SF 298 MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1996	Technical

4. TITLE AND SUBTITLE
Software Reuse

5. FUNDING NUMBERS
DAAH04-95-1-0250

6. AUTHOR(S)
Jacqueline Wall, Regina Ratcliff, Padma Reddy, and Y.B. Reddy

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
Grambling State University
Department of Math and Computer Science
Grambling, LA 71245

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 34157.43-MA-I 52

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Software re-use is defined as the process of creating software systems from existing software rather than building software from scratch. Re-use can occur:

- during maintenance, re-engineering, or in the implementation of new systems
- within a system, between systems, or between a system and a library of reusable components
- at the level of code components or abstract designs.

The properties of reuse and the role of the three R's (Re-engineering, Repository, and Re-use) are essential to software maintenance. We present the statistics of system maintenance, taxonomy of reuse and the sixteen questions about software resue.

14. SUBJECT TERMS	15. NUMBER IF PAGES
Software reuse, Re-engineering, Software maintenance	12
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Enclosure 1

Software Reuse

Jacqueline Wall, Ragina Ratcliff, padma Reddy, and Y.B.Reddy
Grambling State University
Dept. Of Math and Comp. Sci., LA 71245.

The software reuse is defined as the process of creating new software systems from existing software rather than building from scratch. Reuse can occur during the maintenance, re-engineering, or in the implementation of new system. Reuse can also occur within a system, between systems, or between a system and a library of reusable components. In this presentation we will discuss the properties of reuse and the role of the three R's in software maintenance. Then we will brief you on compatibility, errors, how to identify a component and give a further look on repository.

Note: This research is supported by Advanced Distributed Simulation Research Consortium and Office of Naval Research

70th Annual Meeting of the Louisiana Academy of Sciences,
Nicholls State University, Thibodeaux, LA, February 1-2, 1996.

Software Reuse

Jacqueline Wall
Regina Ratcliff
Padma Reddy
Advisor:- Y.B.Reddy

Grambling State University
Dept. of Math. and Computer Science
Grambling, LA 71245

EXAMPLES OF REUSABLE COMPONENTS

- Sort an array
- Solving a system of linear equations
- Data retrieval programs
- Hash tables for managing symbol tables

FACTORS TO KNOW TO REUSE COMPONENTS

- Exact specification
- Precise functionality
- Adaptibility of component
- Component must fit in existing code

REUSE OF ARCHITECTURE

Definition:
The way in which the various parts of a system hang together.

REUSE OF DESIGN (ACTIVITIES)

- Mapping domain
- Translation
- Simplify written text
- Procedures for domain specific algorithm

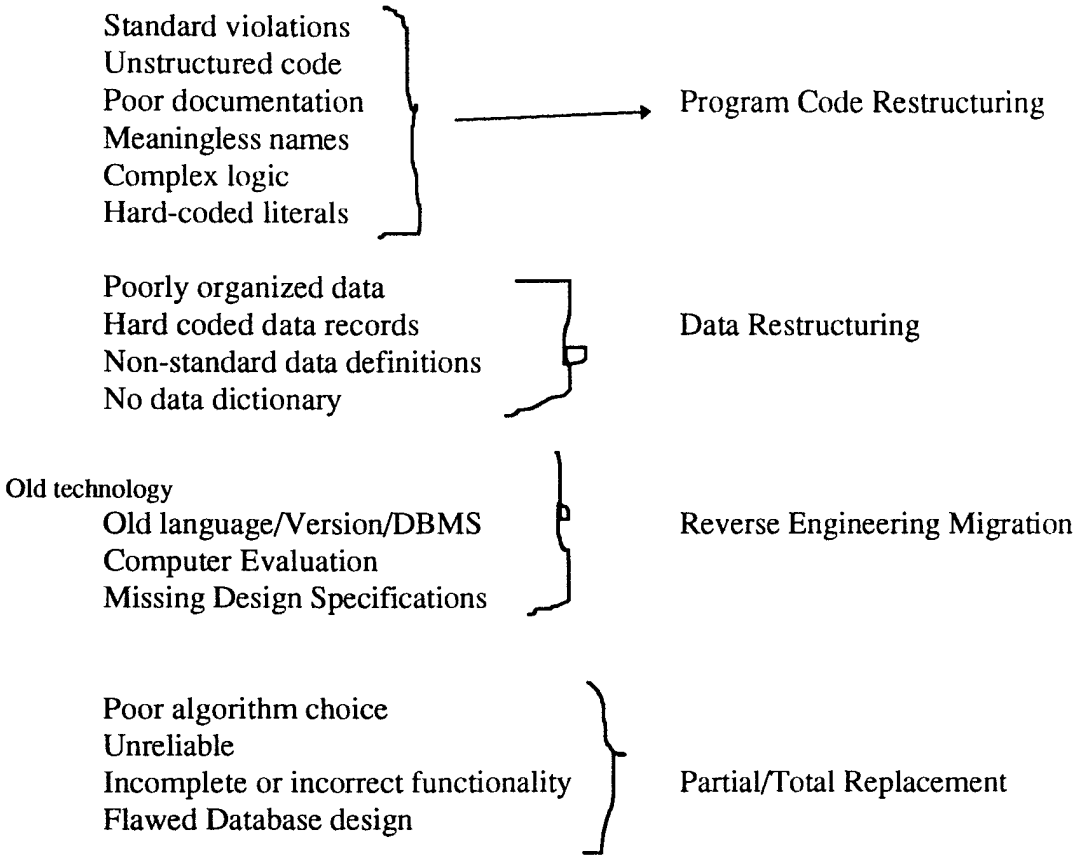
REUSABLE COMPONENT PROPERTIES

- Easy to understand with existing documentation
- Must be a completely tested code
- Must fit in existing code
- Requires no change

REQUIRED PROPERTIES OF REUSABLE COMPONENTS

- Length of component
- Complexity
- Test results
- Errors
- Quality of documentation
- Readability

CHARACTERISTICS TO LOOK FOR TO RE-ENGINEER



THREE R'S
(RE-ENGINEERING, REPOSITORY,RE-USABLITY)

REASONS TO RE-ENGINEER

- Frequent production failures
- Performance problem
- Outdated technology
- System integration problem
- Poor quality code

FRAGILE SYSTEMS LIKELY FOR RE-ENGINEERING

- Critical to the corporation
- Frequent maintenance
- Only understood by few members
- Contain bugs
- Require major enhancement

ERRORS

ERRORS IN REUSABLE CODE
- Library
- User
- System

ERRORS DETECTION

- Invariants
- Function pre-conditions
- Representation invariants

HANDLING ERRORS

- Library invariants
- Correct the problem
- Exit or Abort
- Return error value
- Create nil value

RESOURCE-LIMIT ERRORS

- Stack overflow
- Free-Store exhaustion

COMPATIBILITY

FORMS OF COMPATIBILITY
- Source compatible
- Link compatible
- Run compatible
- Process compatible

EXAMPLES OF COMPATIBLE PRACTICE

- Adding a member function
- Granting a friendship
- Loosening the protection of a member class

DOCUMENTING INCOMPATIBILITIES: every release of a library documented; all notes should be in one place in documentation

UNDOCUMENTED PROPERTIES: WHY WE RELY ON THEM

- The user may have to
- user may rely on undocumented property

Repository: A Further Look

ADVANTAGES

- Multiple Model Versions
- Multiple Architectures
- Multiple Time Management Approaches
- Technology Utilities
- Project Schedule Decoupling
- Data

TECHNICAL CHALLENGES

- Finding Modules
- Understand Module Implementations
- Incorporating Modules
- Building Systems
- Update Rate

ACTIVE REPOSITORY AGENTS

CONTROL MECHANISMS

COMPONENT IDENTIFICATION

Def:
sotware component: a container for expressing
abstraction of data structures and
algorithms

ATTRIBUTES THAT MAKE COMPONENTS REUSABLE

- Usefulness
- Costs (includes cost of extracting)
- Quality
  - correctness
  - readability
  - testability
  - performance
- Criteria

References

1. R.E.Johnson & B.Foote, Designing Reusable classes; Journal of object-oriented programming 1, 1 (1988) 22 - 35
2. Burton, R. Wienk Aragon, S.A.Bailey, K.D.Koehler & L.A. Mayes; The reusable Software Library, IEEE Software 4, 4 (1987), 25-33.
3. Tarumi, K. Agusa & Y. Ohno; A programming environment supporting reuse of object-oriented software; Proceedings 10th International Conference on Software Engineering, IEEE, 1988, pp 265-273
4. Y.S.Maarek, D.M.Berry&G.E.Kaiser; An information retrieval approach for automatically constructing software libraries IEEE Transactions on Software Engineering 17, 8 (1991) 800 -813.
5. Y.Matsumoto; A software factory: An overall approach to software production, 1987 [from Tutorial: Software Reusability, IEEE Catalog nr EZ750, 1987 edited by P.Freeman, pp 155-178.]
6. Carma McClure; The three R's of Software Automation: Re-engineering, Repository, Re-usability; Prentice Hall (1992), ISBN: 0-13-915240-7
7. Martin D. Carroll and Margret A. Ellis; Designing and Coding Reusable C++; Addison-Wesley Publishing Company (1995)
8. Charles W. Krueger; Software Reuse; ACM Computing Serveys, Vol. 24, No.2, June 1992
9. Ruben Prieto-Diaz; Status Report: Software Reusability; IEEE Software, May 1993
10. Deng Jyi Chen and P.J.Lee; On the study of Software Reuse Using Reusable C++ Components; Journal of Systems Software, 1993; 20: 19-36
11. William B. Franks and Christopher J. Fox; Sixteen Questions About Software Reuse; Communications of ACM, June 1995, page 75.
12. Boehm, B; Software Engineering Economics; Prentice Hall, Englewood Cliffs, N.J., 1981.
13. Frakes, W.B., and Fox, C. J.; Software Reuse Survey Report; Software Engineering Guild, Sterling, Va., 1993.
14. Frakes, W.B; Software Reuse as industrial experiment; AM PROGRAMMING. 6, 9 (1993) 27 - 33
15. Franks, W.B., and Isoda, S.; Success factors of Systematic reuse; IEEE softw. 11, 5, (May 1994), 15-19.