NATO
STANDARD FOR
MANAGEMENT OF A
REUSABLE SOFTWARE
COMPONENT
LIBRARY

Volume 2
(of 3 Documents)

- Development of Reusable Software Components
- Management of a Reusable Software Component Library
- Software Reuse Procedures

Issued and Maintained by:

NATO COMMUNICATIONS AND INFORMATION SYSTEMS AGENCY
(Tel. Brussels (2).728.8490)

Table of Contents

Table of Contents (Continued)

APPENDICES

Table of Contents (Continued)

List of Tables

PART I

INTRODUCTION AND BACKGROUND

Section 1

Introduction

The *Standard for Management of a Reusable Software Component (RSC) Library* provides guidance in the establishment and operation of a NATO-controlled resource to support the reuse of software life-cycle products in NATO contracts.

The following subsections describe the purpose of this manual and explain how to use it effectively.

1.1 Purpose and Scope

In order to achieve actual benefits from software reuse, reusable assets and associated resources must be managed and controlled; this manual provides guidance to NATO, host-nation, and contractor personnel on the establishment and operation of a software reuse support organization, hereinafter called a library.

The standard is intended to reflect NATO's current approach to the issues of software reuse and describe the activities by which the personnel of the library address those issues to achieve specific ends.

The guidance is prescriptive to the degree that any management-oriented manual can be; where there are alternatives, the standard presents criteria for deciding among them. The approach is largely independent of specific methods and tools and easily adaptable to project software engineering practice. Issues pertaining to trusted software are beyond the scope of the standard, but may be addressed in the future.

The primary audience is the staff of a library operated by or on behalf of NATO or a host-nation. NATO and host-nation program offices will use the guidance in establishing IFB requirements and in guiding contractors.

This is one of a set of three software reuse standards developed by NACISA. This manual specifically addresses the management of a library of reusable components. The other two documents are standards for the creation of reusable software components and for the reuse of existing software in ongoing projects.

1.2 Guide to Using This Manual

This standard provides specific guidance, organized by library activity, as a basis for establishing individual project practice.

The *Standard for Management of a Reusable Software Component (RSC) Library i*s organized in two parts. Part I provides an introduction to the manual and a brief discussion of general

concepts of software reuse as a frame of reference for the reader, followed by a concise description of the library, its goals, and its activities. Part II is the actual standard. Its major sections address requirement analysis, RSC accession, configuration management, the management of automated library tools, and the management of the library organization as a whole.

Within Part II, each regularly numbered paragraph forms part of the standard, and is considered essential in meeting the reuse objectives addressed by this manual; any deviation must be justified and approved. The standard is augmented by a number of guidelines (indicated by paragraph numbers beginning with the letter "G"). Guidelines support the standard, identifying specific (potentially alternative) approaches to meeting the standard. Adherence to specific guidelines is not considered essential; however some effective approach to meeting the standard must be selected.

The library manager should be familiar with the entire standard and be prepared, especially at the inception of the reuse program, to apply this guidance as flexibly as possible. Priority should be given to setting and achieving realistic goals (tangible benefits of reuse), rather than strict adherence to procedures that are likely to evolve for several years. Never forget that the library is a service and the application engineers seeking reusable components are the library's customers.

Section 2

Applicable Documents

This is one of a set of three documents, specifically addressing the management of a software reuse support organization. The other two documents are standards for the creation of reusable software components and for the reuse of existing software in projects:

Contel Corporation. *Standard for the Development of Reusable Software Components*. NATO contract number CO-5957-ADA, 1991.

Contel Corporation. *Standard for Software Reuse Procedures*. NATO contract number CO-5957-ADA, 1991.

Numerous other references were used in developing this standard, and provide additional guidance in managing a reuse library. Some that may be valuable to the user of this manual are:

SofTech, Incorporated. *Ada Portability Guidelines*. Document number 3285-2-208/1, 1985.

SofTech, Incorporated. *Ada Reusability Guidelines*. Document number 3285-2-208/2, 1985.

SofTech, Incorporated. *RAPID Center Policy Recommendation*. Document number 3451-4-112/5.1, 1988.

SofTech, Incorporated. *RAPID Center Library Procedures*. Document number 3451-4-112/11, 1988.

SofTech, Incorporated. *RAPID Center Reusable Software Component (RSC) Procedures*. Document number 3451-4-326/4, 1990.

Section 3

Basic Reuse Concepts

Software reuse offers tremendous benefits in cost savings and quality; however, it requires technical understanding, changed approaches, and an understanding of potential obstacles.

This section provides a frame of reference for understanding the benefits and challenges of software reuse. It introduces the terminology and concepts used in the remainder of the manual and explains the goals underlying the guidance provided herein.

3.1 Definitions

A consistent terminology is used throughout this and companion manuals.

The following are definitions of the key terms used in this manual:

Reuse—the use of an existing software component in a new context, either elsewhere in the same system or in another system

Reusability—the extent to which a software component is able to be reused. Conformance to an appropriate design and coding standard increases a component's reusability.

Reusable software component (RSC)—a software entity intended for reuse; may be design, code, or other product of the software development process. RSCs are sometimes called "software assets".

Reuser—an individual or organization that reuses an RSC

Portability—the extent to which a software component originally developed on one computer and operating system can be used on another computer and/or operating system. A component's reusability potential is greater if it is easily portable.

Domain—a class of related software applications. Domains are sometimes described as "vertical"—addressing all levels of a single application area (e.g., command and control) and "horizontal"—addressing a particular kind of software processing (e.g., data structure manipulation) across applications. The potential for reuse is generally greater within a single domain.

Domain analysis—the analysis of a selected domain to identify common structures and functions, with the objective of increasing reuse potential

Library—a collection of reusable software components, together with the procedures and support functions required to provide the components to users

Retrieval system—an automated tool that supports classification and retrieval of reusable software components, also called a "repository"

Software life cycle—The series of stages a software system goes through during its development and deployment. While the specific stages differ from one project to the next, they generally include the activities of requirements specification, design, code, testing, and maintenance.

3.2 Expected Benefits of Reuse

Software reuse clearly has the potential to improve productivity and hence reduce cost; it also improves the quality of software systems.

Productivity Improvement. The obvious benefit of software reuse is improved productivity, resulting in cost savings. This productivity gain is not only in code development; costs are also saved in analysis, design, and testing phases. Systems built from reusable parts also have the potential for improved performance and reliability, because the reusable parts can be highly optimized and will have been proven in practice. Conformance to standard design paradigms will reduce training costs, allow more effective practice of quality disciplines, and reduce schedule risk.

Reduced Maintenance Cost. Even more significantly, reuse reduces maintenance cost. Because proven parts are used, expected defects are fewer. Also, there is a smaller body of software to be maintained. For example, if a maintenance organization is responsible for several different systems with a common graphic user interface, only one fix is required to correct a problem in that software, rather than one for each system.

Improved Interoperability. A more specialized benefit is the opportunity to improve interoperability among systems. Through the use of single implementations of interfaces, systems will be able to more effectively interoperate with other systems. For example, if multiple communications systems use a single software package to implement the X.25 protocol, it is very likely that they will be able to interact correctly. Following a written standard has much less guarantee of compatible interpretation.

Support for Rapid Prototyping. Another benefit of reuse is support for *rapid prototyping*, or putting together quick operational models of systems, typically to get customer or user feedback on the capability. A library of reusable components provides an extremely effective basis for quickly building application prototypes.

Reduced Training Cost. Finally, reuse reduces training cost, or the less formal cost associated with employee familiarization with new assignments. It is a move toward packaged technology that is the same from system to system. Just as hardware engineers work with the same basic repertoire of available chips when designing different kinds of systems, software engineers will work with a library of reusable parts with which they will become familiar and adept.

Industry Examples. All of these benefits lead directly to lower-cost, higher-quality software. Some industry experiences have shown such improvements:

- **Raytheon Missile Systems** recognized the redundancy in its business application systems and instituted a reuse program. In an analysis of over 5000 production COBOL programs, three major classes were identified. Templates with standard architectures were designed for each class, and a library of parts developed by modifying existing modules to fit the architectures. Raytheon reports an average of 60% reuse and 50% net productivity increase in new developments.

- **NEC Software Engineering Laboratory** analyzed its business applications and identified 32 logic templates and 130 common algorithms. A reuse library was established to catalogue these templates and components. The library was automated and integrated into NEC's software development environment, which enforces reuse in all stages of development. NEC reports a 6.7:1 productivity improvement and 2.8:1 quality improvement.

- **Fujitsu** analyzed its existing electronic switching systems and catalogued potential reusable parts in its Information Support Center—a library staffed with domain experts, software engineers, and reuse experts. Use of the library is compulsory; library staff members are included in all design reviews. With this approach, Fujitsu has experienced an improvement from 20% of projects on schedule to 70% on schedule in electronic switching systems development

- **GTE Data Services** has established a corporate-wide reuse program. Its activities include identification of reusable assets and development of new assets, cataloguing of these assets in an automated library, asset maintenance, reuser support, and a management support group. GTE reports first year reuse of 14% and savings of $1.5 million, and projected figures of 50% reuse and $10 million savings, in telephony management software development

- **SofTech, Inc.** employs a generic architecture approach in building Ada compiler products. Compilers for new host and target systems can be developed by replacing only selected modules from the standard architecture. This has led to productivity level of 50K lines of code per person-year (10-20 times the industry average). This is typical of compiler developers, as this is a field in which reuse is accepted practice.

- **Universal Defence Systems (UDS)**, in Perth, Australia, develops Ada command and control applications. The company began its work in this business with a reuse focus, and has developed a company-owned library of 396 Ada modules comprising 400-500 thousand LOC. With this base, UDS developed the Australian Maritime Intelligent Support Terminal with approximately 60% reuse, delivering a 700 thousand LOC system in 18 months. A recently begun new project anticipates 50-70% reuse based on the company's asset library.

- **Bofors Electronics** had a requirement to develop command, control, and communications systems for five ship classes. As each ship class was specific to a different country, there are significantly different requirements for each. In order to benefit from reuse, Bofors developed a single generic architecture and a set of large-scale reusable parts to fit that architecture. Because of a well-structured design, internal reuse, and a transition to Ada and modern CASE tools, Bofors experienced a productivity improvement even in building the first ship—from 1.3 lines of code (LOC) per hour previously to 3.28 LOC per hour. Improvements are much greater for

subsequent ships, with a projected productivity of 10.93 LOC per hour for the fifth ship, which is expected to obtain 65% of its code from reuse.

3.3 Dimensions of Reuse

Reuse has several dimensions; the guidance in this manual supports all of these.

Compositional versus Generative Approaches. Approaches to reuse may be classified as either *compositional* or *generative*. Compositional approaches support the bottom-up development of systems from a library of available lower-level components. Much work has been devoted to classification and retrieval technology and to the development of automated systems to support this process. Generative approaches are application domain specific; they adopt a standard domain architecture model (a generic architecture) and standard interfaces for the components. Their goal is to be able to automatically generate a new system from an appropriate specification of its parameters. (The Fourth Generation Languages [4GLs] used in the commercial world can be considered an example of generative reuse.) Such approaches can be highly effective in very well understood domains, but significant effort is required to develop the initial model.

Small-scale versus Large-scale Reuse. Another dimension is the scale of the reusable components. Reuse on a small scale—for example, use of a library of mathematical functions—is practiced fairly widely today. The effort saved from a single reuse is not great; payoff comes from the widespread reuse that is possible. On a large scale, entire subsystems (for example, an aircraft navigation subsystem or a message handling subsystem) may be reused. Here the saving from a single reuse is great; many thousands of lines of code may be reused. However, the opportunities for reuse of a given component are more limited. Large-scale reuse can pay for itself even if a component is only reused once or twice, because of the amount of effort saved.

As-is Reuse versus Reuse with Modification. Components may be reused as is, or may required modification. Generally reusable components are designed to be flexible—for example, through parameterization—but often modification is necessary to meet the reuser's requirement. Modifiability—the capability of a software component to be easily modified—is particularly important in reusable software.

Generality versus Performance. Sometimes there is a trade-off between component generality and performance. A component designed to be general and flexible will often include extra processing to support that generality. Appropriate reusability guidelines help avoid this penalty; guidelines for the reuser can provide mechanisms for coping with performance problems that may arise.

3.4 Forms of Reuse

Reusable components are not necessarily code; they can be specifications, designs, code, tests, or documentation.

Specification Reuse. Reuse of specifications is particularly relevant when aiming for large scale reuse. Large-scale reuse requires up-front consideration during the requirements definition activity. If an entire subsystem is to be designed for reuse, this should be made explicit from the start. The specification is then reusable in systems that will reuse the component, guaranteeing that requirements will match. Reuse of specifications greatly increases the likelihood that design and code will also be reusable. Furthermore, reuse of specifications can reduce time spent on requirements definition and help ensure interoperability, even if neither design or code are reused.

Design Reuse. Sometimes a design can be reused even when the code cannot; for example, the code may not be in the required programming language, or it may have inappropriate environment dependencies. Design reuse can save significant effort in one of the most costly life-cycle phases, provided that the design is specified so as to facilitate reuse. Furthermore, the design phase establishes the software architecture that provides a framework for reuse. Reuse of the software architecture will provide significantly greater code reuse opportunities by establishing a standard functional allocation and uniform interfaces.

Code Reuse. The greatest payoff comes from reuse of actual code. Clearly this is possible only when the specification and design are also reusable. Reusable code should be accompanied by its associated life-cycle products—its requirements and design specifications, its tests, and its documentation—so the reuser will not have to regenerate them.

Test Reuse. Ideally, a reusable code component should be accompanied by test cases that can be used to test it in the environment in which it is reused. A less obvious point is that tests can be reusable even when code is not, with reusable test cases accompanying specification reuse. An example might be the reuse of a specification and a set of test cases for a particular communications protocol. Even if the design and implementation differ from the original, specification and test reuse will save effort and help ensure correctness and interoperability.

Documentation Reuse. Documentation is a major source of software development cost. To be most valuable, a reusable component must be accompanied by appropriate documentation items. Clearly, reuse of a specification or design is only meaningful when the component is in a written form. However, other documentation such as users manuals may also be reusable, even when the code is not.

3.5 Issues in Achieving Reuse

Reuse involves significant change to traditional practice; there a number of challenges to be overcome in achieving its full benefits.

Identifying Opportunities for Reuse. A major technical issue is simply identifying opportunities for reuse. A software engineer may know that similar software has been written before; finding it is another matter. Reuse libraries help solve this problem. Once a component is found, it may be hard to determine if it is indeed a fit, and hard to modify it if change is required. Often software that appears to be reusable in fact will not be—it has inappropriate interfaces, hidden dependencies, inflexible functional limitations, or is simply so difficult to understand that the engineer will be better off simply starting over. The objective of software reusability guidelines is to help avoid these problems.

Investment. Making software that is reusable generally requires investment above and beyond that required for a one-time system. This effort goes into making the software more flexible, ensuring its quality, and providing additional documentation required. Each organization must make decisions about how the investment is supported.

The "Not Invented Here" Syndrome. Sometimes developers are unwilling to reuse software. Software engineers enjoy the creative aspects of their profession, and can feel that these are diminished when reusing software. Management encouragement, training, and perhaps other incentives can help engineers shift to a view of creativity that involves larger "building blocks"—reusable software components.

Estimating and Measuring. Estimating and measuring software development activities has always been difficult, but there are some organizational methods in place that work relatively well. These traditional methods will require modification in a reuse environment, and little data is available to support that modification.

Contractual, Legal, and Ownership Issues. There are a number of contractual, legal, and ownership issues that impact software reuse. Today's usual contracting methods can create a disincentive for contractors to reuse existing software or to provide software for reuse by others. Legal issues arise over liabilities and warranties. Responsibility for maintenance must be identified.

These organizational challenges are, for the most part, outside the scope of this set of manuals. Each organization must develop its own solutions. Managers must be aware of the challenges and address them if reuse is to succeed.

Section 4

Functional Overview of Reuse Library

The library is an organization of personnel, tools, and procedures whose activities are directed at facilitating reuse of software life-cycle products to meet specific goals of cost-effectiveness and productivity. This section presents an overview of the library's activities and resources as a context for the standards and guidelines in the following sections.

4.1 Initial versus Final Operating Capability

This standard concentrates on the initial operating capability (IOC), the actions essential to supporting reuse that must be present when the library is first placed in operation. As reuse becomes standard practice for software engineering and the library becomes well established, additional resources may be allocated so that the library can assume a final operating capability (FOC).

In the remainder of this manual, some FOC activities and issues will be mentioned when necessary to present a clear and understandable picture of reuse issues; in each case, they will be identified as such.

4.2 Primary Function of Library

The main purpose for the existence of the reuse library is ready access to reusable software life-cycle products by the staff of the supported NATO project. Most of the library's activities are directed toward this end.

4.3 Approach

The library staff receives direction in the form of specific operational objectives, principally aimed at making software reuse cost-effective. The staff also manages a continually revised list of RSCs believed to advance those aims which are proposed for inclusion in the library.

This guidance drives a set of coordinated activities with the result that RSCs are added to the library, made available to application engineers for use in projects, and continually improved. These activities include:

Evaluate and rank proposed RSCs. The library evaluates each proposed RSC to note the demand priority, verify assumptions, assess the effort needed to acquire it, and compare it with RSCs already in the library. This allows the proposed RSCs in the list to be ranked by priority so that scarce resources may be allocated for the best return.

Acquire RSCs. Library staff members obtain the actual life-cycle products for each proposed RSC (by priority) by soliciting its development or by purchasing commercial off-the-shelf (COTS) products.

Assess RSC Quality. The staff examine the acquired material for conformance with *Standards and Guidelines for the Design and Development of Reusable Software Components*, completeness, satisfaction of the original requirements and support for the stated objectives, and reusability. RSCs must meet minimum acceptance standards. Each accepted RSC is given a quality rating and deficiencies, if any, are noted in the documentation.

Document RSCs. Like test scripts, adequate documentation should be furnished with each RSC. The staff members verify its completeness, index it as needed to conform with a standard organization, and extract some items for special purposes.

Classify RSCs. To permit automated search and retrieval, each RSC is assigned a set of descriptive keywords corresponding to several different viewpoints. This method, known as a faceted classification scheme, allows reusers to find library RSCs easily by describing their characteristics and is easily extensible to accommodate future RSCs and technologies.

Assimilate RSCs. Once all supporting material is assembled for a given RSC, the staff installs it in the library and enters its description in the catalogue.

Distribute RSCs. The staff ensures that project engineers and managers are made aware of new additions to the catalogue. When a user selects an RSC from the library, certain actions must take place in order to ensure that the complete RSC in its latest version is delivered and to provide for effective future support to this customer. Many of these actions are performed automatically by the library component retrieval system.

Manage RSCs. The library performs various activities in support of configuration management, including keeping track of RSC users and serving as a point of contact for RSC maintenance issues.

4.4 Other Functions in Support of Reuse

In addition to ensuring that RSCs are available, the library is in a position to provide other support to help ensure that the benefits of reuse are realized, including:

Manage Library Tools. As a minimum, the library operates an automated RSC retrieval system to manage the collection of reusable software products. This includes providing user accounts and maintaining the vocabulary of the classification scheme.

Assist Users. The library distributes published manuals like *Standards and Guidelines for Reuse Procedures* and *Standards and Guidelines for the Development of Reusable Software Components*, user documentation for library tools like the automated RSC retrieval system, and the catalogue of RSCs in the library. In addition, some degree of on-call assistance is available to users, as well as a personal walk-through for first-time users of the retrieval system.

Measure Reuse Success. The library collects considerable data for a continuing assessment of the effectiveness of the library's procedures and tools, the usefulness of its RSC collection, the accuracy of RSC classifications, and the general responsiveness of the library to the needs of users. Some of this data is passed on to NATO agencies charged with answering the broader question of the cost-effectiveness of software reuse.

PART II

STANDARD

Section 5

Requirement Analysis

To succeed, reuse support of a software engineering activity must have well-defined, well-understood requirements based on the cost-effectiveness and operational benefits expected. The library staff must understand these goals and translate them into specific supporting objectives attainable in the course of the library's activities.

5.1 General Cost-Effectiveness/Operational Objectives

5.1.1 All library operations will support NATO reuse objectives.

The agency responsible for the success of the reuse effort as a whole provides detailed guidance to the library on the comprehensive reuse benefits expected. In general, these will include:

- Cost savings in software development
- Cost savings in testing and configuration management
- Reduced development time
- Cost savings in software maintenance

For certain projects, or under FOC, additional objectives might include:

- Cost savings in system design
- Higher quality and increased subsystem reliability
- Greater interoperability

G5.1.1.1 Be sure that NATO reuse objectives are understood.

G5.1.1.2 Be sure that NATO reuse objectives are communicated to the library staff.

5.2 Library Supporting Objectives

5.2.1 Define specific objectives, in terms of library actions and goals, that support NATO reuse objectives.

The scope of a comprehensive reuse program is broader than the role played by the library; thus, NATO objectives by themselves are too broad to define the library's mission.

The library staff must define clear, verifiable objectives that will support the directed cost-effectiveness and operational objectives.

G5.2.1.1 Use the following library objectives as a basis:

Relevance of RSC collection. The software life-cycle products in the library must correspond to actual requirements of the supported project and must be described in language recognizable to the project staff.

RSC value. Components selected for the library must provide clearly identified and significant savings. They should be easy to understand, install, and test. They should be complete, including test scripts, test results, and useful documentation.

RSC quality. The quality of RSCs, both as software and as reusable products, must be known, clearly stated, and as high as practical. This includes conformance with development standards and guidelines, documentation accuracy, and satisfaction of stated requirements.

Size of RSC collection. The library should contain a sufficiently wide selection of RSCs to cover the scope of the supported project's requirements and provide choices in design and implementation.

Minimal cost to obtain RSCs. Library tools must be easy to understand and use. RSCs in the collection must be properly classified and clearly described. Once a reuser selects an RSC, there should be minimal procedural burden or delay in obtaining a copy.

Minimal cost to operate library. Recognizing that resources will always be limited, the library should avoid activities that can be deferred or passed on to other agencies, for example, RSC development, maintenance, or reengineering.

Support. The library will be the focal point for "making reuse happen," and the first point of contact for users' concerns. The staff should provide responsive support for perceived problems with RSCs, along with the effective configuration management this implies. The library should retain control over the distribution of RSCs.

Section 6

RSC Accession

Once requirements for suitable RSCs have been identified, actual RSCs meeting these requirements must be obtained, brought into conformance with standards, and added to the library's holdings, ready for retrieval by application engineers. This section describes the series of actions that must take place for this to happen.

6.1 Proposed RSC List

6.1.1 Maintain a list or catalogue of RSCs proposed for accession by the library.

The library receives, from its customers and sponsoring agency, recommendations for RSCs to add to the collection. Proposed RSCs can be existing material that is submitted for inclusion or a required capability that is sought by a customer.

G6.1.1.1 Maintain a file for every proposed RSC.

Since the same or similar recommendations will recur during a project and from one project to the next, keeping a formal file for each proposed RSC avoids duplication of effort and provides a ready reference for all the accession activities described in this section.

G6.1.1.2 Check whether the proposed RSC is really new.

Begin by preparing a tentative classification for every proposed RSC. Using the faceted classification scheme, describe the RSC's known characteristics. Then search for other RSCs matching this classification to determine whether such an RSC, or a similar one, is already on file.

If so, update the file based on any new information. If not, create a new one.

G6.1.1.3 For each proposed RSC, record the requirements of the proposer.

Requirements and perceptions of requirements change with time. In addition, design and implementation considerations can obscure the original intent, with the result that the RSC fails to bring satisfaction. Even when revising requirements, always start with a clear baseline. Include, as a minimum, the following:

Functional description. What kind of product is it? What does it do? How, specifically, will the user determine whether the RSC meets the need?

Availability. Is this existing material that the proposer is recommending for accession? Is it a hypothetical RSC that the proposer is seeking? If so, can the proposer suggest possible sources? What "raw material" (like design, code, documentation, or test objectives) is already available?

Rationale. Why should this RSC be included? Where or by whom will it be used? What benefit will it bring?

6.2 Evaluation and Ranking

6.2.1 Evaluate, qualitatively and quantitatively, every RSC proposed for accession.

The library formally evaluates each proposed RSC in order to:

- Clarify the requirements and ensure that the proposer's expectations are properly understood and achievable
- Broaden the requirements, as appropriate, to meet the needs of multiple users
- Quantify the requirements and estimate the benefits the RSC will bring
- Estimate the costs of acquiring and preparing the RSC for the library
- Rank the proposed RSCs by priority, to allocate scarce resources effectively

G6.2.1.1 Ensure that the requirements address reusability issues.

Usually the proposer will have a particular application in mind. Be sure that the requirements include all likely target applications and environments. If necessary, augment the requirements.

G6.2.1.2 Assess the cost-effectiveness of the RSC.

Using the method described in Appendix A, estimate the reuser's saving, cumulative discounted cash flow, and cumulative discounted cash flow per unit invested. State all assumptions made.

G6.2.1.3 Identify other beneficial characteristics in the RSC.

A number of considerations outside of cost-effectiveness will strongly influence an RSC's desirability. Indeed, the presence or absence of certain "virtues" may determine whether the RSC is reused at all, and in any case will justify some of the assumptions necessary for the cost-effectiveness assessment:

Known demand. An RSC that customers have specifically requested is very likely to be reused, if it is available in time.

Short lead time. This is particularly important when setting up a new library, to meet the goal of an adequately sized RSC collection. In addition, it means that the RSC will be available while the demand is fresh.

Wide applicability. A component that many programmers can use will ensure that many users will learn about the library and begin developing the habit of reuse, regardless of the financial return.

High visibility. An RSC likely to be used in a widely known project can bring favourable publicity with a single reuse.

Compatibility. A component that fits well with or is related to what reusers are already doing appeals to reusers and promotes standards and interoperability.

Low complexity. An RSC that is easily understood and installed is more likely to be reused.

Drudgery avoidance. RSCs that implement necessary capabilities that are tedious to develop and test are likely to find favour with programmers.

"**Glamour**." New and popular technology, well-designed user interfaces, or exceptional performance are appealing for their own sake and encourage the reuse of RSCs that offer them.

G6.2.1.4 Rank the proposed RSCs by priority.

If resources are insufficient to acquire all the RSCs on the list, identify those RSCs which will have first claim on the resources.

6.2.2 Review the evaluation findings with the RSC's proposer.

This helps to ensure that the requirements and expectations were properly understood, corrects unrealistic or unworkable constraints, and enlists the proposer's future support and cooperation.

6.2.3 Retain all RSC evaluation findings.

Retain the evaluation results with the other information in the file for each proposed RSC.

6.3 Acquisition

The library acquires reusable software life-cycle products from numerous sources. In each case, the library staff makes every effort to obtain as complete a product as practical. The greatest degree of control is possible with RSCs obtained from contributors assigned to the supported project or affiliated organizations.

6.3.1 Ensure that RSCs submitted to the library by project engineers conform to the *Standard for the Development of Reusable Software Components*.

That standard specifically addresses the requirements that reusable products must meet in order to support the objectives of the reuse program.

G6.3.1.1 Be sure that all potential submitters of RSCs have ready access to the development standard.

Distribute the standard freely and encourage its incorporation in project software engineering standards.

G6.3.1.2 Be sure that the project's Quality Assurance staff have ready access to the development standard.

If the development standard is part of the supported project's software engineering standards, the actual effort of ensuring compliance is borne by the project's Quality Assurance staff.

6.3.2 Incorporate RSCs by reference, rather than physically, when appropriate.

There is no absolute need for the library to contain a physical copy of each RSC. While in many cases an owned copy is preferred for convenient access and control, there are cases where there are distinct disadvantages.

G6.3.2.1 Incorporate an RSC by reference if:

It is maintained and distributed by an outside organization. Commercial off-the-shelf (COTS) products and many public-domain products are distributed and supported by vendors or individuals. In exceptional cases, the benefits will justify the library's expense of keeping an up-to-date copy for local distribution. Normally, the duplication of effort and potential for obsolescence makes an owned copy impractical. Moreover, COTS products typically have license restrictions.

It carries restricted rights or license agreements. The library cannot assume the responsibility and liability associated with licenses and other legal restrictions on use.

It is considered security-classified material. This is a special case of an RSC that is maintained and distributed by an outside organization. For certain projects, the library may be configured to manage that project's classified RSCs but the library contains only unclassified material.

In the case of software, it is executable code. Transferring executable code is a potential security vulnerability; add only documents and source code to the library.

G6.3.2.2 When an RSC is incorporated by reference, provide instructions for obtaining it.

Recognizing that the purpose of the library is to reduce the cost of reuse, it makes sense to save the user's time and effort by identifying sources and points of contact.

Provide on-line instructions for obtaining the RSC. In some cases, especially where little or no material is furnished directly by the library, the user should be able to read the instructions directly from the retrieval system.

Provide hard-copy instructions for obtaining the RSC. Where significant material (for example, a test plan, reuser manual, or quality assessment) is furnished directly by the library, include the instructions.

In every case, ensure that the use is noted. The library staff needs to track every instance of a reuser obtaining an RSC in order to assess demand, obtain user feedback, distribute change notices, and improve library services. This applies as well to RSCs incorporated by reference.

6.3.3 Encourage the reuse of software life-cycle products from NATO projects.

Projects in the NATO arena, including host nation projects, are likely to have many elements in common. Reusing components among them encourages the evolution of common

architectures, shortens development times, and promotes participation in the library's activities and a strong reuse program.

G6.3.3.1 Always look at the specifications of existing work for opportunities to meet requirements of RSCs proposed for accession by the library.

The library staff should stay informed about NATO and host nation projects; where practical, the library should maintain a reference collection of project functional specifications and other supporting documentation.

G6.3.3.2 Don't overlook the proposer's own project as a source.

Few members of any project's technical staff have an overview of the entire project's design; most work on particular subsystems. Hence, an engineer can be quite unaware that the needed capability is already under development, or even available, as a component of another subsystem.

6.4 Quality Assessment

Library RSCs must be of known quality, as high as practical, but at least meeting minimum acceptance standards. Staff members examine the acquired material constituting the RSC to identify omissions, deficiencies, or discrepancies. The primary object is giving the user a clear idea of what is being provided and what may still need to be done, rather than supplying only components of certifiably high quality.

6.4.1 Allow for incremental enhancement of RSCs.

The library does not normally have the resources to develop or enhance RSCs. This doesn't mean that only complete, highest-quality RSCs are acceptable. Instead, the library accepts any material of significant usefulness, relying mainly on individual reusers to improve RSCs and submit the enhanced product back to the library. The library adds value by identifying omissions, defects, and other shortcomings so that reusers get a product of known status with specific recommendations on additional work needed. Each successive reuser benefits from the enhancements made by the previous customer.

6.4.2 Define minimum acceptance standards for library RSCs.

The fundamental criterion for acceptance is cost-effectiveness for the individual reuser. The cost of obtaining and reusing the RSC, including correcting its shortcomings, must be less than the cost of developing the desired capability directly. The minimum acceptance standards must support this principle in the context of the supported project.

G6.4.2.1 Consider the software engineering standards for the supported project.

Standards for library RSCs are, in general, somewhat higher than those of projects because of the additional requirements reusable products must meet and because deficiencies are multiplied in multiple reuses. This is not to say that the minimum acceptance standards for the library must be higher than project standards; what matters

is the cost to the reuser of bringing the single instance of the reused product up to project standards.

G6.4.2.2 Consider the software engineering process performing the supported project.

An established, experienced engineering team can improve an RSC with little wasted effort; on the other hand, if the RSC implements a technology or design methodology unfamiliar to the reuser, correcting deficiencies may be prohibitively expensive.

G6.4.2.3 Acceptance standards must include the documentation.

No product can be considered reusable if it lacks an understandable description of its capabilities and clear instructions on its use. While a complete reuser's manual goes into considerably more detail, minimally acceptable documentation must include at least that information.

6.4.3 If an RSC is incorporated by reference, modify the quality assessment and testing procedures accordingly.

In some cases, a copy of the latest version may be obtainable, or some version may be accessible for examination at another site. Sometimes a vendor will provide an evaluation version with some features disabled or with a limited life span.

G6.4.3.1 If no reasonable assessment is practical, the RSC cannot be included in the library.

Note this in the RSC's file.

G6.4.3.2 Be sure the quality assessment indicates the basis for the evaluation.

Explain whether a copy was obtained or borrowed, or whether an evaluation version was used. State the release and version numbers.

6.4.4 Examine the acquired material for conformance with the *Standard for the Development of Reusable Software Components*.

The development standard specifically addresses the requirements that reusable products must meet in order to support the objectives of the reuse program. Deviations from the standard are potentially significant and must be noted, but are not necessarily grounds for rejection.

6.4.5 Examine the acquired material for completeness.

Consider all the items a reuser will need to find, understand, install, test, and document the RSC. This list will vary, depending on the RSC's nature.

G6.4.5.1 In general, any library RSC needs the following items to be complete:

- Complete instructions to reuser on characteristics, installation, verification, and operation (the reuser's manual described in Section 8 of the development standard)

- Abstract (described in Section 8 of the development standard)

- Classification
- Actual material to be reused
- Criteria and scaffolding for verification (test objectives, scripts, software, and results)

6.4.6 Assess the general quality and reusability of the complete RSC.

This action calls for a subjective assessment by library staff engineers. While there exist numerous measures relating to software quality (and the library should make use of all that apply), no measure is currently available that concisely and comprehensively answers the questions "How good is this RSC; will it do the job?" Yet these are questions an engineer would not hesitate to ask a colleague, with considerable confidence that the answer will be useful. Even though professionals often disagree, it remains true that there is no automated or "cookbook" substitute for an expert opinion.

G6.4.6.1 Use a simple numeric rating scheme.

Assign the RSC a "general quality and reusability rating" from one to ten. Ten indicates an ideal RSC that exactly meets the stated needs and conforms to the highest standards of modern software engineering practice. One indicates an RSC that just meets minimum standards but still gives the reuser some saving compared with starting over. The range between one and ten might reflect an estimate of the effort completed out of the total estimated effort for the entire development sequence, adjusted up or down for quality, reuse potential, and portability.

Prepare a concise rationale explaining the basis for the rating. Describe the RSC's unusual strengths and weaknesses and the influence of each on the general rating.

G6.4.6.2 Record the quality assessment and recommendations in the RSC's file.

The assessment will be used again by project engineers considering reusing the product, by maintainers modifying or enhancing it, by library staff evaluating proposed RSCs that are similar, and by submitters seeking to correct RSCs that failed to meet minimum standards.

Record the findings using a consistent format. A consistent format, perhaps in the form of checklists or standard outlines, minimizes the time required to review the findings.

Make clear, verifiable recommendations for improvement or further refinement using a directive style. Use imperative sentences like "Include tests for the stated requirements" or "Isolate dependencies on this particular data base management system."

Include a separate summary of recommendations. While individual recommendations will be interspersed with the findings, a separate summary with an assessment of the relative importance of each recommendation helps the reuser decide quickly what will need to be done for this particular reuse.

Keep the rating up to date. To be credible, the rating must reflect the current state of the RSC. Revise it whenever the RSC is improved and whenever actual reusers provide feedback.

Keep the assessment and recommendations up to date. To be credible, the quality assessment and recommendations should reflect the current state of the RSC. Revise them whenever the RSC is improved and whenever actual reusers provide feedback. As a minimum, date each finding and recommendation and add dated revision paragraphs.

6.5 Documentation

The folder for an RSC ready for accession contains considerable information. The staff members verify its completeness and extract some items for special purposes.

6.5.1 Identify documents to be delivered to the reuser with the RSC.

Assemble the following items and review them for readability and consistency. If this RSC is to be delivered in machine-readable form, obtain each item in that form.

- Reuser's manual
- Test plans, objectives, scripts, expected results
- If the RSC *is* a document, the document itself
- General quality and reusability rating and rationale
- Summary of recommendations

6.5.2 Identify documents to be displayed on-line to library users for searching and browsing.

Assemble the following items in machine-readable form, review for readability and consistency, and revise, as appropriate, to facilitate on-line browsing.

- Abstract
- General quality and reusability rating and rationale
- Summary of recommendations

6.5.3 If there is no formal reuser's manual, review the existing documentation and prepare an index.

Except in the case of RSCs developed specifically for the library by project engineers, the reuser's manual will likely be absent. All or most of the information needed by the reuser may be present in the existing documentation.

G6.5.3.1 Index the existing documentation using the organizational format for the reuser's manual.

In general, documentation developed for software is designed to provide an audit trail of the development process and instructions to the end-user. Thus, the information will be organized quite differently from the reuser's manual, which is specifically designed for the needs of the implementor, the reuser. To rearrange the information in a familiar (to the reuser) format, prepare a cross-index arranged like the standard reuser's manual table of contents and indicate where in the available documentation the information actually is.

6.6 Classification

As the library acquires a significant number of RSCs, an automated search and retrieval system becomes indispensable. Several such systems are available as commercial products; many more have been developed by companies for their own use or on behalf of government agencies. Requirements for such systems and other tools are discussed in 8.1, Library Staff Tool Requirements, page 8-1 and in 8.2, Library User Tool Requirements, page 8-3.

6.6.1 Classify every RSC to support automated search and retrieval.

Whatever tool is used, the library must have a way to classify RSCs so that a user can quickly find what is wanted without the frustration and delay of finding bad matches. Sophisticated, expert system, knowledge-based approaches and new technologies for high-speed text search are the subjects of current research efforts. The two methods that have proven themselves in practice are the keyword search and the faceted classification scheme.

The keyword search has the advantages of simplicity and familiarity. In this system, every item to be catalogued has a number of keywords associated with it. The searcher enters words corresponding to the requirements and, typically after many iterations, items in the general category are identified. This method is inefficient, mainly because, without a context, simple words lack precision.

6.6.2 Use a faceted classification scheme to classify RSCs.

A faceted classification scheme places the keywords in context. Moreover, it encourages more precise and accurate classification through the discipline of regarding the item from specific viewpoints, or facets. Numerous tools designed specifically for RSC retrieval use this method. A faceted classification scheme is described in Appendix B.

G6.6.2.1 Encourage submitters of proposed RSCs to include classification descriptors.

The proposer knows the requirements and can describe them clearly, so that the initial search for similar RSCs (Guideline G6.1.1.2, page 6-1) will be effective.

G6.6.2.2 Encourage potential submitters of RSCs to include classification descriptors.

The submitter knows the RSC's characteristics best, and can revise the original proposer's descriptors, if necessary.

G6.6.2.3 Review and revise RSC classifications.

The library staff knows the domain best and is most familiar with the details of the classification scheme. The actual classification that is entered in the retrieval system may call for multiple entries and other subtle usages unlikely to be familiar to users, but which greatly improve the chances of success.

6.7 Assimilation and Distribution

Once the necessary supporting material is assembled for a given RSC, the staff installs it in the library and makes it available to the project engineers.

6.7.1 Assemble each complete RSC for delivery.

G6.7.1.1 A complete RSC includes:

- Abstract
- Classification
- Reuser's manual
- Test plans, objectives, scripts, expected results
- Source code or document to be reused
- General quality and reusability rating and rationale
- Summary of recommendations

G6.7.1.2 Ensure that the RSC package is readily available.

Have machine-readable deliverables organized for efficient retrieval and transmission. For hard-copy deliverables, if reproduction resources limit quick turnaround, have extra copies ready for transmittal.

6.7.2 Enter each RSC's description in the catalogue.

The library staff maintains a catalogue of available RSCs, with periodic updates of new accessions. Once the RSC is assimilated into the library and accessible via the retrieval system, make users aware of its existence by including its description in the next catalogue update.

6.7.3 When new RSCs are acquired, inform interested parties directly.

The person or organization who originally proposed adding the RSC to the collection, anyone who contributed material for the RSC, reusers who sought this or similar capabilities via the

retrieval system and failed, and anyone else who has expressed or implied an interest are all potential reusers of the RSC. Don't wait for them to notice it in the catalogue.

G6.7.3.1 Always note interested parties.

Documents in the RSC's file should always have room for names and dates. The retrieval system should generate a log of unsuccessful searches, and the library staff should check this log frequently.

Section 7

RSC Management

The library assists the supported project's configuration management (CM) team by keeping track of reusable software products and their users and by serving as a point of contact for changes and problem reports.

7.1 RSC Tracking and CM

From the moment an RSC is installed in the library and made available to project engineers, it is managed as a product. The library's incremental enhancement approach and the wide dispersion characteristic of reusable products demand effective CM. The continual process of improvement described in subsection 9.3, Performance Evaluation, page 9-3, depends on a history of the uses of RSCs and the experiences of the reusers.

7.1.1 Provide added value to the maintenance and CM activities.

G7.1.1.1 Don't duplicate the duties of the CM staff.

CM of RSCs is, to a large extent, like CM for all other project products, and is properly the responsibility of the trained CM team. Maintaining separate records introduces more problems than it solves.

G7.1.1.2 Don't try to maintain RSCs.

Normally, and particularly under IOC, the library will have no resources to spare for maintenance, especially since the supported project has the skills available.

G7.1.1.3 Serve as a point of contact for CM and maintenance issues.

The library can provide a service to reusers by acting as a liaison between them and the CM team, as well as coordinating maintenance. While the library staff does not routinely maintain RSCs, it has a responsibility to its customers, the engineers who reuse the components, to ensure quick and effective response to maintenance issues. In general, maintenance of RSCs will be performed by RSC authors or RSC users. Each of these parties has an interest in fixing problems and adding capability; who provides the labor will vary from one case to the next. The library's primary concern is that problems eventually get fixed with no duplication of effort.

G7.1.1.4 Coordinate with the project CM staff to define a problem resolution procedure for RSCs.

The nature of reusable products might call for some changes to the existing problem resolution procedure. Nevertheless, the existing method is the place to start. Be sure the procedure for RSCs takes into account the library's incremental enhancement approach described in 6.4.1, page 6-5.

G7.1.1.5 Maintain and use the logs generated by the automated search and retrieval system.

The retrieval system should track RSC uses, problem reports, search failures, and other data which support CM; improvement of the search mechanisms; and full disclosure to reusers of the current status of each RSC.

7.2 Tracking Reusers

The library staff tracks the transactions of reusers extracting RSCs from the library in various ways. Some of this information supports CM and maintenance; other data, discussed in subsection 9.3, supports performance improvement. Much of the information can be collected automatically by the search and retrieval system; refer to the discussion in subsection 8.1, Library Staff Tool Requirements, page 8-1. This section concentrates on library staff procedures related to CM.

7.2.1 Ensure that every RSC reuse is noted.

The library cannot possibly be an effective point of contact for CM and maintenance for RSCs that are obtained with no record of the extraction. Unfortunately, the strict control measures that would enforce registration of all RSC reuses are not only costly but undermine the goals of reuse by creating disincentives.

G7.2.1.1 Insist that library tools track RSC reuses.

This requirement is important. It allows registration with no burden on the reuser and, in fact, enlists reusers' cooperation through full disclosure. Typically, a reuser has an account for access to the search and retrieval system. The only additional information asked at extraction is an agreement on a "feedback date" when the reuser will be available for an interview (usually telephonic) about the ultimate fate of the RSC and the reuser's experience with it.

G7.2.1.2 Be sure all participants understand the importance of tracking reuses.

Emphasize that the library staff needs to track every instance of a reuser obtaining an RSC in order to assess demand, obtain reuser feedback, distribute change notices, and improve library services. The display that asks for the feedback date might include a statement or "sales pitch" to drive this point home.

G7.2.1.3 Encourage "borrowers" to register uses with the library.

It is likely that engineers working together will share the same RSC for multiple reuses, with the danger that only the original extraction is registered by the library. Be sure everyone understands that there is no cost or penalty for registering the use of an RSC; on the contrary, there are only benefits. Warn the CM team to watch for this.

G7.2.1.4 Don't overlook RSCs incorporated by reference.

Like "borrowed" RSCs, reusable products obtained from sources outside the library may be unregistered. Again, stress the benefits and warn CM. In some cases, not only the reuse, but the very existence of such products may be unknown to the library.

Encourage users of these unregistered software products to propose them for inclusion in the library for the benefit of all.

7.2.2 Ensure that registered but unused RSCs are reported and their use registrations deleted.

If a project engineer obtains a component, registers the use, and then decides not to use it after all, resources will be wasted if the registration remains active. The project engineer will receive unwanted problem reports and change notices as a "reuser," and the RSC will look more attractive than it deserves because, after all, it has been "field-tested!"

G7.2.2.1 Be sure the feedback interview determines the eventual fate of the RSC.

The feedback interview is the best source of information on unused RSCs. Be sure the respondent doesn't feel obligated to pretend it was used.

Section 8

Library Tool Management

Once a reuse program is established and components accumulate, automated search and retrieval tools can make an increasing difference in making these components available to users.

This section describes the requirements for support tools which are unique to the needs of the NATO reuse library. Standard productivity tools such as electronic mail, technical publishing, and spreadsheets should be available as part of the library, but are not discussed here because there are no unique features which the tools must provide.

Requirements expressed explicitly as numbered paragraphs of the standard indicate the minimum military requirement (MMR); the library cannot operate effectively without these. Some features can reasonably be expected in tools, but will not become critical until FOC, and are so noted.

8.1 Library Staff Tool Requirements

The support tools available to the library staff will have direct impact on the effectiveness of the reuse program. Maintenance of the RSCs can become overwhelming if inadequate tools are supplied to the library staff.

8.1.1 Support the cataloguing of RSCs.

Just as libraries of books classify them for easy access, the reuse library must classify its components. There must be an easy-to-use means of associating a classification with an RSC, and this association must support quick searching based on the classification.

The tools should support bulk automatic entry of the RSCs' classification information into the library in order to minimize the data entry burden on the library staff. (FOC)

Initially, a spreadsheet-based package might suffice to track RSCs and their classifications, but a growing collection of RSCs will soon exceed its capacity. A data base management system with an easy-to-use interactive query language would be more powerful and a basis for further enhancement.

G8.1.1.1 Support sublibraries specific to application domains.

The cataloguing tools should support sublibraries for distinct domains such as communications, radar, networks, or logistics, and allow RSCs to belong to more than one domain. (FOC)

8.1.2 Support the maintenance of the classification mechanism.

The classification scheme used to classify the RSCs must not be static. As new technologies, each with its own vocabulary, are identified and added to the library, new elements must be added and deleted from the classification scheme.

8.1.3 Generate transaction and status reports.

The library must be able to generate logs of transactions and produce reports based on these logs. As a minimum, the library must report:

RSC Uses. Report who extracted the RSC, when, and for what project, including a point of contact if different from the user. This facilitates the distribution of problem reports and change notices, demand assessment, and performance measurement.

Classification Scheme. List the descriptive terms used to classify RSCs, by domain.

User Information. Include name, telephone number, mailing address, and project association for each user of the library.

RSC History. Report the RSC change log, which maps to problem reports; feedback log, including inspections which did not turn into actual uses; and relationships to other RSCs, particularly derivations and dependencies.

Problem Reports. Show all outstanding problem reports for a given RSC.

8.1.4 Support CM of the RSCs and of the classification vocabulary.

The library system must prevent unauthorized changes to RSCs in the library (noting that changes by users to extracted RSCs are permitted). The tools must be able to keep track of RSCs and RSC versions, and to track the changes against problem reports or enhancements made by reusers. Similarly, the descriptive terms used to classify products should resist uncontrolled changes.

8.1.5 Support problem-report tracking.

The library must track problem reports to closing, since the library typically acts as a coordinator for problem-report resolution. Also, because of relationships among RSCs and the strategy of incremental enhancements, the library must be able to assess the impact of a problem report against all related RSCs—especially those that have been derived from or are the parents of the RSC the problem report was originally issued against.

8.1.6 Support user and project tracking.

The library must be able to identify the project point of contact for the distribution of new releases of RSCs and for the distribution of problem reports.

The library must be able to identify specific reusers in order to obtain feedback and to determine whether inspections have culminated in actual reuse.

8.1.7 Include adequate tool documentation.

The tools should come with reuser, installation, and maintenance documentation.

8.2 Library User Tool Requirements

The library, with few resources available for labor-intensive tasks, must rely on automation to provide much of the user services that will make reuse successful. While reuse tool technology advances steadily, the MMR can readily be met with currently available tools.

8.2.1 Provide reusers an effective RSC search and retrieval system.

This is the key technology, and it addresses many of the greatest obstacles to reuse by reducing the time, the cost, and the difficulty of identifying and obtaining reusable products. The essential features are:

> **Easy specification.** The reuser should be able to describe the requirements of the desired products without learning a new language or negotiating a cumbersome user interface.
>
> **Iterative refinement.** The system should allow, and preferably assist, the reuser to change the specification, based on the result of a search.
>
> **Detailed information.** After the search has narrowed the list of candidates to a manageable size, the reuser should have access to abstracts, quality assessments, software measures, and other information needed to eliminate unsuitable RSCs and select the best match to the requirements.

These features suggest additional details:

G8.2.1.1 Allow synonyms.

Different domains, and different parts of the same project, use different terminology for requirements that are arguably equivalent. Similarly, reusers with different backgrounds use different terms. An automated thesaurus can bridge the gaps between application jargon, software engineering terminology, plain English, and “engineer’s English.” There is no particular reason why a thesaurus could not accept multiple human languages (for example, English and French) on the same project, although this capability is not readily available today.

G8.2.1.2 Indicate the closeness of each RSC to the stated requirements.

As a minimum, tell the reuser how many terms matched.

G8.2.1.3 Provide tools for browsing RSC information.

The reuser must be able to examine the RSC’s documentation, problem reports, metrics, and log prior to making the commitment to use an RSC. Having this information on-line will greatly increase the likelihood that the information will be accessed.

G8.2.1.4 Allow sorting and other management of the RSC list.

The reuser, trying to narrow the list of candidate RSCs, can keep better track if the list can be rearranged and annotated based on the reuser's reactions to descriptions, problem reports, and software measures.

G8.2.1.5 Support searches across sublibraries or domains.

The searching tools should allow the reuser to search multiple domains for needed components. (FOC)

G8.2.1.6 Support geographically remote reusers.

The searching tools should be easily accessible to geographically remote reusers. The reuse library will be of most benefit if available to engineers through electronic media as part of the development environment. (FOC)

G8.2.1.7 Include a mechanism to provide the reuser a copy of the RSCs requested.

At a minimum, when a reuser selects an RSC from the library, send the reuser a tape or diskette with the requested information. If electronic mechanisms like File Transfer Protocol (FTP) are available, they can be used if the transfer is initiated on the library side of the transfer. (Users cannot be given the file names of the RSCs for CM and security reasons.) Some RSCs, particularly design documents, may not be available in electronic media. In such cases, send a hard copy.

8.2.2 Furnish documentation explaining the features and use of the tools.

G8.2.2.1 Provide on-line help and simple procedures.

Ideally, most users should never need to refer to the published user's manual. More and more modern software is close to this ideal, and today's users resent having to read manuals to perform straightforward tasks.

G8.2.2.2 Provide a detailed user's manual.

On-line help notwithstanding, a manual is still necessary. Some users like to study a manual away from the system to become familiar with the breadth of features available; others find physical pages more comfortable than scrolling text on a screen or navigating menus. Some tasks are complex beyond the capacity of on-line help. Finally, a manual is an excellent place to provide added value in the form of hints and strategies that make the most of the tool's capabilities.

Section 9

Library Organizational Management

A successful reuse program depends critically upon the coordination and the proficiency of the reuse library organization. This section discusses the needed skill mix, user services, and performance evaluation.

9.1 Staff Skills and Responsibilities

In the IOC configuration, which will most likely be the typical configuration for most projects, the library will not have a full-time dedicated staff. Instead, the library tasks will be performed as additional duties by an existing organization with a related mission. A likely candidate is the supported project's maintenance, testing, and validation activity.

Nevertheless, the library's activities call for specific skills and, more importantly, specific viewpoints and priorities. The responsibilities and corresponding skills of the library staff, always focused on cost-effectiveness and operational goals, will evolve as the NATO reuse program matures.

9.1.1 Put the customer first.

The reuse library exists for the benefit of the supported project, and is successful insofar as it increases the project's cost-effectiveness. The project's engineers and managers are the library's customers. Therefore, library activities and procedures give priority to responsive customer service.

This can lead to conflict if, as is likely, the library's duties are performed by the supported project's maintenance, testing, and validation activity. Traditionally, and rightly so, a validation activity calls for a somewhat adversarial position to assure objective assessments.

G9.1.1.1 Make a clear distinction between reuse library activities and conflicting positions.

To the extent practical, separate the activities and, if possible, make this separation clearly apparent to the project staff. Always be responsive to customer needs.

G9.1.1.2 Use creativity and initiative.

While the library functions primarily in a support role rather than as a "reuse advocate," its products and services are the most visible manifestation of the software reuse program. Reuse is an evolving methodology that must overcome many barriers to succeed; if the library is not actively making reuse happen, it is holding it back.

9.1.2 Be conversant with all relevant aspects of the supported project.

Much of the library's added value is the ability to place RSCs in the context of "the big picture," that is, the entire application domain. Library staff members should be able to perceive

opportunities for reuse that may not be apparent to project engineers concentrating on more specialized subsystems.

9.1.3 Know the programming languages and design methods used in the supported project as well as other languages and methods.

Library personnel should recognize commonality and reuse potential even when it is disguised by different design and implementation conventions.

9.1.4 Be thoroughly familiar with the three NATO reuse standards.

Among the library's duties is the application and, where necessary, revision of the three manuals, *Standard for the Development of Reusable Software Components, Standard for Management of a Reusable Software Component Library* (this manual), and *Standard for Software Reuse Procedures*.

9.1.5 Be proficient in modern software engineering practice.

Most technical issues of reuse are nothing more than the systematic application of what is currently considered good practice.

9.1.6 Have a working knowledge of CM principles.

This is essential for the library's role as a point of contact for CM issues pertaining to RSCs.

9.1.7 Have a working knowledge of software testing, validation, and verification procedures.

The library's assessments of reusable products should be done using methods compatible with project practice.

9.2 User Services

The library's main activity is the accession and distribution of RSCs. In support of this activity, the staff performs additional tasks related to CM and maintenance, described in previous sections.

Beyond these activities, the library provides some services directly to the user community.

9.2.1 Publish and distribute a catalogue of available RSCs.

This serves both as a reference, indicating the breadth of offerings, and as advertising, promoting awareness of the support available through the library.

G9.2.1.1 Distribute catalogue updates, announcing additions and changes to the RSC collection, frequently.

Distributing revised versions of the entire catalogue meets the purpose of providing a reference, but fails to make users aware of what is new. The updates might be in the form of a newsletter that includes other reuse-related information of interest to the project staff.

9.2.2 Provide on call assistance.

This is an essential service. Users of the library search and retrieval system and proposers, submitters, and reusers of RSCs will turn to the library staff with their questions and concerns. Be prepared to respond effectively.

G9.2.2.1 Publish answers to frequent questions.

This might include introductory instructions on the use of the tools, an overview of library activities, products, services, and points of contact, and examples of good practice.

G9.2.2.2 Prepare packages of forms, checklists, and instructions for RSC proposers and submitters.

Appendix C includes examples of such forms.

G9.2.2.3 Prepare standard forms for problem reports, RSC enhancements, and other reuser actions.

Appendix C includes suggestions for such forms. Be sure they don't conflict with the project's standard forms and procedures.

9.3 Performance Evaluation

Like any initiative for change, the reuse program must grow into its role; the sponsoring office must continually evaluate the program's success in terms of actual benefits realized and revise standards and practices based on these findings. The library is in a key position to collect data that will assist this evaluation process.

9.3.1 Define specific indicators corresponding to the library's objectives.

Following paragraph 5.2.1, page 5-1, the library staff will:

> Define specific objectives, in terms of library actions and goals, that support NATO reuse objectives.

For each of the objectives defined, the library needs corresponding measures or other verifiable evidence. Note that the indicators need not be numeric, or even particularly quantifiable; for example, a mapping of RSC characteristics to requirements is neither.

G9.3.1.1 For the objectives recommended in Guideline G5.2.1.1, page 5-2, use the following indicators, respectively:

Relevance of RSC collection:

- Fraction of RSCs actually used
- Mapping of RSC characteristics to project requirements and specifications

RSC value:

- Results of evaluation
- Reuser feedback

RSC quality:

- Results of evaluation
- Test results
- General quality and reusability rating
- Problem reports
- Reuser feedback

Size of RSC collection:

- Number of components
- Number of distinct categories of components
- Mapping of categories to application subsystem domains

Minimal cost to obtain RSCs:

- High rate of successful searches by reusers
- Low number of unsatisfied searches
- Low number of unknown or ambiguous terms
- Low search times
- Reuser feedback

Minimal cost to operate library:

- Expenses, by task
- Ratio of library cost to measured benefits of reuse
- Work done outside of IOC requirements

Support:

- Response time for problem reports
- Amount of direct contact with users

9.3.2 Provide regular reports on the performance of the library and its RSCs.

Define and follow a standard procedure for reporting performance to the agency responsible for the reuse program.

G9.3.2.1 Compile the data collected for the above performance indicators.

Use a standard format.

G9.3.2.2 Perform a preliminary analysis.

Annotate the indicators with interpretations and explanations, and combine them, where appropriate, to reduce the volume of data.

G9.3.2.3 Present the results in a summary format for detailed analysis by the sponsoring agency.

Use a standard format.

APPENDICES

Appendix A

Evaluating RSC Cost-Effectiveness

A.1 Overview

Depending on the resources available, mainly labor, evaluating proposed RSCs for cost-effectiveness can range from a rough estimate to a detailed analysis of discounted cash flows. Since consistency is important, the library should select a level of detail that can be sustained over its lifetime and that gives comparable results for all RSCs. The method used to estimate costs should be compatible with software cost estimation methods used by the supported program.

The desired end result is simple to state; there are two quantities of primary importance:

- The net saving to the individual reuser for each instance of reuse of the RSC
- The net saving to the supported program from all reuses of the RSC

The quantities in the analysis can be expressed either in IAU or in labor hours.

A.2 Benefits

Considering likely target applications in the supported project, estimate the following measures of the RSC's usefulness:

> **Saving due to Avoided Cost, S_R**. The sum of costs avoided each time the RSC is reused, that is, the cost that will be incurred if a new component is developed for each instance. Includes design, coding, testing, QA, documentation, maintenance, and CM. Adjust the value if there are multiple likely uses and developers can be expected, in the absence of a library RSC, to salvage existing components to save work.
>
> **Service Life, *L***. The useful lifetime of the component (how long it will be maintained in the library) in years, taking into consideration the nature of the target uses as well as any characteristics of the RSC that might limit its life.
>
> **Demand, *N***. The number of times the RSC is likely to be reused during its service life *L*.

If the estimate takes into account present and future values, use instead:

> **Demand Distribution, N_y ($y = 1, 2, ..., L$)**. The number of times the RSC is likely to be reused in each year of its service life, *L*.

A.3 Costs

Considering the resources available, estimate the following quantities associated with making the RSC available:

Cost to Reuse, C_R. The cost incurred each time the RSC is reused, including identification, retrieval, familiarization, modification (if likely), and installation.

Accession Time, T_A. The amount of time likely to elapse between the decision to acquire the RSC and its availability in the library.

Accession Cost, C_A. The cost to add the RSC to the library, including obtaining raw material, developing the complete RSC, and installing it in the library.

Maintenance Cost, C_M. The cost to maintain the RSC in the library, including CM and change distribution.

If the estimate takes into account present and future values, use instead:

Maintenance Cost Distribution, C_{My} $(y = 1, 2, ..., L)$. The cost to maintain the RSC in the library for each year of its service life L.

A.4 Risks

Assess any risks incurred by the reuser or by the program if the decision is made to acquire the RSC. These may include:

- A dependency on technology or information not yet available
- A dependency on a particular environment or configuration
- Obsolescence of the objects or functions implemented by the RSC
- Obsolescence of the RSC's design or algorithm in areas where technology is advancing quickly
- Legal or contractual obstacles to acquiring or using the RSC

Adjust the cost to reuse, C_R, the accession cost, C_A, or the maintenance cost, C_M (or C_{My}), as appropriate, by an estimate of the cost to overcome identified risks. For risks that can't be overcome, adjust the service life, L, or the demand, N (or N_y)

A.5 Net Saving to the Reuser

Calculate the net saving to the individual reuser for each instance of reuse of the RSC, or the NSR, as the difference between the saving due to avoided cost and the cost to reuse:

$$NSR = S_R - C_R$$

A.6 Net Saving to the Supported Program

The total saving from all instances of reuse is the NSR multiplied by the number of reuses (the demand, *N*).

Calculate the net saving to the supported program from all reuses of the RSC, *NSP*, as the total saving less the accession and maintenance costs:

$$NSP = (NSR \times N) - (C_A + C_M)$$

A.7 Annual Costs and Adjustments for Future Values

It may be useful to calculate the NSP on an annual basis, using the demand distribution and the maintenance cost distribution. For each year y:

$$NSP_y = (NSR \times N_y) - C_{My}$$

This calculation assumes that the accession cost is incurred prior to the beginning of the first year of service and is recorded separately. Alternatively, the accession cost could be amortized over the service life and included as an adjustment to each year's NSP_y.

The cost estimation method may include an adjustment for the fact that future cash flows decrease in value with time. There are many ways to account for this; one method uses a required rate of return or discount rate:

Discount Rate, *i*. The percentage by which future cash flows are discounted each year.

This approach allows the NSP to be expressed as an annual discounted NSP, $DNSP_y$, also called the discounted cash flow:

$$DNSP_y = \frac{(NSR \times N_y) - C_{My}}{(1 + i)^y}$$

A useful measure for comparing RSCs (for example, to determine priorities for scarce resources), is the

Cumulative Discounted Cash Flow, CDCF. This is simply the sum of the annual discounted cash flows, less the accession cost:

$$CDCF = DNSP_1 + DNSP_2 + DNSP_3 + \ldots + DNSP_y - C_A$$

An alternative comparison measure is the

Profitability Index, PI. Also known as the cumulative discounted cash flow per unit invested, this is simply the sum of the annual discounted cash flows divided by the accession cost:

$$PI = \frac{DNSP_1 + DNSP_2 + DNSP_3 + \ldots + DNSP_y - C_A}{C_A}$$

Appendix B

The Faceted Classification Scheme

B.1 How to Classify a Reusable Software Component

Overview

A component classification is a set of {facet, facet term} pairs, also called descriptors, assembled according to the following rules:

1. *Facet* is one of a fixed set of words or short phrases denoting possible aspects, or viewpoints, from which to describe RSCs. The particular words may vary from one installation to another and new ones may occasionally be added during the lifetime of a library system as the need arises.

2. *Facet term* is a word or phrase from the library's list of representative terms for the particular facet. During the startup phase of a new library, the list will grow quickly as the library staff becomes familiar with the users' preferred terminology. However, it is in everyone's interest that the list stabilize soon, after which terms are added only occasionally.

3. There may be any number of pairs in the classification for a particular RSC; any facet can occur any number of times (including zero). In the interest of performance, however, there should always be at least one term for the facet *object* or the facet *function*.

Facets are few in number (between five and ten, occasionally as many as fifteen). Each facet is carefully selected to be clear and unambiguous to the intended user community, but need be neither independent of every other facet nor applicable to every possible RSC.

A typical set of facets might include:

object—a software engineering abstraction implemented by, or operated upon by, the RSC, such as *stack*, *window*, or *sensor*

function—a process or action performed by the RSC, such as *sort*, *assign*, or *delete*

algorithm—any special method name associated with a function or object, such as *bubble* for the descriptor {*function*, *sort*}

rsc type—the particular kind of software life-cycle product that the RSC is, such as *code*, *design*, or *requirement*

language—the method or language used to construct the RSC, for example, *ada*, *c++*, *postcript*, *english*, or *french*

environment—any hardware, software, or protocol for which the RSC is specialized, such as *unix, msdos, postscript,* or *sql*

An RSC is classified by assigning appropriate facet terms for all applicable facets. The objective in classifying the RSC is to make it possible for all users who might in fact have use for the RSC to retrieve it as a result of their requests. Guidelines for this process are given below.

A complete component classification for a sort routine might look like Table B.1:

Facet	**Facet Term**
object	address
object	mail code
function	sort
algorithm	binary sort
rsc type	code
language	ada

Table B.1 - Classification for a Sort Routine

Note that the facet *object* occurs twice (the routine sorts addresses by mail code, but the user might specify either in a search). The facet *environment* is absent (the routine is not tied to any in particular).

A more versatile sort routine might look like Table B.2:

Facet	**Facet Term**
function	sort
algorithm	binary sort
rsc type	code
language	ada

Table B.2 - Classification for a Generic Sort Routine

Here the facet *object* has disappeared, since the object to be sorted is a generic parameter of any type.

The user need not be familiar with all the terms used in classification; many automated systems allow synonyms to be used in searches.

Synonyms are facet terms with the same meaning; a group of synonyms is known as a concept. For example, *delete*, *remove*, *erase*, and *pop* are synonyms. One term from such a group is selected as the representative term (the name of the concept) and serves as the facet term used for the actual classification of RSCs. The remaining synonyms are keyed to the representative term in a list called a thesaurus. When the user enters terms to describe the desired component, any one of the synonyms is equivalent to the representative term.

How to Select Terms, Facet by Facet

1. Look over the available documentation (including the source code) and write a draft abstract. At this point, you should have a clear understanding of the component.

2. Look at each facet's definition and consider how it applies to the RSC. Jot down terms that come to mind, without regard to the list of representative terms.

3. Reconcile your terms with those from the list. If a term you need has no equivalent on the list, prepare a proposed facet term entry:

    - Jot down the term and a definition/rationale (so we can understand why it's different from those on the list).

    - Jot down all possible synonyms. A thesaurus may be helpful. Consider all words a user might employ for this concept, even if they are not true synonyms.

    - Select the original term or one of the synonyms as the representative term. This should be the one with the clearest, least ambiguous meaning within this facet.

    - Write the representative term, all synonyms, the definition/rationale, and the identifier of the RSC for which it is needed on a sheet of paper and submit to the library. New terms will be added periodically and new lists distributed.

4. List the terms for each facet; if any terms are new (proposed to the library), flag them in some clearly visible way.

5. Revise the abstract if you've learned anything new.

Guidelines for Effective Classification

Not all facets need to be employed in classifying an RSC. For example, *algorithm* might not apply. Remember, however, that terms must always be given for *object* or *function*.

More than one facet term may be given for a single facet. For example, a Sort RSC might be classified with {*object*, *address*} and with {*object*, *record*}. Thus, the user who wants to sort addresses will be guided to the RSC, and so will the user who simply asks to sort records. This approach is useful in describing RSCs that can be thought of in either application-oriented or

software-oriented terms. It is also useful if an RSC does more than one thing, runs on more than one system, and so forth.

In classifying packages, subsystems, or other composite RSCs, classify the overall package and each of its constituent components as separate RSCs if it makes sense to do so.

In selecting a facet term, use the representative term rather than one of its synonyms. For example, use *telephone number* rather than *phone*.

Once a component has been classified initially, its classification can be changed or augmented.

If the facet terms that best classify a component are not in the library they can be added.

To classify RSCs most effectively, the librarian must understand the selection process that the user goes through. Librarians should familiarize themselves with this process through actual on-line practice searching for RSCs.

If the system generates a Search Failure Log, it should be reviewed to identify instances in which a user may not be getting a match to an available RSC that meets the needs. This is an indication that the RSC classification needs updating. For example, if a routine that sorts addresses is classified only as sorting {*object, record*}, then the user who requests {*object, address*} will not retrieve it. Noticing this, the librarian might decide to add {*object, address*} to the RSC's classification.

B.2 Maintaining the Library's Classification Scheme

The library classification scheme is designed to allow additions and changes to improve its descriptive power. This subsection addresses the kinds of classification scheme changes available to the librarian.

Maintaining the Descriptor Lists

The descriptor lists are the list of facets and the list of facet terms for each facet. Both can be changed, but the circumstances in which changes are appropriate are very different.

The set of facets provides the basic structure of the classification scheme. They have been carefully chosen to meet a wide range of classification needs, and change should not be undertaken without comparable analysis. Changing the facets involves significant effort, because it means that the classifications of all the RSCs in the library potentially need to be changed. For these reasons, the set of facets should rarely be changed, and such changes should be carefully controlled through appropriate approval policy and access permission. Facet changes may be required when extending the library system to serve a different application area, to meet an installation-specific need, or to address a new policy.

The facet terms, on the other hand, are intended to be added as need arises to classify new RSCs. The following guidelines apply to the addition of facet terms:

- Before adding a new term, be sure that it is not a synonym of an existing term. If so, it should be added as a synonym to that term—see **Maintaining the Thesaurus**, below.

- All terms in a given facet must have unique names, and the names may not be the same as any synonyms for other terms in that facet. For example, there might be two meanings for *state* as an *object* term, a nation and the state of a system entity. *State* cannot appear twice as a term, or once as a term and once as a synonym. The solution is to introduce two terms with unique names like *nation* and *status*, with *state* as a synonym for each.

- Make the new term the same part of speech and case as the existing terms in the facet. For example, the *object* terms are singular nouns like *record* and *address*.

- Always use lower case for portability with systems that are case-sensitive.

- When adding a new term, add obvious synonyms at the same time.

It may be desirable to delete facet terms that have been in the system for some defined period of time without ever being used.

Maintaining the Thesaurus

The thesaurus is the set of synonyms for the facet terms. Synonyms are added to improve the capability of the system to "understand" user requests. The following guidelines apply to adding synonyms:

- In classifying a component, add a synonym if a desired term is a synonym of an existing term.

- Examine the Search Failure Log for instances in which a user failed to get a match when using a synonym for one of the existing terms. It might be appropriate to add that synonym to the thesaurus.

- Avoid adding different parts of speech, plurals, or other complexities to the thesaurus. The user can be expected to conform to the standard for the facet.

- If appropriate, two or more different facet terms can have the same synonym.

As with terms, deletion of synonyms may be appropriate.

Appendix C

Forms and Checklists

This appendix includes the following checklists and forms:

- RSC File Checklist
- RSC File Cover Sheet
- Proposed RSC Requirements Form
- Cost-Effectiveness Evaluation Form
- Conformance Checklist
- Incremental Enhancement Form
- Completeness Assessment Checklist
- General Quality and Reusability Rating Form
- Summary of Recommendations Form

RSC File Checklist

RSC Identifier _____

____ RSC File Cover Sheet

____ Proposed RSC Requirements Form

____ Cost-effectiveness Evaluation Form

____ Evaluation Findings

____ Conformance Checklist

____ Acquisition Instructions

____ Incremental Enhancement Form

____ Completeness Assessment Checklist

____ General Quality and Reusability Rating Form

____ Summary of Recommendations Form

RSC File Cover Sheet

RSC Identifier _____

RSC Name _____

Component Origin _____

Component Type _____

Proposer _____Date:_____

Reviewer _____Date: _____

Change Log		
Date	Change Description	Name

Proposed RSC Requirements Form

RSC Identifier _____

Functional Description:
Availability:
Rationale:

Cost-Effectiveness Evaluation Form, Part 1

RSC Identifier _____

Quantity		Value	Method
Saving due to avoided cost	S_R	_____	
Service Life	L	_____	
Demand	N	_____	
Demand Distribution:		N_y	
	N_1	_____	
	N_2	_____	
	N_3	_____	
	N_4	_____	
	N_5	_____	
(use additional sheet if needed)			
Cost to Reuse	C_R	_____	
Accession Time	T_A	_____	
Accession Cost	C_A	_____	
Maintenance Cost	C_M	_____	
Maintenance Cost Distribution:		C_{My}	
	C_{M1}	_____	
	C_{M2}	_____	
	C_{M3}	_____	
	C_{M4}	_____	
	C_{M5}	_____	
(use additional sheet if needed)			
Risk adjustments:			
	C_R	_____	
	C_A	_____	
	C_M	_____	
	L	_____	
	N	_____	
Discount rate	i	_____	

Cost-Effectiveness Evaluation Form, Part 2

RSC Identifier _____

Calculation	Result	Method
Net saving to reuser ***NSR***	_____	$S_R - C_R$
Net saving to program ***NSP***	_____	$(NSR \times N) - (C_A + C_M)$
Annual net saving:	NSP_y	$(NSR \times N_y) - C_{My}$
NSP_1	_____	
NSP_2	_____	
NSP_3	_____	
NSP_4	_____	
NSP_5	_____	
(use additional sheet if needed)		
Annual discounted net saving:	$DNSP_y$	$NSP_y \div (1 + i)^y$
$DNSP_1$	_____	
$DNSP_2$	_____	
$DNSP_3$	_____	
$DNSP_4$	_____	
$DNSP_5$	_____	
(use additional sheet if needed)		
Cumulative discounted cash flow: ***CDCF***	_____	$\Sigma DNSP_y - C_A$
Profitability index ***PI***	_____	$\Sigma DNSP_y \div C_A$

Conformance Checklist

RSC Identifier _____

Standard	Deviation Description	Analysis

Incremental Enhancement Form

RSC Identifier _____

Deficiency Description	Enhancement Description and Rationale
1. _____	1. _____
2. _____	2. _____
3. _____	3. _____
4. _____	4. _____
5. _____	5. _____
6. _____	6. _____
7. _____	7. _____

Completeness Assessment Checklist

RSC Identifier _____

____ Reuser's Manual

____ Abstract

____ Classification

____ RSC Material

____ Test Material

____ General Quality and Reusability Rating and Rational

____ Summary of Recommendations

General Quality and Reusability Rating Form

RSC Identifier _____

General Quality and Reusability Rating _____

Rationale:

Summary of Recommendations Form

RSC Identifier _____

Priority Rating _____

Recommendations:	Priority: