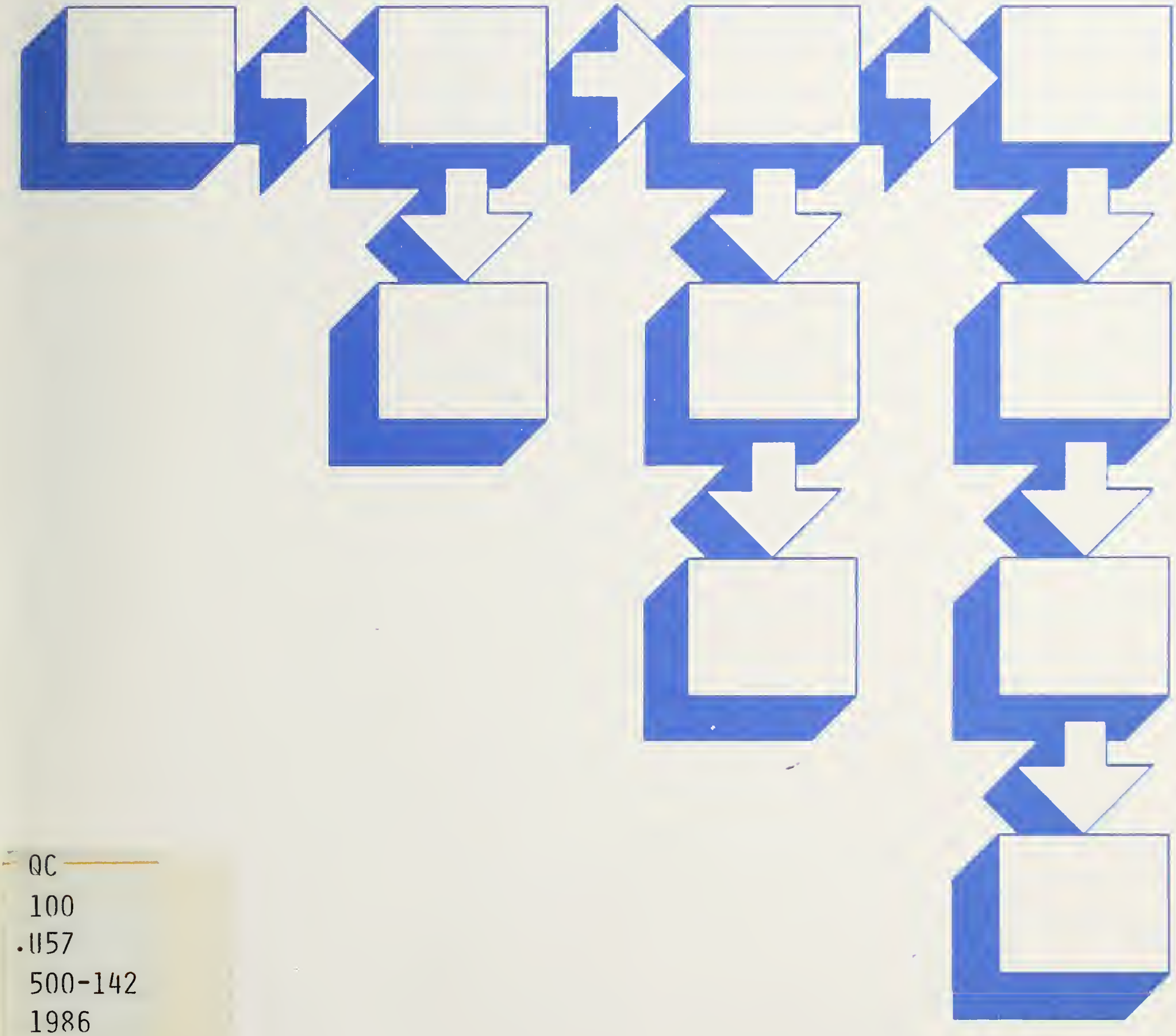
Computer Science and Technology

NBS Special Publication 500-142

A Management Overview of Software Reuse

William Wong



The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Institute for Materials Science and Engineering.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Institute for Materials Science and Engineering

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following Divisions:

- Ceramics
- Fracture and Deformation [3]
- Polymers
- Metallurgy
- Reactor Radiation

[1] Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.

[2] Some divisions within the center are located at Boulder, CO 80303.

[3] Located at Boulder, CO, with some elements at Gaithersburg, MD.

Computer Science and Technology

NBS Special Publication 500-142

A Management Overview of Software Reuse

William Wong

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

September 1986

DEPARTMENT OF COMMERCE • NATIONAL BUREAU OF STANDARDS •

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 86-600581
National Bureau of Standards Special Publication 500-142
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-142, 24 pages (Sept. 1986)
CODEN: XNBSAV

U.S. Government Printing Office
Washington: 1986

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

ABSTRACT

With skyrocketing software costs, both Federal and private sector organizations are increasingly interested in finding ways to improve software quality and productivity, and reduce software risks. Software reuse is one promising method of accomplishing this objective. This report presents a management overview of the problems and issues related to software reuse. It provides a description of software reusability and its scope. The necessity of technical and management involvement to achieve greater levels of software reuse is emphasized.

KEYWORDS

Application domain, design environment, development environment, development methodology, domain analysis, program understanding, software component, software costs, software management, software reusability, software reuse, support tool.

TABLE OF CONTENTS

FIGURES

1.0 FOREWORD

The Institute for Computer Sciences and Technology (ICST) of the National Bureau of Standards (NBS), has a responsibility under Public Law 89-306 (Brooks Act) to promote cost effective selection, acquisition, and utilization of automatic data processing resources within the Federal Government. ICST efforts include research in computer science and technology, technical assistance, and the development of standards and guidelines for data processing equipment, practices, and software. The ICST is developing software reuse guidance designed to assist Federal agencies in improving software quality and productivity as well as controlling software development and maintenance costs.

This report presents a managment overview of the various aspects, problems and benefits of software reuse. Effective reuse of software may involve substantial up-front investment in order to lay the basis for future gains. In addition to technical issues, there are equally important non-technical issues such as lack of standards, resistance to change, data and proprietary rights, and project management problems that need to be addressed and resolved before widespread reuse of software will become a reality. While there is no magic solution to the problem of achieving reusable software, this report provides general guidance in software reuse. Its purpose is to increase awareness of the scope of the issues and the approaches to improving software reuse.

1.1 INTRODUCTION

The cost of computing is now clearly dominated by the cost of software. In the late 1950's software accounted for 20%-30% and hardware 70%-80% of the direct cost of computing [BOEH81]. Today, as a result of technological advances in hardware, these numbers have been reversed. The basic causes of the increased software cost are the explosive growth in size and complexity, the critical nature of modern software systems, and personnel costs. Software costs for both development and maintenance are largely related to the labor-intensiveness of the process and the inadequate use of available technology. Effective reuse of programs, designs, specifications, methods, techniques, and tools that can be adopted from previous work is one way of lowering software costs and reducing software risks. Software reuse has become a key element in efforts to improve quality and productivity, and to reduce software development time and costs.

Reusing well-designed and well-developed software will increase reliability and maintainability not only because the software has been previously tested, but also because it has been used successfully. Developing software for reuse can encourage better designs with greater emphasis on modern development techniques and programming practices.

There is not a strict definition for the scope of software reuse. Some aspects of reusability have been in commerical use for some time, while other aspects are still deeply entrenched in the research community. A common misconception of software reuse is that it is limited to the use of existing source code. Software reuse should be broadly defined as the reuse of any information that may be collected and later used to develop other software. This definition includes reuse of available software development methodologies; software requirements, specifications and design; source code, modules and operating systems; documentation; analysis data; test information; and maintenance information data bases. The reuse of automated tools for generating software and a software support environment to improve software lifecycle processes are also part of the scope of software reuse efforts.

This report is organized into eight sections. Section 1 is this introduction. Section 2 describes the software problem. Section 3 discusses software reusability. Section 4 presents benefits of software reuse. Section 5 discusses obstacles to software reuse. Section 6 describes what is currently being reused. Section 7 discusses what is needed to improve software reuse. Finally, Section 8 summarizes the importance of software reuse.

2.0 THE SOFTWARE PROBLEM

The problems in the development and maintenance of software have increased rapidly over the past decade. There has been an explosive growth in size, complexity, and critical nature of modern software applications and a lack of integrated software development environments for supporting the software lifecycle process. This section discusses several major factors which contribute to the current software crisis. A summary list of the software problems is presented in Figure 1.

FIGURE 1 - THE SOFTWARE PROBLEM

* Increased Complexity Of Software Systems

* Increased Demand For Qualified Software Professionals

* Limited Use Of Software Development Tools and Methodologies

* Frequently Changing Requirements

* Professional Training

* Maintenance

* **Increased Complexity Of Software Systems**

The requirements for new software systems are becoming increasingly complex. Almost every national defense system contains embedded computer software which performs mission critical functions. Many other software systems such as air traffic control, nuclear power plant control, simulation modeling, and manufacturing automation are operated in complex and unpredictable environments. These computer systems have high performance requirements which require the software to be highly flexible and reliable [MART83].

* **Increased Demand For Qualified Software Professionals**

There is a growing shortage of software professionals. The United States Air Force (USAF) Scientific Advisory Board has estimated that the demand for software professionals will continue to exceed available resources through the end of this decade. The shortfall of qualified software professionals may rise to 1.2 million by 1990 if remedial measures are not taken [USAF83, BOEH82]. As a result, the difficulty of developing quality software will continue to rise.

* **Limited Use Of Software Development Tools And Methodologies**

Existing software development tools and methodologies have not been widely adopted and used to develop and maintain software. Many software managers do not know what kind of information is currently available for improving the traditional software lifecycle processes. It is difficult for them to identify the information needed for selecting the right tools and methods without the appropriate information management technique. As a result, software productivity has only increased an estimated

3%-8% per year while the installed data processing capacity has increased at the rate of 40% or more per year [HORO84].

*** Frequently Changing Requirements**
Requirements play a critical role in software development. A major part of software cost, time, and effort is spent on a project's conception, definition, and specification. In fact, most software systems will go through several prototyping and specification revision cycles before implementation is undertaken. In addition, most existing software methodologies do not cope well with changes in requirements and specifications. Software should be designed and developed so that it can be reused and evolved in response to changing needs.

*** Professional Training**
The need to train software professionals and end-users in new computer technology is often overlooked. This is something that should be done through either in-house professional training programs and/or technical educational institutes. Training programs can serve as a feedback mechanism for collecting information from users about their experiences in using the systems, and the problems in understanding and using rapidly changing modern programming techniques and practices.

*** Maintenance**
Software maintenance comprises 60%-70% of the total software lifecycle costs [FIPS106]. The major causes of software maintenance problems are the growth of the inventory of software which must be maintained, and the failure to adopt and utilize improved technical and management methods, techniques, and tools for developing quality software. These software maintenance problems can be addressed through the use of well-designed, well-developed, and well-documented reusable software.

3.0 SOFTWARE REUSABILITY

Software reusability should not be narrowly defined as the reuse of existing source code modules. It should be broadly defined as the reuse of any information that may be collected and later used to develop other software. It includes reuse of:

1) Software development methodology;
2) Requirements, specifications, and design;
3) Source code, and modules;
4) Documentation;
5) Software tools and software support environments;
6) Analysis data;
7) Test information;
8) Maintenance information base.

The reusability challenge must be viewed as one which spans the entire software lifecycle. The reuse of well understood and previously validated specifications and designs can be as important as the reuse of actual high order source code. For example, a well-specified functional requirement for a commonly used function is valuable, especially if it has resulted in useful or effective software, even if the source code can not be reused. Similarly, system design, documentation, development methods and techniques, and software components such as the definition of objects can also be reused. The use of various software support tools from project to project is recommended, from both an economic and reliability perspective.

Furthermore, the software and its supporting information must be easily accessed, understood, and incorporated into a new application. Any modifications required for this integration must be easy to identify, document, and use. Ideally, the software components should be general enough to be used for a variety of related purposes. Before considering reuse as an alternative, it is necessary for a software designer or developer to answer the following questions:

1) What software is available for consideration?
2) Does it meet the requirements (functionality, interfaces, size, costs, debugging aids, operating systems)?
3) What adaptation features are available?
4) What information base is available (specification, design, source, module, executable, documentation)?
5) What changes are pending or under consideration?
6) What restrictions apply to the usages?
7) What support is available if problems are encountered?
8) Who is responsible for notifying the users of changes?
9) How much effort should be devoted to software acquisition and evaluation?

4.0 BENEFITS OF SOFTWARE REUSE

The benefits of software reuse depend on the complexity and size of the software product and the differences between the old and new applications. The incentive for software reuse lies in its increased reliability and quality, reduced development time and lifecycle costs, improved maintainability, and more efficient use of resources.

The more complex a software system is, the higher the anticipated cost to reuse it. Because a significant effort will be required to understand the structure and function of the system, modifications required to reuse a complex system will be made more difficult. Debugging the modifications will be costly. A system with a design and function that is easy to understand will be less costly to reuse than a more complex system. If the software products are reused several times, the incremental cost of creating and cataloging the component can be amortized over the number of times it was used. Similarly, there is benefit in reusing well-developed specifications, designs, tools, analysis data, and development environments. Software reusability can be seen as a capital investment. Figure 2, presents a summary of benefits for the reuse of software.

FIGURE 2 - BENEFITS OF SOFTWARE REUSE

* Economics - Reusing software reduces costs (requirements, design, specification, coding, testing, maintenance, and support tools).

* Reliability - Reusing software products or components which are known to be reliable reduces the potential of unforeseen errors.

* Maintainability - Reusing software products or components which are well-designed and developed for reusability can improve future maintenance efforts.

* Quality - Reusing software products or components can contribute to improved software quality and system performance.

* Development time - Reusing software products can reduce the total time needed to develop and implement a software system.

* Resources - Reusing available software products allows concentration of resources on improvement of the software products and other analysis work.

5.0 OBSTACLES TO SOFTWARE REUSE

Although the concept of reusable software appears attractive from both economical and technical view points, it represents a major deviation from the traditional principles of software production. It may be initially difficult to implement in an organization. This section describes several obstacles to software reuse. Figure 3, presents a summary list of obstacles to software reuse.

FIGURE 3 - OBSTACLES TO SOFTWARE REUSE

- Lack of Confidence In Reusable Software Components
- Little Incentive To Apply The Technology For Reuse
- Resistance To Change
- Software Rarely Designed For Reuse
- Lack Of Standards
- Organizational Issues

*** Lack Of Confidence In Reusable Software Components**
There is resistance from many managers and software developers who doubt that software which is developed by another organization, for another system can be reused in a new system without any modifications. Software developers must be encouraged to reuse existing available software components as much as possible.

*** Little Incentive To Apply The Technology For Reuse**
Software developers may reject the reusability concept, due to their fear of being displaced by it. Many managers also feel threatened with the potential cuts in budgets and resources due to the payoffs of software reusability.

*** Resistance To Change**
A major problem related to software reuse is the "Not Invented Here" syndrome for software developers. In terms of software contractors, the reuse of software may be in conflict with profit for developing, implementing, and maintaining a custom-built software application.

*** Software Rarely Designed For Reuse**
Software that is to be reused must be designed for reuse. Attempting to reuse software components that were not designed for reuse will probably fail. For example, well-designed data structures may have a significant impact on reusability. Generalized data structures which are easy to understand, flexible, and extensible can reduce the costs associated with reusing the software.

*** Lack Of Standards**
Standards problems must be resolved before widespread reuse will become a reality. Standards should be oriented towards defining standard reusable components which can be used by many different software engineering methodologies. For example, there should be an organization specific standard taxonomy for cataloging reusable components in the library. Reusability standards should define what a reusable component is and in what notation it is recorded. This lack of organization-wide standardization also makes it extremely difficult to share software with confidence.

*** Organizational Issues**
Software reuse creates different management problems depending on the number of organizations involved and the relationships among them. For example, if reuse occurs within a single organization which controls both the development of the software components being reused and the decision to reuse, it can be much easier to effect reusability than when multiple organizations are involved. There is also the issue of the desire for a unique software system tailored precisely to the organizations requirements and specifications. In this situation, an off-the-shelf package or software built from reusable components may not quite provide a satisfactory solution without additional modifications.

6.0 WHAT IS CURRENTLY BEING REUSED

There is a wide variety of approaches that address software reusability. The use of subroutine libraries and off-the-shelf software are the most common examples of reusing existing software. The reuse of existing software is becoming more and more practical due to the increasing amount of available, quality software systems. For many commercial applications, modestly priced packages are available and can be incorporated into a software system. Similarly, well-developed existing packages for scientific, government, aerospace, and mission-critical applications are available. It is worth the acquisition and evaluation effort at the beginning of each software development project to determine whether existing software is available for consideration. If candidate software or an appropriate software development methodology is found, a detailed acquisition and evaluation process should be conducted. It is also possible that off-the-shelf software might be acquired as a temporary system while a custom-built system is being designed and developed. This approach can provide the user with an interim solution for refining the requirements and specifications. In addition, it can provided the software developers with a model from which the desirable software system can be designed and built. This may result in reducing software development time and costs. This section briefly describes what is being reused in the various approaches. Figure 4, summarizes what is currently being reused.

FIGURE 4 - WHAT IS CURRENTLY BEING REUSED

* Reusable Design
* Reusable Code
* Application Generators
* Simulation
* Reusable Data

* Reusable Design

Reusable design is an approach advocated by [RICE81] and [PARN83] which consists of performing an analysis of a given domain. This produces a set of concepts and terms which are used when a specific system is to be designed for this domain. The main objective is be able to design programs and components with the potential for reusability and to be able to incorporate reusable components in the design of new applications.

* Reusable Code

The use of subroutine libraries and off-the-shelf software are examples of reusing code. Reusable code has long been an attractive idea, but has yet to be accepted as a practical solution for software development. It requires extensive knowledge of the existing software modules, the programming language, operating system, routine utilities, and input-output devices.

* Application Generators

An application generator is a software package which is designed to help end-users build applications in a given domain. For well-established domains, such as report generation and language parsers, the basics of generating applications in that domain are captured in a tool (the application generator) and only the application-specific details need to be supplied to use the tool to generate the program.

*** Simulation**
Simulation has, for a long time, been recognized as a major type of generic application for software development. Simulation models and test languages are developed as packages and widely used by software project developers. Examples of such simulators are Math Models, Data Processing System Models, Multiple Flight Computer Simulator, Advanced Processor Emulator System (APES) and Statistical Analysis System (SAS).

*** Reusable Data**
A critical problem of the computer industry has been the lack of a standard data interchange format to facilitate both sharing data among applications and systems reusability. Most commercial database systems have data formats that allow many applications to share data under that database. There is, however, no universal data interchange format to allow easy transportability of data from system to system, especially among systems that are competitive in the market place.

7.0 WHAT IS NEEDED TO IMPROVE REUSE

In order to achieve the expected gains in software quality and productivity from the concept of reusability, it is necessary to acquire, explore, evaluate, and use new, innovative software design methodologies and techniques. Furthermore, the iterative enhancement process of a software development lifecycle can be viewed as another effective way of reusing existing software. With the appropriate software development methodology, tools, techniques, and an automation base for capturing the updated software within the existing system, software quality and productivity can be improved substantially. A well integrated software development environment should be provided that supports the entire software lifecycle. This environment can have a significant impact on the ease of developing and maintaining software and on the quality of that software. Use of even a minimum set of commercially available software tools can have a positive impact on the quality of the software and on its reuse. Successful reuse of existing software depends on:

1) techniques used for developing software,
2) methodologies for reusing software in the development lifecycle,
3) integrated software engineering environment and well defined software libraries that promote the reuse of existing software.

This section briefly addresses what is needed to improve software reusability. Figure 5, presents a summary list of what is needed to improve software reuse.

FIGURE 5 - WHAT IS NEEDED TO IMPROVE REUSE

- * Domain Analysis
- * Information Retrieval
- * Program Understanding
- * Interfaces Between Software Components
- * Design Environment
- * Standardization

*** Domain Analysis**

Domain analysis is a generalization of systems analysis in which the objective is to identify the operations and objects (e.g., design, component, specification, requirement, development methods, etc) needed to specify information processing in a particular application domain [NEIG83]. Successful software reusability efforts have occurred in application domains that are well established and well understood. In order to make reusability beneficial, a thorough domain analysis must be performed to identify the basic operations and objects of the domain that have reusability potential. It is also necessary to realize that the application areas that are new or are rapidly evolving may not gain as much benefit from reusability as those areas that repeatedly involve similiar system development efforts.

*** Information Retrieval**

A system of 'software component folders' (e.g., a well defined reusable library) could be organized and cataloged using conventional techniques for indexing information in the area of computer science literature. Having each component in a software library in a form which can be easily retrieved will facilitate reuse.

*** Program Understanding**

In order to reuse software, the software professional must understand how the software works. Software maintenance comprises 60%-70% of the total software lifecycle costs.

Understanding the software, as well as the systems, make up 40%-60% of the software maintenance cost. Thus, a critical issue is the problem of program understanding. Commonality and documentation of software should help to make it easier to reuse. In addition, program functions or descriptions should be coded throughout the system to provide a better understanding of the system.

*** Interfaces Between Software Components**

The interface specifications must specify exactly what the software component does. In addition, the software component must function correctly in that well-integrated software development environment. Interface specifications are very important in providing the basis for schemes of cataloging and retrieving software components.

*** Design Environment**

The design environment is an important factor in software reuse because it provides the foundation on which software is defined, implemented, maintained, and reused. The design environment can enforce the precision required, and provide the support to handle the large volume of information associated with reusable software. Without an adequate design environment, software reuse would be impossible.

*** Standardization**

For effective software reuse, the software support tools such as simulation packages, math model validation aids, High Order Languages (HOL) support tools, database generators, Program Design Languages (PDL) representations, requirements specification aids, and traceability analyzers, as well as hardware components must be standardized. In addition, organizational guidelines must be developed and implemented to deal with all areas of reusable software development, information sharing, and use so that software developers and users alike will have enough confidence to use it.

8.0 SUMMARY

Software development through reuse can substantially reduce software costs and risks, while improving software quality and productivity. The reuse of existing software is becoming more and more practical due to the increasing amount of available, quality software. It is worth the acquisition and evaluation effort at the beginning of each software development project to determine whether existing software or related reusable information is available for consideration. The adaptation of existing software as part of system requirement analysis for developing new systems, the reuse of automated tools for generating software and a software support environment should be encouraged. Studies indicate that many software applications are common and generic [JONE84]. Such source code is a logical target for standard functions, and reusable modules. The potential for sharing software, information, and systems should be a key factor in the decision-making process for future software development and management.

Software and systems can be interchangeable only if standardization and reusability are goals and objectives in the original design. The reusability challenge must be viewed as one which spans the entire software lifecycle. The reuse of well understood and previously validated designs and specifications can be as beneficial as the reuse of source code. The initial reusability thrusts should emphasize understanding the concept of software reuse, and encouraging the use of existing well-developed software, designs, specifications, methods, techniques, and tools to enable economic reuse of software in developing new systems. The benefits of reusing available well-designed and well-documented software can significantly relieve the resource demands for developing timely, cost-effective, reliable software systems.

Effective software reuse requires a substantial investment up-front in order to establish the basis for future gains. The participation of project management and software experts are equally important in the decision process for developing reusable software. Experiences in industry and in Department Of Defense (DOD) underscore the importance of software management representation at top levels within an organization. Top managers must recognize the increasingly critical and pervasive role of software, its characteristics, and the development and selection problems which must be addressed and resolved, in order to be able to make widespread reuse of software a reality.

REFERENCES

[ANDE85] Anderson, C.M, Henne, M, "Reusable Software - A Concept For Cost Reduction", DoD STARS Workshop Reports, April, 1985.

[BARB85] Barbacci, M, Hoberman, A, Shaw, M, "The Software Engineering Institute: Bridging Practice and Potential", Carnegie-Mellon University, 1985.

[BURT85] Burton, B.A, Broido, M.D, "A Phased Approach to ADA Package Reuse", DoD STARS Workshop Reports, April, 1985.

[BUSI84] "Software: the New Driving Force", Business Week, February, 1984.

[BOEH81] Boehm, B.W, Software Engineering Economics, Prentice Hall, 1981.

[BOEH82] Boehm, B.W., Standish, T.A., "Software Technology in the 1990's", Appendix to Software Initiative Plan, 1982.

[CSDL80] The Charles Stark Draper Laboratory, Inc., "A Study Of Software Management And Guidelines For Flight Projects, Final Report", Cambridge, Massachusetts, 1980.

[FIPS106] "Guideline On Software Maintenance", Federal Information Processing Standards Publication 106, National Bureau of Standards, June, 1984.

[FRAN83] Frank, W.L, Critical Issues In Software: A Guide To Software Economics, Strategy, And Profitability, John Wiley, 1983.

[GRAB84} Grabow, P.C, "Reusable Software Implementation Technology Reviews", Hughes Aircraft Company, 1984.

[HORO84] Horowitz, E, "An Expansive View Of Reusable Software", IEEE Transactions on Software Engineering, September, 1984.

[MART83] Martin, E.W, "The Context of STARS", IEEE Computer Society Press, November, 1983.

[JONE84] Jones, T.C, "Reusability In Programming: A Survey Of The State Of The Art", IEEE Transactions on Software Engineering, September, 1984.

[MUXW83] Muxworthy, D.T, Programming For Software Sharing, D.Reidel Publishing Company, 1983.

[NEIG83] Neighbors, J."The Draco Approach to Constructing Software from Reusable Components", Proceedings of ITT Workshop on Reusability in Programming, Newport, RI, September, 1983.

[RICE81] Rice, J.G, Build Program Technique: A Practical Approach For The Development Of Automatic Software Generation System, John Wiley, 1981.

[PARN83] Parnas, Clements, and Weiss, "Enhancing Reusability With Information Hiding", Proceedings of ITT Workshop on Reusability in Programming, Newport, RI, September, 1983.

[STAN84] Standish, T, "Software Reuse", IEEE Transactions on Software Engineering, September, 1984.

[USAF83] USAF Scientific Advisory Board, "Report on the High Cost and Risk of Mission-Critical Software", December, 1983.

[VISC83] Viscomi, A, "What The Software Can't Do", Computer Decisions, November, 1983.

GLOSSARY

algorithm - a finite set of well-defined rules that gives a sequence of operations for performing a given task.

applications software - software which performs a specific task such as word processing, spread sheet analysis, etc. (compare with system software).

compiler - a computer program which translates a high order language program into machine language which can be executed by the central processing unit.

component - a basic part of a system or computer program [*].

custom software - software specially developed for an individual application.

design methodology - a systematic approach to creating a design, consisting of the ordered application of a specific collection of tools, techniques, and guidelines [*].

design specification - a specification that documents how a system is to be built. It typically includes system or component structure, algorithms, control logic, data structures, data set use information, input/output formats, and interface descriptions [*].

development environment - a systematic approach to the creation of software with a set of integrated tools to support the software development lifecycle. The environment includes support tools for requirements and specifications, designing, editing, compiling, testing, configuration management, documentation, and project management.

development methodology - a systematic approach to the creation of software that defines development phases and specifies the activities, products, verification procedures, and completion criteria for each phase [*].

domain analysis - a generalization of system analysis in which the objective is to identify the operations and objects (e.g., design, component, specification, requirement, development method, etc) needed to specify information processing in a particular application domain.

[*] - Adapted from IEEE Standards Glossary of Software Engineering Terminology (IEEE Std. 729) for consistency of definition.

documentation - technical data, including computer listings and printouts in human-readable form which 1) document the design or details of the software, 2) explain the capabilities of the software, or 3) provide operating instructions for using the software to obtain the desired results from computer equipment. It also includes program listings or technical manuals describing the operation and use of programs.

integration - the process of combining software components, hardware components, or both into an overall system [*].

interface - 1) a shared boundary between software modules and/or systems; 2) a hardware component which links two or more devices; 3) that function of a computer program which presents information to an operator and accepts user responses.

module - a well defined section of a computer program with a specific function.

requirement specification - a specification that documents the requirements of a system or system component. It includes functional requirements, performance requirements, interface requirements, design requirements, and development standards [*].

simulation - the representation of selected characteristics of the behavior of one physical or abstract system by another system. In a digital computer system, simulation is done by software [*].

software lifecycle - the period of time that starts when a software product is initiated and ends when a product is no longer available for use. A software lifecycle typically includes phases denoting activities such as initiation, requirements analysis, design, implemenation, test, installation, operation and maintenance.

software product - software that has been developed, tested and documented to a level suitable for delivery to a customer.

software tools - packages, computer programs, and computer systems used to help design, develop, test, analyze, or maintain computer programs, data, and information systems. Examples included high order languages, data base management systems, requirement analyzers, statistical analysis packages, and application generators.

[*] - Adapted from IEEE Standards Glossary of Software Engineering Terminology (IEEE Std. 729) for consistency of definition.

validation - determination of the correctness of the final program or software produced from a development project with respect to the user needs and requirements [FIPS101]. Validation is usually accomplished by verifying each stage of the software development lifecycle.

verification - in general the demonstration of consistency, completeness and correctness of the software at each stage and between each stage of the development lifecycle [FIPS101].

U.S. DEPT. OF COMM.

BIBLIOGRAPHIC DATA SHEET *(See instructions)*

1. PUBLICATION OR REPORT NO.
NBS/SP-500/142

2. Performing Organ. Report No.

3. Publication Date
September 1986

4. TITLE AND SUBTITLE
Computer Science and Technology:
A Management Overview of Software Reuse

5. AUTHOR(S)
William Wong

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
~~WASHINGTON, D.C. 20234~~
Gaithersburg, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered
Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Same as item 6

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number 86-600581

□ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

With skyrocketing software costs, both Federal and private sector organizations are increasingly interested in finding ways to improve software quality and productivity, and reduce software risks. Software reuse is one promising method of accomplishing this objective. This report presents a management overview of the problems and issues related to software reuse. It provides a description of software reusability and its scope. The necessity of technical and management involvement to achieve greater levels of software reuse is emphasized.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

Application domain; design environment; development environment; development methodology; domain analysis; program description; program understanding; software component; software costs; software management; software reusability; software reuse; support tool.

13. AVAILABILITY

☒ Unlimited
□ For Official Distribution. Do Not Release to NTIS
☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
□ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES
24

15. Price

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name ____

Company ____

Address ____

City ____________________ State ______________ Zip Code __________

(Notification key N-503)

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards
Gaithersburg, MD 20899