cal. Super. Ct. 1977); J. & K. Computer Systems Inc. v Paci, 642 P.2d 732 (Utah Sup. Ct. 1982); McCormack & Dodge Corp. v. ABC Management Systems Inc., 222 U.S.P.Q 432 (Wash. Super. Ct. 1983); Aries Information Systems Inc. v. Pacific Management Systems Corp., 366 N.W. 2d 366 (Minn. Ct. App. 1985); Belth v. Insurance Dept., 6 C.L.S.R. 1386 (Sup. Ct. N.Y. 1977); Bishop v. Wick, [1989] Copyright L. Rep. (CCH) 26,467 (N.D. Ill. 1989); Soft Computer Consultants Inc. v. Lalehzarzadeh, [1989] Copyright L. Rep. (CCH) 26, 403 (E.D.N.Y. 1988)

3. Healthcare Affiliated Services Inc. v. Lippany, 701 F. Supp. 1142 (W.D. Pa. 1988)
4. Belth v. Insurance Dept., 6 C.L.S.R. 1386 (Sup. Ct. N.Y. 1977)
5. U.S. v. Perholtz, 1 CCH Computer Cases 46,065 (Ct. App. Dist. Columbia Cir. 1989)
6. Integrated Cash Management Services v. Digital Transactions Inc. 13 U.S.P.Q. 2d 1397 (S.D.N.Y. 1989)
7. Dickerman Associates Inc. v. Tiverton Bottled Gas Co., 594 F. Supp. 30 (D. Mass. 1984); Integrated Cash Management Services v. Digital Transactions Inc., 13 U.S.P.Q. 2d 1397 (S.D.N.Y. 1989)
8. Soft Computer Consultants Inc. v. Lalehzarzadeh, [1989] Copyright L. Dec. (CCH) 26,403 (E.D.N.Y. 1988)
9. Telerate Systems Inc. v. Marshal Caro, 689 F. Supp. 221 (S.D.N.Y. 1988); Secure Services Technology Inc. v. Time and Space Processing Inc., 772 F. Supp. 1354 (E.D. Va. 1989)
10. Telxon Corp. v. Hoffman, 720 F.Supp. 657 (N.D. Ill, 1989)
11. GCA Corp. v. Chance, 217 U.S.P.Q. 718 (N.D. Cal. 1982); Apple Computer Inc. v. Franklin Computer Corp., 215 U.S.P.Q. 935 (E.D. Pa. 1982), reversed 714 F.2d 1240 (3rd Cir. 1983), cert. dismissed 104 S. Ct. 690 (1984); Freedman v. Select Information Systems Inc., 221 U.S.P.Q. 848 (N.D. Cal. 1984); Hubco Data Products Corp. v. Management Assistance Inc., 219 U.S.P.Q. 450 (D.Idaho 1983); Apple Computer Inc. v. Formula International Inc., 562 F. Supp. 775 (C.D.Cal. 1983), affirmed 725 F. 2d 521 (9th Cir. 1984); SAS Institute Inc. v. S. & H. Computer Systems, Inc., 605 F. Supp. 816 (M.D. Tenn. 1985); Fishing Concepts Inc. v. Ross, 226 U.S.P.Q. 692 (D. Minn. 1985); Microsoft Corp. v. Very Competitive Computer Products Corp., 671 F. Supp. 1250 (N.D.Cal 1987); In re Simplified Information Systems, Copyright L.R. (CCH) 26, 255 (Bankr. W.D. Pa. 1988); Soft Computer Consultants Inc. v. Lalehzarzadeh, [1989] Copyright L. Dec (CCH) 26,403; Johnson Controls Inc. v. Phoenix Control Systems Inc., 886 F. 2d 1173 (9th Cir. 1989); Brignoli v. Balch, Hardy & Scheinman, Inc., 645 F. Supp. 1201 (S.D.N.Y. 1986)
12. Synercom Technology Inc. v. University Computing Co., 462 F.Supp. 1003 (N.D. Tex. 1978); Q-Co. Industries Co. v. Hoffman, 625 F. Supp. 608 (S.D.N.Y. 1985); SAS Institute Inc. v. S.&H. Computer Systems Inc., 605 F. Supp. 816 (M.D. Tenn. 1985); Whelan Associates Inc. v. Jaslow Dental Laboratory Inc. F.2d 1222 (3rd Cir. 1986); Pearl Systems Inc. v. Competition Electronics Inc., 8 U.S.P.Q.2d 1520 (U.S.D.C. Fl. 1988); Healthcare Affiliated Services Inc. v. Lippany, 701 F.Supp. 1142 at 1151 (W.D. Pa. 1988); Manufacturers Technology Inc. v. Cams Inco., 10 U.S.P.Q.2d 1321 (D. Conn. 1989); Plains Cotton Co-op. v. Goodpasture Computer Services, 807 F.2d 1256 (5th Circ. 1987); Soft Computer Consultants Inc. v. Lalehzarzadeh, [1989] Copyright L. Dec (CCH) 26,403; Johnson Controls Inc. v. Phoenix Control Systems Inc., 886 F.2d 1173 (9th Cir. 1989); Bull HN Information Systems Inc. v. American Express Bank Ltd. CCC Computer L.R. 46,285 (S.D.N.Y. 1990); Telemarketing Resources v. Symatec Corp. 12 U.S.P.Q. 2d 1991 (N.D. Cal. 1988); Lotus Development Corp. v. Paperback Software International, 15 U.S.P.Q 2d 1577 (D. Mass. 1990); NME Specialty Hospitals Inc. v. Friedman, Comp. Ind. Lit. Rptr. 11,665, (D.C.N.J. 1990)
13. Broderbund Software Inc. v. Unison World Inc., 648 F.Supp. 1127 (N.D. Cal. 1986); Digital Communications Associates Inc. v. Softklone Distributing Corp., 2 U.S.P.Q.2d 1385 (N.D.Ga. 1987); Lotus Development Corp. v. Paperback Software International, 15 U.S.P.Q. 2d 1577 (D. Mass. 1990)
14. Secure Engineering Services Ltd. v. International Technology Corp., 727 F. Supp. 261 (E.D.Va. 1989)
15. Lotus Development Corp. v. Paperback Software International, 15 U.S.P.Q. 2d 1577 (D. Mass. 1990)

specifically disclose and claim software technology without directly referring to hardware other than a conventional computer and peripheral devices.[16] Patents are issuing today that protect virtually every aspect of software. Examples include program algorithms, mathematical formulae used in programs to control processes or program execution, utilities, user interface features, add-ins, compiler programs and operating system programs.

When it is recognized that programs can be protected by the laws governing trade secrets, copyrights and patents, it must be asked how can a library that accepts software components from a variety of sources protect itself from infringing intellectual property rights of parties associated with these components. Title to the software components, or at least the right to grant permission to use software to subscribers, is critical to the library. But, rights in the components will not necessarily be held by the component supplier.

TRACKING THE RIGHTS ASSOCIATED WITH A COMPONENT

A major problem in establishing and operating a reuse library is obtaining the necessary information to track the origin of software to know whether all persons with rights in the software have granted the appropriate rights to the library. Unfortunately this information is often difficult to obtain or to confirm. There are few public records that can be relied upon to trace the origin of software.

In the case of trade secrets, there is obviously no public registry that one can consult, and there are many situations in which a component supplier may not have the rights the supplier might think it has. For example, former employees of a competitor might have been employed by the software developer who wittingly or unwittingly made use of proprietary information in developing the software. Similarly, software developers may have had legitimate access to software developed elsewhere such as through a license, but have made unauthorized use of the software such as by breaching license terms prohibiting reverse engineering.

In the case of copyrights, there are similar problems. Although the Copyright Office maintains a registry, that registry does not include all the information that the library needs to know. There are certain industry practices in developing software, such as the use of clean rooms and reverse engineering, that would not be uncovered in searches of public registries. Because the demarcation between what is and what is not protected in a program is difficult to define, there is no assurance that software created using a clean room procedure is non-infringing. Programs may have been placed in the library which have been created by independent contractors who own the copyright in the programs. If software is taken out from the reuse library, modified by a contractor, and then placed back into the library, issues of ownership of derivative programs will also arise.

In the case of patents, there are also very serious potential problems. One might think that a developer of software knows whether it has infringed the intellectual property rights of another. However, in the case of patents, it can take years after an application is made for a patent for it to issue. A patent search will not locate these patents while they are still in the examination stage. A software developer might well invest significant resources in developing software and later discover that the software infringes a newly issued patent.

What can a reuse library do to protect itself against the risks posed by these situations? What can consumers of the components provided by the library do to protect themselves? Some things can be done, but as a practical matter, many risks will still be present.

The library must consider establishing procedures to investigate the origin of software in the library. It may not be enough for

16. See *10th Annual Computer Law Institute* (Practising Law Institute, 1988), pages 143-172, and "Survey of the United States Software Patents issued from July, 1987 through December, 1987", Proprietary Rights Committee, Computer Law Section, State Bar of Michigan, and John T. Soma and B.F. Smith "Software Trends: Whose Getting How Many of What? 1978-1987", *J.P.T.O.S.* May, 1989 page 415.

the library to do only quality assessments of the software deposited into the library. The library may have to analyze each of the components and require substantial audit information from component suppliers. [17] Both the library and users must know that the component supplier has the right to authorize them to use the software as contemplated by the concept of operations. An analysis of the origin of the software might be done either by the library or by the component users. Component suppliers might, understandably, be reluctant to let component users analyze the origin of their software. Because of this reluctance and the need for reasonable assurance that components in the library are non-infringing, the library may have to accept responsibility and establish proper audit procedures for tracing the origin of the software.

The reuse library should diligently search for all publicly available information concerning the ownership of software to be deposited into the library, even if the information is not complete. Searches at the Copyright Office should be done for every component to be placed into the library. These searches may detect liens, encumbrances and other interests of third parties in the software. When software is transferred into the library recordation of the transfer under the Copyright Act and under the Patent Act should be considered to avoid the loss of rights to subsequent transferees who take interests without notice.

In addition to or as an alternative to investigating the origin of software to be placed in the library, warranty and indemnity provisions relating to intellectual property rights must also be considered. The reuse plan presented by Lieutenant Colonel Morrison includes draft agreements that consider these provisions.[18] For example, in the agreement with the component supplier, there are certain warranties and indemnities with respect to title to the software. There are also warranties and indemnities in the component license. It is not important to analyze these particular provisions at this time. It is more important to recognize that there are risks that must be properly allocated.

It may be reasonable to assume that the component supplier will be in the best position to know whether it has infringed the intellectual property rights of another party and therefore be in a position to provide an indemnity against infringement. The supplier's position will no doubt be, however, that once the component is deposited into the library, there is a potential for open-ended liability to component users and others. In view of these risks the component supplier is unlikely to provide any significant indemnity protection to the library or component users without a substantial economic incentive to do so. To the contrary, component suppliers will demand that their liability be limited. If that potential liability is not limited in some way, it is conceivable that the risk of liability would be too high for many contractors to assume, thus inhibiting the practice of reuse and defeating the purpose of the reuse library.

What rights must the library acquire in software in order to facilitate reuse? It is clear that software reuse requires the right to reproduce, modify, enhance and maintain software. Some users will need to translate software from one computer language to another and to adapt it to operate with different operating systems or hardware. Some users may also need the right to reverse engineer the software and to authorize third persons to exercise the foregoing rights.

The patent and copyright laws, however, operate by conferring exclusive rights on owners of software. For example, the owner of a patent has the exclusive right to use or sell the patented invention. The owner of a copyrighted work has the exclusive right to make copies of the work, to make derivative works, and to authorize third persons to perform these acts. Neither Act was designed to facilitate reuse of software. On the contrary, the exclusive rights conferred by the Copyright Act and the Patent Act give the owners of these rights varying measures of control which necessitate their consent to implement

17. There may be reluctance on the part of component suppliers to disclose the necessary details about the origin of their software. Ultimately, however, the library will have to make a policy decision which balances the need for certainty concerning the origin of the software and the need for maintaining confidential information of the component suppliers.

18. See Appendix C of these Proceedings.

software reuse.

The Copyright Act provides an owner of a copy of a computer program with a limited exemption from infringement to make or to authorize the making of another copy or adaptation of a program provided that such new copy or adaptation is created as essential step in the use of a computer program in conjunction with the machine and that it is used in no other manner.[19] This exemption, which is conferred by Section 117 of the Act, is confined to certain limited and defined situations. It is unlikely that this exemption could be extended to allow the kinds of uses to be made of software needed for reuse without the consent of the owner of the program. Under Section 117 the person seeking to rely on the exemption must be an owner of a copy of the program, rather than a licensee. This is unlikely to be the situation in most cases. Second, any modification must be performed as "an essential step in the utilization of the program in conjunction with a machine." The cases which have interpreted the exemption strongly suggest that Section 117 has limited application for facilitating software reuse.[20] The "fair use" provisions of the Copyright Act[21] permit certain otherwise infringing uses to be made of software, but again, these provisions were not designed with software reuse in mind and are far too limited to allow any meaningful reuse.

Given this background, what rights should the library seek to acquire from component suppliers? Lieutenant Colonel Morrison has suggested that rights in software be conveyed to the library either by a transfer of title or through appropriate licenses.

Consider the first approach suggested in Lieutenant Colonel Morrison's concept of operations in which component suppliers transfer all of their intellectual property rights in the software component to the library. It is questionable whether it will be possible to motivate suppliers to transfer all of their rights in a component to the library. It must be recognized that component suppliers have legitimate interests which militate against their transferring all of their intellectual property rights to the library. For instance, suppliers may need to continue to use software transferred to the library and to maintain that software. Component suppliers may be subject to obligations to third persons which will restrict the component supplier's right to transfer the software. Furthermore, the transfer of title by the component supplier to the library will prevent the component supplier from commercially exploiting the software. A great deal of thought will have to be given, therefore, to the standard terms and conditions and reservations of title that would apply where title (rather than a license) is transferred to the library. It may be appropriate to consider assignments of rights for government purposes but not for commercial ones or for other more specific fields of use.

Licensing software to the library will also raise many issues. In most cases, the term of the licenses will be perpetual and irrevocable in order to protect component users who have made investments in a particular module or a particular part of a program being reused. The library will also have to obtain extensive rights to create derivative works and the right to authorize third persons to create derivative works. The licenses must also address who will maintain software that turns out to be defective. The implications of permitting third parties to have access to the software once it is placed into the library must be considered. Once third parties other than subscribers to the library obtain access to it, there is a potential for loss of control by the library over the use to which the modules are put and to the enforcement of intellectual property rights in the software. This may discourage contractors from becoming involved in the reuse program.

19. 17 U.S.C. § 117

20. CF, Allen-Myland Inc. v. IBM, 746 F.Supp 520 (E.D.Pa 1990); Forsight Resources Corp. v. Pfortmiller, 719 F.Supp. 1006 (D.Kan. 1989); Rav Communications Inc. v. Philip Brothers Inc., copyright L.R. (CC) 26,263 (S.D.N.Y. 1988).

21. 17 U.S.C. § 107

LIABILITY FOR INACCURATE INFORMATION AND DEFECTIVE SOFTWARE

Let me turn now to an issue that is supposed to be near and dear to trial lawyers and that is liability issues involved in software reuse. There are at least two principal forms of liability which arise from software reuse. The first is liability for inaccurate information concerning software that has been provided to the library. The second is liability for defective software including software that is not suited for its intended purpose.

Liability for Inaccurate Information

In addressing the issue of liability for inaccurate information, I do not profess to be able to resolve the myriad of issues that can arise in these situations here this morning. These situations are fact intensive and liability issues will be highly dependent on particular contract terms and the ways in which the parties allocate risks at the time of contracting. There are, however, certain types of situations in which liability can arise: (1) liability of component suppliers to the library and to other subscribers, (2) liability of the library to subscribers, and (3) liability of subscribers to the library, to other subscribers, and to component suppliers.

Liability of the Component Suppliers

Let us consider, for example, the potential liability of the component supplier to both the reuse library and to subscribers of the reuse library. In the suggested concept of operations presented by Lieutenant Colonel Morrison component suppliers will waive all implied warranties of merchantability and fitness for a particular purpose.[22] The proposed agreement between the library and the component suppliers does not, however, confront the problem of liability for the quality of the information provided by the component supplier to the library, even though it is clear that the concept of operations envisions that significant information will be provided to the library by the component suppliers. This information, which will include information concerning the design and compatibility of the program, information about data structures, documentation elements, and deficiency reports, will in turn be disseminated to subscribers. It is clear that there is a potential product liability risk to the supplier of information where a known hazard results from inaccurate information Jeppesen & Company, for example, has been found liable in several cases for providing inaccurate or unsafe instrument approach charts.[23] If these charts, which are information in a material form, are considered "products" under liability law, then data in a computer database, which is also information albeit in a different form, and information provided to a reuse library by the component supplier could well also be considered a product. If so, product liability rules may apply to component suppliers. It will also be argued that component suppliers can be liable in negligence for damages suffered by consumers of components in the library.

Because of this potential liability component suppliers will be reluctant to make software available for reuse unless the allocation of this potential risk is confronted. Agreements between component suppliers, the library and component users must address the warranties to be provided by the supplier, limitations of liability, and, perhaps most importantly, which parties will be indemnified with respect to losses suffered by end users.

Liability of the Library

22. See section 2.8 in the draft transfer agreement (Appendix C).

23. Aetna Casualty and Surety Company v. Jeppesen & Company, 642 F.2d 339 (9th Cir. 1981); Saloomey v. Jeppesen & Co., 707 F.2d 671 (2nd. Cir. 1983); Brocklesby v. United States, 767 F.2d 1288 (9th Cir. 1985)

The concept of operations will impose potential liabilities on the library to component subscribers. There will be a significant amount of information in the database of the library such as information received from component developers, consumers and quality assessments prepared by the library itself. At least four types of liability situations can result if this information is made available to component suppliers or users. They include (1) liability for inaccurate information, (2) liability for incomplete information, (3) liability for failing to disseminate critical or important information, and (4) liability for releasing proprietary information.

Because of the potential liabilities of the library, consideration should be given to the desig of the reuse library data base. It is quite clear that a data base provider must take reasonable skill and care in compiling, entering, retrieving and verifying information that it makes available to subscribers.[24] It is unclear, however, the standard of care to which the library will be held.

Generally, an information provider will be held to a standard of reasonable care. The precise steps which the library must take to insulate itself from liability cannot be precisely articulated. Ordinarily, a myriad of factors are taken into account in determining what the standard of care is in any particular situation. The type of information made available, the importance of the information, the uses to which the information can be put, the practicality of providing error-free information, who has compiled the information, and the common practices among suppliers are all important considerations. One New York case recently suggested that the failure to incorporate features into a system made possible by new technological developments also constitutes negligence.[25]

The library must be particularly careful with quality assessment information that is made available to potential subscribers. Once the library assumes the role of a quality assurer of the software, rather than merely a facilitator of information (such as a lending library), a higher standard of care with respect to the accuracy of the information may be imposed on the library. If the library holds itself out as having particular skill and knowledge to perform the quality assessment it is much more likely to be held liable for errors or omissions in the assessment.

It is even possible that component consumers will be held liable for information they provide to the library. Conceivably, subscribers will be compensated for information provided to the library concerning the software they use. Such information could be relied upon by the library and by other subscribers. In this situation, a component consumer who does not even use a component may be liable - not because the component turns out to be deficient, but because the consumer has provided inaccurate information about the component which causes damage to another user.

Liability for Defective or Unsuitable Software

We can only guess when the first action will be commenced alleging liability for defective or unsuitable software in a reuse situation. The various theories of liability discussed above underscore the importance of allocating, in advance, the risks of this liability. The concept of operations must address these risks and the allocation of these risks must be dealt with in contracts with all parties involved. If these issues are not confronted directly, there will be significant problems of open-ended liability which will result in component suppliers refusing to participate in the library. It is thus essential that these issues be well thought through.

ENFORCEMENT ISSUES

The concept of operations raises significant enforcement issues. For example, the suggested component license contemplates that components be used for only a single

24. CF. Ford Motor Credit Co. v. Swarens, 447 S.W. 2d 53 (Ky. Ct. App. 1969); State Farm Mutual Auto Insurance Co. v. Bochorst, 453 F.2d 533 (10th. Cir. 1972); Pompeii Estates Inc. v. Consolidated Edison Co., 397 N.Y.S.2d 577 (N.Y. City Civ. Ct. 1977)

25. Swiss Air Transport Co. v. Benn, 467 N.Y.S. 2d 341 (N.Y. City Civ. Ct. 1983)

purpose and for no others. The uses to which they are put by the end user will be difficult to enforce, however. Suppose there are trade secrets in a particular component.

Let us further suppose, that some trade secrets will continue to exist in the software even once the software is deposited into the library. How will the component supplier enforce its trade secret rights against a component user who has had access to the component supplier's software, including the underlying algorithms, methodologies, and design, and who then uses this information on a non-government contract? Consider further the situation proposed by Lieutenant Colonel Morrison of a user who retrieves a component but declares his intention not to use it. It is not very clear what, in this context the words "not use" means, but it is possible to envisage enforcement difficulties. A potential component consumer could "borrow" a component from the library, study the design, and learn the methodologies and algorithms employed. This consumer could then return the components to the library and declare its intention not to use the component. Will component suppliers feel comfortable with this situation if the royalties they are paid are based on a declared use of the software?

CONCLUSION

The establishment of a reuse library along the lines proposed by Lieutenant Colonel Morrison will require thinking through a myriad of intellectual property and liability issues. This paper has only briefly touched on a few of these issues. They deserve to be dealt with in much more depth.

QUESTIONS FROM THE AUDIENCE

Differences Between U. S. and Canadian Intellectual Property Law

Jack Kramer, DARPA: You talked about establishing a reuse library between Canada and the U.S. Are there significant differences between the intellectual property laws in Canada and the U.S.? Does this create a problem?

Barry Sookman: There are some differences between Canadian and United States intellectual property laws. Overall, however, there are more similarities in our respective laws than there are differences. Canada does not have legislation governing trade secrets, but trade secrets are protected under the common law. Under Canadian law, computer programs have been protected as trade secrets.[26] In 1988 Canada followed the United States and amended its Copyright Act to expressly protect computer programs as literary works. Canadian case law has now firmly established that the source and object codes of computer programs are protected by copyright, regardless of the material form in which they are expressed or embodied.[27]

26. CF, Lake Mechanical Services v. Crandall Mechanical Systems Inc. (1985, 31 B.L.R. 112 (B.C.S.C.), Ticketnet Corp. v. Air Canada (1987), 21 C.P.C. (2d) 39 (Ont. H.C.); Mortil v. International Phasor Telecom Ltd. (1988), 23 B.C. (2d) 354 (B.C.Co. Ct.); Godin v. Gary Abraham Business Consultants Inc. (1986), 2 Q.A.C. 3176 (Que. C.A.); Spacefile Ltd. v. Smart Computing Systems Ltd. (1983), 75 C.P.R. (2d) 291 (Ont. H.C.); Marque d'Or v. Claymann (1988), 21 C.P.R. (3d) 490 (Que. S.C.); Positron Inc. v. Desroches, [1989] R.J.Q. 1636 (C.S. Que.)

27. CF, Apple Computer Inc. v. Mackintosh Computers Ltd., (1986) 28 D.L.R. (4th) 178 (Fed T.D.), varied (1987), 44 D.L.R. (4th) 74 (Fed. C.A.), affirmed [1990] 2 S.C.R. 209 (S.C.C.); Spacefile Ltd. v. Smart Computing Systems Ltd. (1983), 75 C.P.R. (2d) 281 (Ont. H.C.); F&I Retail Systems Ltd. v. Thermo-Guard Automotive Products Can. Ltd. (1984), 1 C.P.R. (3d) 297 (Ont. H.C.); La Societe d'Informatiques R.D.G. Inc. v. Dynabec Ltee (1984), 5 C.P.R. (3d) 299 (Que. S.C.); Canavest House Ltd. v. Lett, (1984), 4 C.I.P.R. 103 (Ont. H.C.); Gemologists International Inc. v. Gem Scan International Inc. (1986), 7 C.I.P.R. 255 (Ont. H.C.); Selection Testing Consultants International Ltd. v. Humanex International Inc. (1987), 13 C.I.P.R. 27 (Fed. T.D.); Computer Workshops Ltd. v. Banner Capital Market Brokers (1988), 64 O.R. (2d) 266 (Ont. H.C.); Perry Engineering Ltd. v.

Under the Canadian Act the owner of a copyright in a computer program has the exclusive right, among others, to reproduce the software. There is no specific right to create derivative works, but the creation of derivative works almost always involves a reproduction of a substantial part of the underlying work and so effectively, there is no difference. Canadian intellectual property laws permit assignments, partial assignments and licenses of copyrights and patents. Therefore, there should not be major difficulties in structuring agreements to accomplish mutually agreed to objectives.

Farrage (1989), 26 C.O.P.R. 89 (B.C.S.C.); Orbritron Software Design Corp. v. M.I.C.R. Systems Ltd. (1989), 48 B.L.R. 147 (B.C.S.C.); Marquis v. DKL Technologies Inc. (1989), 24 C.I.P.R. 289 (C.S. Que.); Positron Inc. v. Albert Desroches, [1988] R.J. Q. 1636 (C.S. Que.)

PANEL DISCUSSION

LEGAL ISSUES IN SOFTWARE REUSE

Panel Members:

James Baldo, Jr., *Institute for Defense Analyses* (Chair)
Charles Lillie, *SAIC*
Charles McNally, *Westinghouse*
Will Tracz, *IBM*
Bonnie Dancy, *EVB*

Editor's Note: This presents the more significant parts of the panel discussion. The purpose of the panel discussion was to elicit a spectrum of opinions on such issues as software patents, copyrights, protection of user and component interfaces, government data rights and procurement regulations, and reverse engineering of software. The original transcript has been substantially edited for syntax and coherence, and in some cases the order of remarks that were made has been slightly rearranged to smooth the flow of the discussion. Footnotes have been added that cite specific cases and publications and that explain in more detail some key legal and technical terms and issues in law (such as specific sections of the copyright statute) that are mentioned. A few unidentified participants in the discussion are identified as "Speaker A," etc. —C.W.

SOFTWARE PATENTS

Jim Baldo, Institute for Defense Analyses: Many concerns have been expressed recently by the computer industry about software patents. Articles have appeared in the *New York Times*[1] and in technical magazines,[2] and a forum was held at the Massachusetts Institute of Technology in the spring of 1989 entitled *Software Patents: A Horrible Mistake*. Programmers fear that patents will be granted that monopolize the basic concepts that are involved in user interfaces or the algorithms that programmers need to write effective software. It has been claimed that the U.S. Patent and Trademark Office seems to be willing to issue virtually any software patent presented to it, that it lacks the facilities to search out prior art in the field, and that the personnel available are so limited that the Patent and Trademark Office is prevented from effectively examing patent applications. Patent applications stay on file in the Patent and Trademark Office in secret for an average of about three years before they are issued and published. Businesses may therefore invest considerable resources to develop and establish a product that suddenly becomes infringing when a patent springs into existence. The U.S. patent system, unlike those of many other countries, provides a more or less absolute monopoly for the patent holder. A patent holder can charge as high a price for a license as it chooses and can refuse licensing altogether. Yet, algorithms, just as much as any other advances, are deserving of the incentives and rewards of the patent system. The question that I pose to the panel and to this workshop is: "How does the software industry view software patents?"

Chuck McNally, Westinghouse: Westinghouse is a heavy industry and has always been very concerned with patents—that's where we make a lot of our money. The software patent exposure is relatively new to us, but our patent attorneys view software patents as a potential bombshell with respect to cutting down incentives for the development of software technology. This is mainly because there are no clear criteria for discriminating between software that is patentable and software that is not. There is simply not enough of a record of court decisions yet—there are perhaps 15 or 20 court decisions on software

1. Fisher, *Software Industry in Uproar Over at Recent Rush of Patents*, New York Times, May 12, 1989, A1.
2. See, e.g., The League of Programming Freedom, *Software Patents: Is This the Future of Programming?* DR. DOBB'S JOURNAL, October, 1990, 56.

patents.[3]

Ever since the Supreme Court case where software was used in the process of vulcanizing rubber,[4] the Patent and Trademark Office has said: "Okay, software is now patentable, it's fair game." Frankly we at Westinghouse are scared to death of it. Perhaps not quite scared, but definitely worried about what software patents will do to the inventiveness that is involved in software. Small businesses can't afford the legal fees that go along with patents—for patent applications, enforcing one's patents when necessary, and defending against patent infringement lawsuits filed by others. Even for large businesses, patents for software can become a legal nightmare, because these companies can make huge investments, many millions of dollars to produce products based on software and all of a sudden something can come out of the blue that is a patent infringement. We at Westinghouse have a real concern about this.

Chuck Lillie, SAIC: My concern would be how to protect the person who has invested money in building software from scratch. I certainly see that having software patents is going to stifle creativity at some point, but if someone is investing money in building a product, there has to be some protection that allows him or her to get a return on that investment.

McNally: Most of my lawyer friends are rubbing their hands with glee, because they see a great deal of litigation that is going to arise here with all the complications that are involved with third and fourth parties. It's going to be great. It's not so great for corporations, and especially small companies—they can't afford it, given today's legal fees.

Bonnie Dancy, EVB: My only exposure has been to the article in *Dr. Dobb's Journal*,[5] and frankly as a small business we are very worried.

Barry Sookman, McCarthy and Tetrault: The question we are dealing with here is a very basic one: What rights should inventors of technology have in their inventions? If you look at the history of protection of software in the United States, there were many cases in the early part of the 1980s where it was argued that computer programs should not be protected by copyright, in part because it would give too great a monopoly to computer programmers. Another argument that was made at that time against software being protected by copyright, however, was that the patent system was a more appropriate system for the protection of software. According to this argument, software is primarily concerned with algorithms and processes that are implemented, these are ideas that are applied in a certain way in a technological device, and that this was therefore a proper subject matter for patent protection. The patent system had safeguards—a way of assessing prior art, a higher standard for giving protection, and a shorter term of protection than copyright. The fundamental issue for copyright protection of software from the developer's perspective that is now being confronted in the courts is just how far section 102(b) of the U.S. Copyright Act will extend. That is the section that exempts from copyright protection processes, ideas, and methods.

Editor's Note: Section 102(b) defines legislatively the "idea-expression" dichotomy of copyright law, in which the expression of an idea is protected, but the underlying idea is not. This section states that "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described,

3. The patent cases McNally is referring to were primarily in the 1970s, and all were litigation between patent applicants whose applications were rejected by the Patent and Trademark Office. With the 1981 Supreme Court decision, the Patent and Trademark Office dropped its long-standing opposition to software patents and drastically changed its policies. Because after the 1981 decision software inventions were rarely rejected because of a software patentability issue, litigation largely stopped. In the years since that decision the PTO policy has become in the view of some considerably more liberal than the Supreme Court decisions would have reasonably mandated. This is potentially unstable because there could be a new round of lawsuits concerning patentability. this time patent infringement lawsuits in which defendants with substantial resources may argue strongly against certain forms of software patentability.

4. Diamond v. Diehr, 450 U.S. 173 (1981).

5. See note 2, *supra*.

explained, illustrated, or embodied in such work."

The Congressional Committee report accompanying this change to the copyright law[6] states: "Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the 'writing' expressing his [or her] ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law."

Sookman: If certain aspects of a program should not be protected by copyright, then the most natural regime to protect it would be the patent system, which has the safeguards I just mentioned. It is certainly true that in the 1960s and 70s the Patent and Trademark Office was opposed to software patents, and if you look at the litigation you'll see that over and over again, in the early years, the PTO refused to issue patents for software. Frequently, the PTO was overturned when the case went up to the Court of Customs and Patent Appeals, and finally the Supreme Court dealt with it in a series of decisions.[7] There is, however, at this stage, no *sui generis*[8] protection for computer programs—protection outside of the existing intellectual property laws. The United States could lead the way by developing a separate scheme for protection of computer software, and then lobby for an international treaty that would give protection not only in the United States but in other countries of the world. But unless and until the U.S. does that, from a pragmatic point of view we have to rely on the current patent system. I would think it would take fairly strong reasons for an exception to be made for software-related patents.

Will Tracz, IBM: I was involved in the Intel-NEC microcode court case and am the Chairman of the Association for Computing Machinery Special Interest Group in Microprogramming. We have a forum at our annual workshop and have been talking about proposals to give microcode some quasi-patent protection—above copyright protection—but not seventeen years, more like five to seven years protection. This is in response to the finding that microcode has been officially deemed to be software. I would be interested in knowing whether there is a serious movement afoot, or even discussion of creating a separate category of intellectual property rights particularly tailored toward software.

Editor's Note: The creation of a special, or *sui generis*, form of protection began to be seriously raised in the mid-1980s, with several law review articles discussing the idea at length. Pamela Samuelson, for example, has argued for such a law based on what she sees as the success of a special law protecting semiconductor chips passed in 1984[9], as well as her view that copyright ought not to protect utilitarian entities, and, particularly, object code that does not contain human-readable information.[10]

Leo Raskind, discussing the same issue, argues that it is unclear whether a *sui generis* approach is needed or whether the issues can be effectively resolved by incremental changes within the current structure.[11]

Richard Stern, in a third article, proposes a system that protects "noncode aspects of software" that would be somewhere between copyright and patents in terms of the originality required for protection and rights granted.[12]

There appears, however, to be no major movement to create a *sui generis* law at the present time. In a survey of members of the Computer Software Committee of the American Intellectual Property Law Association, members were asked whether they favored a *sui generis* system, or would prefer to continue with the present copyright and patent system for protecting computer software. Only about 20% favored a *sui generis* approach. Some others suggested that a *sui generis* system might eventually be desirable if the scope of copyright protection continues to be extended by the courts.[13]

A *sui generis* law does not necessarily imply a completely new conceptual framework—one

6. H.R. Rep. No. 94-1476, 94th Congr.

7. Gottschalk v. Benson, 409 U.S. 64 (1972). Parker v. Flook, 437 U.S. 584 (1978). Diamond v. Diehr, 450 U.S. 173 (1981).

8. *Sui generis* means: "Of its own kind or class, i.e., the *only one* of its own kind; peculiar." BLACK'S LAW DICTIONARY 1234 (6th ed. 1990) (emphasis in original).

9. Semiconductor Chip Protection Act of 1984, P.L. 98-620, 17 U.S.C. § 901-914 (1988).

10. Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs*, 50 MINN. L. REV. 471 (1985).

11. Raskind, *The Uncertain Case for Special Legislation Protecting Computer Software*, 47 U. PITT. L. REV. 1131 (1986).

12. Stern, *The Bundle of Rights Suited to New Technology*, 47 U. PITT. L. REV. 1229 (1986).

13. Samuelson, *Survey on the Patent/Copyright Interface for Computer Programs*, 17 AIPLA Q.J. 256 (1989).

motivation may simply be to change some aspect of the law that might be written into international treaties providing reciprocal recognition of copyrights internationally. The new semiconductor protection law used a *sui generis* approach that allowed it, for example, to have a 10-year term of protection rather than the much longer term used in copyright law, but is otherwise not very different from the copyright law. The *sui generis* question is also discussed in a recent National Research Council report.[14]

Baldo: I will finish up the discussion of software patents by relaying some suggestions that have been made that could improve the current patent system. Some people suggest, for example, reducing the current life of patents from 17 years to a shorter period. The patent lifetime could also begin at the date of application, rather than the date of issue, since some patents spend so much time in the application process that their life nearly doubles.[15] Requiring the mandatory licensing of patents is another idea, while still another is to restrict the rights of patent holders that are not making use of the patent. One of the worst abuses of the patent system is the establishment of companies that buy patents and then seek license fees from other companies.

PROTECTION FOR USER INTERFACES AND COMPONENT INTERFACES

Baldo: Most large DoD systems that involve software also require user interfaces. User interfaces for computer programs are often significant in determining how easy it is to learn to use and how efficiently users can utilize the programs. They are therefore significant factors in commercial success for computer software. Should the law protect screen displays and other aspects of user interfaces from imitation by competitors? If so, what is the best mechanism to do this?

Sookman: In many cases in the United States the courts have held that in principle the interface of a program is protected by copyright.[16] Many of the early cases were decided in real piracy situations and so the courts were constrained to do everything they could to protect something that looks like a sufficient amount of labor had gone into creating. Unfortunately, the implications for copyright law were that in the much tougher cases that came along later, those principles established in the earlier cases have been applied. In my view, interfaces are better regarded not as fanciful devices for users to look at, but as functional devices designed to do something, which is something that copyright has traditionally not protected. If you ask what types of protection you could have gotten for machines designed 30 or 40 years ago to do the same things that software is doing today, you might easily conclude that the levers you would have pulled and buttons that you would have pressed would not be things that you could copyright. If you had a particularly inventive system you could perhaps get a patent for it, but not a copyright. One recent case, for instance, involved a computer program that printed cards, and the interface was held protected by copyright. I can't help thinking that if that case had come up 40 years ago, and someone had had a manual printing press method with a nice interface for controlling it, that they never would have gotten copyright protection for that interface. Viewing interfaces in the way the courts do today results in extending an awful lot of protection, both duration of protection and scope, to people who spend money developing such interfaces. While I think that such interfaces should be protected, I think they should be protected by

14. COMPUTER SCIENCE AND TELECOMMUNICATIONS BOARD, INTELLECTUAL PROPERTY ISSUES IN SOFTWARE. Washington, DC: National Academy Press, 1991.

15. An example of a patent that took a long time in the application process is one issued in July, 1990 to Gilbert Hyatt covering the invention of the microprocessor (Single chip integrated circuit computer architecture, U.S. Patent 4,942,516.). The inventor's first application was in 1970. However, in Hyatt's case, even if his patent is successfully defended he will get royalties only for seventeen years after the patent was issued, since he was not involved in the commercial development of the microprocessor and has as yet received no royalties. Some developers can keep their invention a trade secret while in the application stage, thus effectively lengthening the life of the patent.

16. For a discussion of these cases see Samuelson, *Why the Look and Feel of Interfaces Should Not be Protected by Copyright Law*, 32 COMMUNICATIONS OF THE ACM 563 (May, 1989).

patents and not by copyright. Unless a case gets to the Supreme Court, however, it is going to be very difficult to overturn the very many Circuit Court of Appeals decisions that already exist on this subject. Barring such a Supreme Court case, or legislative changes enacted by Congress, I don't think the issue is whether interfaces should be protected—they already are. The issue is the scope of that protection—what sorts of things are protected and what sorts of things are not—and if you are a competitor, how do you design a compatible interface so as not to run afoul of existing case law?

Tracz: Are you speaking mostly about graphical user interfaces, rather than the internal software building block interfaces that are used in libraries that maintain reusable components?

Sookman: Yes. I had in mind graphical user interfaces. There would be a similar type of analysis for determining whether copyright protected other types of interfaces. And to date, there hasn't been as much case law on that.

Tracz: Given the creativity that goes into making a user interface, I believe that it is reasonable to protect it with copyright law like other textual representations. My concern is whether that protection will spill over to command sets or to internal interfaces. For example, if EVB[17] has a subroutine library—a package library in Ada—are the interfaces to those Ada bodies afforded copyright protection? Is there anything that prevents some other company from marketing a competitive set of plug-compatible Ada package specifications? The competitors might simply copy the package specifications, so the interface would be the same, but the actual implementation of the packages would be different or even have extended capabilities. I would like to ask our colleague here from EVB to comment on how they would feel if suddenly someone marketed either another Ada package or perhaps a C++ or Modula-2 or Modula-3 package of the equivalent Ada components that EVB markets?

17. EVB Software Engineering, Inc. is a commercial developer and distributor of reusable software components.

Dancy: We would be protected under copyright law, although I do not know to what degree copyright protection would extend if a competitor chose to produce, say, a C++ version of our Ada packages. I guess I would have to go to our attorney and seek to have our license prohibit someone else from benefiting from the work we have already done.

Tracz: I would like to make what I think to be a relevant analogy to hardware. If Brad Cox were here he would be talking about software ICs—that the analogy to an AND gate or an OR gate or a multiplexor chip or a shift counter or similar types of off-the-shelf hardware components could be extended to software. Implementing a software component in Ada rather than C++ is analogous to implementing hardware in TTL logic rather than CMOS or Schottky or some other technology. Is there case law that has looked at this interface issue in terms of possible expression-idea merger[18] associated with these internal interfaces? We know from the Intel-NEC case that in some situations of alleged infringement where a sequence of only 6 or 8 instructions were involved these particular instructions were obviously the only way to do it.

Sookman: The Intel-NEC case probably did about as an exhaustive analysis of the topic as any other case. If you look at the cases that have examined this idea-expression dichotomy in the context of the merger doctrine, you'll find that it hasn't been a very successful defense. The test has been to look at whether there are other ways of designing a component or programming it. If the judge or a clever trial attorney can demonstrate that there are other ways of achieving the same function, then the merger defense won't succeed. The trick in some of these cases is not to argue it on the merger doctrine, but to argue the notion that the

18. *Idea-expression merger* is based on the notion that copyright law protects only the expression of an idea, not the idea itself. If there is only one way to express an idea, the idea and expression are said to have merged, and copying would not in this case constitute infringement. See, e.g., Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738 (9th Cir. 1971).

interface is an uncopyrightable process or idea, and that analysis was done by Judge Keeton in the *Lotus* case.[19] It didn't do *Paperback* any good, unfortunately, but I think that the *Lotus* case may now be the threshold for analyzing these types of issues. This case is better than the Whelan case,[20] which applied an incorrect method of distinguishing between idea and expression in the computer context.

Tracz: The issue of copyright for an interface is significant for reuse because if interfaces were not copyrighted it could provide a way to avoid the risk of patent or copyright infringement—including the possibility of a court injunction—resulting from attempts to use a government repository. The repository could say: "If we can't authenticate the lineage of this software, we will fund independent development based on this consistent interface and then we will know that will have no infringement problems."

Jack Kramer, DARPA: Isn't it true that the European Economic Community is going the other way? I think they have come out and said that interfaces are not to be protected for many of the reasons that you raised. The U.S. position has been I believe at least so far to disagree with the European position.

Tracz: Right, there was an article in *IEEE Spectrum* that summarized the exact argument that we are going through here.[21]

Kramer: I can understand at a low level why somebody like EVB would like to protect the interfaces as well as the components themselves, but suppose I begin to build software that is derived from these components. In that case, it would be desirable to have alternate implementations. I am not sure that I like the idea that the first person who happens to come up with an interface owns the rights to it. In a reuse library, many people may build software around that interface, even build entire applications around it. All of this would be dependent on that particular interface. It would be better for other people to put packages in the reuse library that are competitive and evolvable and alternate implementations. For these reasons, I tend to back away from thinking that the interfaces should be protected.

Dancy: In the past, we have licensed our components to industry and to contractors and we have had no difficulty with regard to derivative rights, but that is going to be a different issue when it comes to a library situation. We know that we are going to enter into a growing competitive market. And we want to be in that market, and we know that we will be in essence competing with our own products, downstream. We want the protection that is due us, in the copyright law, but we don't seek to foreclose anyone else from going into the same area.

Jesse Abzug, IBM: It is important to realize, though, that were it not for copyright law protection, those EVB interfaces may never have been created in the first place.

Kramer: In order for software reuse to become a standard way of doing business, you have to get general agreement on what the architecture is. Unfortunately, the interfaces between components really define the architecture, or an awful lot of it, and I guess I tend to lean against protecting the interfaces. I agree with what you are saying and that's why I said at one level of granularity of components, I think that protecting interfaces is a good idea. But when I think of the higher levels of abstraction that you'd like to move to, I get very concerned because I think that protecting interfaces at this higher level of abstraction becomes a detriment to creating a marketplace. It would be desirable for users to be able to use alternately the Booch parts and the Grace parts[22] in one particular

---

19. Lotus Development Corp. v. Paperback Software International. Case No. 87-76-K, U.S. District Court, Massachusetts. 11 COMPUTER LAW REPORTER, 839 (1990), 15 U.S.P.Q., 2d, 1577 (D. Mass. 1990). In this case, Paperback Software, developer of a clone of the Lotus 1-2-3 spreadsheet program, was sued by Lotus for copyright infringement based on the similarity of the *Paperback* user interface to that of the *Lotus* product. The District Court held that there was in fact infringement. See Samuelson, *Why the Look and Feel of Software User Interfaces Should not be Protected by Copyright Law.* 32 COMMUNICATIONS OF THE ACM 563 (1989), and Samuelson, *How to Interpret the Lotus Decision (and How Not to)*, 33 COMMUNICATIONS OF THE ACM 27 (1990).

20. Whelan Associates, Inc. v. Jaslow Dental Laboratory, 797 F.2d 1222 (3rd Cir. 1986).

21. Chen and Zorpette, *Microelectronics and Computers* 27 IEEE SPECTRUM 32, June, 1990. The *Spectrum* article refers to a draft European Community directive that "would allow computer makers to copyright the coding—but not the ideas, logic, or algorithm—to interface the operating system with the hardware." The directive would also "ban reverse

application and still have the same interface.

Tracz: *Right, and let the markets drive* the approach, the quality factors, the intangibles, the performance attributes. There are still many dimensions that could be addressed that would distinguish the implementation of components from the interface.

GOVERNMENT DATA RIGHTS AND PROCUREMENT REGULATIONS

Baldo: Government procurement regulations are different from other systems for allocating intellectual property rights and responsibilities in that they are written by the consumer. As a result, some of the procurement regulations have seemed unbalanced in favor of the government as contrasted with more traditional areas of intellectual property law which tend to favor the creators of works. Emphasizing the need to provide incentives for creative, skilled people to motivate them to continue to produce items that would benefit society, the software industry has claimed that broad claims of rights by the government inhibit their ability to commercialize software technology and thereby recoup their investment. For example, industry is frequently concerned that if they permit the government to have access to valuable source code and other technical data containing proprietary information, this material may end up in the hands of their competitors. What impact is this effect having on planning for software reuse by U.S. industries?

Lillie: Reuse can be addressed from two perspectives: internal reuse and external reuse. Internal reuse, in which the company reuses components internally, we can all agree is happening today and there are no major problems regarding intellectual property rights. External reuse, though, will have to occur if we are going to create a reuse industry and I think in order to make that happen, the government is going to have to allow industry to maintain rights to the intellectual property that they have created.

McNally: I agree with that. One of the big problems is industry's tendency to identify its proprietary rights by designating it as limited rights when it delivers software to the government, and third parties that are not government contractors are precluded from getting that data. If we are going to have a reuse library that's going to be effective, we should have a level in that library where the development of derivative software by third parties does not require any passage of royalties or incentives back to the contractor that developed the original software. Otherwise you will have an administrative nightmare—you'd have a developer and then the first derivative, the second derivative, and the third derivative, and you wouldn't know where you were, in this line of regression. One of the recommendations that came out of the JIAWG committee was that maybe after the second derivative contractor any reimbursement to the developer would stop. Beyond the second derivative, the administrative cost would be likely to exceed the remuneration to the developer. As far as government infringement on the proprietary rights of the software developed by contractors, that depends very much on the particular agency involved. We deal with many parts of the Air Force, and some treat this proprietary information very well and some do not. It's the same with the Army, the Navy, and others. In general, I don't think that the government regulations are too restrictive, but I do think that they are too confusing. There is a good set of FAR regulations, but a lousy set of DFARS regulations,[23] that no one understands, including the government. We would be a lot better off if we did away with about 90% of

engineering" of programs.

22. These are competing libraries of reusable software components. The Booch components were developed by Grady Booch; the Grace components are sold by EVB.

23. "FARS" refers to the Federal Acquisition Regulations, which is part of the *Code of Federal Regulations* that applies to contracts with government agencies generally. "DFARS" refers to the Defense Federal Acquisition Regulation Supplement, which are analogous to the FAR regulations but which apply only to the Department of Defense.

the DFARS.[24]

Tracz: I once wrote a microcode translator to be delivered under a government contract, and there was some concern that I had used a translator writing system, YACC.[25] The problem was that I was not delivering the tool that one uses to generate the actual translator and therefore there was a question about whether the government could maintain the compiler that I had written. As it turned out, I did not have to deliver the tool to generate the software, but resolving this issue in general would seem to be significant. Applications generators[26] are a particularly interesting approach to reuse. It may make good economic sense for a developer to invest in application generator technology. This allows you to characterize an application domain, create the necessary parameters, and automatically generate a program—like stamping out cookies with a cookie cutter—and then deliver the representative system to the government. This seems to me to be a very attractive way of maintaining intellectual property rights over your reuse technology, because you've just delivered an instance of the system rather than delivering reusable components. All the reuse technology, your generic architecture and what have you, is kept in-house as a tool that you maintain control of. What I don't know is what the government regulations have to say about this approach.

McNally: The government loses a lot of material that is developed but not specified as a deliverable item in the contract. They just never ask for it.

Kramer: I think it is a very intriguing idea that the government gets rights to the tools that we use to produce our systems. There was a great push several years ago for the government to do that. On the other hand, if I am a commercial company and I have built a good tool—say an application generator—there is no way I'm going to bid that tool on a government contract if I am going to lose the tool. Another twist to that is if I have a product, I might like to use some government money, to adapt it to a new application. If the government expects me to turn over the whole product to them, I won't go that route. If the entire product was developed under government money I think it is reasonable for the government to get rights to it. But the contracting data rights regulations have to address this issue of mixed investment between the government and the contractor and we need to protect and encourage investment by the contracting community in tools to support the development and maintenance of software.

Speaker A: I am inclined to agree with Jack Kramer, but the problem that I have is that if a tool has been built with government money, it should be usable. As an example, in the SIMTEL 20 repository a Fortran to Ada converter that was built with public money is unusable. The reason it is unusable is that there is an essential database that is a part of that tool that was not delivered. Where do we draw the line? Do we deliver everything that helps to make this tool usable, just part of it, or what? What was the purpose of having this tool built to begin with?

Kramer: In that case, the government did itself in. It is probably one of those situations where the government didn't go back and get all the rights and data that they should have that was clearly developed as part of that particular activity. The situation that I was talking about is where a contractor has invested their own funds—maybe they weren't even a government contractor originally. Let's consider that same situation.

---

24. This is in part happening, at least for software. Under proposed regulations, the DFARS rules for software and technical data will be eliminated and replaced by a modified set of FAR rules that will apply to both civilian and defense agencies. See *Federal Acquisition Regulation (FAR); Rights in Technical Data, Advanced Notice of Proposed Rulemaking*. FEDERAL REGISTER, October 15, 1990, 41788.

25. A *translator writing system*, or "compiler compiler", is a computer program that, given a set of rules that define the syntax and semantics of a programming language, produces as its output a compiler for that language. It is an example of an application generator (see next footnote). YACC stands for Yet Another Compiler Compiler, a utility that is used with the Unix™ operating system.

26. An *application generator* is a computer program that has considerable knowledge about a relatively narrow application domain and that can, given a set of rules and parameters, generate a program that solves a particular problem in that domain. Application generators are frequently used to create financial applications such as accounting programs.

Assume that they had developed that database and were using it for whatever other purposes that they had. Now they win a government contract. I think the government, in negotiating for the product, needs to open up its eyes and see whether there is a necessary proprietary piece of information that is not going to get delivered. We should protect the contractor's right to retain its rights to that proprietary thing, but also buy rights to the database so that we end up with a useful tool.

Speaker A: I think the case here was that the contractor wanted to make a product out of this tool, so they held back that database. I think the government should have said, "we bought it, it is ours."

Baldo: Suppose that a government contractor developed software at private expense, and reused this software in a system that was later delivered to the government under a development contract. Suppose further that another party used the Freedom of Information Act to gain access to that software. Can that third party make use of that software, even use it in a product?

McNally: If this was developed at private expense, it was represented to the government as being proprietary, and the government treated it under its data rights provision under DFARS, then the government erred in letting it go out and it's not subject to the Freedom of Information Act.

Speaker B: One would assume that the original developer would have copyrighted his [or her] efforts and that only the modifications would be delivered with unrestricted rights to the government. The company should have protected their rights when they delivered the software to the government, clearly distinguishing between what was proprietary and what portion of the derivative work the government owned.

McNally: We have many, many government contracts where we have submitted data with proprietary rights. So far, we have had no problem with dissemination of that data to third parties outside of the government. Now the government can give it to government contractors and government employees to use. But all the agencies I have dealt with have been very observant of proprietary data, when it is in the Request for Proposal or the contract. If you come in after the fact and say "hey, that's proprietary", they sort of look askance and say, "well, you've already lost your rights." Be aware that you must mark everything that is proprietary.

REVERSE ENGINEERING OF SOFTWARE

Baldo: An important issue is the extent to which programmers are allowed to "reverse engineer" software, for example, by decompiling or disassembling it. A programmer working on a government project might buy a copy of a commercial program in a computer store, decompile it, and use the techniques from the program in developing software for the government. In this case, the commercial program would usually come with a so-called "shrink-wrap license," in which by the act of opening the software package, which is wrapped in plastic, the user is supposedly agreeing to the terms of the license, which generally prohibits reverse engineering. Some of the significant questions here are: Is this sort of shrink-wrap license actually valid? What kind of reverse engineering is legal under copyright law, which would determine the user's rights if the license was not valid?

Sookman: I cannot tell you whether every shrink wrap license will be valid or not. There are a couple of cases that have actually addressed that issue. There is a Canadian case decided by an obscure securities judge in Alberta that held in a particular instance that a shrink wrap license was not valid.[27] I think what is important to know is the way in which the shrink wrap license is brought to the attention of the user at the time of licensing. There is a series of patent cases in both our countries that have addressed the issues raised when a patent holder has tried to restrict the ways in which a product is used. The general law is that a patent holder cannot restrict the way in which a product can be used once it is lawfully put on the market, unless notice of those restrictions is brought to the attention of the buyer at the time of

27. North American System Shops, Ltd. v. King (1989) 27 C.P.R. (3d) 367 (Alta. Q. B.)

contracting. The important point is whether or not the shrink wrap terms are brought to the attention of the prospective buyer at the relevant time, and, if they are, whether or not an appropriate mechanism is used to get the consent of the buyer to the terms of the shrink wrap.

McNally: I think what the courts are going to look at is, "What is the normal use in trade?" They are going to go back to the old common law approach of asking the question, "What is normal in this particular type of trade for protection in laws?"

Speaker C: What does the shrink wrap license say? We have trade secrets in here we don't want you to discover? Suppose the product was Coca Cola, and someone chemically analyzed the Coke and finally derived formula X7, which is the essence of how they make Coke. X7 is a trade secret—there is no patent or copyright on it. Coca Cola, lots of luck, it's gone. That's what a trade secret is, it must be kept secret. If developers put a technique in their program and reverse engineering can be applied to discover the secret, I don't think they have any recourse. If you put a trade secret out there where someone can look at it, and reverse engineer it, tough luck. If, on the other hand, you are a commercial company, and you provide us with proprietary information, that is, trade secrets, and we accept this, and we turn around and provide it to another contractor under very strict guidelines that they can only use it in a certain way, and one of their employees goes out and sells it to someone, we would be liable. That is why we say if it's a trade secret we don't necessarily want to use it.

Baldo: Let me put this question to the panel. You are all software program managers and say you are developing a particular software system for the federal government. Before the project starts, are you going to warn your programming staff to not look at any copyrighted or shrink wrapped software? Don't look at it or decompile it and put what you've learned about it in the code you write because we will be held liable. Would you give such a warning?

McNally: No, I would not.

Tracz: Yes, that is common sense. Such a warning should be part of the standard new employee indoctrination and violating that warning should be spelled out clearly as a cause for termination.

McNally: What a shrink wrap license is, is a notice, it tells a person: "Here is something special." It is like putting a sign on your fence: "Beware of the dog." If they go in there, they have been warned. This is old common law too, goes way back. I think it is important because I think in the future this is the way the courts are going to decide this. They are not going to go into a lot of theories about reverse engineering. They don't understand that stuff. That is technical stuff. And courts don't like that. They look at what is the function of this thing. What is it for? And basically they say, hey, watch out buddy. This is something special and should be treated that way.

Sookman: The issues may be technical. But I think you can be sure that trial lawyers are not going to stop bringing these sorts of things to court just because they are technical. We have had lots of litigation already that has gone way beyond the capabilities of the judges to understand the issues and to decide them. But that has not stopped the courts from ruling. There was a recent patent decision involving Motorola where the judge made the comment that there was simply no way that this case should have been in his court because he did not know the first thing about patent law and he did not know the first thing about the kind of technology that was involved. But he said, "the parties are stupid enough to bring the case before the court, then I have no choice. So, here is my decision."[28] The courts that have addressed the issue of reverse engineering in cases that have so far gotten to court have not approved the practice. In one case, the court mentioned that it was common practice in the industry to reverse compile. That court found that there was infringement, but on the basis of substantial similarity, not reverse engineering.[29] In an earlier case, a case called *Hubco,*[30] the reverse decompilation was a

28. Motorola v. Hitachi, Ltd. 2 CCH COMP. L. CASES 46, 280 (W. D. Tex. 1990).

29. E. F. Johnson Co., v. Uniden Corp. of America. 623 F. Supp. 485.

30. Hubco Data Products Corp. v. Management Assistance, Inc. 219 USPQ 450 (D. Idaho 1983).

major reason for the finding of infringement. In the Intel-NEC case it was quite clear that there had been reverse engineering and there was still no finding of infringement.

Tracz: In the Intel-NEC case, reverse engineering was involved but that was not one of the issues that the court was being asked to be resolved. I just wanted to clarify that.

Sookman: Yes, you are right about that.

Tracz: They were arguing that the microcode was hardware and that people reverse engineer hardware all the time.

Sookman: But the point there is—and the court even cited a case, *Eden Toys,*[31] to the effect that even if there is some infringement at an intermediate stage, that does not mean that the product that eventually is created is infringing. To establish infringement you must establish several things, and one thing is that there must be a resemblance between the original work and the work that is finally created. If there is no such resemblance, the work that is created may not be infringing. This still does not deal with the trade secret aspect of it. I would say there would still be some fair exposure in taking the position that reverse decompilation is sanctioned, for example, under § 117. This may be true, but there are no cases yet that have held that.

Editor's Note: Section 117 of the U.S. Copyright Act provides that "it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided: (1) that such new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other manner, or (2) that such new copy or adaptation is for archival purposes only..."

The intention of § 117 (1) is murky and appears in part to have been motivated by the view that loading a computer program into main memory constitutes making a "copy." This literal a view has been questioned, since a "copy" under copyright law involves fixing it in a "tangible means of expression," implying a rather more permanent form. Section 117 (1) has generally been interpreted as allowing users to fix bugs and to make changes to allow a program's use under a new operating system, but not to add new capabilities to a program.[32]

However, the language in § 117 results from recommendations made by the Ccommission on New Technological Uses of Copyrighted Works, charged by Congress in the late 70s to study the problem. The report of this commis. on suggests that "conversion of a program from one higher-level language to another to facilitate use would fall within this right [of adaptation] as would the right to add features to the program that were not present at the time of rightful acquisition," as long as the resulting adapted program was not sold.[33] There is, however, a more recent trend to interpret § 117 more broadly.[34]

Kramer: Translations of literary works from one language to another are protected, right? If you translate something, you retain the rights to it. In a sense, though, translation is a form of reverse engineering. You have to understand it to translate it to a new language.

Sookman: If one person has a copyright in a dramatic work and then someone else makes an audiovisual work from that dramatic work, there can be copyright in each. However, if the person creating the audiovisual work does not have a license or permission from the owner of the dramatic work, then the audiovisual work is going to be infringing.

Speaker D: You have stated that reverse compilation is prohibited under these licenses, period. Is that enforceable? There are a lot of reasons to decompile things. Suppose that you worry about whether it has a virus in it, or a bug? Or you simply want to modify it faster than the vendor would do so. That is different than stealing it and reselling it. The idea that you cannot decompile, period, is sort of like you bought a mechanical box, with a prohibition against opening it up. I just cannot believe that.

McNally: We license to people and part of our license agreement may say you can decompile this to fit it into a particular user

31. Eden Toys Inc. v. Marshall Field & Co. 675 F. 2d 498 (2nd Cir. 1982).

32. See, e.g., Hubco Data Products Corp. v. Management Assistance, Inc., 219 USPQ 450 (D. Idaho 1983).

33. *Final Report of the National Commission on New Technological Uses of Copyrighted Works.* Washington, DC 1978.

34. See, e.g., Stern, *Section 117 of the Copyright Act: Charter of Software User's Rights or an Illusory Promise?* 7 W. NEW ENGLAND LAW REVIEW 459 (1985).

requirement. And that is not unusual.

Dale Henderson: There is emerging case law that says that that form of modification and adaptation would be permissible under §117 of the copyright statute [allowing copying of a program if essential to its use or for archival purposes]. Out in the Ninth Circuit, in a case I am looking at now called Foresight Resources, a District Court in Kansas said that that kind of adaptation right was permissible.[35] So this is getting expanded right now with new case law.

McNally: I think, and my Canadian friend might agree with me, that most of the courts are getting embroiled up to the gills with technical approaches. And they are going to look at the purpose of the use of the software rather than how it is used. And if the purpose is legitimate, I think that they will decide on the part of the defendant. What do you think? This is a broad brush, but?

Sookman: To date, we have not had a strong enough argument made to a court in the right type of case which analyzes historically what a person should be able to do with a work that is covered by copyright. The purpose of copyright was always to promote science and the useful arts under the U.S. Constitution, and if you buy a book you can read it and use it for other purposes. The argument has been made forcefully in the academic journals, and I am sure that if Pamela Samuelson was here, she would argue fairly forcefully that an owner of a copy of a program should have the right to study it and to learn from it. It seems to me that one day a court is going to look beyond § 117 [copying when essential to use of a program or for archival purposes], which may be fairly narrow. They will take a good hard look at the fair use doctrine, § 107.

Editor's Note: Section 107 provides that "the fair use of a copyrighted work, including such use by reproduction in copies ... for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use upon the potential market for or value of the copyrighted work." House Report No. 94-1476 states that "the claim that a defendant's acts constituted a fair use rather than an infringement has been raised as a defense in innumerable copyright actions over the years, and there is ample case law recognizing the existence of the doctrine and applying it." Section 107 represents legislative recognition of this case law, and the four "factors to be considered" codify the criteria that have been used by the courts in applying the doctrine.

Sookman: I know that Pamela Samuelson has also argued that there should be, beyond those statutory exemptions, recognition of a personal right which an owner of a copy of a work like a program should have. To establish a right like that, we are going to have to have the right kind of case with a sympathetic defendant, in which it does not look like the defendant is doing anything dishonest but is only using his program for the purpose of study and research and then ultimately comes up with a product which would be non-infringing based on traditional copyright principles. If we get that kind of case, we could well see this area of the law clarified and perhaps expanded.

Kramer: As we move into parallel architectures, I think we are going to see more and more cases where the owner of a licensed or a copyrighted work wants to modify that work. Suppose I build an application, and later want to move it to one of the parallel architectures. The application may have some EVB components in it that were licensed. I would now like to reverse engineer them so as to make them work on my new architecture. We are going to see, I think, more and more of that need on the part of the user for fair use of particular implementations. On the other hand, we want to make sure that we protect the rights of the companies that are investing in the base product.

Dancy: Well, in our case, you would not have to reengineer them, because the purpose of the library itself is to allow you to do that. I would like to raise a different issue.

35. Foresight Resources Corp. v. Pfortmiller. Case No. 88-2641. U.S. District Court, Kansas.

In the scenario involving liability for software malfunction [distributed before the workshop, and reproduced elsewhere in this proceedings] a point was raised about specifications and their role in liability when failure of software occurs. In our case, we rely on the specifications as being the product itself. I would really like to get an opinion on that.

Henderson: I think that an important point that should be brought out is that no software in object code form only should be permitted into the repository. If you want to enter object code, you have to enter source code also.

Tracz: Just the religious tirade that I was about to go on. The most successful examples of reuse have been scientific subroutine libraries and operating system utilities. Why have they been successful? It is because they have been a very stable technology and because they are very well understood concepts. I don't see why there cannot be black box reuse. One of the DARPA initiatives is called software understanding. DARPA is challenging the academic community and industry to team together to come up with an expressive interface which states in no uncertain terms—through examples, test cases, formal specifications, and textual descriptions—exactly what a component does. By having behavioral specifications and having families of implementations that describe and supplement the interfaces of particular modules you can have a library and not need to look at the code. You can have several different implementations and be successful at it. If you did a proper domain analysis or modeling, you can provide the user with the proper types of modification mechanisms through parameters. The user might then use an expert system to help select the particular options needed or rely on the defaults. The expert system would not allow you to specify parameters that would create a form of the component that does not make sense, given the abstract model the component is based on.

Henderson: I just think that is plain wrong—the only complete description of an algorithm *is* the algorithm. If you look at software as a Turing Machine, you can't write an abstract description which is going to necessarily be fully adequate in all situations with all data.

Tracz: I am not saying that the design might not be specified in Ada, but it should be abstract enough so that it is not dependent on particular operating systems or data structures—for example, whether it works with a linked list or an array. If I am doing a Fourier analysis or a matrix transform, there are different approaches. We can have different algorithms for calculating square root, yet one should be able to characterize the memory or time resource tradeoffs. That is all I care about, not what square root does. What all this leads to is reuse occurring at design time rather than code time and that we should really reuse the designs of the algorithm by recognizing when we need that particular concept.

Henderson: For practical purposes in most cases, I think what you are saying is true. But I disagree that it is true in principle.

Tracz: I'm assuming a domain-specific software architecture in which the person already knows the context to which this algorithm is applied. There is a certain framework that it all fits in so it makes sense. You are not just shooting in the dark for something, you know how to apply it because someone has already put it in context.

Kramer: I would like to side with Will in the following sense. None of us knows the internal details of an Intel XYZ chip. And yet, we all put our lives daily on the line without detailed understanding of that particular implementation. I feel very comfortable with the idea that we will have components that we will use that are black box kinds of components. They must be produced right and it helps to have a company reputation behind them and other things. But I think it is important that we break away from the idea that we've taught programmers that they need to see the detail of every piece of software that they are going to use. Most people never look at the libraries and people that have been looking at the Ada language and Ada libraries and the ISO work in numerics are beginning to realize how catastrophic those libraries are, that we are dependent on in all of our computer systems today. They are in horrible shape, they make horrible mistakes, make horrible assumptions. In twenty years of programming, none of us have ever really worried about defects in software components except where we ran into a particular

problem and then we went to fix it. I think it dangerous to think that we need to know all the details of a particular software package.

Henderson: I have a question: How should the government acquire rights to the subcontractor's components that were developed at private expense?

McNally: The same way they do for the prime.

Henderson: Why should the government acquire rights to the subcontractor's components developed at private expense through a prime contractor, since they are already developed? Shouldn't the government be going out and negotiating with the subcontractor directly?

McNally: The government does not have time.

Kramer: Also, it would then be a government GFE [Government Furnished Equipment] item.

McNally: Yes. And the subcontractor is supposedly supplementing something that the prime is doing, and there has to be an interface established there so when the government gets the delivered article the darn thing works. Once you start separating components and various manufacturers you have a horrendous interface problem.

Henderson: In a normal contractor situation the government is going out and contracting for a specific program or a specific piece of software. But we are talking about a whole different animal. The right to reuse software in any fashion, not for a specific contract. So to try to fit that procurement in the usual mold may be causing more problems than it solves.

McNally: One reason the government uses prime contractors is because they can use their management expertise in a particular area. If the government is buying an electronic warfare system, we at Westinghouse are experts. We act as a filter of problems coming from subcontractors to us before they get to the government. The government relies on the prime contractors to look at the big picture and filter out these problems and give them something that works.

Henderson: If the government wants to go out and buy a thousand packages of Lotus 1-2-3, they go to Lotus. They don't go to a prime contractor to go negotiate with Lotus. Why shouldn't these reuse components be handled in the same fashion?

Jack Kramer: If you consider a reuse industry as at least I would envision it, in which there would be many sources for reusable components, I think it is unrealistic to expect the government to have licenses to every possible library that may exist. I believe there will be many cases in which a particular prime contractor may know about a library that is somewhere out in the commercial world, and that may be the only system that that library is applicable to, a particular kind of satellite or something like that. It is very appropriate for the prime contractor to negotiate. I think it is wrong to assume that the government will go out and necessarily negotiate with every library that may exist just to have it on the shelf. I think the government ought to look at it and when the prime contractor says "I want to bid on the following components out of this library," it might very well be in the government's best interest either through the prime contractor or directly to negotiate because they see a much wider applicability of those components. But I don't think that is necessarily the first way that you would get introduced into a library or actually see it introduced into a particular DoD weapons system.

Speaker D: The ultimate answer to any dispute over rights to reuse a component ultimately becomes programming it yourself. And the economic realities will drive the decisions that are made—if a prime contractor is obligated to deliver something and he cannot get the rights that he is promised from a subcontractor then he will in all probability make arrangements to program it himself. And while this defeats the purpose of reuse, these are still the economic realities under which we will be operating for at least the next few years.

SCENARIOS EXERCISE

1. SCENARIOS RAISING LEGAL ISSUES IN SOFTWARE REUSE

The following scenarios were supplied to participants before the workshop as examples of very specific situations in which legal difficulties might be raised by certain reuse practices. The scenarios were selected to reflect a broad range of issues that are likely to be raised by reuse practices, including particularly likely problems and those that seemed especially difficult to resolve as a result of recent changes to or uncertainty in the law. The first six scenarios include an analysis written after the workshop that reflects the results of discussion at the workshop as well as further investigation by IDA, while the last two scenarios include responses from the workshop discussion. This approach was taken in part because of the complexity of some of the scenarios and in part because technical difficulties resulted in loss of audio recording of some of the scenario workshop discussions. The scenarios and analysis involving copyrights, patents, and reverse engineering were written by Craig Will; those involving liability for software malfunction and contractor-subcontractor relationships were written by Jim Baldo. The wording of some of the scenarios has been edited slightly after the workshop to improve clarity.

1.1 SCENARIO 1: COPYRIGHT INFRINGEMENT

1.1.1 Description of Scenario

In August, 1990, a contractor obtains software from a reuse library that consists of code that has been modified several times by different government contractors, uses it in a software system, and delivers the software system to the government. No copyright notices are included in the software, and it is assumed to be in the public domain. A small software developer, Compusoft, Inc., not known to have been involved in the development of the library code, claims that it owns the copyright for a portion of this code. They claim their code was developed in April 1989, and thus was presumably added to the code in the reuse library after that date. This is after the effective date of the 1988 amendments to the copyright law providing that copyright notices are no longer required. Compusoft sues for copyright infringement.

1.1.2 Questions About Scenario

1. What kind of evidence does Compusoft have to provide to initiate, and to win, such a lawsuit?

2. Can Compusoft realistically obtain an injunction against use of the software by the government until the copyright infringement lawsuit is settled?

3. Can Compusoft obtain an injunction against use of the software in other, future systems by the same contractor and against distribution of the software by the reuse library, even though the (relatively small) Compusoft code is crucial to the operation of a large module?

4. What damages is Compusoft likely to win should it show that the reuse library in fact infringed Compusoft's copyright, and thus that the contractor did as well in using it to construct an application system?

5. If the reuse library and contractor continue to make use of the Compusoft software after being notified that they are infringing, are they liable for "willful infringement" damages? What kind of damages might be reasonable and how might the reuse library protect itself if the Compusoft code must be continued to be used until a rewritten version that does not cause an infringement is substituted?

6. In the *Sony v. Universal Studios*[1] case the Supreme Court held that the court is "to look beyond actual duplication ... to the products or activities that make such duplication possible" for contributory infringement. What situations might occur in which operating a reuse library could result in the contractor or government being held liable for contributory infringement? Is this possible even though the reuse library itself did no illegal copying (such as a reuse library that only stored descriptions of programs rather than the programs themselves)? What sort of license agreements and procedures for entering components into the library would avoid or mitigate this problem, and how effective might they be?

1 4.3 Analysis and Discussion

This scenario presents an example of a potential copyright infringement problem that can occur with reuse libraries that results from recent changes to U.S. copyright law that removed the necessity for marking software with the usual copyright notice (e.g., "© 1991 George Bush") in order to obtain copyright protection. The effect of this change has been considered a significant threat to public domain software because of the risk of infringement lawsuits and, particularly, the possibility that someone who felt their copyright was being infringed could obtain an injunction against use or further development of software pending a trial, potentially preventing use or development for years.

1. Sony Corp. v. Universal Studios, Inc. 464 U.S. 417 (1983).

The changes to the copyright statute were made as part of a 1988 law[2] that was intended to make U.S. law conform to international treaties that require that copyright holders should not have to pursue "formalities" in order to enjoy their rights. The principal changes to the statute relevant here include making the copyright notice optional rather than mandatory, and, in the case of copyrights originating in foreign countries, removing the requirement that copyrights be registered before lawsuits are brought for infringement.[3] In an effort to provide motivation to copyright holders to continue to place copyright notices on works, another change was made that reduced the ability of defendants accused of infringement to claim that their infringement was innocent if copyright notices were in fact placed on the works.[4]

The copyright law generally provides for three kinds of remedies for infringement: (1) liability for damages; (2) impoundment and eventual destruction of copies; and (3) injunctions against infringement. The law generally provides that "... an infringer of copyright is liable for either—(1) the copyright owner's actual damages and any additional profits of the infringer ... or (2) statutory damages ..." Statutory damages can range from $500 to $20,000, and up to $100,000 if the infringement was "committed willfully."[5]

For a lawsuit to be brought, the copyright must be registered with the Copyright Office at the Library of Congress. This need not be done when the work was produced, but can be done at any time prior to instituting a suit (and is not required for most works originating outside of the U.S.).

The copyright law provides that injunctions can be granted that prevent infringement. Preliminary injunctions can be obtained under certain conditions to prevent infringement pending resolution of a lawsuit (which could take years). However, the standards for obtaining preliminary injunctions in any lawsuit are high, with the person bringing the suit required to show strong probability of success in winning the eventual trial and "irreparable injury" to the copyright holder resulting if there is no injunction.

2. P.L. 100-568, the Berne Convention Implementation Act of 1988.

3. 17 U.S.C. § 401 now states that copyright notices "may be placed on" copies, while § 411(a) has an exception for "Berne Convention works whose country of origin is not in the United States."

4. Subsection 401(d) was added, as follows: "(d) EVIDENTIARY WEIGHT OF NOTICE.—If a notice of copyright in the form and position specified by this section appears on the published copy or copies to which a defendant in a copyright infringement suit had access, then no weight shall be given to such a defendant's interposition of a defense based on innocent infringement in mitigation of actual or statutory damages, except as provided in the last sentence of section 504(c)(2)." [The "last sentence" refers to a section limiting infringement liability for nonprofit institutions and libraries.]

5. 17 U.S.C. § 502, 503, and 504 (1988).

In the case of infringement by the government or government contractors, injunctions can usually not be obtained.[6] (The law concerning government exemption against injunctions is more fully developed in the case of patents. For details, see the discussion below in scenario 3 concerning patent infringement.) Generally, government contractors are not subject to injunctions if the government accepts a software product containing copyrighted material or authorizes development of software using specific copyrighted material. In the case of inadvertent use of copyrighted software in a software development project funded by the government, however, if the contractor was not authorized by the government to use that software the government exemption would not apply, and the contractor would presumably be liable for infringement.

Another danger here appears to be that a software developer faced with an infringement lawsuit and an injunction against commercial distribution of a product could decide to simply stop development of that product even if an injunction against development was not obtained. This risk would presumably be highest for products in an early stages of development and for products that had more of a commercial than a government market. The risk would be especially high for use of products in which the infringed copyright originated outside of the United States.

The risks of infringement in this scenario are real for any software procurement. They pose particular problems for software reuse in situations in which (1) attempts are made to use standard reusable components that have both commercial and government use; and (2) reuse libraries or other situations are involved in which software is successively modified by different groups of people.

In the specific scenario in question, the risks are real but can be minimized. Given both the law itself and the legislative history of the changes to the copyright law, courts are likely to be reluctant to issue injunctions in cases where reuse libraries exercised diligence in attempting to avoid copyright infringement. It does appear desirable that reuse libraries develop procedures that track the source of components and modifications so as to avoid the problem posed in the scenario to the extent possible.

In general, reuse libraries themselves appear to be potentially liable for contributory infringement, even though they may have no knowledge of specific infringing acts. Contributory infringement is generally defined as the act of "one who, with knowledge of the infringing activity, induces, causes or materially contributes to the infringing conduct

6. U.S. law provides that "... whenever [a] copyright ... shall be infringed by the United States ... or by a contractor, subcontractor, or any person, firm or corporation acting for the Government and with the authorization or consent of the Government, the exclusive remedy of the owner of such copyright shall be by action against the United States in the Claims Court for the recovery of reasonable compensation as damages for such infringement ..." 28 U.S.C. 1498(b) (1988). These provisions "shall not apply to any claim [for infringement] arising in a foreign country." 28 U.S.C. § 1498(c) (1988).

of another," who "may be held liable as a 'contributory' infringer."[7] In the *Sony v. Universal Studios*[8] case, even though the court recognized liability for infringement beyond those who actually participated in the infringement, Sony was held blameless for manufacturing video recorders. "The sale of copying equipment ... does not constitute contributory infringement if the product is widely used for legitimate, unobjectionable purposes. Indeed, it need merely be capable of substantial noninfringing uses."[9] In most reuse libraries, the library itself is likely to be making copies, and therefore if there is infringement, it would be direct. If a reuse library only stored descriptions of programs, a library would seem to be relatively free of any risk of either direct or contributory infringement (assuming the descriptions themselves did not infringe). However, it is possible that a person who felt that a particular product referenced in the library infringed on his or her product could demand that the first product be removed from listing or that the library at least post a notice indicating that a dispute over copyright had been raised. It is conceivable that the library might be held liable for contributory infringement if the library did not follow a practice of doing so.

1.2 SCENARIO 2: COPYRIGHT INFRINGEMENT

1.2.1 Description of Scenario

Suppose the recent *Lotus*[10] "look-and-feel" decision is affirmed by a Supreme Court decision that takes an extremely broad view of what is protected by look-and-feel. Meanwhile, as software reuse technology develops, complex interfaces between software components begin to be used. These interfaces are not static, but can involve one component having some intelligence and communicating with another component by means of a succession of commands to and responses from the second component. Such a component might have the look-and-feel (pardon the expression) of a human at a human-computer user interface.

1.2.2 Questions About Scenario

1. Is this kind of interface likely to be copyrightable?

2. How complex might an interface need to be before it can be copyrighted?

3. Suppose look-and-feel is rejected. Would that mean that interfaces between components are likely not to be copyrightable?

---

7. Gershwin Publishing Corp. v. Columbia Artists Management, Inc. 443 F.2d 1159 (2nd Cir. 1971).
8. Sony Corp. v. Universal Studios, Inc. 464 U.S. 417 (1983).
9. 464 U.S. at 442 (1983).
10. Lotus Development Corp. v. Paperback Software International. Case No. 87-76-K, U.S. District Court, Massachusetts. 11 COMPUTER LAW REPORTER, 839 (1990), 15 U.S.P.Q., 2d, 1577 (D. Mass. 1990).

1.2.3 Analysis and Discussion

The conditions under which component interfaces are protectable by copyright have not been resolved, since there appear to be no cases that have directly involved copyright protection of software component interfaces, and it is difficult to predict what the limits of that protection might be, as well as answer questions about how complex an interface might have to be to be protected by copyright.

Answers to these questions depend on the more general issue of what aspects of a computer program, beyond the specific program statements, are protected by copyright. This has been litigated since the early 1980s, when court cases began to interpret the law. There have been three waves of cases: the first simply verified that computer software, including source code and object code, was copyrightable.[11] The second wave of cases established that higher-level aspects of program organization were protected, specifically the program's "structure, sequence, and organization."[12] The third wave, now being litigated, is aimed at testing expanded notions of copyright protection, including the limits of "substantial similarity" of two programs, and whether the so-called look-and-feel, or user interface, of a program can be copyrighted.[13]

The issue of protection of component interfaces may be decided (as the scenario suggests) by analogy with protection for user interfaces, if courts see little difference between interfaces between components and a human-software interface. In the *Lotus* case, for example—the principal case argued on the basis of look-and-feel where there is at least an initial decision—the major issue was similarity between the commands of the Lotus 1-2-3 spreadsheet program and those of a clone developed by Paperback Software.[14] These commands seem not very different than what one might see in a component interface, except for the sequence of actions that one might argue is part of the "feel" of a user interface. The sequence may not be critical, however, and, as the scenario stated above suggests, more complex, intelligent components may duplicate this aspect as well. In the *Lotus* case, the District Court held that there was in fact infringement. However, the decision itself is very ambiguous and has tended primarily to further confuse the issue.[15] The case is being appealed, and there is another, similar case being brought in California. The issue may not be fully resolved until one of these cases is brought to the Supreme Court, which may take many years.

---

11. See, e.g., Apple Computer Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3rd Cir. 1983).
12. See, e.g., Whelan Associates, Inc. v. Jaslow Dental Laboratory, 797 F.2d 1222 (3rd Cir. 1986).
13. Lotus Development Corp. v. Paperback Software International. Case No. 87-76-K, U.S. District Court, Massachusetts. 11 COMPUTER LAW REPORTER. 839 (1990), 15 U.S.P.Q., 2d, 1577 (D. Mass. 1990).
14. Id.
15. Samuelson, *How to Interpret the Lotus Decision (and How Not to).* 33 COMMUNICATIONS OF THE ACM 27 (1990).

It is also possible that the courts will see component interfaces as quite different from user interfaces. As Pamela Samuelson has noted,[16] one reason that the courts have so far allowed relatively broad protection for user interfaces is because they have seen these interfaces as artistic and aesthetic (in part because graphics are often involved), rather than as functional entities, which they really are. This has important implications for copyright law, since a traditional assumption of copyright law is that it does not apply to "utilitarian" entities. For example, the architectural drawings of a house can be protected by copyright, and others prevented from copying them, but it is not an infringement to construct a house that is a duplicate of another house.[17]

It does appear that relatively simple interfaces are not protectable by copyright. There are two issues here: the doctrine of "idea-expression merger," and the extent to which an "original" work need be "creative" to be copyrightable. Idea-expression merger is based on the notion that copyright law protects only the expression of an idea, not the idea itself. If there is only one way to express an idea, the idea and expression are said to have merged, and copying would not in this case constitute infringement.[18] This doctrine would seem to make the simpler component interfaces uncopyrightable, although Sookman has commented[19] that idea-expression merger has not been a very successful defense against infringement, and that arguing that the interface is an uncopyrightable process or idea, as was done in the *Lotus* case, is more likely to be successful.

Whether developing the interface was "creative" enough to be "original" as required by the copyright statute is also an issue, given a recent Supreme Court ruling involving the compilation of telephone directories.[20] The court ruled that the alphabetical listing of names and addresses in the white pages of a telephone directory was not copyrightable, concluding that "copyright rewards originality, not effort." This rejected the previous "sweat-of-the-brow" doctrine by which copyrightability was assured based on effort and investment. While the impact of this decision on the copyrightability of interfaces is not clear, it could imply that very simple interfaces are not copyrightable.

It may be, then, that the more complex interfaces between components that are likely to be used for reuse of large components are likely to be copyrightable, but simpler interfaces are not, and interfaces that appear to be more like languages may also not be copyrightable. However, these issues are yet to be clearly resolved.

16. Samuelson, *Why the Look and Feel of Software User Interfaces Should not be Protected by Copyright Law,* 32 COMMUNICATIONS OF THE ACM 563 (1989).

17. Imperial Homes Corporation v. Michael M. Lamont. 458 F.2d 895 (1972).

18. See, e.g., Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738 (9th Cir. 1971).

19. Panel discussion, this proceedings.

20. Greenhouse, *Copyright Protection Limited: Justices Say Phone Lists Lack Creativity,* New York Times, March 28, 1991, D1.

It is also of interest to consider that trade secret law clearly protects interfaces between components, and thus if a software manufacturer can enforce a license agreement, any interface can be protected. Such enforcement requires the interfaces to be kept secret, however, and this may be incompatible with broad distribution of knowledge about the component and how it can be used. Trade secret protection has different characteristics than copyright, however, and can be susceptible to reverse engineering. Interfaces can also be patented, and, in a very small sample of software patents studied, about 5% appeared to be interface or data storage standards.[21]

1.3 SCENARIO 3: PATENT INFRINGEMENT

1.3.1 Description of Scenario

In February, 1989, a programmer writes a reusable software component, and sends it to a reuse library. Following the standard practice of the library, the component is tested, documented, and evaluated for quality. In addition (this is a very careful library), a patent search is done to see if any patented processes are used in the component. The search comes up negative. The component is installed in the library in September, 1989.

In June, 1992, a programmer retrieves the module from the library and uses it, without modification, in software delivered to the government. Unknown to any of these participants, a patent issues in March 1991, to a party uninvolved in the reuse effort for a process used in the component.

1.3.2 Questions About Scenario

1. Who can the patent holder sue for infringement (direct or contributory)? The original programmer? The programmer who reused the component? The library? The government?

2. In the case of the programmer reusing the component, is it a defense to infringement that the patented process was contained in the internal logic of the component, and that there was no way for the programmer to tell whether a patented process was used in the component or not?

3. If a suit were brought, could the patent holder obtain an injunction against use of the patented process pending resolution of the infringement lawsuit? Is this likely? Would there be any motive for him or her to do so?

---

21. Will, *Software Patents and Economic Competitiveness.* PROCEEDINGS OF THE WASHINGTON ADA SYMPOSIUM, June 17-21, 1991, McLean, VA, Association for Computing Machinery, 136.

4. If the reuse library staff discovered the existence of the patent when it issued, but felt that the patented process was in fact a standard industry practice at the time the patent application was filed, what should they do? Continue to distribute the component as before? Distribute the component but with a warning about possible infringement? Negotiate a license from the patent holder? File a request with the Patent and Trademark Office to reexamine the patent citing evidence of prior art?

1.3.3 Analysis and Discussion

This scenario poses one rather extreme example of the kind of problems that have raised much recent concern about software patents. Unlike copyright, where it is assumed to be impossible to infringe without actual access to the original material, patent infringement can result from use of a protected technique even if the person causing the infringement conceived of it on their own.

The traditional way of protecting against patent infringement is by a patent search, but this has limitations, particularly in the case of software. Patent searches performed at the time of entering a component into a library will not necessarily protect against infringement, since information about a patent being examined is not available. The examination and processing of a patent application takes 18-24 months on the average, but can take several years in some cases and, infrequently, a decade or more. At the present time, about 500-1000 software-related patents are being issued per year[22], and it may still be feasible to monitor such new patents and compare them against a library, but it is time-consuming and, as the number of patent applications increases, the cost of such a practice may become prohibitive.

In general, there is some risk of failure of a patent search, particularly in the case of the general-purpose techniques used in software and the difficulty of describing a software technique in the existing patent classification system, which is organized largely around specific applications, rather than the more general-purpose techiques often characteristic of software.

The patent statute provides that anyone who "without authority makes, uses, or sells any patented invention" infringes the patent.[23] Furthermore, "[w]hoever actively induces infringement of a patent" is also liable as an infringer.[24] Finally, the patent statute provides that "[w]hoever sells a component of a patented machine ... knowing that" the component was "especially made or especially adapted for use in an infringement of such patent" is liable for "contributory" infringement.[25]

The patent holder can bring suit against anyone who infringes the patent and potentially obtain monetary damages that are "in no event less than a reasonable royalty

for the use made of the invention by the infringer, together with interest and costs..."[26] In the case of deliberate or willful infringement, treble damages may be assessed.[27] In addition, the patent holder can, if desired, bring suit to enjoin the infringer from making further use of the invention.[28] Claims against the U.S. government or federal contractors operating with appropriate authorization from the U.S. government, however, are limited to recovery of "reasonable ... compensation for ... use,"[29] and injunctions against infringement presumably cannot be brought, although there are definite risks related to issues of proper authorization.

Because software patents are new there may be some uncertainty about what it means to "manufacture" or to "use" a piece of software. For example, retrieving a patented program from a reuse library could be said to be "manufacturing" it, and executing the program could be said to be a "use" of the process. However, courts would presumably most likely look for money changing hands and consider infringement to only occur when it was clear that the patent was being used for profit. The original programmer (or his or her employer) of a component, the library, the programmer (or his or her employer) who reused the component, the library, and the government might all, depending upon the specific circumstances, be legitimate targets for a lawsuit. Because the statute pretty clearly makes component developers only liable for "contributory infringement" if they know their component was designed for a specific patented invention (rather than being a staple component with other uses), a component developer does not appear to be liable for contributory infringement if the component itself does not infringe a patent.

In the case where a programmer reuses a component with no knowledge of its internal mechanisms, such ignorance does not protect him or her from patent infringement. Historically, patent holders have filed lawsuits for infringement either against manufacturers of a component or the government, presumably because of the difficulty of pursuing large numbers of consumers, but developers of systems using patented components should also fear lawsuits, particularly as court actions become more aggressive, as they have in recent years.

---

22. Id.
23. 35 U.S.C. 271(a) (1988).
24. 35 U.S.C. 271(b) (1988).
25. 35 U.S.C. 271(c) (1988).
26. 35 U.S.C. 284 (1988).
27. Leinoff v. Luisin Milona & Sons, Inc., 726 F.2d 734 (Fed. Cir. 1984).
28. 35 U.S.C. 283 (1988).
29. 28 U.S.C. 1498 (1988).

A patent holder could under certain conditions obtain a preliminary injunction against use of a patented process while an infringement lawsuit was pending. While most lawsuits for patent infringement have sought only royalties, there is an increasing trend to use patents as a means of establishing a monopoly for a particular invention, as, for example, Intel has done with some of its microprocessors. This is legal as long as "misuse" of the patent does not occur. Misuse classically consists of three prohibited acts: "(1) requiring the purchase of unpatented goods for use with patented apparatus or processes; (2) prohibiting production or sale of competing goods; and (3) conditioning the granting of a license under one patent on the acceptance of another and different license."[30]

To obtain a preliminary injunction, a patent owner must establish "a strong probability of success on the merits," and "irreparable injury," with the patent owner required to make a "clear showing" that the patent is valid and infringement has occurred. In deciding whether to issue an injunction, the court "balances the hardships the respective parties will suffer from granting or withholding the injunction and considers possible effects on third parties and the public interest."[31]

Injunctions against patent infringement cannot be obtained against the government or, frequently, contractors supplying the government. For a contractor to be immune from an infringement lawsuit, it must have "the authorization or consent of the government."[32] In the case of products delivered to the government, acceptance by the government of a product using a patented invention generally constitutes such consent.[33] However, in the case of a contractor doing work for the government, the situation is less clear, and depends upon specific contract wording, whether such use is required by contract specifications, and whether written instructions requiring use of the patent are provided by the contracting officer.[34]

In cases where a manufacturer supplies both the government and a commercial market, an injunction can be obtained against commercial sale but not sale to the government.[35] Use of commercially marketed components may be dangerous to government consumers, however: a supplier may choose not to fight an infringement lawsuit and instead stop development and support of a product, or postpone further development of a

30. CHISUM, PATENTS: A TREATISE ON THE LAW OF PATENTABILITY, VALIDITY, AND INFRINGEMENT. § 19.04. New York: Matthew Bender & Co., 1991.

31. Id., § 20.04.

32. 28 U.S.C. 1498 (1988).

33. CHISUM. Id., § 16.06.

34. CHISUM. Id., § 16.06.

35. CHISUM. Id., § 16.06.

product until after settlement of a lawsuit.

Much concern has been raised about the Patent and Trademark Office issuing patents for software techniques that are standard practices at the time the application was issued. There are two issues here: is the invention for "novel" and is it "unobvious"? Both of these terms have very specific meanings in patent law. There are a number of issues concerning the patentability of an invention that has been previously known or used. It is clear, for example, that if an inventor "abandons" an invention by failing to seek a patent, then a second inventor can legitimately obtain a patent. However, if an inventor uses the invention in a product, but without patent protection, a second inventor cannot obtain a patent. The uncertainty appears when the invention is used in a product in a manner that is undetectable, such as an internal process in software. There have been conflicting court decisions (in cases not involving software), and the law is not presently very clear, although this issue may be resolved in one of the current software patent infringement cases. Of course, if the technique has been published in the open literature, then it is unpatentable (as long as the inventor does not file a patent application within a year after publication). In cases where a technique has not been published, but is very widely known and considered a standard industry practice, it could be argued that its "reinvention" would be obvious and thus unpatentable.

Appropriate policy for a reuse library in cases where patents are suspected to be invalid is particularly difficult to determine. Clearly, the library should protect itself and its customers from the treble damages possible from deliberate infringement by analyzing the claims and validity of the patent. Furthermore, any decision to continue to distribute the software should include warnings to customers concerning the risk of patent infringement. One strategy, assuming that the patent holder is interested only in royalties rather than a monopoly on the practice, is to attempt to get the supplier of the component to indemnify the library and its customers against claims for patent infringement. In cases where the validity of the patent is in question and where relatively clear evidence of its lack of validity is available (such as a publication of the technique in the open literature that was overlooked by the patent examiner), it is appropriate to provide this information to the Patent and Trademark Office, who may reexamine the patent based on this information. Under the patent statute, any person can request a reexamination provided a reexamination fee is paid.[36]

36. 35 U.S.C. §301, 302 (1988).

1.4 SCENARIO 4: PATENT INFRINGEMENT

1.4.1 Description of Scenario

A government contractor is hired to implement a set of reusable components and deliver them to a reuse library operated by another government contractor. Neither contractor actually embeds the components in a real application that does useful work, although they do run the processes as part of a test procedure.

It turns out that some of the components use patented processes.

1.4.2 Questions About Scenario

1. Are the contractors exempt from infringement liability based on the experimental exemption from case law?

2. Would the answer to this change if the experimental exemption in the patent law revisions now being considered by Congress becomes law?

1.4.3 Analysis and Discussion

Although it is not contained in the patent statute, there is a right recognized in case law of an "experimental exemption" in which a patented machine or process can be manufactured and used if it is done only for purposes of research or for evaluating the invention for possible use, and that such manufacture and use is not done for profit.[37]

A relatively recent case involving the testing of a patented drug, *Roche Products,*[38] somewhat narrowed this exemption. The defendant used a drug patented by the plaintiff to obtain clinical data for submission to the U.S. Food and Drug Administration to show the safety and efficacy of the drug; its goal was to obtain such approval so that it could manufacture and sell the drug when the patent expired. The court said that although the defendant was not selling the drug for profit directly, this use was beyond what had been traditionally allowed as an experimental exemption, and thus there was an infringement.

One reasonable interpretation may be that manufacturing reusable components and placing them in a library would still be an infringement, since it is done for profit and with the intention that the components would be used, even if none of the patented components are actually retrieved and embedded into a system.

---

37. Hantman, *Experimental Use as an Exception to Patent Infringement.* 67 JOURNAL OF THE PATENT AND TRADEMARK OFFICE SOCIETY. 617 (1985). The original 1813 decision establishing the precedent exempted "philosophical experiments" and "the purpose of ascertaining the sufficiency of the machine to produce its desired effects."

38. Roche Products, Inc. v. Bolar Pharmaceutical Co., Inc. 221 USPQ 937 (Fed. Cir. 1984).

To date, the only patent law revisions dealing with experimental exemption that have been enacted have been a provision that overturns the *Roche Products* case and allows clinical testing of patented drugs for purposes of obtaining FDA approval.[39] This provision is narrowly drawn for the case of patented drugs and does not affect other types of inventions such as software.

1.5 SCENARIO 5: PATENT INFRINGEMENT

1.5.1 Description of Scenario

A patented process is implemented in software, and an agreement is made with the patent holder that any user who executes the proper license agreement can obtain the components from a library. A user does so, then notices that the patent claim in question uses a mathematical algorithm in combination with a ROM. It appears that, given court decisions (such as the *Iwahashi*[40] case) on the patentability of algorithms, the patent would likely not have issued without the ROM being part of the claim. The user refuses to pay royalties, on the grounds that the invention is only valid when it uses a ROM, and that when it is implemented completely in software, the user is using only a part of the invention—a part that is unpatentable.

1.5.2 Questions About Scenario

1. Is the programmer correct? Is the patent holder likely to lose a patent infringement case?

2. Suppose the reuse library distributed a component that implemented the patented system, but left out the part of the system that was implemented in ROM, leaving it up to the user to either use a ROM and obtain a patent license, or implementing the missing piece in software and run the risk of patent infringement. Can the reuse library or programmer of the component that implements most of the patented system be held liable for contributory infringement?

1.5.3 Analysis and Discussion

This scenario results from the extremely confusing question of where to draw the patentability line for inventions involving mathematical equations, which is the principal barrier to patenting software.[41] In the case referred to, *In re Iwahashi*,[42] an applicant for a patent on an invention in which certain parameters are computed for pattern recognition of a signal was denied a patent by the PTO. The applicant then appealed to the Court of Appeals for the Federal Circuit (CAFC). The invention, if described as a process (method), would be unpatentable under previous court decisions, because it consisted of a mathematical equation. However, the patent claim is written as an apparatus

(machine) consisting of a combination of component parts, with each component described in terms of its function, except for one component that consists of a read-only memory (ROM). The court held that because the claim defined an apparatus, it was patentable as a machine.[43]

According to the doctrine of equivalents, a patent claim covers not only the defined invention, but also inventions that "perform substantially the same function in substantially the same way to achieve the same result."[44] This suggests that an invention that implemented the same algorithm in software only would infringe the patent, even though (paradoxically), a patent application that defined the invention in terms of software only would have been rejected as unpatentable.

The most likely result may be that a software-only implementation would infringe the patent. This issue is very close to the boundary line of patentability, and the law is very inconsistent and court decisions hard to predict. In addition, some attorneys have suggested that the software patents issue will likely raise many new issues related to the doctrine of equivalents, and that changes in the law in this area resulting from current and future litigation might be expected.

In the case of a component that implemented most (but not all) of a patented invention, the reuse library could be held liable for contributory infringement, since it distributed a component that was specifically designed to implement a particular patented invention, and which had no other use. However, a court would have to find direct infringement by someone else (which is uncertain), since there cannot be contributory infringement without direct infringement.

---

39. 35 U.S.C. 271(e) (1988). Not allowing clinical testing by a competitor before the patent expired was considered unfair because it effectively extends the life of the patent until such testing is completed and FDA approval is granted.

40. In re Iwahashi. 888 F.2d 1370 (Fed. Cir. 1989).

41. For a description of the present PTO guidelines on patentability of software, see *Report on Patentable Subject Matter: Mathematical Algorithms and Computer Programs.* 1106 *Official Gazette* 5. (August 9, 1989). See also the following critique and discussion of these guidelines: *The Patentability of Computer Programs: The PTO Guidelines, In re Grams and In re Iwahashi.* 6 COMPUTER LAWYER 21 (December, 1989). For discussions of the software patentability issue, see the following papers by Chisum and by Samuelson. Chisum takes the position that algorithms ought to be broadly patentable, while Samuelson, based on what she views as industry preference for not patenting software, argues that there is legal justification for not allowing software patents. Chisum, *The Patentability of Algorithms.* 47 UNIVERSITY PITTSBURGH LAW REVIEW 959 (1986). Samuelson, *Benson Revisited: The Case Against Patent Protection for Algorithms and Other Computer Program-related Inventions.* 39 EMORY LAW JOURNAL 1025 (1990).

42. In re Iwahashi. 888 F.2d 1370 (Fed. Cir. 1989).

43. There is some controversy over the meaning of the Iwahashi decision and over whether the PTO's interpretation of it, described in a notice published in the *Official Gazette*, is a fair characterization. See *Notice Interpreting In Re Iwahashi,* 112 OFFICIAL GAZETTE 16 (March 13, 1990), and *Commentary: The PTO and In Re Iwahashi.* IDEA: The Journal of Law and Technology (1990).

44. CHISUM, PATENTS: A TREATISE ON THE LAW OF PATENTABILITY, VALIDITY, AND INFRINGEMENT. § 18.04. New York: Matthew Bender & Co., 1991.

1.6 SCENARIO 6: SHRINK-WRAP LICENSING AND REVERSE ENGINEERING

1.6.1 Description of Scenario

A programmer employed by a federal contractor buys a piece of commercial software packaged with a shrink-wrapped license in a state that has no law regulating shrink-wrap license agreements. The agreement states that if the purchaser breaks the seal and loads the program that action constitutes agreement to the terms of the license. The agreement itself along with the shrink-wrapped program diskette is packaged in a box that is itself shrink-wrapped. The agreement gives the purchaser the right to return the software for a full refund if the license is not agreed to. The agreement prohibits decompilation, disassembly, and reverse engineering of the object code. The software bears a copyright notice.

The programmer decompiles the program, uncovers a useful new method, and applies that method in developing software that is delivered to the government. None of the original code or program organization is made use of in the new software, only the abstract method.

The developer of the original software discovers the action, and sues the contractor for theft of trade secrets. He also sues the government, requesting an injunction to halt use, and copying of the software, and asks that all copies be destroyed.

1.6.2 Questions About Scenario

1. Is the shrink-wrap license agreement basically valid?

2. If not, would it be valid if the license agreement was displayed on the package or otherwise displayed to the user at the time of sale?

3. Even if the shrink wrap license is valid, is the restriction on reverse engineering applicable even for object code that is copyrighted?

1.6.3 Analysis and Discussion

A "shrink-wrap license" is a written contract that accompanies software that states the terms of a license to use the software together with a provision that the act of breaking the seal (or, alternately, loading the software into a machine) constitutes acceptance of the terms of the license agreement. The terms always state that title (ownership) of the software is retained by the manufacturer, but that the consumer is granted a license to used the software in accordance with the stated conditions. Title is important because retaining title allows the manufacturer to protect the software under trade secret law,

which allows broader protection (such as allowing the protection of algorithms) than does copyright.

Whether shrink-wrap licenses are valid is essentially unresolved, since there has apparently only been one court decision on the topic, in a case specifically tied to a Louisiana statute that attempted to legalize shrink-wrap licenses.[45] However, several analyses in law reviews, discussed below, argue that shrink-wrap licenses are probably not enforceable in the form that they are practiced today, but are likely enforceable under other conditions.

Shrink-wrap licenses are legally "contracts of adhesion"—contracts that use standard forms and are generally presented on a "take-it-or-leave-it" basis. Although some have argued that such contracts are unenforceable,[46] the conventional doctrine is that contracts of adhesion are legal as long as they are not "unconscionable,"[47] although the law interpreting this has often been applied in an inconsistent manner.[48]

Shrink-wrap licenses, in addition to being contracts of adhesion, tie acceptance not to a signature but to a specific action, which adds still another level of uncertainty, although courts have been consistent about approving similar contracts, such as limitations of liability printed on labels, and based on such evidence two analyses of shrink-wrap licenses for software, by Puhala and by Ryan, have concluded that they probably are, in principle, enforceable contracts.[49]

However, these analyses also conclude that the practice of packaging the license so that it can be read only after the consumer has purchased the software and opened the box—by far the most common practice—probably does render the shrink-wrap agreement unenforceable. In many similar situations, such as automobile sales where a seller attempts to present the buyer with a warranty disclaimer after the sales transaction has been completed,[50] modifications of sales contracts have not been allowed.

In an attempt to resolve the issue of the enforceability of shrink-wrap licenses by legislation, Louisiana passed in 1984 a statute that attempted to legalize the practice. The law was then tested in court when one software company, Quaid Software, developed a copying program that could unlock data security software developed by Vault

45. Software License Enforcement Act, LSA-R.S. § 1961-1966.

46. Rakoff, *Contracts of Adhesion: An Essay in Reconstruction.* 96 HARVARD LAW REVIEW 1174 (1983).

47. Williams v. Walker-Thomas Furniture Co. 350 F.2d 445 (D. C. Cir. 1965).

48. See Rakoff, op. cit.

49. Puhala, *The Protection of Computer Software Through Shrink-Wrap License Agreements.* 42 WASHINGTON AND LEE LAW REVIEW 1347 (1985); Ryan, *Offers Users Can't Refuse: Shrink-Wrap License Agreements as Enforceable Adhesion Contracts.* 10 CARDOZO LAW REVIEW 2105 (1989).

50. See, e.g., Taterka v. Ford Motor Co., 25 U.C.C. REP. SERV. (CALLAGHAN) 680 (1978).

Corporation that was intended to prevent such copying. The program was developed by reverse engineering of Vault's software, which was sold with a shrink-wrap license that included a provision against decompiling or disassembling the software. Vault sued Quaid claiming that Quaid's reverse engineering of their software was a theft of trade secrets. Although neither company was based in Louisiana, Vault's license agreement specified that the laws of Lousiana would apply, and sued in federal district court in Louisiana to take advantage of the Louisiana statute. The district court, without any reasoning or citation of authority, ruled that the license agreement "could only be enforceable if the [Louisiana law, the Software License Enforcement Act] is a valid and enforceable statute."[51] The court, however, then went on to note that the Louisiana statute allowed license agreements to prohibit copying "for any purpose," and the creation of derivative works. The court found that because federal copyright law regulates copying and the creation of derivative works, the Louisiana statute "has invaded the exclusive province of the federal Copyright Act, and has gone beyond trade secrets law by outlawing reverse engineering."[52] The Louisiana law was thus unenforceable as conflicting with federal law, and the court found that the license agreement was invalid and that Quaid's reverse engineering was legal. A federal appeals court later agreed, and concluded that "at least" the provision of Louisiana law concerning reverse engineering was unenforceable.[53]

The question of whether shrink-wrap licenses for software are enforceable and can prevent reverse engineering is thus still unresolved. The *Vault* decision indicated primarily that such licenses cannot be made enforceable by legislation at the state level, although they may still be enforceable in the absence of such legislation. While the district court did state that shrink-wrap licenses would not be valid without such a law, this assertion without discussion is a weak aspect of the decision and is not very helpful as precedent, although it is consistent with other analyses that suggest that such licenses are not valid if they are only available after the sales transaction has been completed (which is the case in *Vault.*)

1.7 SCENARIO 7: LIABILITY FOR SOFTWARE MALFUNCTION

1.7.1 Description of Scenario

Company XYZ is a commercial supplier of real-time communication protocol software. The company's products are almost exclusively integrated in systems that support real-time distributed applications and they are typically designed for reuse.

51. Vault Corp. v. Quaid Software Ltd. 655 F.Supp. 750. (E.D. Louisiana 1987).
52. Vault Corp. v. Quaid Software Ltd. 655 F.Supp. at 763. (E.D. Louisiana 1987).
53. Vault Corp. v. Quaid Software Ltd. 847 F.2d 255 (5th Cir. 1988).

Company ABC has just been awarded a major defense contract for a data fusion system for a large command and control center (CCE). ABC's proposal stated it would integrate XYZ's real-time communication software with it's custom CCE application programs. The proposal shows that the XYZ software meets system requirements for communication and is much cheaper to buy than develop.

XYZ adheres to existing software standards and practices for development and reuse of software components. Testing meets all government standards. In fact XYZ provides additional testing that is beyond government requirements. The system is delivered and goes into operation with minor problems, which XYZ resolves within a reasonable time frame.

Fourteen months after the system has been in operation, communication errors rise above acceptable operational requirements and force the system to be shut down. ABC traces the problem to company XYZ's communication software—a major error is discovered that requires a major rewrite of the code.

The customer, the DoD, wants the system fixed with no charge to the government. ABC claims that XYZ is liable for the cost of fixing the error since there is a contract between XYZ and ABC that guarantees that their product meets certain performance requirements if used within specifications. XYZ claims that the software was used in an application that was outside their specifications for the communication software subsystem. In fact, XYZ claims that ABC *was at fault because they improperly* integrated the XYZ software into the system.

Several real-time communication software experts are asked to review the case. After a thorough review of the specification of the real-time communication software subsystem, questions are raised concerning XYZ's interpretation. However, all experts agree that XYZ was not negligent in developing the software but exercised prudence and diligence in its software engineering practice.

1.7.2 Questions About Scenario

1. Who is liable for the cost to repair this error?

2. The real-time communication software subsystem has a specification that attempts to define its functions and behavior. The developers of these software specifications may know the specifics of the application domain in which their component will be used (such as the case of company XYZ). However, it is impossible to predict or list all possible applications in which the software will fail (as was the case for ABC). From a practical view, users of reusable components will attempt to use a component in a system by attempting to remain with the specification constrains. Since exhaustive testing is

usually not possible (again this is a technical limitation), errors can show up after a system has been integrated and fielded. If both ABC and XYZ adhered to industry development standards and practices, should both parties be responsible for the software fix?

3. Can software specifications be considered a legal contract?

4. Does the government have no recourse since the system it contracted to build was at the limits of software reuse technology?

1.7.3 Discussion

The liability scenario identified the following concerns: 1) Was the component owned by the government or by the commercial company? and 2) How was maintenance described in the license agreement? In this scenario, it was not explicitly stated if the component was government or privately owned. For the case of a component being owned by the government, several attendees concluded that the government will have to put in place a mechanism that ensures that users of that component can get responses to maintenance problems. If the component was developed by a private company (i.e., XYZ) and integrated in the system by a prime contractor (i.e., ABC), then it would be the responsibility of the prime contractor to have the problem fixed (i.e., possibly their license with XYZ would cover the maintenance of the component).

The maintenance portion of the reusable component license is critical to avoid having the problems that were described in this scenario. Analysis must be performed to identify types of errors that can be detected or predicted after the system is fielded and language should be drafted for the license that explicitly determines the organization responsible for maintenance. It should be noted that drafting maintenance language for software reusable components is relatively new and the complexities of such drafting are not well understood.

A particular problem when responsibilities are not clear is that long delays may occur before components are repaired while different parties are arguing about the responsibilities of the respective parties. Several attendees were concerned that these delays may discourage software reuse in DoD systems. This is particularly a concern at the present time because there is much uncertainty about the law concerning liability for software malfunction, with no known cases having gone to trial that specifically involve software reuse.

1.8 SCENARIO 8: CONTRACTOR AND SUBCONTRACTOR RELATIONSHIPS

1.8.1 Description of Scenario

A large prime contractor negotiates a contract with the government, and agrees to deliver software with certain rights. That contractor then delivers to the government software that includes components developed by a subcontractor. It turns out that the rights that the subcontractor has agreed to deliver to the prime contractor for those components are less than the rights the prime contractor has agreed to deliver to the government.

1.8.2 Questions About Scenario

1. Will subcontractors who have a vested interest in maintaining some type of limited or restricted rights on their technology, their products, or components be willing to work in an environment in which a prime contractor will force them to give up more proprietary rights than they would in a normal commercial deal?

2. How should the government acquire rights to the subcontractor's components that were developed at private expense?

3. Why should the government acquire rights to the subcontractor's components developed at private expense through a prime contractor? Since they are already developed, why shouldn't the government go out and negotiate with the subcontractor directly?

4. How complex are negotiations likely to be for proprietary rights passed through a prime contractor—subcontractor relationship in programs such as JIAWG?

1.8.3 Discussion

The workshop attendees primarily viewed the data rights responsibility between the DoD, prime contractor, and subcontractor as that of the prime contractor. Specifically, the prime contractor must obtain via its subcontract the proprietary rights that are in the FAR and DFARS. If the subcontractor has proprietary rights, it can negotiate a license agreement with the prime contractor that protects those rights. The prime contractor then passes on the rights and restrictions to the DoD in such a way that the subcontractor is protected.

A major issue that was discussed in terms of this relationship was the problem of operating a joint venture in the context of the DoD (or government). Joint ventures are groups of companies working together to develop a product or service (e.g., a defense system). Joint ventures have been used extensively in the commercial world, however, they are relatively new for the DoD. The problem with joint ventures in the DoD contract

environment is that many of the corporations involved have technological expertise that is critical to their commercial business. They have a major concern that doing business with the government in a joint venture situation may result in proprietary information being passed on to other partners in the joint venture that may risk their competitive position.

Reusable components were also considered from a different view in comparison to the traditional DoD acquisition process. In a normal contractor situation, the government contracts for a specific system or piece of software. The software is part of that specific system and the rights and responsibilities for maintaining it are well defined. Reusable software, however, has the potential to be used in many systems and by many users. Therefore, a complex set of licenses may be required to protect the developer's data rights. One view on this issue was that it would be unrealistic to expect the DoD to have licenses for every software reuse library or component available. Instead, the DoD would negotiate licenses for reusable components that a prime contractor proposes for use on a specific system.

WORKING GROUP REPORTS

INTRODUCTION TO WORKING GROUPS

The working groups were instructed to focus on the National Test Bed Software Reuse Library Concept of Operations in their discussion and analysis. The reusable components would be used in a context called large scale software reuse, where a prime contractor and a large number of subcontractors would be teamed to build a major DoD system. Since each working group reviewed the plan from a specific legal perspective, they were to assume that the type of software being developed for the reuse library was to have potential use in both the contractor's commercial and DoD business areas.

Each working group produced a summary report, which appears in this section. The workshop planning committee recommended that the reports contain the following information:

a. Working group purpose and scope;

b. An identification of existing legal or contractual mechanisms for the group's specific area;

c. An identification of existing legal or contractual issues that may have an impact on software reuse; and

d. A set of recommendations based on the issues the group identified.

Each group was also asked to comment on any major changes needed to support large scale software reuse from an organizational or regulation (i.e., FAR, DFARS, etc.) perspective.

—J.B.

COMPONENT LICENSING / INTEGRATION CONTRACT WORDING WORKING GROUP

James Baldo, Jr.

Institute for Defense Analyses
Alexandria, Virginia

PURPOSE AND SCOPE

The purpose of the Component Licensing/Integration Wording Working Group was to analyze a set of existing software license and contract mechanisms for issues pertaining to software reuse that may impact the NTB Software Reuse Library Concept of Operations. The analysis was limited to new laws, integrating software between systems being built for the three services, and license agreements that would be needed for libraries.

EXISTING MECHANISMS

The group identified the following mechanisms for software licenses and contracts:

a. Contracts can have a value added clause to provide a contractor with an award fee if during development the contractor determines that reusing a component(s) provides a cost savings. One example of such an award fee is that the government splits the cost savings with the contractor.

b. During the course of developing a system, a contractor proposes a set of reusable component(s) to be developed with additional effort. If the government concludes that the proposed reusable components have potential value on future systems, the contractor is awarded additional money.

c. A contractor includes in their proposal the rationale, design, and implementation of a proposed set of reusable components. The additional cost for the reusable components is included in the contract. It should be noted that the government is encouraging software reuse in RFPs.

d. A contractor's contract specifies that components in a government reuse library will be available to the contractor for an agreed-upon time period.

ISSUES

After analyzing the NTB Software Reuse Library Concept of Operations, the group identified the following license and contract issues that may inhibit its operation:

a. Since a large number of components will be under the auspices of the NTB, the problem of software maintenance was considered intractable from a license and contract mechanism as described in the concept of operations.

b. Since consumers of reusable software components will have varying and unique needs specific to the system they are applying the component to, the NTB plan for generic license between the consumer and NTB Reuse Library will probably be too limited. The generic license plan advocated in this plan provides a license for reusable components between the NTB Library and the consumer is probably not feasible. This is an issue because consumers have different needs that may require various types of services from a developer that a generic license could not support.

c. In the NTB plan, classified and unclassified components were treated the same. There was no separate mechanism for

either category. The group agreed that classified reusable components would have a different set of issues than unclassified.

RECOMMENDATIONS

Near Term

The following recommendations were derived by the group as near-term (2-5 years) solutions for contracts and licenses incorporating software reuse:

a. Contracting Agencies should utilize input from industry during the early stages of acquisition for incorporating software reuse. For example, an agency may bring together a group of contractors (e.g., avionics contractors) to assess potential reuse standards and libraries available.

b. Contracting Agencies should use two-phase RFPs. Software reuse is evaluated during phase one for feasibility. Phase two of the RFP incorporates reuse only if feasibility is justified in phase one. The group concluded that once software reuse becomes more mature, this mechanism would not be needed.

c. License agreements for reusable components in software reuse library should be negotiated between the consumer and the developer.

d. There should be explicit language in the contract that requires developers to include the following life-cycle phase information: domain knowledge, design, implementation, testing, and usage information (when possible).

e. Contracts should require developers of reusable components to supply development information. For example, development statistics such as the number of engineering hours required to design and build a component. Currently, the government has three years to acquire such information after a contract has been completed. However, due to a shortage of resources, most of this data goes unclaimed. Requiring such data for inclusion into the reuse library provides a way to collect such information. It should be noted, that to determine if building a component from scratch or reusing an existing component is cost effective.

Long Term

The following recommendations were derived by the group as long-term (5-10 years) solutions for contracts and licenses incorporating software reuse:

a. A section covering software reuse should be addressed as a separate item in the DFARS under the software section.

b. Software reuse should be considered in the maintenance phase, such as planned upgrades to the system. Software maintenance contracts should include language that encourages the use of reusable components.

LEGAL OBLIGATIONS, SUBSCRIBERS, AND DEVELOPERS WORKING GROUP

Will Tracz

IBM Federal Systems Division
Owego, New York

PURPOSE AND SCOPE

The working group focused on identifying legal issues that could arise in establishing a government run software repository. The scope of the problem was constrained by defining a realistic Library/Clearing House Model, which was considered scalable.

EXISTING MECHANISMS

The group based the Library/Clearing House Model on the RAPID and proposed NTB- Operational Concept. Copyright and patent law, as well as the FAR and DFARS and proposed legislation were taken under consideration as being applicable.

ISSUES

The following section lists the assumptions the working group made with regards to the model being analyzed, then identifies the legal issues concerning the four parties involved in such an enterprise:

a. contracting agency

b. supplier/developer

c. clearing house/library owner

d. consumer/subscriber

Assumptions

The working group assumed that the Clearing House/Library would be created under the following conditions:

a. The contracting agency would stipulate that the software developer supply reusable software according to some "threshold requirements".

b. The Clearing House/Library would set up some threshold requirements for the software it supports.

c. The Clearing House/Library would define a classification scheme, similar to the RAPID approach, for the reusable software based on the "quality" of documentation, testing, etc.

d. The Clearing House/Library would not keep source or object code for COTS (Commercial Off The Shelf) and joint software Only pointers to how to license it from its owners would be available upon query.

e. There could be possible incentives for joint and private software owners to place their software in the Clearing House, such as reduced licensing agreements.

f. The software in the Clearing House/Library would be for use on government contracts only.

g. The contracting agency would mandate the use of software resources from the Clearing House/Library, unless justification were presented by the contractor/consumer for waivers.

h. The government runs the Clearing House/Library, or contracts to have it done.

i. The government charges some fees for

use of the repository. Use is described as access for query and distribution of artifacts.

j. The repository would maintain security classification and ITAR restriction type information.

k. Access to the repository is given on a "need to know" basis. In particular, when RFPs are let out, access rights would be given to potential bidders, subject to the security classifications appropriate to the program.

Contracting Agency Legal Issues

a. The Government Accounting Office would need to approve the contracting agency setting deposition ratios of reusable software because a contractor might argue that it costs more (50-200%). But is was felt by the group that this was a short term cost that might, in reality, be absorbed in large projects with immediate benefits.

b. If the government (contracting agency) requires the use of certain software from the repository, and "validates/certifies" its functionality, then if the contractor demonstrates a deficiency that causes some impact to cost and schedule, the government would need to work with the contractor to resolve this problem and would probably bear the cost.

Software Supplier Legal Issues

a. Perhaps the most serious responsibility for the software supplier is to provide "clear" title to the software. That is to assure the government that the software does not infringe on the copyrights of any other software (e.g., a derived work). In the event of inadvertent or malicious copyright infringement, the supplier would be responsible for clearing up the problem.

b. The supplier has no legal liability for operational deficiencies, once the software has been accepted (meets certain threshold requirements set by the repository managers). In essence, the software is supplied "as-is".

Clearing House/Library Legal Issues

a. One issue that requires further investigation centered on the government's liability or role if the software that was in its library turned out to not have clear title (infringed on someone else's software). If the supplier was available to deal with the matter, then, the government could mediate. But if the supplier was not available, then there is some question as to how to proceed. Furthermore, while it is the case that if the software in question infringed on a patent, then FAR 52.227-1, Authorization and Consent Clause stipulates that the contractor may not be sued for patent infringement (but the government can be, if royalties need to be paid), there is no (known to the working group members) current provision for copyright infringement to prevent injunction.

b. There is a possibility that the Freedom Of Information Act might be invoked to access information in the repository. Under these circumstances the security classification would need to be maintained, as well as the access fee policy applied.

c. If the government or contractor who operates the repository does not follow the security classification or ITAR (International Traffic in Arms Regulations) protection, then they could be prosecuted for gross negligence.

d. Similarly, if the government or contractor who operates the repository does not properly apply the threshold or evaluation criteria, then they could be prosecuted for gross negligence.

e. The government is responsible for reporting all errors found in the software it supports to all know subscribers (due diligence).

f. There is a possibility for conflict of interest in the case where the repository is

Government Owned and Contractor Operated (GOCO) in applying the threshold and evaluation criteria in an unbiased, objective manner.

g. There may need to be some statuary basis for the government to collect fees or pay royalties for the software in the repository.

Consumer/Subscriber Legal Issues

a. The subscriber is responsible for protecting ITAR protected and classified artifacts.

b. The subscriber is responsible for using the repository for government related contracts only.

c. The consumer is liable for any "as-is" repository software not stipulated as GFE.

d. The issue of injunction and third party damages for copyright infringements needs to be further investigated, but could be an issue.

e. The consumer is responsible for paying all usage fees.

RECOMMENDATIONS

The working group recommended a full task force of between 5-10 IPL (Intellectual Property Law) and contract attorneys spend 3-6 months focus on these issues and do more research into the possible implications.

In the longer term, possible statutes might be instituted to collect and distribute fees/royalties, if so desired. Also approval may be sought from the GAO to require deposition and reuse quotas on contracts. Furthermore, it is not certain that current revisions of the FAR may be addressing the issues raised by the working group.

ROYALTIES AND INCENTIVES WORKING GROUP

John F. Kramer

Defense Advanced Research Projects Agency
Arlington, VA

PURPOSE AND SCOPE

The charter of the Royalties and Incentives Working Group was to develop recommendations for providing incentives to contractors to develop reusable software components and to reuse components in building software systems.

We first identified and discussed the principal issues involved in providing incentives, and then developed a set of specific recommendations.

ISSUES

The issues are as follows, in order of decreasing significance:

1. Liability Associated with Intellectual Property. This involves claims or lawsuits for patent or copyright infringement or theft of trade secrets that might result from placing components in a library. How can such liability be minimized and how can it be allocated between developers, the library, and users so that developers are not discouraged from developing reusable components or reusing components?

2. Liability for Defective Software. This involves liability resulting from failure to meet performance standards for software or malfunctions of software that results in damage. How can the risk of liability be allocated between developers and users of reusable components so that neither are not discouraged from practicing reuse?

3. Loss of Proprietary Rights. This involves possible loss of proprietary (intellectual property) rights resulting when contractors place components in the library. How can these rights be protected so that developers are willing to allow their components to be entered in the library?

4. Rewards for Developers. What forms of rewards are best for developers? Examples include royalties based on number of times used, assessments of quality, etc.

5. Ease of Use. What can be done to make the library easy to use?

6. Form and Granularity of Components. What should be in the library? Which of the following representations should be included: source code, object code, designs, requirements? Should components in the form of object code also have source code available? Documentation?

RECOMMENDATIONS

We developed the following recommendations:

1. Multilayered Library. The library should be organized with a minimum of three different layers. The layers are:

a. Public Domain Layer. This layer to allow access to software in the public domain, which would be available to anyone who is a subscriber to the library.

b. Limited License Layer. This layer would allow access to software that was included in a standard license agreement.

c. Clearinghouse Layer. This layer would provide services similar to that of a card catalog, with the user being required to negotiate a license agreement with the vendor themselves.

2. National Asset Library. A national library should be established to serve as a Defense Department asset, to be managed by

the Defense Logistics Agency or similar organization. This library should have interconnectivity with other libraries, including those for SDIO, industry (such as the Reusable Ada Avionics Software Packages, or RAASP), university, and the STARS library. This interconnectivity would allow the library to be easily accessible. One issue that must be addressed for this library is how to provide appropriate security so that users can access appropriate layers.

3. Needed Infrastructure. Many things must be done to provide an infrastructure that can promote reuse. These include:

a. DoD Policy. Policies and directives at the Defense Department level must be developed that define how reuse is to be done.

b. Evaluation Criteria. A standard set of evaluation criteria must be developed that can measure reuse practices, both for evaluating proposals and for calculating award fees.

c. Training. Education and training is necessary to provide program managers, contractor management, and engineers with good approaches to reuse.

d. Insurance. Some form of insurance should be provided to cover the risk of defective software or liability associated with proprietary rights. This could involve the insurance industry or a self-insurance program managed by the library.

e. Standards. Standards should be developed for the library so that functional components and documentation are created in a standard way.

f. Procedures and Criteria. Software engineering procedures and evaluation criteria, such as the Software Engineering Institute evaluation criteria, should be modified to include reuse practices.

4. Core Set of Agreements. A standard set of licenses and fee calculation agreements must be developed that implements the means selected for the library to protect the rights and provide incentives for developers. These include:

a. Subscriber License. This license covers everyone who interacts with the library.

b. Standard License Agreement. This is used as a framework for licenses to allow components to be entered into the library or to use components.

c. Set of Royalty Agreements. These agreements could be accessed by users of the library and disclose to them the fees necessary to make use of particular components.

d. Incentive Fees. This fee might, for example, be paid to a consumer who, in the process of reusing a component, fixes a bug in it.

e. Award Fees. This fee is for the development of a reusable component.

All of these agreements must be mapped out and agreed to in advance by all parties.

CONCLUSIONS

In general, both management and technical personnel must be involved in developing the necessary policies, procedures, guidelines, and license agreements for reuse. However, the fundamental issues involved in incentives are economic ones, and these issues must be resolved or a reuse program will not succeed.

POTENTIAL APPROACHES TO SOFTWARE INDUSTRY PARTICIPATION WORKING GROUP

Charles Lillie

SAIC
McLean, Virginia

PURPOSE AND SCOPE

Software reuse is a new discipline just beginning to evolve into a recognized industry. To be successful, software reuse must be embraced by the government and supported by the software industry. The purpose of the Potential Approaches to Software Industry Participation group is to identify mechanisms and issues, and make recommendations that will help the government stimulate a software reuse industry. The term "software reuse industry" includes government sponsored and supported software asset libraries as well as industry libraries of commercially available reusable software assets. A software asset includes requirements, specifications, design, source code, object code, test cases, documentation, descriptions, evaluation, pointers to other libraries, or any component related to software development.

The scope of this group is to identify government actions and activities that will stimulate a software reuse industry. The group recognizes that the software industry plays an important role in stimulating a reuse industry, but time constraints forced us to focus on government activities. To further limit our scope, most recommendations are directed to the Strategic Defense Initiative program. The group feels that with minimal modifications the recommendations could be scaled up, or reused, to be applicable to the Department of Defense.

EXISTING MECHANISMS

A mechanism is an element that can be used to encourage or discourage a successful software reuse industry. A mechanism can be something as abstract as competition or as concrete as a request for proposal (RFP). The group identified existing and potential mechanisms that provide either positive (encouraged) or negative (discouraged) application of software reuse.

Encouraging Mechanisms

Encouraging mechanisms can be used to apply, enforce, or stimulate software reuse within a given project, across projects, or across the software industry. Table 1 lists mechanisms that encourage software reuse and help stimulate a reuse industry.

Table 1. Encouraging Mechanisms

a.	Competition
b.	RFP/Source Selection Criteria
c.	Library Management Tools
d.	Software Engineering
e.	Emphasis on Prototyping
f.	Government Risk
g.	Contracting Machanisms
h.	Standards
i.	Education

Competition is the corner stone of the American marketplace. Successful software reuse will help the software industry reduce cost, improve quality, and sustain realistic development schedules. Private industry will support a national reuse program to maintain a strong competitive edge. By effectively using competition the government can stimulate the software industry to reuse software assets.

An important part of government procurement is the set of criteria used to evaluate proposals. The government can stimulate and encourage the software industry to reuse software assets by including evaluation

criteria in RFPs that specifically address software reuse. Source selection evaluations for software intensive programs could be structured to award points to bidders that identify and justify reusing software assets.

Adequate technology is needed to establish and expand an industry. Software reuse needs tools that require less time to find and retrieve assets than it would take to build the asset from scratch. Library management tools have matured to the point that they encourage software reuse.

Professional software engineering practices provide techniques that help build reusable assets as well as encourage the use of existing assets. By emphasizing software engineering in all software development projects the government can help stimulate industry to not only take advantage of reusable assets, but to create higher quality software products.

Prototyping is used to help refine requirements. Effective and efficient prototyping programs will take advantage of existing software components and establish demands for reusable software assets.

A useful mechanism that will stimulate a software reuse industry is government commitment. DOD program managers must be willing to take some risk to encourage the creation and application of reusable assets. Industry will respond positively once DOD has shown its commitment to reusing software assets. In addition, the DOD needs to take advantage of existing contracting mechanisms as well as create contracting mechanisms that encourage reuse across programs. A successful reuse program requires innovation in contracting activities.

Standards help stimulate reuse by providing a basic structure for the reusable product and confidence that the product will have a long life. Standard programming languages like Ada encourage developers to create reusable assets while standard component definitions and library architectures give the developers the longevity structure to realize a return on investment.

Education plays a key role in establishing a reuse industry. Upper management must be aware of the benefits and risks of reusing software assets so that they can make sound business decisions. Designers and implementors must be aware of the technical benefits and risks of incorporating reuse into their design and implementation methodologies so that a solid product will emerge. One mechanism that has proven useful on past technology insertion efforts is conveying success and failure stories via lessons learned documents, seminars, and conferences. In addition, technical tutorials that address reuse design techniques will help engineers learn techniques to expand reuse applications.

Discouraging Mechanisms

Discouraging mechanisms can be used to apply or enforce practices that depress software reuse within a given project, across projects, or across the software industry. Table 2 lists mechanisms that discourage software reuse and help dishearten a reuse industry.

Table 2. Discouraging Mechanisms

a.	Program Manager Pressures
b.	Current Source Selection Criteria
c.	Competition
d.	Lack of Critical Mass of Reusable Components
e.	State-of-Practice in Software Development
f.	Support Issues

Pressures are placed on program managers to stay within schedule and budget. New unproven technology is not readily embraced for fear that the new technology, although theoretically superior to existing technology, may cause schedule slippage or an increase in budget.

Corporate culture, including both government and industry, have investments to protect and are suspicious of technology from other sources. The mindset of not using technology that is "not invented here" must be changed before software reuse will be embraced by government and industry.

The idea behind reusing software assets is to provide software components developed in such a way that the components can be used by other developers on other projects. The government needs to change the policy requiring unlimited data rights and allow the

developers the right to reuse or resell software assets developed for the government to other government projects or commercial ventures.

Current source selection criteria do not award developers for reusing software assets. In some cases the developer is penalized for bidding the application of existing software assets. Source selection criteria, especially those criteria used for software intensive programs, must realize the importance of reusing existing assets, and give extra points to the bidders innovative enough to reuse existing software assets.

In some software domains competition is so intense that program managers are not willing to take additional risks involved with introducing new technology. Software reuse initiatives must be introduced that convince the program managers that software reuse will improve their competitive edge.

A set of reusable assets is necessary before reuse can be implemented. Although there are some commercially available software assets, there are not nearly enough to provide the components Brad Cox identified as being necessary to stimulate a software revolution. In addition, the government could "prime the pump" by providing initial libraries and software assets that can help spawn a software reuse industry.

Current state-of-the-practice in software development does not promote reuse. In fact, current state-of-the-practice makes it more difficult to reuse external assets than it is to recreate that asset. Integrated software development tools and on-line software libraries are needed to help promote software reuse within and across projects.

Although reusable software assets exist today, responsibility for supporting those assets are not clear, especially if the assets are part of a government supported library and data rights issues make asset ownership unclear.

ISSUES

For software reuse to become a viable industry the following issues must be addressed. The issues are divided into two groups, issues that are critical to successful reuse and issues that inhibit reuse.

Critical Issues

Issues that are critical to successful reuse are those issues that have a potential to prevent the occurrence of large scale software reuse. Table 3 lists the issues that must be resolved in order to establish a successful reuse industry.

Table 3. Critical Issues

a.	Establish a critical mass of reusable assets
b.	Demonstrate that software reuse works
c.	Reuse standards
d.	Certification mechanism
e.	Legal

a. Establish a critical mass of reusable assets. Quality reusable assets must be available before designers and implementators can benefit from reuse technology. Government and industry developers must stimulate the build-up of this critical mass of reusable software assets for large scale reuse to occur. Questions such as how does one stimulate the build-up to a critical mass, what amount of assets comprise a critical mass, and what types of packages should constitute a critical mass must be answered.

b. Demonstrate that software reuse works. For wide spread, large scale software reuse to occur, improved benefits and reduced risks offered by the technology must be demonstrated. These demonstrations must show at a minimum that project costs will decrease and project schedules will not expand.

c. Reuse standards. Software reuse standards are needed to provide structure to reusable assets. Standards that provide structure for interfaces, asset design, asset implementation, and library storage and retrieval will help promote reuse and make assets accessible to a wider community of software developers.

d. Certification mechanisms. Certification mechanisms will provide a means to evaluate assets and supply a degree of confidence in quality and reliability. These mechanisms could provide services like validation, library acceptance, and national registry of lineage. Without certification mechanisms large scale software reuse will be critically hampered and industry growth suppressed.

e. Legal. There are many unknowns with respect to interpretations for software reuse in the FAR and DFARS. Contractors have concerns about ownership rights which raise questions about the extent of their intellectual property rights (i.e., copyright and patents). Also, liability and licenses agreements need to be established or addressed.

f. Technology transfer mechanism. The transition of new technology from conception to widespread use takes from 12 years to 24 years. Technology transfer mechanisms are needed to help expedite the use of reusable software assets. One method is referred to as the Johnny Appleseed approach where people with reuse expertise act as catalyst for reuse on specific projects. When reuse culture is established they move on. Others methods like education and training, seminars, conferences, and word of mouth are also effective technology transfer techniques. Experience at the Software Engineering Institute (SEI), the Software Productivity Consortium (SPC), and Microelectronics and Computer Consortium (MCC) offer insight into issues of technology transfer.

Reuse Inhibitors

Issues that can potentially inhibit large scale software reuse are listed in Table 4. These issues will not prevent the occurrence of large scale reuse, but unless resolved will hamper timely development.

Table 4. Reuse Inhibitors

a.	Economic model of reuse
b.	Reuse metrics
c.	Security
d.	Access to/Protection for Proprietary Software

a. Economic model of reuse. An economic model of reuse is needed to effectively evaluate the benefits and risks of developing reusable assets and reusing existing assets. A number of models have been proposed and used, such as IBM's frequent flier point system, the Joint Integrated Avionics Working Group model, the National Test Bed model, the Co-op for Small Companies model. These models should be evaluated to provide prospective users a means of selecting the model that best meets their needs.

b. Reuse metrics. A reuse measurement plan is needed to effective evaluate an emerging technology like software reuse. The measure plan would provide a domain analysis, propose a measurement process, identify metrics to apply, and furnish a plan to analyze and apply the results.

c. Security. Two concerns need to be addressed by this issue. First, physical security, that is, protecting classified software while making it available for reuse needs to be addressed if the software industry is to take full advantage of reusable software assets. Second, protecting software from intentional malice such as computer viruses is important to the success of software reuse. Keeping software "healthy" and having a method to check that health will promote confidence in reusable assets and increase reuse applications.

d. Access to/Protection for Proprietary software. For software reuse to be successful, public access to reusable assets is essential. Proprietary software issues must be addressed that would allow this

public access but also protect the investment by the developers.

RECOMMENDATIONS

The following recommendations are offered to help expedite software reuse technology initiative specifically and DoD in general, they are equally applicable to the software industry. Each recommended is targeted to one or more issues and mechanisms identified in sections 2.2.2 and 2.2.3.

a. 1. Create Reuse Standards Committee (DOD Wide)

(1) Encouraging mechanism: Software Engineering

(2) Discouraging mechanism: State-of-Practice in Software Development

(3) Critical issue: Reuse Standards

(4) Reuse inhibitor: Reuse Metrics

b. Reinforce Priority of SDI Reuse Committee

(1) Encouraging mechanism: Library Management Tools

(2) Discouraging mechanism: State-of-Practice in Software Development

(3) Critical issue: Certification mechanism

(4) Reuse inhibitor: Economic model of reuse

c. Develop Incentives to Foster Growth of Reuse Industry

(1) Encouraging mechanism: Standards

(2) Discouraging mechanism: Lack of Critical Mass of Reusable Components

(3) Critical issue: Establish a critical mass of reusable assets

(4) Reuse inhibitor: Economic model of reuse

d. Certification of Software Originality

(1) Encouraging mechanism: Library Management Tools

(2) Discouraging mechanism: State-of-Practice in Software Development

(3) Critical issue: Certification mechanism

(4) Reuse inhibitor: Security

e. Incremental Implementation Strategy

(1) Encouraging mechanism: Government Risk

(2) Discouraging mechanism: Program Manager Pressures

(3) Critical issue: Demonstrate that software reuse works

(4) Reuse inhibitor: Economic model of reuse

f. Reuse Consortia

(1) Encouraging mechanism: Software Engineering, Standards

(2) Discouraging mechanism: Corporate Cultures

(3) Critical issue: Technology transfer mechanism

(4) Reuse inhibitor: Economic model of reuse

g. Sponsor Domain Analysis

(1) Encouraging mechanism: Emphasis on Prototyping

(2) Discouraging mechanism: Lack of Critical Mass of Reusable Components

(3) Critical issue: Establish a critical mass of reusable assets

(4) Reuse inhibitor: Economic model of reuse

h. Sponsor Pilot Program to Demonstrate Reuse

(1) Encouraging mechanism: Education

(2) Discouraging mechanism: Program Manager Pressures

(3) Critical issue: Demonstrate that software reuse works

(4) Reuse inhibitor: Economic model of reuse

i. Programmer Development Kits (Standard Parts and Interfaces)

(1) Encouraging mechanism: Standards

(2) Discouraging mechanism: Support Issues

(3) Critical issue: Reuse standards

(4) Reuse inhibitor: Reuse metrics

j. Electronic Marketplace/Catalogs/Other Distribution Media

(1) Encouraging mechanism: Competition

(2) Discouraging mechanism: Competition

(3) Critical issue: Technology transfer mechanism

(4) Reuse inhibitor: Economic model of reuse

k. Adequate Weighted Software Reuse

(1) Encouraging mechanism: RFP/Source Selection Criteria

(2) Discouraging mechanism: Current Source Selection Criteria

(3) Critical issue: Establish a critical mass of reusable assets

(4) Reuse inhibitor: Access to/Protection for proprietary software

CONCLUSIONS

The working group thinks that software reuse will be an accepted technology in the 1990s. Although the critical issues must be resolved for large scale reuse to become a reality, software reuse will evolve without resolving the issues, just much slower.

APPENDICES

APPENDIX A

WORKING GROUP AND PLANNING COMMITTEE MEMBERS

Component Licensing/Integration Contract Wording Working Group

(1) Charles McNally - Westinghouse (chair)
(2) James Baldo Jr. - POET/IDA
(3) Bonny Dancy - EVB
(4) Hugh Klipp - USAISSDCW
(5) Frank Poslajko - USASDC/CSSD-SP
(6) Bob Saisi - DSD Labs
(7) Terry Sigoren - NRL
(8) Steve Wynn - USASDC

Legal Obligations, Subscribers, and Developers Working Group

(1) Will Tracz - IBM FSD (chair)
(2) Jesse Abzug - IBM IPL
(3) Murray Baxter - JAL-IP
(4) Russ Geoffrey - NTBJPO
(5) Robert Hamilton - USASDC
(6) Bill Zanca - SDIO

Royalities and Incentives Working Group

(1) Jack Kramer - DARPA (chair)
(2) Rich Bertel - Martin Marietta/ISG
(3) Eric Besser - Westinghouse Aerospace Software Eng.
(4) Vincent Bia - NTI
(5) Gary Cox - IBM
(6) William Farrell - DSD Labs
(7) Dale Henderson - SDRO/PTN
(8) Kerry Kent - MITRE
(9) Chuck Knostman - MITRE
(10) Kerrie Light - Martin Marietta
(11) Bob McCauley - Martin Marietta
(12) Joanne Piper - USAISS DCW
(13) Keith Rathjen - Rockwell
(14) James Sutton - AFSPACECOM/JA
(15) Brian Whitney - Martin Marietta
(16) Joseph Zimmerman - ESD/PKRX

Potential Approaches to Software Industry Participation Working Group

(1) Charles W. Lillie - SAIC (chair)
(2) Capt Emily Andrew - NTBJPO
(3) Ron McCain - IBM
(4) LtCol John Morrison - NTBJPO
(5) Diana Quinn - IBM
(6) Otis Vaughn - Teledyne Brown Engineering
(7) Alison Wildblood - EVB Software Engineering Inc

Planning Committee

(1) Capt Emily Andrew - NTBJPO
(2) James Baldo, Jr - IDA
(2) LtCol John Morrison - NTBJPO
(3) Will Tracz - IBM
(4) Craig Will - IDA

APPENDIX B

ATTENDEES LIST

Jesse Abzug
IBM
6600 Rockledge Drive
Bethesda, MD 20817
Phone: (301) 493-1214

Capt. Emily Andrew
NTBJPO
Falcon AFB
Colorado Springs, CO 80912
Phone: (719) 380-3255
Fax: (719) 380-3255

James Baldo
IDA/POET
1801 N. Beauregard Street
Alexandria, VA 22311
Phone: (703) 845-6624
Fax: (703) 845-6848

Murray B. Baxter
JAL-IP
5611 Columbia Pike
Falls Church, VA 22041
Phone: (703) 756-2622

Rich Bertel
Martin Marietta/ISG
4795 Meadow Wood
Chantilly, VA 22021
Phone: (703) 805-5601

Eric Besser
Westinghouse Aerospace Software Eng.
Defense & Electric Center
P. O. Box 746, MS-432
Baltimore, MD 21203
Phone: (301) 765-1010

Vincent Bia
NTI
P. O. Box 100
Leupp, AZ 86035
Phone: (602) 686-6391

Gary Cox
IBM
Falcon AFB
Colorado Springs, CO 80912
Phone: (719) 380-2463

Bonnie Dancy
EVB
5303 Spectrum Drive
Frederick, MD 21701
Phone: (301) 695-6960

William Farrell
DSD Labs
75 Union Avenue
Sudbury, MA 01776
Phone: (617) 443-9700

Russ Geoffrey
NTBJPO
Falcon AFB
Colorado Springs, CO 80912
Phone: (719) 380-2146

Robert Hamilton
USASDC
P. O. Box 1500
Huntsville, AL 35807
Phone: (205) 895-4680

Dale Henderson
Los Alamos
National Laboratory
PO Box 1663
A-DO Mail Stop F606
Los Alamos, NM 87545
Phone: (505) 665-2151

Kerry Kent
MITRE
Falcon AFB
Colorado Springs, CO 80912

Phone: (719) 380-3319

Hugh Klipp
USAISSDCW
Stop H-4
Ft. Belvoir, VA 22060
Phone: (703) 815-0898

Jack Kramer
DARPA
801 N. Randolph St.
Suite 400
Arlington, VA 22203
Phone: (703) 243-8655
Fax: (703) 528-2627

Chuck Lillie
SAIC
1719 Goodridge Drive
McLean, VA 22102
Phone: (703) 827-4799

Bob McCauley
Martin Marietta
Falcon AFB
Colorado Springs, CO 80912
Phone: (719) 380-2175

Joanne Piper
USAISS DCW
Stop H-4
Ft. Belvoir, VA 22060
Phone: (703) 285-9007

Frank Poslajko
USASDC/CSSD-SP
P. O. Box 1500
Huntsville, AL 35807
Phone: (205) 722-1995

Keith Rathjen
Rockwell
3370 Miraloma, P.O.B. 3105
Anaheim, CA 92803
Phone: (714) 762-1186

Chuck Knostman
MITRE
Falcon AFB
Colorado Springs, CO 80912
Phone: (719) 380-3305

Kerrie Light
Martin Marietta
Falcon AFB
Colorado Springs, CO 80912
Phone: (719) 380-2378

Ron McCain
IBM
3700 Bay Area Blvd.
Houston, TX 77058
Phone: (713) 335-3427

Chuck McNally
Westinghouse
Defense & Electric Center
P. O. Box 746
Baltimore, MD 21203
Phone: (301) 765-0769

LtCol John Morrison
Technology Transfer International
6736 War Eagle Place
Colorado Springs, CO 80912-1634
Phone: (719) 260-0925

Diana Quinn
IBM
Falcon AFB
Colorado Springs, CO 80912
Phone: (719) 380-2191

Robert Saisi
DSD Labs
75 Union Avenue
Sudbury, MA 01776
Phone: (508) 443-9700

Theresa C. Sigoren
NRL
4555 Overlook Ave.,
Washington, DC 20375
Phone: (202) 767-0329

James O. Sutton, III
AFSPACECOM/JA
Peterson AFB
Colorado Springs, CO 80914
Phone: (719) 554-3844

Otis Vaughn
Teledyne Brown Engineering
300 Sparkman Dr., POB 070007
Huntsville, AL 35807
Phone: (205) 726-1611

Alison Wildblood
EVB
4121 Leafy Glade Place
Casselberry, FL 32707
Phone: (407) 699-8110

Bill Zance
SDIO
Pentagon
Washington, DC 20301-7100

Barry Sookman
McCarthy and Tetrault
Code 5570 P.O. Box 48
Toronto Dominion Center
Toronto, Canada M5K1E6
Phone: (416) 362-1812
Fax: (416) 868-1891

Will Tracz
IBM
MD 0210
Owego, NY 13857
Phone: (607) 751-2169

Brian Whitney
Martin Marietta
Falcon AFB CO 80912
Colorado Springs, CO 80912
Phone (719) 380-2379

Steve Wynn
USASDC
P. O. Box 1500
Huntsville, AL 35807
Phone: (205) 895-4680

Joseph Zimmerman
ESD/PKRX
Hanscom AFB, MA 01731
Phone: (617) 271-4716

APPENDIX C

CONTRACTS FOR REUSE LIBRARY

Reproduced here are the draft contracts for the National Test Bed Simulation Reuse Library that were presented for review to the workshop by John Morrison. This includes:

a. A subscription contract.
b. A transfer of the title agreement.
c. A component license.

1. Subscription Contract

The purpose of the subscription contract is to bind the Government (responsible for operating and maintaining the Simulation Reuse Library) and the subscriber (who uses the library to locate and obtain software components in order to build systems) to a set of terms and conditions.

1.1 Products and Services

The NTB Simulation Reuse Library service (the "Service") consists of computing services, software, documentation, and databases which are made available to subscribers by the Government. The service includes:

- Passwords and identification numbers provided by the Government;
- An electronic database, or "library" of software components;
- Software for accessing and retrieving items from the database;
- An accounting system for
  - — Debiting user accounts for services and delivered components;
  - — Crediting user accounts for library additions;
- Authentication numbers used for charging software licensing costs;
- A user support "hotline" service;
- Librarian services;
- Software quality assessment services;
- An electronic bulletin board system;
- An electronic mail system.

1.2 Definitions

a. The terms "products" and "computer programs" shall include all data items associated with a software identification number or numbers.

b. The term "Government" shall refer to the National Test Bed Joint Program Office of the Strategic Defense Initiative Organization.

c. The term "subscriber" refers to the organization or individual who uses the National Test Bed Simulation Reuse Library in accordance with the terms and conditions of this subscription agreement.

d. The term "derivative work" shall apply to software derived from other software.

e. The term "Service" shall refer to the NTB Simulation Reuse Library service.

1.3 Terms and Conditions

a. These terms, together with operating rules published by the Government, will constitute the entire agreement for the Service and will supercede all prior agreements, communications, statements, and documents.

b. Upon notice published over the Service, the Government may modify these terms, the operating rules, or the Service. Such modifications may include, without limitation, price changes, implementation of user priorities, and discontinuance of parts of the Service. Upon at least six months prior notice published over the Service, the Government may terminate the Service.

c. Subscriber's use of the Service will be subject to any credit limits established for the customer's account.

d. The Service is provided on an "as is, as available" basis. Neither the Government nor its contractors make any warranties, express or implied, including without limitation those of merchantability and fitness for a particular purpose, with respect to the Service.

e. The Government and its contractors shall not be liable for any direct, incidental, or consequential damages caused by events beyond the Government's or its contractor's control. Such events shall include, but not be limited to, Acts of God, communications line failures, theft, destruction or unauthorized access to or use of subscriber's records, files or programs, or unauthorized access to records, programs or services contained in the Simulation Reuse Library or National Test Bed.

f. This agreement is effective from the date of acceptance by the Government of a copy signed by the subscriber, and it may be terminated by either party without cause upon

thirty days after receipt of written notice of termination. The Government shall have the right to terminate this agreement without notice and in its sole discretion in the event that the subscriber:

1. Materially breaches the agreement in any way;
2. Has used the Service for unlawful or improper purposes, or in any way which the Government, in its sole discretion, considers to jeopardize the security system of the Service; or
3. Has interfered with or threatened to interfere with any other person's privacy or proprietary right or use of the Service.

It is Government policy that malicious acts by individuals or organizations against any National Test Bed asset shall be treated as a national security matter.

g. The subscriber hereby agrees to indemnify and hold harmless the Government and its contractors from and against any claim, loss, or liability (including reasonable attorney's fees) arising from the violation by any subscriber of any third party rights, including copyright or patent infringement, privacy or proprietary rights. This indemnification obligation shall survive the expiration or termination of this agreement.

1.4 Responsibilities of Subscriber

a. The Subscriber agrees to binding Government arbitration in the case of disputes arising from the sale or use of software components obtained under the Service.

b. The Subscriber shall provide all equipment necessary to access the service from the subscriber's site.

c. The Subscriber shall not reproduce, sell, publish, or in any manner commercially exploit any information obtained through the Service or participate in or allow such reproduction, sale, publication, or exploitation by any person, except as specified under the terms associated with specific software component licenses.

d. The Subscriber agrees to prompt payment of charges billed to Subscriber's account by the Government, consistent with the terms and conditions of this agreement.

e. Subscriber agrees to protect any and all classified material accessible through the Service in the manner described below:

1. Subscriber shall protect classified Simulation Reuse Library products against unauthorized disclosure in a manner consistent with the security classification of the component, including all stated security caveats and restrictions.
2. The acceptable rules and procedures for protection shall be those adopted and/or sanctioned by the U.S. Department of Defense and/or the Defense Investigative Service.

f. The Subscriber agrees to notify the Government immediately if classified or company proprietary software is compromised, suspected to be compromised, or if security and/or confidentiality procedures are otherwise breached.

g. The Subscriber agrees to take prompt action to limit the scope of a security or confidentiality breach if such a breach is detected.

h. Subscriber is responsible for: (1) confidentiality and use of subscriber passwords, identification numbers, and authentication numbers, including all charges to subscriber's account; and (2) any violation of this agreement by anyone using subscriber's account. Subscriber shall immediately notify the Government if password, identification number, or authentication number has been lost or stolen, or if subscriber suspects that someone has unauthorized access to them.

i. Subscriber agrees to comply with Government operational procedures published on the Service.

j. Subscriber agrees to promptly notify the Government of any instances or attempts to circumvent published operational procedures, gain improper entry or make improper use of the Service, or introduce or spread malicious logic via the Service.

1.5 Responsibilities of Government

a. The Government agrees to operate and maintain the Simulation Reuse Library in a manner consistent with the following Government goals. These are:

1. Reduce the cost, improve the quality, and protect the security and integrity of software systems developed under Government contract;
2. Provide a means for the rapid incorporation of new software capabilities and technologies into Department of Defense programs;
3. Provide a service to commercial industry which supports U.S. defense programs, and which is open to any and all qualified organizations. Such qualification shall be based on the existence of a Government contract and appropriate security facility clearances for the applying organization.

b. In order to protect the security, integrity and lawful use of the Service, the Government, or at the Government's direction its contractor, may review any material stored in files or programs to which all subscribers have access, and shall remove any material which

the Government, in its sole discretion, believes may be unlawful or otherwise objectionable.

c. The Government agrees to pay or otherwise reimburse the subscriber for consumer reports, deficiency reports, and re-engineering cost reports which the subscriber provides concerning software components maintained in the Simulation Reuse Library. The formula for such reimbursement shall be as described in operational procedures published on the Service, and is subject to change at the sole discretion of the Government.

1.6 Miscellaneous

a. CHOICE OF FORUM: This agreement is made in, governed by, and the performance thereof shall be construed in accordance with the laws of the state of Colorado. Any action or proceeding arising out of this contract shall be brought in a federal or state court of competent jurisdiction in the state of Colorado, and in no other jurisdiction.

b. SEVERABILITY: If any provision of this agreement shall be held to be invalid, illegal, or unenforceable, the validity, legality and enforceability of the remaining provisions shall in no way be affected or impaired thereby.

c. MERGER OR INTEGRATION: Subscriber represents to have read this agreement and is in agreement with all of the terms and conditions herein. This agreement constitutes the entire agreement between the parties and may only be modified by a written instrument executed by seller and the Government. Neither subscriber nor the Government shall be bound by any oral agreement or representation. This is the complete and exclusive statement of contract between the Government and subscriber and no verbal or other written communication supercedes or modifies this agreement.

2. Transfer of Title

The purpose of the title transfer is to clearly identify the ownership of new software components entering into the software reuse library.

2.1 Program Products & Transfer of Title

The product seller, _____, hereby transfers title, patent and copyright rights of documentation, computer programs, and related data described below to the United States Government under the terms stated below in this agreement, in consideration for $_____.

ITEM	NAME/ DESCRIP.	SW IDENT NUMBER
_____	_____	_____
_____	_____	_____

2.2 Definitions

a. The terms "products" and "computer programs" shall include all data items associated with the above software identification number(s).

b. The terms "buyer" and "Government" shall refer to the National Test Bed Joint Program Office of the Strategic Defense Initiative Organization.

c. The term "seller" refers to _____.

d. The term "derivative work" shall apply to software derived from other software. The process by which the Government determines whether software is a derivative work and the extent to which it is derivative is contained in the NTB Software Center Operating Instructions, incorporated by reference.

2.3 Effective Date

The transfer of title shall be effective on the date of signature of this document.

2.4 Usage and Royalties

a. The Government agrees to pass to the seller royalty fees collected by the Government from third party use of the product while the product is maintained in the Government reuse library.

b. The seller agrees that the Government shall establish a maximum royalty fee based on a formula which includes:

1. Completeness of the product, as determined by the Reuse Librarian;
2. Product quality, as determined by an independent assessment team;
3. Development cost (as can be verified by the developer).

The precise formula is incorporated by reference.

c. The royalty fee for all potential users, or for any selected set or sets of potential users may be reduced from the Government-established maximum fee at the discretion of the seller. Any such modification to the royalty fee is as incorporated by reference.

d. If the product is a derivative work, then the seller agrees to a Government formula for compensating the original developer or developers out of the total royalty fee. The precise formula for compensation is incorporated by reference.

2.5 Access and Usage Restrictions

The Government agrees to limit access to and usage of said product in accordance with the stated desires of the seller, as incorporated by reference.

2.6 Warranty of Title

Seller hereby warrants that it holds unencumbered title and/or copyright rights to all software

sold and delivered pursuant to this agreement, and that it conveys said title for all software sold.

Seller represents that title to certain programs in the software is held by the United States Government, as indicated below:

PROGRAM	SIN	DESCRIP.
_____	____	_____
_____	____	_____

2.7 Indemnification

The seller hereby indemnifies and holds harmless the buyer from any claim by a third party that the software supplied hereunder infringes upon a patent, copyright, trade secret, or similar proprietary right of a third party. This indemnification shall be for all damages, including reasonable attorney's fees.

2.8 Warranties/Limitation of Liability

The seller disclaims all implied warranties of merchantability and fitness for a particular purpose.

2.9 Miscellaneous

a. CHOICE OF FORUM: This agreement is made in, governed by, and the performance thereof shall be construed in accordance with the laws of the state of Colorado. Any action or proceeding arising out of this contract shall be brought in a federal or state court of competent jurisdiction in the state of Colorado, and in no other jurisdiction.

b. SEVERABILITY: If any provision of this agreement shall be held to be invalid, illegal, or unenforceable, the validity, legality and enforceability of the remaining provisions shall in no way be affected or impaired thereby.

c. MERGER OR INTEGRATION: Buyer and Seller represent to have read this agreement and is in agreement with all of the terms and conditions herein. This agreement constitutes the entire agreement between the parties and may only be modified by a written instrument executed by seller and the Government. Neither seller nor the Government shall be bound by any oral agreement or representation. This is the complete and exclusive statement of contract between the Government and seller and no verbal or other written communication supercedes or modifies this agreement.

d. LIBRARY SERVICES: The seller shall assist the Government in classifying the product and establishing the manner in which it may be used within the reuse library.

3. Component License

The purpose of the component license is to define the terms and conditions for using software components from the reuse library.

3.1 Program Products

The United States Government hereby grants to licensee, and licensee hereby accepts a nontransferable, non-exclusive license for documentation, computer programs, and related data described below to the licensee, _____, on project _____ under contract _____ for the stated price, payable in accordance with the terms stated below in this agreement.

ITEM PRICE	SW IDENT NUMBER	NAME/ DESCRIPTION
____	_____	_____
____	_____	_____

3.2 Title

Licensee hereby acknowledges that the United States Government claims ownership, without limitation, of the design and documentation and other information relating thereto.

3.3 Grant of License

The United States Government hereby grants to licensee, and licensee hereby accepts a nontransferable, non-exclusive license to use documentation, computer programs, and related data in accordance with the terms of this agreement. Said use is restricted to a single project and a single contract as stated above, and use on any other project or contract must be authorized by another agreement or amendment to this one.

The license shall commence upon two conditions: (1) acknowledgment by the licensee that licensee intends to accept this license; and (2) the delivery of the licensed product to the licensee.

Licensee shall acknowledge intent to accept this license through one of two acceptable methods: (1) licensee may electronically enter the licensee's unique authentication number to initiate the component retrieval transaction within the Simulation Reuse Library; or (2) licensee may sign a license form obtainable from the Simulation Reuse Library librarian, and submit said form to the librarian.

3.4 Derivative Works

The license granted shall include the nonexclusive right to make derivative works based on the licensed product, and to grant sublicenses under such right. The original developer and the derivative work developer shall share license fees based upon the licensed software in accordance with the percentage contributed to the derivative work. Percentages will be determined by the Government in accordance with a formula incorporated by reference.

3.5 Definitions

a. The terms "products" and "computer programs" shall include all data items associated with the above software identification number.

b. The terms "licensor" and "Government" shall refer to the National Test Bed Joint Program Office and Strategic Defense Initiative Organization.

c. The term "licensee" shall refer to the entity obtaining the software license under this agreement.

3.6 Prices and Payment

Licensee shall pay the Government all fees set forth in Clause 1 of this agreement.

3.7 Nondisclosure

The licensor agrees not to disclose or use the product outside of the agreed project without authorization from the Government.

3.8 Security

a. The licensee agrees to protect the product against unauthorized disclosure in a manner consistent with the security classification of the component, and with any and all stated security caveats and restrictions.

b. The acceptable rules and procedures for protection shall be those adopted and/or sanctioned by the U.S. Department of Defense.

c. The licensee agrees to notify the Government immediately if the software is compromised, suspected to be compromised, or if security procedures are otherwise breached.

d. The licensee agrees to take prompt action to limit the scope of a security breach if such a breach is detected.

3.9 Import/Export

a. The licensee agrees to protect the product in accordance with any and all import/export/and technology transfer caveats and restrictions.

b. The acceptable rules and procedures for protection shall be those adopted and/or sanctioned by the U.S. Department of Defense.

c. The licensee agrees to notify the Government immediately if import/export or technology transfer procedures are breached.

d. The licensee agrees to take prompt action to limit the scope of a procedural breach if such a breach is detected.

3.10 Warranty of Title

The U.S. Government warrants that it holds title and/or copyright rights to all products sold pursuant to this agreement, and that it conveys good title for all software sold. The United States Government agrees to indemnify licensee for any and all claims or lawsuits challenging the Government's right to sell or license the products which are the subject of this agreement.

3.11 Indemnification

The U.S. Government hereby indemnifies and holds harmless licensee from any claim by a third party that the software supplied hereunder:

a. Infringes upon a patent, copyright, trade secret, or similar proprietary right of a Third Party;

b. Is solely responsible for failure of a system designed by the licensee under the Government contract for which this license applies;

c. Fails to perform in accordance with its documentation;

d. Has a designated security classification level which is inconsistent with its true security classification level.

This indemnification shall be for all damages, including reasonable attorney's fees, and is conditioned upon licensee:

a. Giving the Government prompt notice of any such claim or demand;

b. Giving the Government control over all actions, including settlement, of such claim or demand;

c. Giving the Government full cooperation in pursuing any such claim.

3.12 Breach and Termination

In the event of breach of this agreement by licensee, licensee shall:

a. Immediately cease to have any rights to the use of products under this agreement;

b. Return or certify destruction of all copies of products covered by the terms of this agreement;

c. Reimburse the Government for all costs incurred because of breach of this agreement, and be subject to such damages as may be allowed by law, including the costs of enforcing the terms of this agreement, attorneys fees, and punitive damages if applicable;

d. Reimburse the Government for all license fees lost as a result of the unauthorized distribution of said products.

The Government may terminate this agreement and the license in whole or in part if the licensee shall breach or fail to perform any obligation or condition of the license agreement.

If this agreement and license is terminated by the licensee, or by the Government due to a violation by the licensee, no refund of monies paid will be due.

3.13 Warranties

The Government warrants the product shall operate as specified in the documentation. The forgoing warranties are in lieu of all other warranties expressed or implied, including without

limitation, the implied warranties of merchantability and fitness for a particular purpose. Under no circumstances shall the Government or its contractor be responsible for consequential damages, lost profits, or other special damages, even if the Government or its contractor has been advised of likelihood of same. In no event shall the liability of the Government or its contractor arising out of or based upon this agreement regardless of the form in which any legal or equitable action may be brought, including, without limitation, any action in tort or contract, exceed the license fee paid by the licensee to the Government.

3.14 Software Usage

The licensee may modify the product and/or merge it into another computer program for use in accordance with the terms of this license. All modified and incorporated software remains the property of the Government.

3.15 Return of Software

At any time following purchase of license, licensee may apply to the Government for return of the license fee. The Government shall return the fee provided the following conditions are met:

a. The licensee otherwise met the terms and conditions specified in this agreement;

b. The licensee certified in writing the return or destruction of all copies of the product licensed under this agreement;

c. The licensee certified in writing that the product was not used to meet the terms of the contract specified in clause 1 above, and was not used in any other contract, sale or transaction.

3.16 Miscellaneous

a. CHOICE OF FORUM: This agreement is made in, governed by, and the performance thereof shall be construed in accordance with the laws of the state of Colorado. Any action or proceeding arising out of this contract shall be brought in a federal or state court of competent jurisdiction in the state of Colorado, and in no other jurisdiction.

b. SEVERABILITY: If any provision of this agreement shall be held to be invalid, illegal, or unenforceable, the validity, legality and enforceability of the remaining provisions shall in no way be affected or impaired thereby.

c. MERGER OR INTEGRATION: Licensee represents to have read this agreement and is in agreement with all of the terms and conditions herein. This agreement constitutes the entire agreement between the parties and may only be modified by a written instrument executed by licensee and the Government. Neither licensee nor the Government shall be bound by any oral agreement or representation. This is the complete and exclusive statement of contract between the Government and licensee and no verbal or other written communication supercedes or modifies this agreement.

APPENDIX D

NATIONAL TEST BED
SOFTWARE REUSE LIBRARY
CONCEPT OF OPERATIONS

NATIONAL TEST BED
SOFTWARE REUSE LIBRARY
CONCEPT OF OPERATIONS

6 July 1990

John S. Morrison, Lt Col, USAF
Director, System Engineering and Development
National Test Bed Joint Program Office
Mail Stop 82
Falcon AFB, CO 80912-5000
Tel: (719) 380-3267
Fax: (719) 380-3303

SOFTWARE REUSE LIBRARY LEGAL AND CONTRACTUAL CONCEPT OF OPERATIONS

- What assumptions underly contractual approach?
- Who are the players?
- What contracts are involved?
- What are the objects of these contracts?
- What processes invoke contracts?
- What payments and fees are incurred?

SOFTWARE REUSE LIBRARY LEGAL AND CONTRACTUAL ASSUMPTIONS

LIBRARY OPERATION:

- The reuse library is in a secure, Government-owned, contractor operated facility, with daily Government management oversight
- The library is operated as part of the National Test Bed (NTB) Integration Contract
- The contract has provisions and procedures for handling activities with potential for Organizational Conflict of Interest

Rationale:

- A secure facility on a Class-A Government installation provides the best protection for such a sensitive software repository
- The NTB Integration Contract is in place
- OCI activities have been successfully handled under provisions of the contract

SOFTWARE REUSE LIBRARY LEGAL AND CONTRACTUAL ASSUMPTIONS

SOFTWARE COMPONENT DEVELOPMENT:

- Reusable software components will be developed by a variety of organizations, to include: Army, Navy, Air Force, and Department of Energy
- Reusable software components will be shared by these same agencies to build systems

Rationale:

- The SDI program requires integration of software from these multiple sources
- Transfer of technology across the program also demands such sharing and exchange of software

SOFTWARE REUSE LIBRARY
LEGAL AND CONTRACTUAL ASSUMPTIONS

OWNERSHIP:

- Software developed by independent commercial vendors would continue to be owned by the vendors
- The Government must license such software in order to make available in a reuse library
- The license should provide the Government with the ability to sub-license to reuse library subscribers

Rationale:

- Procurement of copyrights and patent rights on such software would be prohibitively expensive
- Licensing is the traditional approach
- Sub-licensing allows the reuse library to deal with subscribers in a uniform way (Licensing is the instrument by which Government-owned and non-Government owned software is made available through the library)

SOFTWARE REUSE LIBRARY
LEGAL AND CONTRACTUAL ASSUMPTIONS

OWNERSHIP:

- Software developed under Government contract is owned by the Government
- Title, to include patent and copyright rights, should be explicitly transferred to the Government

Rationale:

- Traditional Government view
- Simplifies potential liability problems in a reuse context
- Clarifies data [illegible]s issues

SOFTWARE REUSE LIBRARY
LEGAL AND CONTRACTUAL ASSUMPTIONS

INCENTIVIZATION FOR REUSE:

- Both development and use of reusable software components will be incentivized by the SDI Program
- Government contracts for system developers will include:
  - Requirement to subscribe to the reuse library
  - Requirement to train people on use of reuse library
  - Requirement to train people on system design from reusable components
  - Incentives to produce software cheaply
- Consumers of components will pay royalties to the Government
- These royalty fees will "pass through" to component developers
- Consumer reporting will also be incentivized

Rationale:

- Remove barriers to reuse, make it work

SOFTWARE REUSE LIBRARY
LEGAL AND CONTRACTUAL ASSUMPTIONS

[illegible]ERIVATIVE WORKS:

Encouraged for Government-owned software

A derivative work is subject to a "pass-through" of royalties to previous developers, based on Government determination of "value added."

Rules for royalty pass through are included in the subscription contract

[illegible]ationale:

Encourage evolution of library components in the direction of increasing quality

Make room for innovation and competition

Avoid problems associated with single-source development

NEW PROCEDURES MUST ACCOMPANY SOFTWARE REUSE (2)

IV&V Specialists
- Define how to test and assess a system where software is a hybrid of new and reused parts

Library Staff
- Acquire and maintain parts
- Operate library

Component Assayer
- Determines the Part Quality Index of components entering into the library

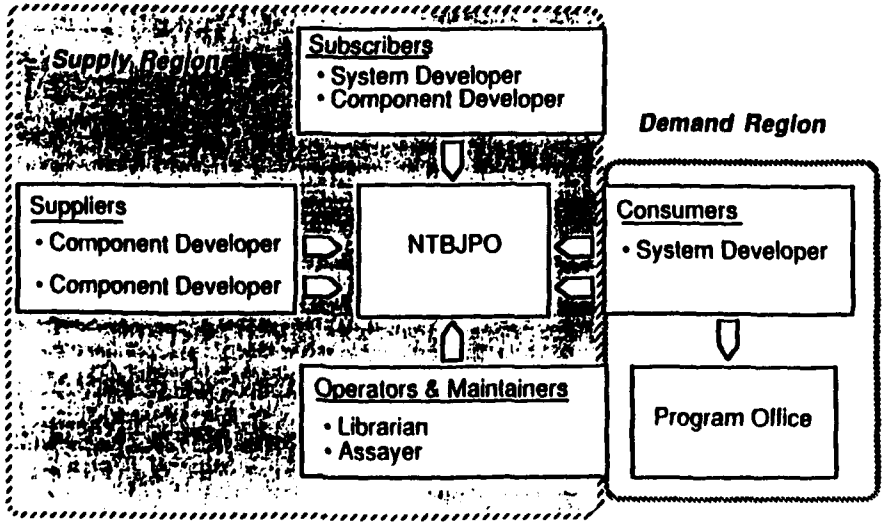
NEW PROCEDURES MUST ACCOMPANY SOFTWARE REUSE (1)

Acquisition Specialists
- Define contractual wording for reuse incentives
- Write contractual language for liability avoidance
- Develop evaluation factors

Contract Evaluators
- Understand reusable part evaluation factors
- Recognize increased costs to build reusable parts
- Recognize decreased costs to build from parts
- Use Part Quality Indices to assess risk

Program Managers
- Monitor software reuse metrics on their projects
- Assess and mitigate software risks

SOFTWARE REUSE CONTRACTING --Overview--



NEW PROCEDURES MUST ACCOMPANY SOFTWARE REUSE (3)

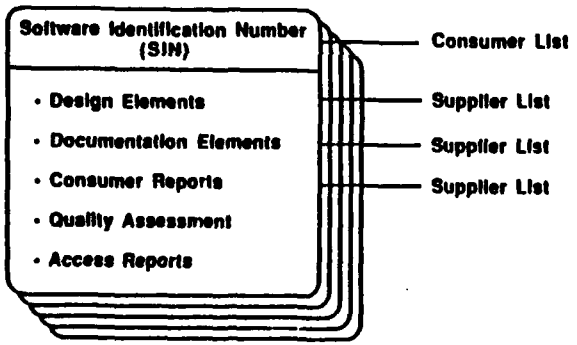
System Engineers
- Balance between top-down design and bottom-up capability of reusable parts inventory

System Software Architects
- Program primarily in parts, not in high order languages

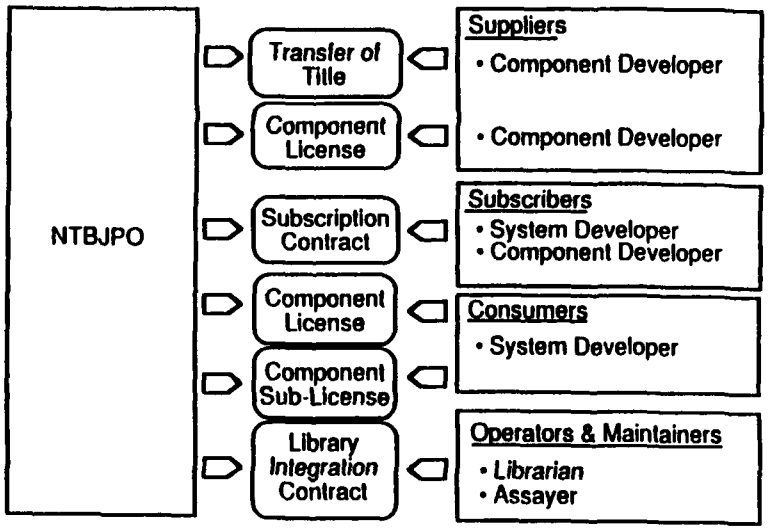
Component Developers
- Construct reusable software components

Company Management
- Measures profits in terms of cost savings in system development

SOFTWARE REUSE LIBRARY IS A DATA BASE OF COMPONENTS



SOFTWARE REUSE CONTRACTING --Supply Side View--



DOCUMENTATION ELEMENTS

SUPPLIER: Component Developer

ELEMENTS:

- 2167A
  - Software Development Files
  - Version Description Document (Component)
  - Other (if available)
- Development Cost (in manhours)
- Status of any unclosed Deficiency Reports

DESIGN ELEMENTS

SUPPLIER: Component Developer

ELEMENTS:

- Source Code, annotated to show compiler used and hardware compatibility
- PDL/Formal language description
- Graphic design representation(s)
- Data structures
- Simulation

CONSUMER REPORTS

SUPPLIER: Component Consumers

REPORTS:

- Blue Book (Cost of adapting a component for other projects)
- Deficiency Reports
- Consumer Assessments

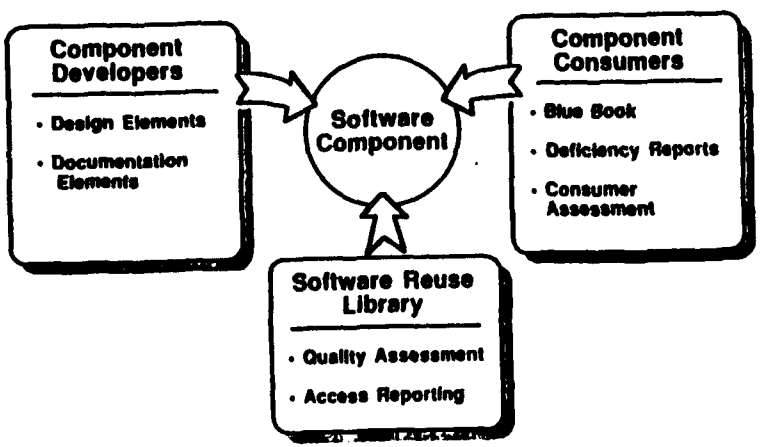
BLUEBOOK

PURPOSE: Provide consumers with a basis for estimating the cost to re-engineer the component.

EXAMPLE:

Component Software Identification Number (SIN): nnn				
Descriptive Statistics: • Mean cost to re-engineer: nnn Man-hours • Standard deviation: + / - nn Man-hours • Sample Size: xxx				
Basis of Estimate:				
Re-engineering Cost	Date	Project	POC	Remarks
xxx man-hours	10/12/91	Sys Sim	Schultz	
yyy man-hours	7/6/91	PCC	Arnold	
zzz man-hours	5/5/91	C2 Gaming	Arnold	

COMPONENT CONTRIBUTIONS



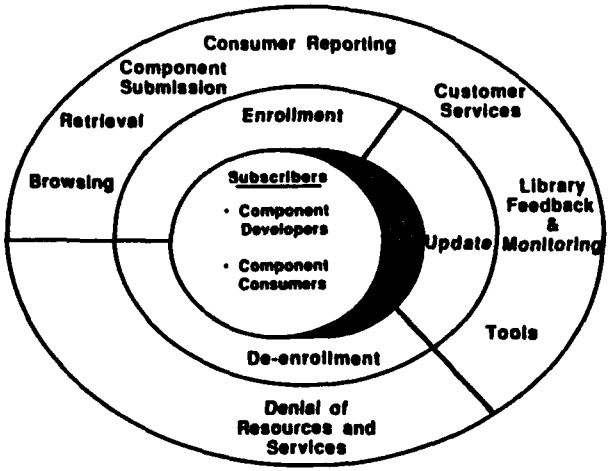
OTHER REPORTS

SUPPLIER: Software Reuse Library

REPORTS:

- Quality Assessment Report
- Access Reports

SUBSCRIBER-CENTERED SOFTWARE REUSE LIBRARY PROCESSES



REUSE LIBRARY PROCESSES AND TRANSACTIONS

PROCESS: Set of related reuse library activities, such as

- Library Enrollment
- Updates to Authorized User List
- Library De-enrollment
- Component Submission
- Component Removal
- Library Browsing
- Component Retrieval
- Consumer Reporting
- Library Feedback and Monitoring

TRANSACTION: Activity occurring as part of a library process. Transactions may result in binding contracts, and may incur costs, fees, and royalty payments.

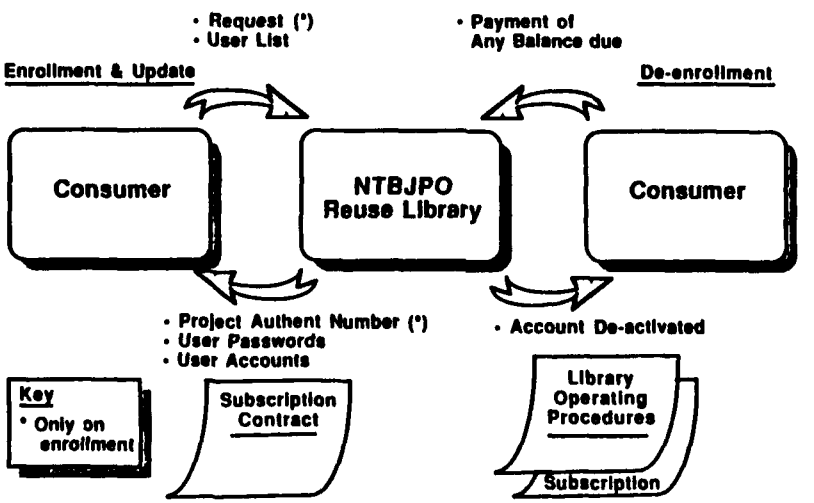
ENROLLMENT PROCESS

PROCESS DESCRIPTION: This process establishes a user account for the software reuse library. The enrolling organization pays any equipment or communication costs associated with establishing the library link.

TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [M] Potential subscriber submits a written request for a library account, and a signed subscription contract
2. [M] Librarian verifies access requirements through SDIO sponsor, site security accreditation through NTBJPO security.
3. [M] Librarian assigns project authentication number used to electronically "sign" component licenses and accomplish other contractually binding electronic transactions.
4. [M] Librarian assigns personal identification numbers and temporary passwords to list of potential users submitted by the enrolling organization
5. [E] Librarian activates user accounts and project authentication number

ENROLLMENT, DE-ENROLLMENT AND UPDATE



USER UPDATE PROCESS

PROCESS DESCRIPTION: *This process updates the list of authorized users* associated with a subscribing organization.

TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [E] Member of a subscribing organization submits a revised list of *authorized users, their security clearances, and any access/usage* restrictions they might have. List is authenticated using project authentication number.

2. [M] Librarian notifies NTBJPO security and operations.

3. [E] Librarian assigns personal identification numbers and temporary passwords to list of new users, and de-activates numbers and passwords for dropped users.

4. [E] Librarian activates new user accounts and de-activates dropped accounts.

DE-ENROLLMENT PROCESS

PROCESS DESCRIPTION: This process closes a user account for the software reuse library. The de-enrolling organization pays any equipment or *communication costs associated with de-activating the library link.*

TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [M] Librarian notifies SDIO sponsor, NTBJPO security. and operations

2. [E] *Librarian de-activates project authentication number used to* electronically "sign" component licenses and accomplish other contractually binding electronic transactions.

3. [E] Librarian de-activates personal identification numbers and passwords associated with users in de-enrolling organization

4. [E] Librarian de-activates user accounts and terminates any flow of royalties to the de-enrolling organization

5. [E] Librarian places a notice of de-enrollment on the library electronic bulletin board system (BBS), and identifies which components the de-enrolled organization had submitted

6. [E] Librarian notifies consumers of affected components by message

ENROLLMENT PROCESS LEGAL IMPLICATIONS

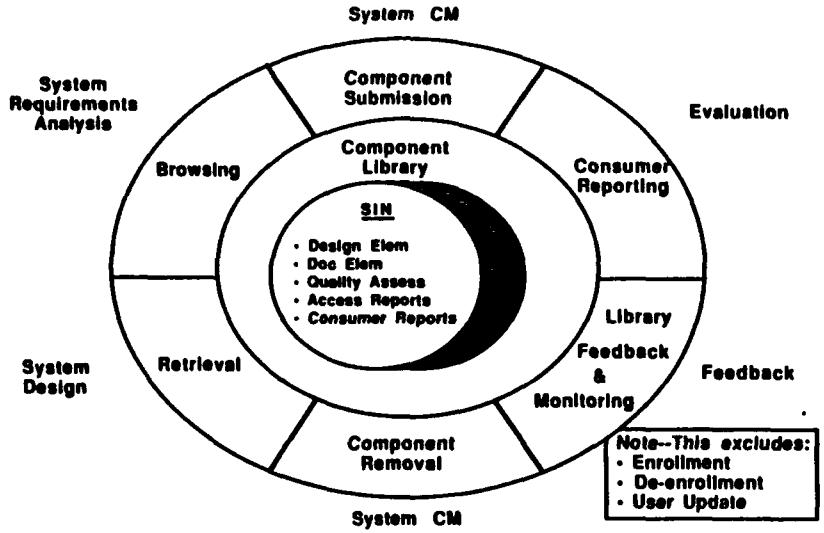
Subscription Contract:

- Binds Government and subscriber to terms, conditions and responsibilities
- Establishes library usage fees and charges
- Expands user access to reuse library resources

USER UPDATE PROCESS LEGAL IMPLICATIONS

- Modifies who has valid access to library resources

LIBRARY-CENTERED SOFTWARE REUSE LIBRARY PROCESSES



DE-ENROLLMENT PROCESS LEGAL IMPLICATIONS

- Modifies who has valid access to library resources
- Any component royalty flow-through to the de-enrolled organization reverts to the Government if the Government has title to a component
- Sublicenses (in the case where the component is owned by a vendor):
  - *Remain valid unless the vendor company dissolves*
  - If the vendor company dissolves, the Government retains the right to sublicense the component, with royalties reverting to the Government
- Component Licenses (where the Government licenses components which it owns) are not affected

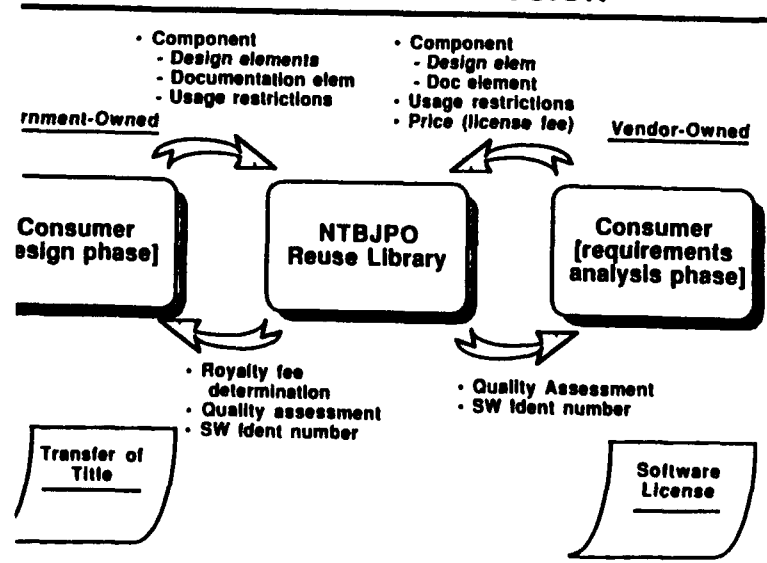
COMPONENT SUBMISSION PROCESS

PROCESS DESCRIPTION: This process introduces a new or modified software component into the reuse library.

TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [E] Component developer submits design and documentation elements.
   - Submission includes any access limitations or restrictions
2. [E, M] Librarian performs component check-in.
   - Assigns Software Identification Number (SIN)
   - Inventories items/ checks for completeness
   - *Deposits crypto-sealed copy into "data vault"*
   - Deposits working copy into Software Assessment Library
3. [E, M] *Component Assayer performs software assessment*
   - Evaluates deliverables
   - Creates Component Assessment Report
   - Determines Part Quality Index (PQI)
4. [E, M] Reuse Librarian puts the component "on-the-shelf"
   - Design, documentation, and assessment report put in library
     - Access limitations established
     - Automatic access reporting system activated
     - Consumer reporting system activated
   - Royalty fees established based on a formula which includes completeness, quality, development cost and (possibly) source
   - Librarian and developer establish component licensing wording
   - Transfer of title or component license inked

COMPONENT SUBMISSION



BROWSING PROCESS

PROCESS DESCRIPTION: This process allows a consumer, searching for a component, to view documentation elements, quality assessment report, access report, and consumer reports. These transactions do not by themselves generate a flow of royalty fees, but there may be a flat fee associated with use of library resources.

TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [E] Library subscriber performs component search and browse
   - Library search tools and decision aids assist the consumer
   - Search is constrained by security, other access limitations
   - Consumer is cautioned that use of any documentation element requires payment of a royalty fee [Selection Option]
   - *Component Licensing terms and conditions are displayed with* data on each component
   - Default assumption is that consumer does not desire to use any of the browsed documentation

BROWSING PROCESS LEGAL IMPLICATIONS

- Consumer may incur flat fee for use of library resources
- Consumer may incur penalty if browsed data is used without notification / licensing
- Government must caution browsing consumer about data rights and data usage restrictions
- Government must make component license terms, conditions, and costs visible to consumer

RETRIEVAL PROCESS

PROCESS DESCRIPTION: This process allows a consumer to retrieve a particular component or set of components, incurring royalty fees.

TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [E] Consumer views and/or retrieves design elements
   - Consumer must enter project authentication number, agreeing to component licensing terms and conditions
   - Consumer is charged the full royalty fee
   - During retrieval:
     - Copies of design design and documentation elements are made from the data vault copy
     - Copies are shipped electronically to consumer terminal

2. [E] Consumer may declare intent not to use any portion of the *component, and forego royalty charges* [Option]
   - Such a declaration must be authenticated using consumer personal identification number and project authentication number

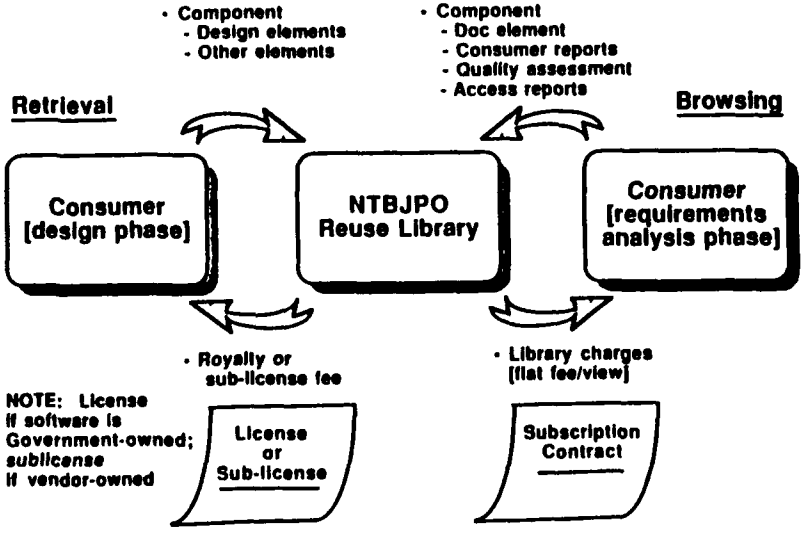
RETRIEVAL PROCESS LEGAL IMPLICATIONS

- Creates binding component license or sub-license
- Incurs royalty payment
- Safety valve feature: Consumer may formally reject use of the retrieved component and thereby revoke the license

COMPONENT SUBMISSION PROCESS LEGAL IMPLICATIONS

- Transfer of Title (Component source is Government contract)
  - Transfers copyrights and patent rights to the Government
- Component License (Component source is commercial vendor)
  - Licenses the component to the reuse library
  - Permits sub-licensing to reuse library subscribers
- Government Quality Assessment
  - Determines royalty payment for components developed under Government contract
  - Is linked to the component and may therefore influence consumer demand for the component
- Determines terms and conditions for component licenses and sub-licenses

COMPONENT RETRIEVAL & BROWSING



COMPONENT REMOVAL PROCESS

PROCESS DESCRIPTION: This process removes a software component from the reuse library.

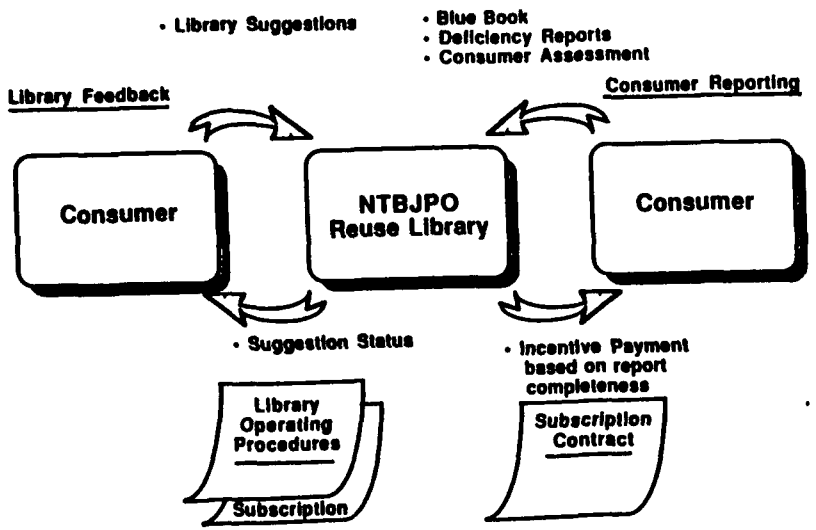
TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [E] Reuse Librarian removes components if:
   - Malicious logic has been detected within any of the design elements
   - Serious faults have been detected in design elements
   - Serious inaccuracies have been detected in documentation elements
   - The component has not been accessed within a prescribed time
   - The developer requests it and the request is approved by SDIO
   - The SDIO requests it
2. [E] Reuse Librarian annotates data vault copy with reason for removal.
3. [E] Reuse Librarian places notice of removal on the library electronic bulletin board system (BBS)
4. [E] Reuse Librarian sends messages to consumers who have paid royalty fees for use of the component.
   - Notifies consumers of component removal
   - States rationale for removal
   - States whether or not royalty fees will be refunded (NTBJPO Decision)

COMPONENT REMOVAL PROCESS LEGAL IMPLICATIONS

- May result in suspension or revokation of component licenses
- May result in refund of royalty fees
- Government is obliged to notify all affected parties

CONSUMER REPORTING AND LIBRARY FEEDBACK



CONSUMER REPORTING PROCESS

PROCESS DESCRIPTION: This process allows a consumer to supply feedback on components. *These reports may be freely submitted by anyone at any time,* using electronic reporting procedures and report templates established within *the reuse library.* Following submission, an automated validation step occurs, which checks to see whether the report comes from a member of an organization which had retrieved a component. If validated, these transactions generate a reverse flow of royalty fees (partial payback) to incentivize comments. The amount depends on the completeness of the report.

TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [E] Consumer submits "Blue Book" reports (Cost of re-engineering)
2. [E] Consumer submits Deficiency reports (problems with the component)
3. [E] Consumer submits Consumer Assessment reports (evaluations)

CONSUMER REPORTING PROCESS LEGAL IMPLICATIONS

- Payments flow to consumer to incentivize reporting
- Deficiency reports may decrease assessed quality of a component
- Royalty fees are based on a formula tied to the assessed quality of a component

LIBRARY FEEDBACK & MONITORING PROCESS

PROCESS DESCRIPTION: This process supplies feedback on the library. Some reports are generated automatically by the library system; others by subscribers. These reports are not incentivized.

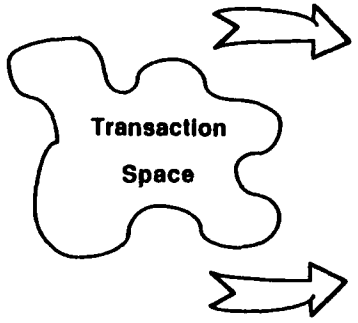
TRANSACTIONS: [E] = Electronic, [M] = Manual, [E, M] = Mix

1. [E] Consumer submits reports containing problems or recommendations on the library, to include:
   - Tools
   - Procedures
   - Reporting systems
   - Management
2. [E] System tracks and logs use of the library, to include:
   - Component access [records, statistics are retrievable by consumers during browsing]
     - Who/what organization
     - When
     - For what purpose
   - Library Use [records, statistics are retrievable only by library personnel]
     - Who/what organization
     - When
     - For what purpose

LIBRARY FEEDBACK & MONITORING PROCESS LEGAL IMPLICATIONS

- The Government must take prompt action to correct any library deficiencies which impact on security, data integrity, or proprietary exclusions

LIBRARY MONITORING



Component Monitoring

- Who accesses?
- What project?
- When?
- What purpose?
- What charges?
- Summary statistics

Library Monitoring

- Who accesses?
- What project?
- When?
- What purpose?
- What components accessed?
- What charges?
  - Browsing
  - Retrieval
  - Consumer Reporting
  - Library feedback

Component candidates for deletion

SOFTWARE REUSE LIBRARY NEEDED CAPABILITIES

- Software Identification Number (SIN) numbering and classification scheme
- Procedures and tools for Quality Assessment of software components
- Part Quality Index scheme
- Formula for royalty fee determination
- Formula for royalty fee "pass through" on derivative works
- Tools and algorithms for objectively determining whether or not a component was "derived" from another component *[measures of conceptual distance]*
- "Drop logic" for unused components
- Library and component monitoring tools
- Detailed operating procedures

REUSE LIBRARY CASH FLOW SUMMARY

ACTIVITY	$	FROM	TO	REMARKS
Enrollment	+	Subscriber	Government	Initiation of service
De-enrollment	?	Consumer	Government	Royalty for component may revert to the Government
Component Submission				
• Transfer of Title	-	Government	Component Developer	Nominal fee for copyright & patent rights
• Component License	-	Government	Component Developer	Government right to sub-license
Component Removal	?	Component Developer	Government	Royalty refunds may go to consumers
Browsing	+	Subscriber	Government	Use of library resources
Component Retrival	+/-	Subscriber	Government	Pass through of royalties to component developers
Consumer Reporting	-/+	Government	Consumer	Incentive for reporting (charge component developers)

Distribution List for IDA Document D-1004

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Lt Col James Sweeder Chief, Computer Resources Engineering Division SDIO/ENA The Pentagon, Room 1E149 Washington. DC 20301-7100	3
Other	
Mr. John Ablard USN/NRL Contracts Division Code 3200 4555 Overlook S.W. Washington, D.C. 20375	1
Mr. Jesse Abzub IBM 6600 Rockledge Drive Bethesda, MD 20817	1
Mr. Dennis Ahern Westinghouse Electronic Systems Group Aerospace Software Engineering, MS-432 P.O. Box 746 Baltimore, MD 21203-0746	1
Mr. Axel H. Ahlberg GE Aerospace National Test Facility, MS-N9020 Falcon AFB, CO 80912-5000	1
Mr. Ted Allen Teledyne Brown Engineering Cummings Research Park 300 Sparkman Drive MS-174 Huntsville, AL 35807-7007	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Don Alley Software Productivity Consortium SPC Building 2214 Rock Hill Road Herndon, VA 22070	1
Dr. Dan Alpert, Director Program in Science, Technology & Society University of Illinois Room 201 912-1/2 West Illinois Street Urbana, Illinois, 61801	1
Capt. Emily Andrew NTBJPO Falcon AFB, CO 80912	1
Mr. Murray B. Baxter JAP-IP 5611 Columbia Pike Falls Church, VA 22041	1
Mr. Rich Bertel Martin Marietta/ISG 4795 Meadow Wood Chantilly, VA 22021	1
Mr. Eric Besser Westinghouse Aerospace Software Eng. Defense & Electric Center P.O. Box 746, MS-432 Baltimore, MD 21203	1
Mr. Vincent Bia NTI P.O. Box 100 Leupp, AZ 86035	1
Mr. Robert Bowes DSD Labs 75 Union Avenue Sudbury, Mass 01776	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Richard Bremner G.E. Aerospace Suite 800 5933 W. Century Blvd Los Angeles, CA 90045	1
Ms. Gina Burt WRDC/AL-A Wright Patterson AFB Dayton, OH 45433 513-255-9614	1
Mr. Richard Cheston United States General Accounting Office (GAO) Room 4073 441 G. Street, NW Washington, DC 20548	1
Dr. Sholom Cohen Software Engineering Institute Carnegie Mellon University 4500 Fifth Avenue Pittsburgh, PA 15213-2691	1
Mr. Gary Cox IBM Falcon AFB Colorado Springs, CO	1
Mr. Jay Crawford Naval Weapons Center Code 31C China Lake, CA 93555	1
Ms. Bonnie Dancy EVB 5303 Spectrum Drive Frederick, MD 21701	1
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Mr. Dennis S. Deutsch Gallo Geffner and Fenster 235 Main Street Hackensack, NJ 07601	1

NAME AND ADDRESS	NUMBER OF COPIES
Capt Tony Dominice Air Force ATF Liaison Army Aviation Systems Command AMCPEO-LHX-TM 4300 Goodfellow Blvd St. Louis, MO 63120-1798	1
Mr. Tom Durek TRW MS FVA6/2050C 2701 Prosperity Avenue Fairfax, VA 22031	1
Mr. Dan Edwards Boeing Military Airplanes P.O. Box 7730, K80-13 Wichita, KS 67277-7730	1
Mr. William Farrell DSD Labs 75 Union Avenue Sudbury, MA 01776	1
Ms. Marielena Finn SAIC 1710 Goodridge Drive MS T2-8-2 McLean, VA 22102	1
Cpt Tim Fisk HQ SDIC Division/CN1 P.O. Box 92960 Los Angeles, CA 90009-2960	1
Ms. Diane Foucher Naval Weapons Center Code 251 China Lake, CA 93555	1
Dr. Joe Fox Software Architecture and Engineering, Inc. 1600 Wilson Blvd., Suite 500 Arlington, VA 22209	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Bill Frakes Software Productivity Consortium SPC Building 2214 Rock Hill Road Herndon, VA 22070	1
Mr. John Gaffney Software Productivity Consortium SPC Bldg 2214 Rock Hill Road Herndon, VA 22070	1
Mr. Harold Gann UIE 1500 Perimeter Parkway Suite 123 Huntsville, AL 35806	1
Mr. Russ Geoffrey NTBJPO Falcon AFB Colorado Springs, CO 80912	1
Mr. Terry Gill CMRI/CEC Martin Marietta NTB MS 8700 Falcon AFB, CO 80912-5000	1
Mr. Todd Goodermuth GE Aerospace P.O. Box 1000 Blue Bell, PA 19422	1
Mr. Jack Gorman NASA Johnson Space Center NASA Road 1 Houston, TX 77058	1
Dr. Ron Green Deputy Commander USA/SDC ATTN SFAE-SD-GST-D 106 Wynn Drive Huntsville, AL 35807	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Joe Greene Real-time Solutions, Inc. P.O. Box 1466 Annapolis, MD 21404-1466	1
Mr. Harley Ham NAC-825 Naval Avionics Center 6000 East 21st Street Indianapolis, IN 46219-2189	1
Mr. Robert Hamiliton USASDC P.O. Box 1500 Huntsville, AL 35807	1
Mr. Ron Halbgewacks POET, Suite 300 1225 Jefferson Davis Hwy Arlington, VA 22202	1
MAJ Gary Hausken Intellectual Property Law Division Office of the Judge Advocate General ATTN USA/JALS-IP 5611 Columbia Pike Falls Church, VA 22041-5013	1
Mr. Dale Henderson Los Alamos National Laboratory P.O. Box 1663 A-DO Mail Stop F606 Los Alamos, NM 87545	1
Mr. Jim Hill Martin Marietta Falson AFB, CO 80912	1
Mr. Bill Hodges Boeing Aerospace and Electronics P.O. Box 3099 Mail Stop 9Y38 Seattle, WA 98124-2499	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Robert Holibaugh Software Engineering Institute Carnegie-Mellon Pittsburgh, PA 15213	1
Mr. Danny Holtzman Vanguard 10306 Eaton Place, Suite 450 Fairfax, VA 22030	1
Prof. James Hooper UAH Computer Science Building Room 111 Huntsville, AL 35899	1
Mr. Richard Iliff SDIO ENA Room 1E149, The Pentagon Washington, D.C. 20301 703-693-1826	1
Mr. William Jarosz 14800 Aviation Avenue L.A. Air Force Base Hawthorne, CA 90009 213-363-8699	1
Mr. Raj Kant Honeywell Systems and Research Center 3660 Technology Drive Minneapolis, MN 55418	1
Ms. Kerry Kent MITRE Falcon AFB, CO 80912-5000	1
Mr. Jack Kleinert SAIC 1710 Goodridge Drive P.O. Box 1303 McLean, VA 22102	1
Mr. Hugh Klipp USAISSDCW Stop H-4 Ft. Belvoir, VA 22060	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Chuck Knostman MITRE Falcon AFB Colorado Springs, CO 80912	1
Ms. Virginia P. Kobler Chief, Technology Branch Battle Management Division US Army Strategic Defense Command P.O. Box 1500 Huntsville, AL 35807	1
Mr. Ed Koss Advanced Technologies Inc. 14 Cove Rd Melbourne Beach, FL 32951	1
Dr. John F. Kramer STARS Technology Center 3701 North Fairfax Drive Arlington, VA 22204-1714	1
Dr. Larry Latour Department of Computer Science University of Maine Neville Hall Orono, ME 04469-0122	1
Mr. Doug Lea Visiting Researcher Syracuse University CASE Center 2-173 Science & Technology Center Syracuse, NY 13244	1
Mr. Ronald Liedel SDC ATTN CSSD-SA-EE P.O. Box 1500 Huntsville, AL 35807 205-955-3972	1
Ms. Kerrie Light Martin Marietta Falcon AFB Colorado Springs, CO 80912	1

NAME AND ADDRESS	NUMBER OF COPIES
Dr. Charles Lillie SAIC 1710 Goodridge Drive P.O. Box 1303 McLean, VA 22102	1
Mr. Gary Mayes SDC/GSTS PO U.S. Army ATTN SFAE-SD-TST 106 Wynn Drive Huntsville, AL 35807-3801	1
Mr. Ron McCain IBM Corporation 3700 Bay Area Blvd Houston, TX 77058-1199	1
Mr. Bob McCauley Martin Marietta Falcon AFB Colorado Springs, CO 80912	1
Mr. Charles McNally Westinghouse Electronics Systems Group Contracts P.O. Box 746, MS-1112 Baltimore, MD 21203-0746	1
Mr. Stan McQueen MITRE 1259 Lake Plaza Drive Colorado Springs, CO 80906 719-380-3325	1
Mr. Mike Mitrione Dynamics Research Corp. 1755 Jefferson Davis Hwy, Suite 802 Arlington, VA 22202	1
Mr. John Morrison Technology Transfer International 6736 War Eagle Place Colorado Springs, CO 80912-1634	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Robert Nelson Information Systems Management NASA Space Station Program Office 10701 Parkridge Blvd. Reston, VA 22091	1
Ms. Terri Payton UNISYS 12010 Sunrise Valley Reston, VA 22091	1
Ms. Joanne Piper CIVPERS Stop H-3 Ft. Belvoir, VA 22060-5456	1
Mr. Frank Poslajko SDC MS CSSD-SP 206 Wynn Drive Huntsville, AL 35807	1
Dr. Ruben Prieto-Diaz SPC 2214 Rock Hill Road Clarendon, VA 22070	1
Ms. Diana Quinn IBM Falcon AFB Colorado Springs, CO 80912	1
Mr. Don Reifer Reifer Consultants, Inc. 2550 Hawthorne Blvd. Suite 208 Torrance, CA 90505	1
Mr. Ernie Roberts McDonnell Douglas Corp. P.O. Box 516 D309/B66/L2N/Room 210 St. Louis, MO 63166	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Jim Robinette DCA/Z4S/SMBA 3701 N. Fairfax Drive Arlington, VA 22203	1
Mr. Jack Rothrock SofTech, Inc. 2000 North Beauregard Street Alexandria, VA 22311	1
Mr. Rich Saik Teledyne Brown Engineering Cummings Research Park 300 Sparkman Drive MS-174 Huntsville, AL 35807-7007	1
Mr. Robert Saisi DSD Labs 75 Union Avenue Sudbury, MASS 01776	1
Ms. Pamela Samuelson School of Law University of Pittsburgh 3900 Forbes Avenue Pittsburgh, PA 15260	1
Dr. John Salasin GTE Govt. Systems Corp. 1700 Research Blvd Rockville, MD 20850	1
Mr. Robert Schafrit TA&T, Inc. 133 Defense Hwy Suite 212 Annapolis, MD 21401 301-261-8373	1
SDIO Technical Information Center DRC 1755 Jeff Davis Highway Suite 802 Crystal Square 5 Arlington, VA 22202	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Robert Seltzer Meta Software Corporation Cambridge, MA 02140	1
Ms. Theresa C. Sigoren NRL 4555 Overlook Avenue Washington, DC 20375	1
Dr. John Solomond AJPO OUSDRE/R&AT The Pentagon, Room 3E114 Washington, D.C. 20301-3081	1
Mr. Barry Sookman McCarthy and Tetrault P.O. Box 48 Toronto Dominion Center Toronto, Canada M5K1E6	1
Mr. Terry Starr GE P.O. Box 1000 Blue Bell, PA 19422	1
Ms. Geree Streun General Dynamics/Ft. Worth Div. P.O. Box 748, MZ 4050 Ft. Worth, TX 76101	1
Mr. James O. Sutton, III AFSPACECOM/JA Peterson AFB Colorado Springs, CO 80914	1
Dr. Ted Tenny General Dynamics/Ft. Worth Div. P.O. Box 748 MZ 1761 Ft. Worth, TX 76101	1
Mr. Will Tracz IBM Corporation Mail Drop 0210 Route 17C Owego, NY 13827-1298	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Otis Vaughn TBE Cummings Research Park 300 Sparkman Drive MS 198 Huntsville, AL 35807-7007	1
Dr. Anthony Wasserman, President Interactive Development Environments, Inc. 150 Fourth Street, Suite 210 San Franciso, CA 94103	1
Prof. Bruce W. Weide Dept. of Computer and Information Science The Ohio State University 2036 Neil Ave. Mall Columbus, OH 43210-1277	1
Mr. Nelson Weiderman Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213-3890	1
Mr. Brian Whitney Martin Mariett Falcon AFB Colorado Springs, CO 80912	1
Mr. Paul Wilbur Teledyne Brown Engineering Cummings Research Park 300 Sparkman Drive MS-174 Huntsville, AL 35807-7007	1
Ms. Alison Wildblood EVB 4121 Leafy Glade Place Casselberry, FL 32707	1
Mr. Joan Winston Office of Technology Assessment (JTET) Project Director Room 309 U.S. Congress 600 Pennsylvania S.E. Washington, DC 20510-8025	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Neal Winters TBE Cummings Research Park 300 Sparkman Drive MS 174 Huntsville, AL 35807-7007	1
Mr. Steve Wynn USASDC P.O. Box 1500 Huntsville, AL 35807	1
Mr. Bill Zance SDIO The Pentagon Washington, DC 20301-7100	1
Mr. Joel Zimmerman ESD/Contracts Division (PK) Hanscom AFB, MA 01731-5000	1
IDA	
General Larry D. Welch, HQ	1
Mr. Philip L. Major, HQ	1
Dr. Robert E. Roberts, HQ	1
Ms. Ruth L. Greenstein, HQ	1
Dr. Richard Ivanetich, CSED	1
Ms. Anne Douville, CSED	1
Mr. Terry Mayfield, CSED	1
Dr. Richard L. Wexelblat, CSED	1
Dr. Dennis Fife, CSED	2
Mr. James Baldo, CSED	5
Dr. Craig A. Will, CSED	3
Dr. David Carney, CSED	1
Mr. Michael Nash, CSED	1
Dr. Reginald Meeson, CSED	1
Dr. Robert Turner, SED	1
Ms. Sylvia Reynolds, CSED	2
IDA Control & Distribution Vault	3