Repository-based Software Engineering Program

CONCEPT DOCUMENT

Applied Expertise, Inc.

4/14/92

(NASA-CR-190384) REPOSITORY-BASED SOFTWARE ENGINEERING PROGRAM: CONCEPT DOCUMENT (Research Inst. for Computing and Information Systems) 54 p

N92-29200

Unclas

G3/61 0096745

Cooperative Agreement NCC 9-16
Research Activity No. RB.04

NASA Johnson Space Center
Information Systems Directorate
Information Technology Division

Research Institute for Computing and Information Systems
University of Houston-Clear Lake

TECHNICAL REPORT

Repository-based Software Engineering Program

CONCEPT DOCUMENT

RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Applied Expertise, Inc. Dr. E. T. Dickerson served as RICIS research coordinator.

Funding was provided by the Information Technology Division, Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA research coordinator for this activity was Ernest M. Fridge III, Deputy Chief of the Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

University of Houston - Clear Lake
Repository-based Software Engineering Program

CONCEPT DOCUMENT

April 14, 1992

Prepared by:

Applied Expertise, Inc.
1925 N. Lynn Street
Arlington, VA 22209
(703) 516-0911

NASA Cooperative Agreement No. 9-16, Subcontract No. 101

University of Houston - Clear Lake
Repository-based Software Engineering Program

CONCEPT DOCUMENT

April 14, 1992

APPROVED BY:

[illegible]

Frank E. Peñaranda/Level I
Director, Technology Utilization Division/
Deputy Assistant Administrator (Programs)
NASA Office of Commercial Programs

[illegible]

John R. Garman/Level II
Deputy Director, Information Systems Directorate
NASA Johnson Space Center

[illegible]

Dr. E.T. Dickerson/Level III
RBSE Program Manager
University of Houston - Clear Lake

Charles W McKay

Dr. Charles W. McKay
RBSE Chief Scientist
University of Houston - Clear Lake

CONTENTS

EXECUTIVE SUMMARY

THE PROBLEM

Most engineering disciplines reuse common practices, architectures and off-the-shelf parts to build products. This is not the case with software. Current software development practices lack the clarity, consistency and predictability that other, more mature engineering disciplines provide as a matter of course. As an industry expert points out, few construction companies build on-site steel mills to forge custom girders, yet in the software industry, an analogous practice of building software from scratch is common. All too often, fundamental practices of software engineering are dropped to meet immediate schedule or budget pressure. Decisions like this take place even though software is known to be a key to success or failure for virtually every U.S. product or service. This significantly increases costs to the consumer and taxpayer, reduces the competitiveness of U.S. products and services, and creates chilling risks to public health and safety.

Reuse of software engineering assets is widely recognized as a critical factor in solving this problem. Through reuse, software developers can engineer and integrate products a module or subsystem at a time rather than crafting software products line by line. However, interrelated and entrenched technical and non-technical barriers stand in the way. For example, reuse technology is often inaccessible to software practitioners, and customer acquisition policies often discourage reuse.

DEFINITION, GOALS AND OBJECTIVES

No single effort can address all the barriers to reuse throughout government, industry and academia. NASA's Repository-Based Software Engineering (RBSE) Program will fill a unique, high-value role in solving this problem through serving clearly focused customer segments, fostering innovation and working closely with other reuse initiatives.

RBSE introduces and supports common, effective approaches to engineering software. The basis for this Program is the process of conceiving, designing, building and maintaining systems using software assets that are stored in a specialized operational reuse library, or life-cycle repository, accessible to the public. The Program builds upon experience gained through operation of its prototype repository, AdaNET.

RBSE seeks to improve software practices among its target customers -- software developers in key segments of U.S. industry, government and academia -- so their software engineering efforts parallel the clarity, consistency and predictability of other engineering disciplines. As a result, customers will be able to develop better and safer software while saving time and money.

Through its strategy of cooperative participation in national programs designed to encourage the mainstream adoption of software reuse, RBSE will make a significant contribution toward this goal by --

- Delivering and supporting a robust set of reuse products geared to help customers incorporate effective reuse into the way their organizations do business. (Section 3.3.3, Operations)

- Improving the effectiveness of the AdaNET repository (the product delivery mechanism) through application of research results, customer feedback and off-the-shelf software. (Section 3.3.2, Systems Development)

- Filling in critical technology gaps through research, such as methods for organizing and managing software assets and process models. (Section 3.3.1, Research Activities)

- Broadening the customer and supplier base by supporting interoperability. (Section 3.3.4, Interoperability)

RBSE was one of the first programs to evolve the concept of life-cycle repository-based software engineering, which is now being implemented by a number of organizations in government and industry. This breadth of research and development programs, balanced with cooperation, is critical while the technology and practice of reuse matures. It allows the individual reuse libraries to customize technology and services to the needs of their customers, share the advances of other repositories and expand the base of reuse library suppliers and customers. To this end, RBSE participates in the Reuse Library Interoperability Group (RIG), a government and industry organization that promotes reuse through development of consensual standards.

No matter how sophisticated the repository becomes, team members' commitment to serving customers will mean the difference between major organizations' mainstream adoption and occasional use of RBSE technology. Similarly, a program-wide commitment to quality is key to success in any and all Program objectives and goals (Section 3.3.5, Total Quality Management). This commitment is evidenced by the extent to which the Program:

- Is customer-driven, providing customers with what they expect and need.

- Focuses on efficiency, i.e., providing products and services at a minimum cost while ever more effectively increasing bottom-line benefits to target customers.

- Measures its impact using well-defined criteria.

USERS

In order to leverage the greatest impact from its efforts, RBSE seeks to develop a loyal customer base within three overlapping customer groups, in addition to continuing response to requests from the public at large. Those groups are (i) NASA and civilian aerospace, (ii) builders of software-intensive, safety-critical systems, such as railroad, medical, nuclear and hazardous material and (iii) educational organizations interested in reuse. The pull of customer requirements will also lead RBSE and its operational AdaNET repository to fill additional unique, high-value market niches.

IMPACT

Three features together distinguish the RBSE program from other reuse efforts. The Program is committed to:

- **Becoming the supplier of choice for reusable assets within civilian aerospace application domains.** RBSE is in a unique position to offer NASA components to customers throughout NASA and the aerospace and avionics industries.

- **Improving civilian software-intensive, mission- and safety-critical systems.** Recent studies in NASA's Software Engineering Laboratory show that benefits realized by increasing software reuse include a reduction in cost to develop the software by one-third and an improvement in reliability (as measured by defect rates) of 87 percent. These are realized benefits, not potential improvements. RBSE has targeted key, but limited, segments of U.S. industry and academia that maintain a competitive edge and from which technological successes are likely to spread quickly. RBSE, in concert with other reuse and software engineering initiatives, can effect broad and positive changes in the way U.S. industry develops software.

- **Removing the barriers that inhibit introduction of reuse into all phases of the software life cycle through research.** Just as the spreadsheet provided a model that made personal computing accessible to a target market, so can RBSE's research program provide a model that will make reuse attainable for targeted customers in the software engineering community. In addition, demonstration programs and cooperation with universities can aid in training and awareness of reuse practices among succeeding generations of software developers.

In summary, software theory and practice lack several essential elements common to more mature fields of engineering. By building a strong repository-based program, RBSE will promote the reuse of common models, standards, practices, guidelines and life-cycle components necessary to get software engineering accepted as an engineering discipline.

RBSE cannot achieve this broad goal by itself; therefore, the Program is working in concert with other reuse efforts.

In particular, NASA is in a position to make a major contribution to adoption of reuse, both as a source and a reuser of reliable, high-quality software. By making NASA assets as well as other mission- and safety-critical assets available, RBSE will fill a key niche in the software development marketplace. In addition, the Program's research will focus on removing barriers to the introduction of reuse. Through pragmatic research and development, RBSE will leverage resulting innovations through technology transfer to high-impact customer groups. Program success will be judged by how well the needs of these target customers are served and how efficiently reuse products and services are provided to the software engineering community.

1.0 INTRODUCTION

1.1 Identification

This is the Concept Document for the Repository-Based Software Engineering (RBSE) Program. The Program is conducted by Johnson Space Center Information Systems Directorate and the Research Institute for Computing and Information Systems (RICIS) at the University of Houston - Clear Lake (UHCL), under the direction of NASA's Office of Commercial Programs, Technology Utilization Division (Code CU).

1.2 Scope

This document provides the context for RBSE's evolving functional and operational product[1] requirements, and it is the parent document for development of detailed technical and management plans. When furnished, Requirements Documents will serve as the governing RBSE product specification. The RBSE Program Management Plan will define resources, schedules, and technical and organizational approaches to fulfilling the goals and objectives of this Concept.

This document complies with the NASA Software Documentation Standard (NASA-STD-2100-91) and has been tailored to provide a comprehensive understanding of RBSE.

1.3 Purpose and Objectives

The purpose of this document is to provide a concise overview of RBSE, describe the rationale for the Program, and define a clear, common vision for RBSE team members and customers. The document also provides the foundation for developing RBSE user and system requirements and a corresponding Program Management Plan. The Concept will be used to express the program mission to RBSE users and managers and provide an exhibit for community review. The document will be changed in response to community feedback and Program adjustments.

1.4 Status

This Concept Document was baselined (reviewed and approved by NASA) on April 14, 1992.

1.5 Organization

The Document comprises nine sections including this introduction section. The other sections are:

[1] Product here refers to the repository, which itself acts to deliver end products to customers.

- **2.0 Related Documentation** - This section identifies laws, regulations standards and other documents that provide background, historical perspective and related technical information. It includes all documents referenced within this Concept.

- **3.0 Definition of the RBSE Program** - This section provides Program background; describes the Program rationale and mission; identifies RBSE products, services and benefits; and describes Program research, development and operational approaches.

- **4.0 User Definition** - This section identifies and classifies RBSE users, and it describes the general capabilities and characteristics required by those users, as well as the unique requirements of targeted, high-priority RBSE customers.

- **5.0 Capabilities and Characteristics** - This section presents general RBSE capabilities and characteristics and describes expected life-cycle repository usage by different classes of users.

- **6.0 Operational Scenarios** - This section presents a number of scenarios illustrating user interaction with the repository.

- **7.0 Abbreviations and Acronyms** - This section lists definitions for abbreviations and acronyms used in the Document.

- **8.0 Glossary** - This section lists definitions for special or unusual terms used in the Document.

- **9.0 Appendices** - These sections contain material that enhances and expands the reader's understanding of the Document.

2.0 RELATED DOCUMENTATION

This section includes citations for documents that are referenced by or directly applicable to the Concept, or contain policies or other directive matters that are binding upon the Concept.

2.1 Applicable Policies and Standards

1. Section 203(a)(3), National Aeronautics and Space Act of 1958 (42 U.S.C. 2473).

2. National Aeronautics and Space Administration (NASA) Federal Acquisition Regulations, Supplemental Directive, September 30, 1989, Subpart 18-27.372, Policy.

3. NASA Software Documentation Standard, NASA-STD-2100-91, July 29, 1991, Software Engineering Program, Office of Safety and Mission Quality, Washington, D.C.

4. NASA Management Instruction 2210.2B, "Distribution of Computer Programs," December 13, 1990, CU/Technology Utilization Division, NASA, Washington, D.C.

2.2 RBSE Program Documents

1. AdaNET Service Version 2.0 (ASV2) User's Guide, April 24, 1990, Subcontract No. 044, Cooperative Agreement NCC-9-16, University of Houston - Clear Lake, Research Institute for Computing and Information Systems.

2. Design and Implementation Document, AdaNET Service Version 2.0 (ASV2), April 4, 1989, Subcontract No. 002, Cooperative Agreement NCC-9-16 University of Houston - Clear Lake, Research Institute for Computing and Information Systems.

3. Focused Ada Research/Applied Expertise, AdaNET Concept Document and Program Management Plan: Focused Ada Research Comments, March 4, 1991.

4. McKay, C.W., "Conceptual and Implementation Models Which Support Reusability of Processes and Products in Computer Systems and Software Engineering," SERC/RICIS Report, AIRMICS™HTL Grant 2-5-51537, October 1988.

2.3 Other Documents

1. ACM, Proceedings of the 4th International Software Process Workshop: Representing and Enacting the Software Process, Software Engineering Notes (14), June 1989.

2. Ada Information Clearinghouse (Ada IC) Newsletters published by Illinois Institute of Technology Research Institute (IITRI), Lanham, Maryland.

3. Ada Software Repository Newsletters, published by Management Assistance Corporation of America, White Sands Missile Range, New Mexico.

4. Biggerstaff, T.J., Richter, C., "Reusability Framework Assessment and Directions," Software Reusability, Volume 1: Concepts and Models, Biggerstaff, T. J. and Perlis, A. J. (eds.), Addison Wesley, 1989.

5. Boehm, B., "What We Really Need Are Process Model Generators," Proceedings of the 11th International Conference on Software Engineering, IEEE, May 1989.

6. Booch, Grady, Object Oriented Design with Applications, Benjamin/Cummings, 1991.

7. Brooks, Fred, "No Silver Bullet," Computer, April 1987.

8. "Can the U.S. Stay Ahead in Software?" Business Week, March 11, 1991.

9. Committee on Science, Space and Technology, U.S. House of Representatives, "Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation," 1989.

10. Computer Science and Technology Board, National Research Council, Workshop Report, "Scaling Up: A Research Agenda for Software Engineering," 1990.

11. Computer Software Management Information Center (COSMIC) Software Catalog, 1990 Edition, NASA-CR-185888.

12. Common Ada Missile Packages (CAMP), AFATL-TR-85-93, May 1986, Air Force Armament Laboratory, Eglin Air Force Base, Florida.

13. DOD, Department of Defense Software Master Plan, Preliminary Draft, U.S. Department of Defense, February 1990.

14. DOD, Software Technology Strategy, Draft, December 1991.

15. DOD, Total Quality Management: Master Plan, U.S. Department of Defense, August 1988.

16. DOD-STD-2167A, Defense System Software Development, February 29, 1988.

17. Humphrey, W.S., Managing the Software Process, Addison-Wesley, 1989.

18. Humphrey, W.S., Kellner, M.I., "Software Process Modeling: Principles of Entity Process Models," Proceedings of the 11th International Conference on Software Engineering, IEEE, May 1989.

19. Jost, J., "The Successful Introduction of Software Reuse Across an Entity," Proceedings of the Sixteenth Annual NASA/Goddard Software Engineering Workshop, December 4-5, 1991.

20. Kellner, M.I., "Software Process Modeling: Value and Experience," Technical Review 1989, Software Engineering Institute, 1990.

21. McDonnell Douglas, "CAMP: Developing and Using Ada Parts in Real Time Embedded Systems," 1990.

22. McGarry, Frank and Waligora, Sharon, "Experiments in Software Engineering Technology: Recent Studies in the SEL (1990-1991)," Proceedings of the Sixteenth Annual NASA/Goddard Software Engineering Workshop, December 4-5, 1991.

23. Memorandum from the Under Secretary of Defense entitled "Total Quality Management (TQM) in Acquisition and the Transition from Development to Production," January 12, 1989.

24. Out of the Crises, Deming, W. Edwards, MIT Press, 1986, ISBN 0-911379-01-0.

25. "Report of the Joint Logistics Commanders' San Antonio I Software Reusability Panel," January 28-February 1, 1991.

26. Reuse Library Interoperability Group, RIG_TC1 Glossary, Draft, January 22, 1992.

27. Riddle, W.E., "Session Summary-Opening Session", Proceedings of the 4th International Software Process Workshop: Representing and Enacting the Software Process, ACM Software Engineering Notes (14), June 1989.

28. Rothrock, Jack, "Reusable Ada Products for Information Systems Development (RAPID) Reuse - Year 2000," Washington Ada Symposium Proceedings, June 1990.

29. Sullivan, L. J., Presentation at Johnson Space Center on the "Johnson Space Center Team Excellence Initiative," July 31, 1990.

30. System Administrator's Guide for the Reusable Ada Packages for Information Systems Development (RAPID) Center Library System, 3451-4-112/9.3, January 13, 1989, Contract No. F33600-87-D-0337, Contract Data Requirements List (CDRL) Seq. No. A013, U.S. Army.

31. Tracz, W. "Where Does Reuse Start?" Software Engineering Notes, Association for Computing Machinery, 1990.

32. Willmott, Thomas H., "Software Solutions for the IBM PC," Prentice-Hall, 1983.

33. Wood, H.M., "Computer Society President's Message: Meeting the Technology Challenge," Computer, August 1990.

3.0 DEFINITION OF THE REPOSITORY-BASED SOFTWARE ENGINEERING PROGRAM

RBSE is a research and development program that will upgrade and evolve a public-domain software engineering repository called AdaNET. Repository-based software engineering refers to the process of conceiving, designing, building and maintaining systems using software assets that are stored in a specialized operational reuse library, or life-cycle repository. A software asset is a reusable entity. Assets include software components as well as supporting material such as standards, informational documents, test data and even conference announcements.

The repository will acquire, store and disseminate reusable assets from across the software engineering life cycle to customers in industry, government and academia.[2] In addition to software components, RBSE will maintain standards, practices and guidelines relating to repository-based approaches to software engineering and to life-cycle repositories. Through targeted research, the Program will address critical technology gaps in order to promote customer adoption of reuse in all phases of the software engineering life cycle.

3.1 Purpose and Scope

The purpose of RBSE is to support the adoption of software reuse through repository-based software engineering in targeted sectors of industry, government and academia. The Program provides a repository that facilitates the selection, acquisition, integration and reuse of software components, provides proven architectures upon which to assemble systems and promotes common software engineering practices and standards.

3.1.1 Background

Software plays an expanding and increasingly critical role in the safety, quality and competitiveness of U.S. products and services. Software is critical to the missions of virtually all U.S. private and public organizations. Even small improvements in large software programs can lead to significant savings and improvements in mission capabilities.

A recent Business Week article pegs the 1991 U.S. share of the world market for off-the-shelf software at $55 billion. This is just the tip of the iceberg. According to the article, "High-quality software is the key to running everything from personal computers to Patriot missiles.

[2] The software engineering life cycle describes a series of stages for software development from the time a concept is defined through software implementation. NASA and the Defense Department generally adhere to the following eight-step life cycle:

1 - Concept and Initiation	5 - Implementation / Coordination
2 - Requirements	6 - Integration and Test
3 - Architectural Design	7 - Acceptance and Delivery
4 - Detailed Design	8 - Sustaining Engineering and Operations

In automated factories, stock exchanges, banks, airlines and just about every other business, America's software edge is helping" to maintain U.S. competitiveness in world markets. However, the article describes the all-too-familiar software development practice of "shipping the product and getting the details right later.... Quality could be the Achilles' heel of the U.S. software industry -- and industry executives know it."[3]

RBSE and related programs can transform the approach for software development and support from a "software cottage" to a quality-driven "software factory" and enable needed improvements in software engineering practice, thus filling a critical technology gap and improving national competitiveness.

The RBSE team developed a prototype software repository located in Dellslow, West Virginia, called AdaNET Service Version 2 (ASV2). It has been operational since October 1989 and supports a growing number of customers in government, industry and academia. AdaNET also provides business and technical support to companies adopting reuse. Current sources for software engineering components include the Army's Ada Software Repository, NASA's Jet Propulsion Laboratory, and DOD's Software Technology for Adaptable Reliable Systems (STARS) program. The repository also includes non-source code assets, including standards, educational tools and abstracts from over 60 government and industry software periodicals. AdaNET staff participate in demonstrations and presentations to promote the Ada language and reuse among educators and students.

3.1.2 Problem Statement

Most engineering disciplines reuse common practices, architectures and off-the-shelf parts to build products. This is not the case with software. Current software development practices lack the clarity, consistency and predictability that other, more mature engineering disciplines provide as a matter of course. For example, buildings, bridges and even computer hardware are built by using common models, practices, guidelines, interfaces and off-the-shelf parts. These elements, taken for granted in other engineering disciplines, are in their infancy for software engineering.

Grady Booch, a top industry expert who specializes in the design of huge software systems, points out that few construction companies build on-site steel mills to forge custom girders, yet in the software industry such practice is common.

All too often, fundamental software engineering practices are dropped to meet immediate schedule or budget pressure. Booch observes:

Rarely would a builder think about adding a new sub-basement to an existing 100-story building; to do so would be very costly and would undoubtedly invite failure.

[3] "Can the U.S. Stay Ahead in Software?," Business Week, March 11, 1991.

Amazingly, users of software systems rarely think twice about asking for equivalent changes. Besides, they argue, it is only a simple matter of programming.[4]

Decisions like this take place even though software is known to be a key to success or failure for virtually every U.S. product or service. This significantly increases costs to the consumer and taxpayer, reduces the competitiveness of U.S. products and services, and creates chilling risks to public health and safety (see Appendix A, The Software Crisis).

The level of software practice falls far short of the level of the more mature engineering disciplines. Solving this problem is not a simple task, nor are all solutions within the scope of any single government or industry effort.

Reuse of software engineering assets is widely recognized as a critical factor in solving this problem. Through reuse, software developers can engineer and integrate products a module or subsystem at a time rather than crafting software products line by line. However, interrelated and entrenched technical and non-technical barriers stand in the way. For example, reuse technology is often inaccessible to software practitioners, and customer acquisition policies often discourage reuse.

The Joint Logistics Commanders' San Antonio I Software Reusability Panel identified 24 barriers to effective software reuse that fall into three categories: business and non-technical problems, software development process deficiencies, and gaps in supporting technology. In particular, the panel noted these barriers:

- *Software reuse is not integrated into a good overall software engineering process.*
- *There are no identifiable incentives for anyone in either government or industry to create reusable software assets.*
- *The way recoupment policy is administered discourages contractors from investing in reusable software assets.*
- *There is a widespread lack of confidence in the quality and suitability of assets available for reuse.*
- *The issues of liability and data rights appear to have a direct effect on reuse.*
- *The lack of a centralized asset library was noted as a perceived barrier, though the panel warned against viewing central catalogs as a panacea.*[5]

RBSE will fill a unique, high-value role in promoting software reuse, and the Program will <u>increase U.S. competitiveness through effective government to industry technology transfer</u>.

---

[4] Booch, Grady, <u>Object Oriented Design with Applications</u>, Benjamin/Cummings, 1991.

[5] "Report of the Joint Logistics Commanders' San Antonio I Software Reusability Panel," January 28 - February 1, 1991.

3.1.3 The Unique Role of RBSE

Advancing U.S. software practice to the level of more mature engineering disciplines requires a coordinated attack on a broad set of entrenched, interrelated problems. U.S. government agencies and contractors as well as private industry have launched several efforts to advance software engineering, but no one effort can target all areas of the problem or carry out all activities designed to solve it. These activities range from fostering software domain analysis to developing training programs for acquisition personnel. Within the combined efforts of interrelated programs required to address all facets of the software engineering problem RBSE fills a critical niche.

RBSE will contribute to improved software engineering through participation in the following efforts:

- Introduce standards, architectures, processes and models to facilitate exchange and reuse of life-cycle products.
- Build in West Virginia libraries of quality software reuse products tailored to specific application domains.
- Replace government acquisition barriers with incentives to reuse software (see Section 3.4 Policies)
- Build confidence in the quality and suitability of assets available for reuse through application of Total Quality Management principles (see Section 3.3.5 Total Quality Management).

Within this set of activities, RBSE has targeted three areas of focus, which taken together make the Program unique. It is committed to:

- **Becoming the supplier of choice for reusable software assets within civilian aerospace application domains.** The program will acquire life-cycle products in response to the demands of customers in the U.S. aerospace industry and in civilian aerospace agencies. Specifically, RBSE's repository is better able than other government programs to provide NASA software components and documents to customers throughout NASA and the aerospace and avionics industries.
- **Improving civilian software-intensive, mission- and safety-critical systems.** RBSE will play a leadership role in serving industries that are concerned with other types of software-intensive, safety-critical systems. The program initially will select a few specific industry segments such as the medical equipment or nuclear power industries, work to identify software reuse needs and system requirements of those industry customers, and tailor products and services to fill those needs (Section 5.0, Capabilities and Characteristics).

- **Removing the key barriers that inhibit customers' introduction of reuse into all phases of the software life cycle.** Other than code, few if any life-cycle assets have been distributed by most reuse libraries. This is attributed to complex technical problems and the reluctance of people to accept new technology. However, some innovations can foster acceptance of new technology. For example, the spreadsheet made computing technology accessible and useful to millions of professionals by adapting the computer's resources to a model that was familiar to these professionals (Section 3.3.1, Research Activities).

3.2 Goals and Objectives

RBSE will help organizations achieve and improve upon software engineering quality goals that are specific to organizational objectives. RBSE products and services support customers' continuous-improvement efforts to produce measurable results in line with their key objectives.

3.2.1 Long-Term Goals

An ongoing and long-term goal is to improve the quality of software-intensive systems developed by RBSE's customers and to increase RBSE's responsiveness to those customers' demands for products and services that support software reuse. To support this goal, RBSE customers may target more specific goals that address, for example, reducing development and maintenance cost:

- A ten-year goal is to help reduce the life-cycle costs of sustaining engineering for customers' future systems by at least 50 percent of today's costs. Such a reduction will not be possible for the "brittle" software-intensive systems produced by past practices. Systems developed using the life-cycle repository's resources will be more flexible than those developed with traditional methods and technology.

- A ten-year goal is to help reduce the time and human effort needed to create new systems that are safe, adaptable and affordable by at least 20 percent of today's costs and reduce errors by a factor of ten.

These goals are in line with those articulated by DOD's Software Technology Strategy,[6] published in 1991. Several studies, cited in Section 3.2.3 below, have demonstrated that these goals are currently achievable. While the cost reduction goals are well within reach for organizations developing systems with the repository-based approach advocated by RBSE and others, they will only be achieved *if customer organizations have also adopted the necessary software engineering practices*. In other words, these organizations must be prepared to make significant cultural changes in how they develop software.

[6] "Draft Department of Defense Software Technology Strategy," December 1991.

3.2.2 Specific Objectives

RBSE will make significant progress toward its goals by --

- Delivering and supporting a robust set of reuse products geared to help customers incorporate effective reuse into the way their organizations do business. (Section 3.3.3, Operations)

- Improving the effectiveness of the AdaNET repository (the product delivery mechanism) through application of research results, customer feedback and off-the-shelf software. (Section 3.3.2, Systems Development)

- Filling in critical technology gaps through research, such as methods for organizing and managing software assets and process models. (Section 3.3.1, Research Activities)

- Broadening the customer and supplier base by supporting interoperability. (Section 3.3.4, Interoperability)

3.2.3 Measurements and Benefits

Effective measurement is critical to achieving RBSE program goals. To effectively measure, it is necessary to translate Program goals into specific criteria. Criteria define program success in clear terms that individuals and organizations within the Program can effect through their day-to-day activities. This brings "quality improvement" out of the conceptual realm and into everyone's daily lives. In this way criteria help track progress, communicate management's definition of success, increase productivity and employee satisfaction and lay the foundation for continuous improvement.

The ability to provide products and services at a minimum cost while ever more effectively increasing bottom-line benefits to target customers -- that is, program efficiency -- is a key criterion for RBSE. It is important to measure efficiency impacts because (i) the problem is large and resources to deal with it are limited, (ii) constant adaptation to customer needs is critical and (iii) commercial viability is a goal.

There is evidence that these benefits are achievable. The Software Engineering Laboratory at NASA's Goddard Space Flight Center Flight Dynamics Division has kept meticulous data on the software development process and the software products of the Flight Dynamics Division for about 15 years. A recent presentation at the 1992 SEL Software Engineering Workshop compared and contrasted the current software development environment with data collected on FORTRAN projects over a 15-year period within the Flight Dynamics Division at NASA. The study showed that, as a direct result of increased reuse:

- The effort to deliver a line of code decreased from .65 to .42 staff hours per statement -- <u>an improvement of 35 percent</u>.

- Productivity was raised from 12.4 to 19.0 statements per day -- <u>an improvement of 53 percent</u>.

- The errors per delivered statement fell from 3.9 to .5 -- <u>an improvement of 87 percent</u>.[7]

A Hewlett-Packard Logic Systems Division study also indicated an increase in software productivity and reliability. The study documented that projects in which the firm did not reuse software components took 60 percent longer to develop than those that did. In addition, when reuse was built into the design of an entire product line, pre-release defects dropped from more than 180 in the first product to less than ten in the tenth and subsequent products.[8]

These two studies stress the multiple benefits resulting from software reuse: decrease in cost, increase in reliability and decrease in schedule for systems developed by reusing existing components. RBSE will help its customers realize these benefits.

It is essential that RBSE monitor progress against specific, measurable criteria in order to ensure that it will fulfill its significant goals. For instance, the RBSE team will ask such questions about efficiency as:

- Did the technology transfer facilitate cost-effective reuse of software assets?

- Was RBSE the supplier of choice for target customer groups?

- Do products and services map directly to program objectives? and customer requirements?

- Do asset acquisition policies map to program objectives and customer requirements?

- Did investments, products and services provide high value relative to their cost?

Criteria that assess efficiency and RBSE fulfillment of customer needs will set guideposts for continuous program improvement. Criteria, along with corresponding measurements and goals, will be laid out in the RBSE Program Management Plan.

---

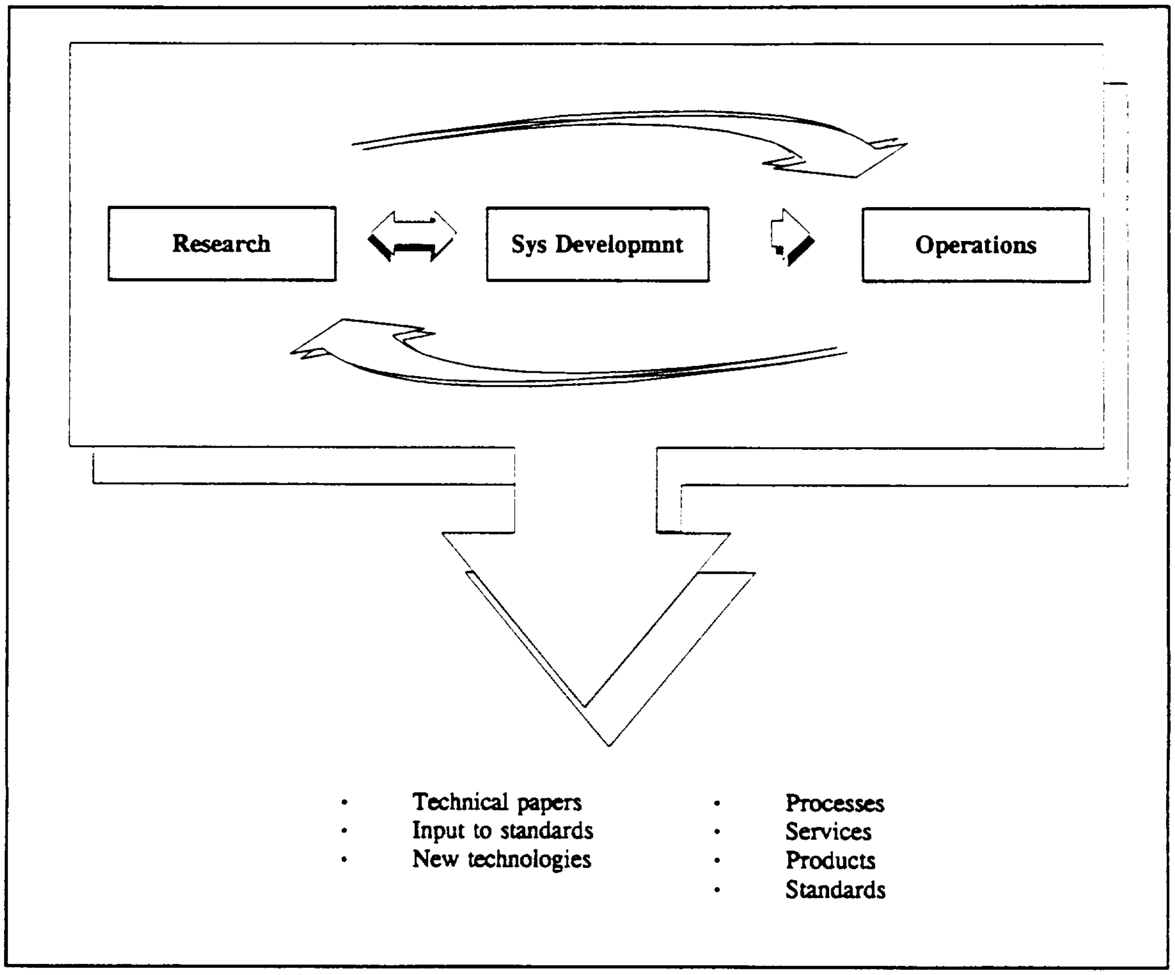
[7] McGarry, Frank and Waligora, Sharon, "Experiments in Software Engineering Technology: Recent Studies in the SEL (1990-1991), <u>Proceedings of the Sixteenth Annual NASA/Goddard Software Engineering Workshop</u>, December 4-5, 1991.

[8] Jost, J., "The Successful Introduction of Software Reuse Across an Entity," <u>Proceedings of the Sixteenth Annual NASA/Goddard Software Engineering Workshop</u>, December 4-5, 1991.

3.3 Program Description

The RBSE Program has three primary components, which are coordinated to deliver maximum benefit to AdaNET customers: (i) operation of a national center in West Virginia for the collection and dissemination of software engineering assets and information, (ii) system development, responsible for building and enhancing the system that supports repository operations, and (iii) research into software reuse technology and information access and distribution (see Figure 1).

Figure 1 -- RBSE Process and Products



3.3.1 Research Activities

In order for reuse to be adopted by mainstream software producers and maintainers, theories and practices must evolve to support software reuse. Research can provide a sound and proven theoretical foundation for solid engineering. Perhaps equally important, research can stimulate innovation, which leaps over entrenched barriers to reuse and, in some cases, quickly enables widespread adoption of new science and technology.

Today, the technology and practice that support proper software reuse is hard to grasp and even harder to standardize. Similarly, fifteen years ago, computers were completely intimidating to all but a few. Then, VisiCalc released the first version of the computerized spreadsheet.[9] The spreadsheet (while not funded as research) made computer technology accessible and useful to millions. Spreadsheets exemplify enabling technology: They make the raw power of computers available through an experience model that is very familiar. In particular, the user knows intuitively how to get the results he or she wants.

The creation of a model of software development, a process model based on the reuse of components from each phase of the life cycle, will provide the framework in which theory and practice can evolve. Within each phase of the life cycle, the requirements, architectural framework and design decisions will be captured in a clear, unambiguous form. Issues of representation are intimately tied to the model of development, nature of the phases and context for development products. There is currently no clear candidate for a general model that everyone accepts as a solution, nor are there techniques of representation, which must accompany the model to facilitate clear and precise communication.

RBSE was one of the first to enunciate and develop the concept of life-cycle, repository-based software engineering. RBSE researchers proposed to develop a generalized model of the software life-cycle in 1988. The model (see Appendix B, Generalized Model of Computer Systems Life Cycle) and its evolution is intended to be consistent with and supportive of the four broad objectives of the process modeling work of the Software Engineering Institute:

1. Increase understanding regarding a process.
2. Support evolutionary improvements to a process.
3. Enable processes to be formally defined and applied prescriptively.
4. Facilitate effective management of a process.

RBSE research will identify, develop and evolve models that express the repository-based life cycle and techniques of access and representation that support those models. RBSE will use semantic models, which operate according to precisely defined rules that can be understood by humans and processed by computers.

The RBSE approach will organize software engineering assets within a model of the software life-cycle that provides users with familiar and intuitive methods to find items of interest. After identifying items of interest, RBSE will permit users to move to related information by traversing reference links associated with the life-cycle model. For example, a user may identify a software concept for development of a telemetry system, then traverse links to find requirements for the satellite downlink acquisition subsystem. Further traverse could lead to specifications for signal filter components, software code and test data. RBSE's use of a general model of the software engineering life cycle facilitates access and retrieval of assets, and it promotes adaptation of the repository to new products, methods, interface standards or other

[9] Willmott, Thomas H., "Software Solutions for the IBM PC," Prentice-Hall, 1983.

technology. Models are critical for efficient, effective and sustainable research and development.

In addition to developing the life-cycle process model, RBSE research will integrate and apply off-the-shelf products and outside research to increase the accessibility and impact of the AdaNET repository. For example, RBSE research is currently working to apply the human factors engineering "Principle of Least Astonishment" to make simple, cost-effective improvements to the AdaNET interface. Simply stated: "What you expect to happen should happen." Applying this principle should aid in identifying interface issues that seriously discourage use and are relatively easy to fix.

To realize its full potential, the research component cooperates with other research organizations. This cooperation includes technology exchange so that advancements made elsewhere can be integrated into the life-cycle repository. RBSE research produces papers on topics of interest to the software engineering/reuse community at large, provides input to relevant standards groups and acquires innovative technology and practical solutions. The most important input for research, however, comes from analysis of customer needs and system usage. With this input, RBSE research pragmatically supports AdaNET development and evolution.

Research proposals and results are evaluated against the following criteria:

- Benefit to repository operations and system development.
- Impact on customers' ability to benefit from repository products.
- Contributions to and benefits from collaborative efforts that directly affect AdaNET and its customer base.

RBSE research is working to create a reuse library that is as user-friendly as an automated spreadsheet. Just as spreadsheet software put a PC on every middle manager's desk, so too can innovative solutions put repository components into every software engineer's computer toolkit.

3.3.2 Systems Development

The current system, AdaNET Service Version 2.0 (ASV2), is an automated computer-based repository with Internet and dial-in communications access. While it contains both source code and related life-cycle documents, the system does not make explicit the links between specific assets. For example, an ASV2 user cannot intuitively and logically traverse from a first-stage software concept to later-stage components such as software specifications and source code.

Successors to ASV2 will show relationships among repository assets and enable users to move easily from one life-cycle component to another. New versions will also facilitate user selection of assets that are not held in the RBSE repository by providing direct or indirect links to other repositories that contain the desired assets. System developers will incorporate user feedback and make the system more responsive to user needs.

RBSE's system developers integrate the repository host computers and communications network. They set up the system, install upgrades and provide sustaining engineering. Developers enhance repository efforts, incorporating the inputs of research and operations as appropriate. RBSE developers also provide feedback and input to the research team.

3.3.3 Operations

Operation of the RBSE repository comprises the following functions:

- Collecting, assessing, cataloging, and maintaining general schema (general organization, keywords, thesaurus, etc.) associated with repository holdings maintained for the user community.

- Maintaining user support services, including a help desk to assist users in connecting to the system and accessing desired information.

- Analyzing system usage and customer needs. This provides critical feedback to the research component and helps guide incremental evolution of RBSE.

- Establishing and maintaining procedures and policies to govern user access and authority, data rights, warranties and liability, interlibrary sharing, configuration and change control management, and system operations and maintenance.

Operations receives the inputs of both the research and systems development functions, and it supplies information to the research team about customer response. RBSE operations already has put in place a service organization that is well equipped to respond to customer requests. As the program sharpens its focus on serving selected customers with specific expectations for products and services, operations will refine the process by which assets and related products are identified for acquisition and system features are considered for incorporation into research plans and development.

3.3.4 Interoperability

RBSE plays a leading role in the Reuse Library Interoperability Group (RIG), a broadly based organization that is dedicated to enabling interoperability among reuse libraries through standardization of appropriate software component classifications, communication and transfer protocols and related mechanisms, and interoperation policies. Its members represent a diverse set of government and industry software engineering reuse libraries. The multiplicity of these programs, when balanced with cooperation, is critical while repository-based software engineering matures in technology and practice.

Within the framework of evolving consensual standards, the RIG allows each of the individual reuse libraries to customize its technology and services to the needs of its customers. Each repository, in developing creative approaches to practical problems, benefits from the advances and inventiveness of other repositories through the RIG.

The RIG also promotes broader, more comprehensive repository-based software engineering by facilitating the exchange of assets among heterogeneous repositories. This increases the impact of RBSE while saving time and money. Interoperability holds immense promise for promoting reuse, extending the capabilities and collections of individual software libraries and expanding the reuse library customer base in general.

3.3.5 Total Quality Management

No matter how sophisticated the repository becomes, team members' commitment to serving customers will mean the difference between major organizations' mainstream adoption of RBSE technology and their occasional use. Similarly, a program-wide commitment to quality is key to success in any and all Program objectives and goals. This commitment is evidenced by the extent to which the Program:

- Is customer-driven, providing customers with what they expect and need.
- Focuses on efficiency, i.e., providing products and services at a minimum cost while ever more effectively increasing bottom-line benefits to target customers.
- Measures its impact using well-defined criteria.

Similarly, while research innovations make technology more accessible, they cannot in and of themselves ensure quality. In particular, customer confidence in the integrity and quality of RBSE products is critical. As stated by the JLC panel, "widespread lack of confidence in the quality and suitability of assets available for reuse" is a significant barrier to reuse.[10] Quality will engender mainstream acceptance of RBSE technology through its products and services.

DOD defines total quality management (TQM) as "the disciplined process of continuous improvement in performance at every level and at all levels of responsibility.... Improved performance is directed toward goals assigned to cost, schedule, mission need, and operational suitability. <u>Increased user satisfaction is the paramount objective</u>...."[11] [emphasis added]

The very point of TQM is to reduce the cost of better service, develop a loyal customer base and build and provide products that get used. TQM calls for meaningful measurement of customer satisfaction, monitoring those measures and clearly stated goals for improvement. All these elements should be incorporated into RBSE's Program Management Plan. The development of these quality metrics and goals will be developed by all the members of the project team and adopted by the group as a whole.

[10] "Report of the Joint Logistics Commanders' San Antonio I Software Reusability Panel," January 28 - February 1, 1991.

[11] Memorandum from the Under Secretary of Defense entitled "Total Quality Management (TQM) in Acquisition and the Transition from Development to Production," January 12, 1989.

3.3.6 Cost Recovery

DOD is actively considering fee-for-service arrangements in the operation of its own repository-based programs. Many legal and technical issues related to cost recovery in reuse operations remain unresolved, such as licensing considerations or other incentives for sharing reusable software components and related products. Consequently, it makes sense to hold off on formulation of a fee-for-service plan and evaluate DOD's plan when it is developed. As the underlying issues are resolved, RBSE program management will reconsider institution of cost recovery methods.

3.4 Policies

The RBSE Program is an integral part of NASA's Technology Utilization program. It derives its authority from NASA's founding act, the National Aeronautics and Space Act of 1958 (as amended). The basic responsibility for a continuing program in the transfer of technology is stated in Section 203(a)(3):

> The Administration in order to carry out other purposes of this Act, shall-- (3) provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof.

The most recent version of NASA's procurement regulations clearly states NASA's policy toward technology transfer:

> The objectives of NASA policy...are to obtain the prompt reporting of inventions, discoveries, improvements, and innovations made in the performance of any work thereunder (whether or not patentable) in order to protect the Government's interest therein and to provide the widest practicable and appropriate dissemination, early utilization, expeditious development, and continued availability thereof for the benefit of the scientific, industrial, and commercial communities and the general public.

NASA Management Instruction 2210.2B establishes NASA policy and procedures regarding the distribution of computer programs developed by NASA. Currently, the Computer Software and Management and Information Center (COSMIC), operated by the University of Georgia and under contract to NASA's Technology Utilization Division, is directed to manage this distribution. However, NASA management is considering revision of the NMI to distinguish the role RBSE plays in the distribution of software assets.

The procedures for submission of life-cycle assets and policies for acceptance of them must be designed to encourage both active and passive participation. Active submitters knowingly contribute information to the life-cycle repository, while passive submitters provide public information, which operations staff members capture and incorporate into the repository. All information submitted for inclusion in the repository, from conference announcements to software source code, undergoes qualification and review as a precondition to acceptance.

Current policies allow the life-cycle repository to handle unclassified software and information with unrestricted distribution within the United States. Several legal and security issues require resolution if the repository is to handle information that is classified, restricted, licensed, copyrighted or falls under the Arms Export Control Act (Title 22, U.S.C. 2751 et seq), Executive Order 12470, or Department of Defense (DOD) Directive 5230.25.

An RBSE Technical Advisory Committee, composed of members from agencies and organizations heavily involved in software development, will provide further guidance toward the development of RBSE program policies. The formulation of official RBSE policies will coincide with the development of user requirements.

Acquisition issues pose some of the greatest perceived and actual barriers to reuse. To address the issue of replacing acquisition barriers with incentives to reuse, RBSE has been working with groups within the federal reuse program arena to provide action-oriented proposals to senior managers. A prime example of these efforts is the Reuse Acquisition Action Team, which is sponsored by the ACM SigAda Reuse Working Group. By March 1992, the group defined key reuse inhibitors within the Defense Department, recommended actions to remove those inhibitors, identified action agents with ability and authority to take action, and appointed group members to approach those agents. Proposals to remove similar barriers in existing NASA acquisition policy is a logical next step.

4.0 USER DEFINITION

Users of the repository can be described in two ways: (i) by user category, or type, which differentiates users by the functional roles they play in searching, populating or maintaining the repository, and (ii) by target customer groups, which distinguishes users by their work domains. RBSE has targeted customer groups it is best positioned to serve and will determine how to meet the specific needs of those groups.

4.1 User Categories

The RBSE library will be available to three broad categories of users. They are reusers, submitters and repository staff.

- Reusers include software engineers, system architects, managers, educators, authors and conference organizers. The reuser locates and retrieves components that can be used and/or adapted for use in the development of a new software engineering system (Section 6.1, Reuse of Repository Contents).

- Submitters include government and commercial software developers, authors, conference organizers, and marketers. The submitter provides the library a set of components for qualification and admission into the repository (Section 6.2, Submission to the Repository).

- Repository staff includes the RBSE operations and support staff. Staff will support customers, and they will acquire, classify, qualify and maintain the repository contents and organization (Section 6.3, Repository Maintenance).

4.2 Target Customers

RBSE targets three overlapping customer groups as both sources of repository products and as reusers of these products. RBSE's target customers are (i) NASA and civilian aerospace software engineers, (ii) builders of software-intensive, mission- and safety-critical systems and (iii) educational organizations interested in reuse.

RBSE is uniquely positioned to become the supplier of choice for NASA-developed software assets. Because of its focus on promoting software practices to produce more reliable systems, RBSE also can serve a wider community of users that build mission- and safety-critical systems.

Although DOD builds many systems that require extremely high software reliability and integrity, they are not the only group concerned with these requirements. Customers needing this degree of reliability and integrity may be found in many industries, including manufacturing, railroads, medical equipment, nuclear engineering and hazardous material management, as well as aviation. All these potential customers need to create flexible, reliable, maintainable, quality software in order to make their systems as safe as possible.

Educational institutions that are interested in reuse are a potentially large and high-leverage customer group. The ubiquity of the UNIX operating system attests to the sensibility of getting RBSE technology into the hands of teachers and students. (AT&T distributed UNIX source code free to universities.)

While RBSE focuses on specific customer groups, the repository continues as a public access facility. As a result, customers may apply RBSE products to unforeseen and innovative applications. Similarly, the pull of customer requirements will lead RBSE and its operational AdaNET repository to fill additional unique, high-value market niches for software reuse technology.

5.0 CAPABILITIES AND CHARACTERISTICS

RBSE will replace the current ASV2 prototype repository, first by rehosting the system to a more powerful open systems environment, then with a fully capable life-cycle repository over the next two to three years. Capabilities described here assume that automated support is available so these services can be provided efficiently.

5.1 Capabilities

AdaNET capabilities focus on the services and data required by the three types of users. They are reusers, submitter/suppliers and RBSE staff.

5.1.1 Reuser

AdaNET will help reusers to find and acquire the assets they need efficiently. AdaNET will provide flexible search capabilities and rich asset descriptions that let the user quickly narrow his or her set of potential acquisitions. It will then allow the reuser to browse the assets themselves, as well as related files. These files may contain test logs, bug reports, user feedback, lessons-learned reports or the number of copies in current use.

If required assets are not present, AdaNET will log reusers' requirements and identify potential outside sources. If a candidate asset is found, AdaNET may, under a cooperative agreement with the outside library, provide the reuser with suitable asset descriptions. Alternately, through the Software Referral Service, AdaNET may refer the customer to the appropriate outside source.

Once assets are selected, AdaNET will enable reusers to acquire the assets and corresponding support (from the supplier) they need. AdaNET and the reuser will conclude appropriate agreements with the reuser based upon the restrictions placed upon the asset, e.g. royalty, license fee, proprietary restrictions. The reuser may wish to acquire the asset as-is or with ongoing support (if available).

AdaNET will build customer confidence in the integrity of its asset descriptions and the consistency of its products. AdaNET will enforce a number of processes that ensure that when customers view asset descriptions, "what they see is what they get," for example, ensuring that:

- Assets are not modified by any person or at any time other than what is specified during storage or transmission.

- Assets are classified in appropriate categories and that these categories are useful to the selection of assets.

- Attributes, such as certification level, are consistent with certification criteria.

For example, builders of safety-critical systems require configuration management and security from a repository. These and most other system builders also need to know that the software was not tampered with; consequently, they require assurance of the software's pedigree.

Providing these capabilities involves human and automated qualification and classification of assets as well as follow-up to ensure that the assets lived up to their descriptions. (More detail is provided under Section 5.1.3, RBSE Staff.)

After reusers acquire assets, AdaNET builds upon the customer relationship. AdaNET solicits customer feedback and provides this feedback to asset sources and other reusers. For example, when serious bugs are reported for an asset, AdaNET notifies customers who have acquired that asset as well as the supplying library or developer. AdaNET also makes this feedback available to potential users and modifies the asset's certification level.

5.1.2 Submitter/Supplier

AdaNET makes acquisition efficient for itself and its suppliers. AdaNET publicizes its unique niche and current requirements. This enables potential suppliers to better qualify AdaNET as a potential customer. It also makes it easier for AdaNET to find the assets it requires and reduces the need to sift through poorly written or inappropriate materials.

Submitters may convey assets and asset descriptions to AdaNET electronically through file transfer protocol (FTP) or on selected magnetic media. AdaNET will evaluate assets and notify suppliers of acceptance or rejection. Once assets are accepted, AdaNET and submitters will agree upon terms such as whether the asset description or the entire asset will be accessible to AdaNET users and under what conditions.

AdaNET will team with suppliers to ensure that customers have the highest possible regard for suppliers' products and services. AdaNET will immediately notify suppliers of reported bugs, discrepancies, bug fixes and change requests.

5.1.3 RBSE Staff

AdaNET automates tedious and complex repository maintenance and management tasks, and it enforces access control and security procedures required by RBSE. For example, AdaNET --

- Protects against viruses.
- Expedites the process of evaluating, qualifying and classifying assets.
- Monitors asset distribution to support possible tracking and billing of royalties and license fees, and customer follow-up.
- Notifies customers when serious bugs are reported or upgrades are available for assets they have acquired.

- Maintains customer feedback logs that can be sorted by asset, author, supplier or customer.

- Monitors requests for assets that are not currently present in system.

- Maintains information on assets that exist in cooperating library systems.

- Assists staff in locating assets from cooperating library systems for customers.

In summary, the life-cycle repository will contain a set of browsing, cataloging, reporting, retrieval and configuration management tools to act on the organizational schema and the reusable objects in the repository. The repository collection may include objects that are referenced in the catalog but are stored remotely or are archived off-line. For example, the catalog may list commercial software and restricted information, although complete publications, source code, or documentation may not be available on-line. In addition, an intuitive graphical user interface will be evolved to enhance user power in searching and browsing. Figure 5-1 shows these capabilities by the three user categories outlined in Section 4.1.

On-line, context-sensitive help and a telephone help desk will be available to further assist the repository user. Electronic mail enhances communication between customers and the RBSE team. Figure 5-2 below summarizes the expected use of the repository resources by the three user types.

5.2 Characteristics

The system is generally characterized by an open architecture that is responsive to customer needs.

5.2.1 Phased Implementation

The capabilities' supporting characteristics stated here are those envisioned in 1992. They are based upon several years of practical experience with the ASV2 prototype, combined with industry experts' ideas about what public reuse libraries should provide and reports from successful reuse libraries within organizations. These characteristics will serve to establish the phased implementation of a next-generation version of the repository over the next two to three years. As software reuse and its commercial support matures, AdaNET will adapt its capabilities and characteristics to meet customer demand.

Capability	Reuser	Submitter	RBSE Staff
artifact search	•	•	•
catalog search	•	•	•
artifact browser	•	•	•
catalog browser	•	•	•
catalog and index management	•		•
standards conformance analysis	•	•	•
data format conversion	•	•	•
interlibrary import and export	•	•	•
file transfer	•	•	•
circulation monitoring			•
electronic mail	•	•	•
bulletin board	•	•	•
online help	•	•	•
reference to other technology transfer systems	•	•	•
artifact traceability analysis	•	•	•
artifact traceability maintenance			•
artifact cataloging			•
artifact classifications	•	•	•
classification schema creation and modification			•
artifact qualifications	•	•	•
qualification schema creation & modification			•
version control			•
utilization monitoring and reporting			•
security and access control	•	•	•
user management			•
graphical user interface	•	•	•
command line interface	•	•	•

Table 5-1 Life cycle Repository Capabilities and Users

Resources	Reuser	Submitter	RBSE Staff
reusable software life cycle products	•		
reusable software life cycle processes	•		
reusable software life cycle interfaces	•		
guidelines for contributors	•	•	•
software engineering guidelines	•	•	•
electronic forums	•	•	•
postings of software engineering news, conferences, and meeting announcements	•	•	•
commercial tool and training artifact references	•		
schema design for managing information	•	•	•
software engineering help desk	•	•	•
software engineering policies	•	•	•
software engineering practices	•	•	•
software engineering standards	•	•	•
life cycle repository concepts	•	•	•
life cycle repository principles	•	•	•
life cycle repository models	•	•	•

Table 5-2 Life cycle Repository Resources and Users

RBSE will replace the current ASV2 prototype with a production-quality library management system. ASV2 is based upon an outdated Data General MV 8000 platform and the AOS/VS II operating system. Current search capabilities are limited to those provided by the operating system. The outdated architecture severely limits system response. For example, if one user initiates a comprehensive search, the system virtually stops until the search is complete. Four to six simultaneous casual users are about all the current system can accommodate.

The MV8000 will be replaced with a modern high-performance platform that supports open systems such as UNIX or POSIX. The current Data General CEO file management system will be replaced by the UNIX/POSIX-based NASA Electronic Library System (NELS) Version 1.2, a library management system developed by NASA. The replacement system is called AdaNET Service Version 3 (ASV3) and will serve as an interim platform while a fully capable software engineering life-cycle repository, AdaNET Service Version 4 (ASV4), is developed and deployed. ASV3 is expected to provide acceptable response when loaded with five active users and two active librarians.

Enhancements in the repository's telecommunications equipment and networks are planned to provide the necessary throughput as usage increases. As the size of the collection and the number of users increases, the system will be scaled up to maintain acceptable performance.

Because AdaNET is derived from a deployed system, implementation will be phased to provide an appropriate level of continuity through system upgrade and replacement. Functional and technical requirements are reviewed and agreed upon by operations to ensure a smooth transition.

5.2.2 Architecture

ASV3 and ASV4 are characterized by an open-systems computer and communications architecture. AdaNET's platform will be UNIX/POSIX-compliant. Users may access AdaNET through SprintNet (a commercial data network), Internet (the global telecommunications backbone) or the NASA Program Support Communications Network (PSCN). PSCN provides high-speed access to AdaNET customers at Johnson Space Center, other NASA field centers and laboratories and aerospace contractors with PSCN links.

Most users are expected to access the life-cycle repository using an engineering workstation or a personal computer, a modem and common terminal emulation software. Co-processing user interfaces may provide additional capability for network and remote users. With co-processing, the user's computer would provide most of the interface processing and display functions.

5.2.3 Customer Focus

AdaNET will tailor products and services to the needs of its target customers. To determine the needs of its target customer groups, RBSE will first identify key organizations and associations to which those customers belong. RBSE program participants will attend meetings of these groups to determine issues and capabilities that are critical to its members. Then these critical issues and capabilities can be translated into specific needs that RBSE can address. One of the key organizations for the avionics industry, for instance, is the Airlines Electronics Engineering Committee. AEEC members have stressed issues related to FAA certification requirements and conformance to ARINC guidance documents (de facto standards). To serve these customers, products that support their domain-specific focus will be acquired for the repository.

6.0 OPERATIONAL SCENARIOS

Three operational scenarios for the future life-cycle repository illustrate the user's interaction with the repository. The scenarios do not imply specific requirements. In the first scenario, a reuser locates and retrieves components that will be reused and adapted for a new system development. In the second scenario, a user submits a set of components for qualification and admission to the life-cycle repository. In the third scenario, a repository staff member modifies the classification of repository assets.

6.1 Reuse of Repository Contents

In this scenario, a repository user seeks software components that will help construct a graphics import and export function. The graphics export-import functions are part of a larger system that provides text and graphics processing functions. The user would like to obtain requirements, design information, code and test procedures to be used to create the export-import functions of the new system. The user has already identified many of the characteristics of the desired components.

The user logs on to the life-cycle repository via Internet. The user has a workstation with a graphical user interface, perhaps a Macintosh or an X-Windows system on a Unix machine. The user selects the cooperative processing interface to take advantage of the AdaNET graphical interface capability. This allows simple, direct browsing of repository semantic structures.

Of the various searching methods provided by the life-cycle repository, the user elects to use faceted classification augmented with a lattice structure. From prior analysis, the user has identified several facet values that characterize the desired components. The search element of the repository obtains the characterizations from the user. The user has provided the characterizations of the code components for graphics import-export. The search mechanism identifies several candidates and presents them as a set of nodes in the lattice on the user's workstation screen. The primary nodes are the code components identified by the search, and secondary nodes connected to the primary nodes are other repository assets that have traceability relationships with the primary nodes. The secondary nodes in this example are design information, requirements statements, test procedures and the standards, methods, tools and environment used for the export-import code components. Using the graphical browsing functions, the user examines the various sets of related components. Information windows for each component display the component characterizations and their relationships to other components. During the examination process, the user selects components that will be transmitted at the end of the session. Now, at the end of the search and retrieval session, the repository transmits the export-import packages, documents, designs and test procedures to the user using FTP over the Internet.

During this process the life-cycle repository has collected metrics on the use of the search methods and their results. The AdaNET system tallies the download counts of the components automatically and generates usage reports on a regular basis. These usage reports assist the RBSE team in the management of the repository and in their customer accounting.

A user involved in requirements definition may search the repository collection for a document with similar requirements. Once found, the user should be able to quickly find related documents such as the concept document, the design document and the test results; references to the methods employed in development or management; references to the training methods employed by the project team; and descriptions of the computer-aided software engineering (CASE) tools that were a part of the development environment.

For example, an individual interested in exploring the merits of a new method for the detailed design of software components may access the repository to examine a list of lessons learned by other users of the method. As a repository user, this individual may browse the inputs used and outputs produced by the method on a project, learn what automated tools were used or examine the interface and interactions with the other methods and standards used on that project.

6.2 Submission to the Repository

A submitter has identified a set of life-cycle products (candidate reuse components) consisting of the following:

1) Software requirements in document form,

2) Ada designs (represented by Buhr diagrams in this scenario),

3) Ada code (packages and programs),

4) Test data and programs for the code above,

5) Development methods, tools and standards used,

5) Traceability relationships among all these items.

We shall assume that the submitter is familiar with the submission standards, policies and procedures of the life-cycle repository and has ensured that the component set complies with those policies. Qualification of the components by the repository operations staff verifies compliance of the components with the submission standards, policies and procedures.

The submission standards identify the various data representation standards supported by the repository. These standards prescribe the format and data representation for the various classes of assets accepted by the life-cycle repository. For example, text documents such as the requirements that the submitter has prepared should be represented in a standard form such as Standard General Markup Language (SGML), Rich Text Format (RTF), or simple American Standard Code for Information Interchange (ASCII) text. Design information or graphical components may be stored using data representation standards such as Standard Entity Rendering Interchange Format (SERIF) or Computer Design Interchange Format (CDIF). Ada code should conform to style standards or be processed by a formatting program to obtain stylistic conformance. Additional meta information may be semantically modeled in an Information

Resource Dictionary System (IRDS). Ada code may have to exhibit certain metric values for qualification. The representation standards of the life-cycle repository ensure that components are understandable to later reusers.

In this scenario, the collection of components arrives at the repository by an electronic file transfer (File Transfer Protocol is likely for Internet) initiated by the submitter. The artifacts are treated as a single logical entity. The submission package contains information required by the repository operations staff to catalog and characterize the components correctly according to the various semantic models used for the repository.

The repository staff uses tools to assist in the qualification of the submission packages. Standards-checking tools are straightforward and determine that the components abide by the representation standards for their component class. The staff can correct minor problems identified in the components, for example, formatting Ada code to conform with style standards. Submission packages with major problems are not accepted, and the submitter is notified accordingly.

At this point, the components have met the qualification standards. The repository librarian now characterizes and catalogs the components. It is likely that several semantic models for the repository contents will be used. For each semantic model supported by the repository, analysis of the submitted components will yield the needed descriptive information. This information may be facet categories and terms, or component class and subclass categorizations, or other software organizations. Much of the information required for the repository semantic models will be produced automatically by analysis tools and programs.

The submitter supplies the traceability relationships among the components. The repository staff browses the relationships among the components to verify their correctness. Automated tools may be used for this, and such tools will also identify relationships to other repository elements. These qualified, related and characterized components are now published in the repository catalog and are ready for customer access and reuse.

6.3 Repository Maintenance

In this scenario, the repository staff modifies the faceted classification system to accommodate a new domain of components to be added to the repository. A major software project has notified the repository operations team that its life-cycle products will be submitted to the repository. In this example, the new components are elements of an extensive remote sensing imagery analysis system (e.g., EOSDIS).

The staff recognizes that new facets and terms must be added to the faceted classification system to accommodate the imagery analysis applications. However, other classification and characterization methods used by the repository require no particular adaptation for the new application domain.

The staff uses tools to create a preliminary analysis of the imagery components. The tools identify candidate facets and terms for the imagery domain. Other assets already in the repository are used as much as possible to assist in the analysis. The staff compares the candidate facets and terms with the existing set to determine which existing facets require new terms and to determine what new facets to create. Having identified the needed modifications to the classification scheme, these changes are installed in the system.

The staff then analyzes the imagery analysis components that have been submitted with the aid of classification and analysis tools. The components are characterized and loaded into the repository. As noted earlier in Scenario 1, the components are checked for conformance with the repository semantic representation standards. The components are characterized for each of the methods supported by the repository. Finally, the components are published to the general users of the life-cycle repository, becoming visible for search, retrieval and reuse.

With a significant new domain of life-cycle assets, operations staff design a strategy to target a new customer group in the area of environmental applications.

7.0 ABBREVIATIONS AND ACRONYMS

Ada IC	Ada Information Clearinghouse
ACM	Association for Computing Machinery
ADP	Automated Data Processing
AEEC	Airlines Electronics Engineering Committee
ARINC	Aeronautical Radio Inc.
ASCII	American Standard Code for Information Interchange
ASV2/3/4	AdaNET Service Version 2/3/4
CAMP	Common Ada Missiles Package
CASE	Computer-Aided Software Engineering
CDIF	Computer Design Interchange Format
CDRL	Contract Data Requirements List
COSMIC	Computer Software Management Information Center
COTS	Commercial, Off-The-Shelf
DOD	Department of Defense
FAA	Federal Aviation Administration
FTP	File Transfer Protocol
JLC	Joint Logistics Command
JSC	Johnson Space Center
IEEE	Institute for Electrical and Electronics Engineers
IRDS	Information Resource Dictionary System
NASA	National Aeronautics and Space Administration
POSIX	Portable Operating System Interface for Unix
QFD	Quality Function Deployment
RAPID	Reusable Ada Packages for Information Systems Development
RBSE	Repository-Based Software Engineering Program
RICIS	Research Institute for Computing and Information Systems
RIG	Reuse Library Interoperability Group
RTF	Rich Text Format
SERIF	Standard Entity Rendering Interchange Format
SGML	Standard General Markup Language
SMAP	Software Management and Assurance Program
SPC	Statistical Process Control
STD	standard
TC	Task Committee
TQM	Total Quality Management
UHCL	University of Houston-Clear Lake

8.0 GLOSSARY

This glossary contains definitions produced by the Reuse Library Interoperability Group for terms relevant to the RBSE program and to the Concept.

Access Control	Security feature for managing access to assets; pertains to library mechanisms Security feature for managing access to subscriber services; pertains to operational reuse libraries
Adaptability	A measure of a library mechanism's capability to represent multiple data models, user defined data models and other user defined tailoring, and the ability to support multi-organization's policies and to incorporate new technology; pertains to library mechanisms
Asset	A reusable entity
Asset Certification	The process of validating an asset with respect to predefined criteria appropriate to that type of asset; pertains to operational reuse libraries
Asset Certification Level	Attribute of an asset that describes the degree of certification the asset has undergone; pertains to operational reuse libraries
Attribute	Characteristic of an asset; pertains to library mechanisms
Auditability	A measure of the library mechanism's capability to support the capture and analysis of usage statistics, behavior patterns, and other metrics; pertains to library mechanisms
Browse	Ability to navigate through a library and examine information about assets; pertains to library mechanisms
Catalog	The collection of asset descriptions that an asset library maintains about its assets; pertains to operational reuse libraries
Classification	The process of organizing assets according to a defined classification method; pertains to operational reuse libraries
Classification Methods	Defined methods of classifying assets supported by a library mechanism; pertains to library mechanisms (Faceted, Hypertext, Keyword, Class Object Hierarchy, Rule Based)

Classification Scheme	Organizational approach to utilization of classification methods; pertains to operational reuse libraries (Architecture Based, Component Based)
Configuration Management	i.e. archiving, version control, etc.; pertains to operational reuse libraries
Component	See asset
Domain	A set of problems with similar requirements for which a corresponding set of similar software systems (a family of systems) might be developed. Domains have been characterized as application, horizontal or vertical, technology, computer science, execution, execution models, etc.; pertains to operational reuse libraries
Domain Analysis	The process of identifying objects and operations of a class of similar systems in a particular problem domain; pertains to operational reuse libraries
Expandability	A measure of performance (cost and system response) relative to the number of assets in the operational reuse library and across distributed heterogeneous asset libraries; pertains to operational reuse libraries
Extensibility	A measure of the ability to modify and enhance the operational reuse library's data model and contents while minimizing interruption to subscribers; pertains to operational reuse libraries
Interoperability	A measure of the capability to perform common functions or processes across the boundaries created by the connections between homogeneous and heterogeneous asset libraries; pertains to library mechanisms
Library Data Model	The organizing principles and concepts underlying structured data in an operational reuse library and the means of representing that structure; pertains to operational reuse libraries
Library Mechanism	An automated tool that supports classification, search and retrieval of assets
Library Meta Model	A set of rules for building library data models
Library Metrics	Quantitative measures related to library operations

Notification	Informational message posted to users of assets regarding extracted assets; pertains to operational reuse libraries
Operability	A measure of the ease of learning versus ease of use of the library mechanism's capability to support searches, retrievals, extractions and contributions; pertains to library mechanisms
Operational Reuse Library	A collection of assets + a community of subscribers (even if restricted) + a set of operational policies and procedures + support services for subscribers.
Platform	The basic set of system resources on which the library mechanism executes (hardware, operating system and DBMS or object management system, proprietary and COTS software); pertains to the library mechanism
Query	Specification of a set of criteria used in searching for assets; pertains to library mechanism
Reliability	A measure of how well a mechanism's performance consistently matches its specifications; pertains to library mechanisms
Reusability	Characteristics of an asset that make it easily adaptable for use in different contexts
Reuse	The use of an existing asset in a new context, either elsewhere on the same system or in another system
Reuser	An individual or organization that reuses an asset
Search	Capability for locating assets within an operational reuse library; pertains to library mechanisms (for methods see Browse and Query)
Search Refinement	Capability to assist the user in refining/clarifying a search; pertains to library mechanisms (Conceptual Closeness; Saving query, query results or paths; Proximity)
Subscriber	User of the services of an operational reuse library; pertains to operational reuse libraries
Usability	A measure of the ease of use versus the effectiveness of subscriber services; pertains to operational reuse libraries

9.0 APPENDICES

Appendix A: The Software Crisis

The need to improve America's economic growth, productivity and competitiveness is becoming acute. One of the requirements for any future improvement will be significant advancements in the engineering and application of computer-based systems.

Unfortunately, today's computing systems are often brittle. In the face of increasing requirements to cope with complexity, systems are too often delivered late, over budget, with a compromised subset of the needed capabilities and with uncertainties about reliability (1, 2). When a computing system is deployed and operating, attempts to inexpensively and reliably modify the system to meet new or changing requirements are often unsuccessful. As a result, the life-cycle expenses (that is, expenses from the time the system was proposed until the time it is retired) incurred for initial development are often 10% to 15% of the total system cost while 85% to 90% of the life-cycle expenses are for operations and sustaining engineering.

These exorbitant post-deployment expenses are not a complete measure of the true costs. The true costs begin when early attempts to modify the brittle system first reveal to management the magnitude of the costs, difficulties and risks of making changes in response to changing needs and opportunities. The mission of the organization becomes both dependent upon and endangered by computing systems that decline rapidly from being an essential part of the solution to become an unadaptable, expensive part of the problem. Productivity, competitiveness and market share begin to erode as organizations in other countries build and sustain systems that are better engineered for life-cycle adaptability, reliability and cost-effectiveness.

Still another cause for concern is the growing need for systems that are critical to the safety of life, health, property and the environment. The use of such systems is increasing even though loss of life and the destruction of property have been traced to errors in computer system software (3). This questionable ability to cope with the complexity of safety-critical systems is posing a dilemma for American institutions. The fear of our inability to properly engineer such systems, combined with liability issues, is clearly restraining the growth rate of markets for such systems. While such restraints are often relevant, many systems could be made safer if proper methods for producing and sustaining computing systems were consistently applied.

Brittle, unsafe and inadequate computer-based systems point to the need to improve quality across the life cycle. The greatest impediment and risk in the improvement of quality is the absence of the integrated advancement of software engineering throughout the life cycle (4, 5, 6). Such integrated advancement requires a coordinated technical and managerial process. Partially in response to such needs, the Department of Defense initiated a Total Quality Management (TQM) program to infuse and coordinate process improvement in all its activities (7). In February 1990, the DoD Software Master Plan focused this TQM commitment on the advancement of software engineering within the DOD and among its contractors and vendors (8). In July 1990, NASA's Johnson Space Center, presented its plans to infuse and coordinate TQM in software engineering (9).

Major improvements have been made in the past decade in those portions of the software engineering life cycle that deal with programming languages. In 1983, the Ada programming language was adopted as a national standard in reaction to a proliferation of languages that did not support the application of modern software engineering principles. To leverage the improvements made possible by well-engineered Ada source code components, several government-sponsored projects were established to encourage component reusability. The Army Ada Software Repository was established as an electronic network node for the collection and dissemination of voluntarily or contractually contributed components and tools. As the use of Ada spread, other more domain-specific projects were initiated to improve the quality, availability, classification and organization of source code artifacts within particular application domains. The Reusable Ada Packages for Information Systems Development (RAPID) project is an example for the information systems domain, and the Common Ada Missiles Package (CAMP) project is an example for the domain of missile control systems (10,11).

Although such projects are facilitating improvements in source code quality and availability for reuse, the impact upon the overall life cycle is small (5, 6, 12). For computer-based systems and applications, both the DOD standard 2167A and the NASA Software Management and Assurance Program (SMAP) standard accommodate eight major process divisions of the life cycle. These standards define the expected products and guidelines for each of the eight major software system development processes (13, 14). Figure A-1 below illustrates the processes defined by SMAP and 2167A.

Source code implementation is the sixth of these processes and typically represents 7% of the cost of developing software. Testing and integration of this source code is the seventh of these processes and typically accounts for 13% of these development costs.

Since the initial software development costs are often only 10% of the total costs of the full software life cycle, the combined impact of achieving 100% reusability of source code on a new project would affect only 20% of the development costs and, potentially, only 2% of the life-cycle costs. In addition, there are costs associated with designing the component to be reused and with classifying, storing, finding and reusing it. Yet, total reusability is unrealistic. It would require that no new components be produced and that no modifications to existing components be made. Nearly every instance of reuse involves modification and enhancement of the reused code. Code reusability alone cannot address the problems of the software crisis.

Thus, the solution to the "software crisis" demands that more than just source code components be made available for reuse. The life cycle of a complex computing system depends on the quality of, and interfaces among, many software engineering processes and products. The long-term solution requires the development of sufficiently general system and software life cycle, process and information models that are based on strong theoretical foundations (4). Such generalized models would greatly facilitate codifying and disseminating software engineering knowledge. Other benefits would also be gained, such as the development, preservation and reuse of high-quality software engineering artifacts. In particular, emerging techniques of modeling software engineering processes offer enormous potential for the integrated advancement of software engineering and quality throughout the life cycle of critical computing applications

(4, 5, 15, 16, 17, 18). Appendix B discusses the basis for the development of generalized system and software life-cycle product, process, interface and information models and their application to the development and organization of the life-cycle repository.

Life Cycle Processes

P0:	Concept Exploration
P1:	Systems Requirements
P2:	Sw, Hw and Human I/F Requirements
P3:	Design Specification
P4:	Detailed Design
P5:	Development
P6:	Testing
	--P0 through P6 is 10% of the life cycle costs--
P7:	Deployment, Operation, Sustaining Engineering
	--P7 is 90% of the life cycle costs--

Figure A-1 Software Life Cycle Costs

Appendix B: Generalized Model of Computer Systems Life Cycle

The research component of the RBSE Program will concentrate on the development and implementation of generalized system and software life-cycle product, process, interface and information models and standards. As noted in the recommended actions resulting from the National Research Council's 1990 workshop on "Scaling Up: A Research Agenda for Software Engineering," the long-term solution requires the development of general models based upon strengthened theoretical foundations. The report notes that such general models will greatly facilitate codifying and disseminating software engineering knowledge and provide other benefits such as the development, preservation and reuse of high-quality software engineering artifacts. Several other studies have produced similar conclusions (4, 5, 15, 16). There is also a growing consensus on some of the most highly desirable capabilities for the automated support of creating and sustaining such models (17, 19, 20, 21).

A general software life-cycle model believed to support the life-cycle repository needs for precise modeling of interrelated product, process, interface and information artifacts was proposed in 1988 (5). The model relies on the concept of engineering for reusability to develop and sustain quality across the life cycle of computer-based systems and applications. As defined in this model, engineering for reusability is the process of creating and sustaining life-cycle products, processes and standard interfaces with quality worthy of reuse.

If a growing national interest can be focused upon improving quality through engineering for reuse, then many of the problems described earlier in Appendix A can be attacked through the development of a national life-cycle repository with associated information services. Organizations and projects needing timely access to high-quality, reusable products, processes and standard interfaces can contribute to the repository collection as well as obtain artifacts from it.

Because the repository collection will be organized and managed in accord with a generalized model, two major benefits are derived. The first benefit is that products, processes and interfaces are traceable across the life cycle. A user involved in requirements definition may search the repository collection for a document with similar requirements. Once found, the user should be able to quickly find related documents such as the concept document, the design document, the test results, references to the methods employed in development or management, references to the training methods employed by the project team and descriptions of the computer aided software engineering (CASE) tools that were a part of the development environment.

The second benefit of using a generalized model to organize and manage the collection of life-cycle products is the ability to promote localized improvements where they are most needed. For example, an organization needing to explore the merits of a new method for the detailed design of software components may access the life-cycle repository to examine a list of lessons learned by other users of the method, see the inputs used and outputs produced by the method on a project, learn what automated tools were used and with what effects and examine the interfaces and interactions with the other methods and standards used on that project.

One attribute of an appropriate generalized model is the emphasis on quality that permeates all processes throughout the software life cycle. Such a model should be flexible and precise in that it explicitly represents the application of the best software engineering knowledge. Such a model should also have explicit rules for expansion and modification as knowledge is advanced (5).

The model should have significant explanatory power with respect to the management of the importation, evaluation and reuse of high-quality engineering artifacts produced under other life-cycle models and methods. Such management will be context-sensitive since high-quality artifacts for one application domain might be completely inappropriate for another application domain. Therefore, the model must flexibly support the representation of a rich body of meta-information that characterizes and describes the life-cycle products, processes and interfaces in addition to support for the representation of the life-cycle products, processes and interfaces themselves. The model must also support the representation of projects that are classified as major modifications to existing systems as well as new systems.

REFERENCES

The following documents are referenced in Appendix A and B.

1. Wood, H. M., "Computer Society President's Message: Meeting the Technology Challenge," Computer, August 1990.

2. Brooks, Fred, "No Silver Bullet," Computer, April 1987.

3. Committee on Science, Space and Technology, U. S. House of Representatives, "Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation," 1989.

4. Computer Science and Technology Board, U. S. National Research Council, Workshop Report: Scaling Up: "A Research Agenda for Software Engineering," 1990.

5. McKay, C. W., "Conceptual and Implementation Models Which Support Reusability of Processes and Products in Computer Systems and Software Engineering," SERC/RICIS Report, AIRMICS™HTL Grant 2-5-51537, October 1988.

6. Biggerstaff, T. J., Richter, C., "Reusability Framework Assessment and Directions," Software Reusability, Volume 1: Concepts and Models, Biggerstaff, T. J. and Perlis, A. J. (eds), Addison Wesley, 1989.

7. DoD, Total Quality Management: Master Plan, U. S. Department of Defense, August 1988.

8. DoD, Department of Defense Software Master Plan, Preliminary Draft, U. S. Department of Defense, February 1990.

9. Sullivan, L. J., Presentation at Johnson Space Center on the "Johnson Space Center Team Excellence Initiative," July 31, 1990.

10. Rothrock, Jack, "Reusable Ada Products for Information Systems Development (RAPID) Reuse - Year 2000," Washington Ada Symposium Proceedings, June 1990.

11. McDonnell Douglas, "CAMP: Developing and Using Ada Parts in Real Time Embedded Systems," 1990.

12. Tracz, W. "Where Does Reuse Start?" Software Engineering Notes, Association for Computing Machinery, 1990.

13. DoD-STD-2167A Defense System Software Development, 29 February 1988.

14. Information System Life Cycle and Documentation Standards, Release 4.3, 2/28/90. NASA Office of Safety, Reliability, Maintainability, and Quality Assurance, Software Management and Assurance Program (SMAP), Washington, D.C.

15. Boehm, B., "What We Really Need Are Process Model Generators," Proceedings of the 11th International Conference on Software Engineering, IEEE, May 1989.

16. Humphrey, W. S., Kellner, M. I., "Software Process Modeling: Principles of Entity Process Models," Proceedings of the 11th International Conference on Software Engineering, IEEE, May 1989.

17. Kellner, M. I., "Software Process Modeling: Value and Experience," Technical Review 1989, Software Engineering Institute, 1990.

18. Humphrey, W. S., Managing the Software Process, Addison- Wesley, 1989.

19. Rombach, H. D., Mark, L., "Software Process and Product Specifications: A Basis for Generating Customized Software Engineering Information Bases," Proceedings of the 22nd Annual Hawaii International Conference on System Sciences, Vol II, IEEE, 1989.

20. A.C.M., Proceedings of the 4th International Software Process Workshop: Representing and Enacting the Software Process, ACM Software Engineering Notes 14, June 1989.

21. Riddle, W. E., "Session Summary-Opening Session", Proceedings of the 4th International Software Process Workshop: Representing and Enacting the Software Process, ACM Software Engineering Notes 14, June 1989.

ACKNOWLEDGEMENTS

This Concept Document is based on the foundation work of Dr. Charles McKay, as enunciated in prior versions of the Concept developed by Softech's Dave Auty and others. The revised Concept could not have been produced without the active participation of all the RBSE team members: MountainNet, GHG and Softech. In particular, we would like to thank Karen Fleming, Linda Braun, Dave Auty, Dave Henning and Professor Dave Eichmann of West Virginia University for their helpful comments and input. Members of Applied Expertise who contributed to the drafting of the document include Dave Dikel, Jim Wilson, Valerie Norville, George De Marco, Dr. David Preston and Ann Eustice Brandon.

Definitions from the draft Reuse Library Interoperability Group's Task Committee 1 (RIG TC1) glossary were particularly helpful in adding clarity and precision to this document. Linda Braun and Karen Fleming of MountainNet were instrumental in producing the RIG glossary.

Recognition must also be given to RBSE program manager, Dr. E.T. Dickerson, for ensuring that sufficient time and resources were allocated so a thorough, well-coordinated document could be produced.