REUSABLE Ada PRODUCTS FOR INFORMATION SYSTEMS DEVELOPMENT (RAPID)
Reuse - Year 2000

Jack Rothrock

U.S. Army Information Systems Software Development Center - Washington
ATTN: ASQB-IWS-R STOP H-4
Fort Belvoir, VA. 22060-5456
Autovon 356-6202 Commercial (703) 285-9043

INTRODUCTION

The next century, out of necessity, will see a tremendous increase in software reuse. Fiscal and manpower constraints will require innovative ways to "do more with less." The advent of new software engineering methodologies combined with the Ada programming language has created an environment in which significant reuse appears achievable. Few have challenged the fact that there is significant productivity, quality, and cost savings potential associated with large scale reuse. However, whether or not large scale reuse is achievable remains controversial.

> Reduced DoD budgets will increase automation needs to reduce inefficiencies and personnel costs, thus creating further demands on software. Affordability will drive DoD toward common modular components, with flexible software support. Dod Software Master Plan (Preliminary Draft, 1990)

Software development organizations have been practicing forms of reuse for many years. Many of the world's leading technology companies, to include IBM [9], GTE [6], Toshiba [8], MITRE [2], and Raytheon [7] have published works on the virtues of reuse since the mid 1980's. In 1981, computer literature was already boasting significant productivity improvements with reuse [11]. The Software Productivity Consortium (SPC), established in Reston, Virginia, is focusing on making prototyping and reuse a routine part of the development and maintenance of complex, embedded software systems. Fourteen major United States aerospace companies currently financing the SPC are banking on significantly improved productivity, quality, reliability, and maintainability as a result of their efforts [13].

It is safe to say that reuse is not a new philosophy. Why, after all these years, has it been so difficult to make reuse an integral part of the DoD's software development efforts? The answer is complex and could easily be the subject of a future research paper. However, let's look forward to the next century at RAPID's vision of the future. What issues do we in the RAPID program view as our major challenges of today in order to meet the requirements of the future?

RAPID AND REUSE - YEAR 2000

This 21st century Systems Support Environment (SSE) will provide "cradle-to-grave" life cycle support for systems. "Systems" is emphasized because the integration of "systems engineering" rather than "software engineering" will become the norm. Recognition of the interrelationships between the hardware, software and people making up the system will become critical for successful design and development of tomorrow's systems. We will move away from the concept of developing software as only a collection of software parts.

An enormous amount of knowledge must be available to those developing systems in the future. Thus, the most significant attribute of the SSE will be the availability of knowledge that can be applied throughout the system's life cycle. SSE knowledge of past development efforts as well as knowledge of all current "assets" available for reuse will be coupled with an intelligent interface to the developer. This availability of knowledge will allow the developer the opportunity to accurately assess alternatives provided by the SSE throughout all phases of systems development. Prototyped alternatives can be generated automatically and evaluated for system performance characteristics as well as on a variety of other requested metrics. The availability of knowledge coupled with powerful automated prototyping capability will be used effectively to determine the user's true requirements as well as for negotiating design and development alternatives on the basis of cost, complexity, resource requirements, and time to develop. Based on the selected prototyped design, the SSE will generate standardized code, documentation, and test scripts. Actual reuse of life cycle products will eventually become transparent to the developer. All information from the requirements specification to the final user's manual is mapped together and therefore traceable.

There will be three significant changes in systems development philosophy in the next century:

1. Initial user requirements specifications will become short and concise. The SSE will evaluate the specifications and provide prototyped alternatives based on reusable knowledge. These alternatives will document resource and cost requirements, performance issues, and required development time. Detailed requirements will be automatically generated by the SSE.

2. Maintenance organizations will cease to exist as we know them today. System change (enhancement and modification) will be expected. Flexibility to accommodate change will be built into the system. Maintenance will consist of only repairing "latent" errors found in the system. All enhancements and modifications will be considered new development and be accomplished through the SSE. This will not only provide information on alternatives to the developer, but will provide complete traceability and automatic generation of changed life cycle products.

3. Documentation will be streamlined, machine readable, and nearly paperless. Due to the cost associated with the tonnage of "never-read" paper documentation, DoD will totally revamp their documentation requirements providing for automatic generation of concise documentation by SSEs.

This is obviously a very high level overview of a technically complex future system. Speculating on future systems is easy and exciting. The challenge remains that we must resolve significant management and technical issues to make the future become reality. RAPID is dealing with these issues to make reuse an integral part of systems development in the next century.

IS TECHNOLOGY THE ANSWER?

Technology has been at the forefront of the headlines and has gained tremendous ground in support of software development. Research, development, and insertion of new technology at the pace witnessed in recent years seems to spawn a feeding frenzy of the "power hungry" technologists. This "frenzy" is readily accepted and endorsed by management (as long as the "other guy" pays the R&D bill) who are eager to find new solutions to increasingly complex and difficult problems. Let's look at technology from four perspectives:

Environments

Significant work is being done by the Software Engineering Institute (SEI) on Ada Programming Support Environments (APSE) and in 1988 they published their "Perspective on Reuse" [12] which spawned new strategies and methodologies for software development. The Software Technology for Adaptable Reliable Systems (STARS) program is on the forefront of developing powerful Software Engineering Environments supporting the development of highly complex, real time embedded systems. RAPID is working with STARS to investigate methods to access the STARS national repository using tools such as RAPID. Both SEI and STARS integrate the concept of reuse throughout their environments.

The TRW Corporation, in development of the Army WWMCCS Information System (AWIS), has created a sophisticated programming support environment [15] where designs are conceptually modeled in graphic format. Compilable Ada Design language is automatically generated by the system to evaluate design alternatives through prototyping. Designing for reuse plays an important role throughout TRW's development effort.

These are but a few examples of the tremendous interest and technological advancement in programming support environments. This appears to be a natural evolutionary step toward the "systems" support environment of the future.

Engineering

"Engineering" is becoming common place and has been firmly attached to the word "software." This semantical association of software and engineering has not been defined clearly though everyone admits its importance. Classical "applied science" engineering methods such as modeling through abstraction and reuse is becoming common place. It has been stated that "reuse is not a goal, but a byproduct of classical engineering" [4]. RAPID continues to promote the advantages of "engineering" software with the resulting benefits of quality, reliability, and maintainability that engineering and reuse provide.

The sense of classical engineering we are beginning to apply to software development today will lead to classical systems engineering in the next century. Adherence to the rigor of engineering will propel the requirements for reusable knowledge to great heights. RAPID must continue to be evolutionary and dynamic to meet these future requirements. Certified reusable knowledge will be the key ingredient to successful systems development in the next century.

Tools

The draft DoD Software Master Plan has stated that "software tools are beginning to play a major role in the support of system design, development, test, evaluation, operations, and maintenance processes" [3]. This is no surprise. The lack of tools and efficient compilers supporting Ada development has been a significant problem for several years. Technology, however, is moving quickly to alleviate past shortcomings in this area.

RAPID has been involved in several in-depth reviews of a variety of CASE tools supporting software development and reuse. As a result, several tools are being enhanced to better support reuse. RAPID requires tools that allow detailed evaluation of graphical designs and source code. Since RAPID's primary sources of reusable components are existing systems, reverse engineering tools are also needed. RAPID is finding that the quality and capability of CASE tools is improving dramatically.
Based on the number of advertised tools, the level of interest at symposiums and expositions, and RAPID's own experience in evaluating a wide variety of tools, it appears that tool and compiler development may now be one of the fastest growing segments in the commercial Ada market.

Standards

There is a tremendous amount of work being put into the establishment of standards. Rightly so, because the establishment of standards is key and requisite to supporting distributed APSEs, accessing distributed reusable resources, and the concept of portability. Progress is being made to standardize areas such as: data dictionaries, Portable Operating System Interface for Computer Environments (POSIX), Government Open Systems Interconnect Profile (GOSIP), Ada Programming Support Environments (APSE), Common APSE Interface Set (CAIS), documentation (DoD Standard 2167A and 7935A), portability, Ada/SQL binding, and others.

RAPID has drafted a command-wide reuse policy that is currently being staffed. This is a major step to formalizing an organizational reuse program that incorporates specific reuse standards and procedures for software development.

The emergence of new technologies will continue to proliferate. The establishment of standards will allow integration of tools such as RAPID into systems support environments. This will also provide the capability to access distributed reusable resources which will provide power and flexibility for systems development in the future.

RAPID firmly believes that technology will provide the "high powered" tools necessary to support sophisticated systems development and reuse in the future. The question is ... will management be prepared?

MANAGEMENT'S ROLE

Management issues are rarely in the headlines and are normally found in the shadows of technology's limelight. Technology will give us the tools and environments to make true systems engineering a reality but the most critical resource is still the people who use those tools and environments to specify, design, code, test and integrate systems.

In the March 1990 issue of Communications of the ACM [14] an interesting "software crisis" analogy was presented based on the Chinese character for crisis. The character for crisis is composed of characters that make up both Chinese characters for danger and opportunity. The wisdom of this is ever present in the software development community. We are at a point of transition where management must assess a variety of risks, react appropriately to not only minimize the danger but to enhance our opportunities for success. Management actions based on the assessment of risk associated with training, methodologies, incentives, and legal issues will drive the success or failure of reuse and true systems engineering in the future.

Training

There is continued emphasis on the development of sophisticated tools and integrated programming support environments to eventually solve the vast majority of our software engineering woes. This is well justified in the realm of technology advancement, however, we must keep technology in perspective. Tools only support highly skilled engineers that design and develop systems. Skilled craftsmen can build wonderful things using powerful tools however, a child with a power saw is destined for tragedy. Ensuring that our software, hardware, and system engineers have the training and skills to keep pace with technology is vital. The primary emphasis should be placed on integrating systems engineering curriculum at the undergraduate level. Academia must take the lead to meet this challenge.

It is essential that we educate our users and developers on what reuse is, how it works, and what benefits will be achieved as a result of reusing life cycle components. RAPID has placed a tremendous amount of emphasis on ensuring that both our customers, programmer/engineers, and managers are thoroughly familiar with each project's reuse strategy. This strategy encompasses the following areas:

1. Reuse goals and objectives
2. Identification of reuse opportunities (domain analysis)
3. Design approach
4. Reusable component resources
5. Development of reusable components
6. Reuse standards and guidelines
7. Use of the RAPID center
8. Incentives and legal requirements

If the systems development community does not understand the benefits of reuse, then getting reuse integrated into the systems development methodology will not happen. RAPID will continue to be an "aggressive advocate" of software engineering and reuse. It is imperative that we maintain the momentum until the philosophy of engineering and reuse becomes a natural extension of our development mentality.

Methodology

Technology is providing an assortment of sophisticated tools and environments that can support a variety of development methodologies. Once again, the role of management is critical in determining how reuse is being integrated into that environment.

RAPID continues to emphasize the philosophy that reuse is essential to the entire systems development process. Management must understand the role reuse will play in their methodology and must move beyond the assumption that reuse applies only to code. Reuse of knowledge and components is an effective means of technology transfer. It is intuitive that the reuse of quality knowledge and information leads to quality decision making. It is equally intuitive that we can apply that same principle to our systems development methodologies.

RAPID continues to develop processes that support effective reuse and the integration of those processes into the development methodology. Most importantly, the process of domain analysis which determines potential reuse areas within and across systems. Once domain analysis has been accomplished RAPID has developed processes to identify potential sources of reusable components and support the actual reuse of the components. Our emphasis today is on reuse of design components which are linked to the implementing code, documentation, and test data that support the design. Evaluating lessons learned during this phase of RAPID's program will give us the information necessary to evolve toward the reuse of requirements and eventually the reuse of "knowledge" in the systems support environment.

Incentives

This is probably the most difficult issue that impacts on successful reuse. How can we create an environment where both the contractor and the DoD will benefit? Can we achieve reuse between contractors? How do we get beyond the "not invented here syndrome?" Relatively simple questions on the surface but the solutions can quickly become extremely complex and difficult.

There are two things that must happen to incentivize reuse successfully. First, this is a capitalistic society and reuse must be profitable. Second, it must be easier to "reuse" than to build it from scratch.

The government will move, out of necessity, toward relinquishing the requirement for unlimited rights to all delivered software products. As defined in Subpart 227.471 of the Defense Federal Acquisition Regulation, Government Purpose License Rights (GPLR) will likely become the norm. This gives the government unlimited rights for "government use only" but allows the freedom of "commercialization" to the contractor.

Contracts must change. A Cost-Plus contract is a great incentive to redevelop rather than reuse. A Firm Fixed Price contract is an incentive to reuse existing components, however, it is also a disincentive to develop reusable components because it costs more. Does it make sense to pay for "lines of code"? RAPID doesn't think so. New contractual methods, such as paying for delivered functionality, are being explored. Considerable work remains to resolve the many issues surrounding contracts and reuse.

RAPID is working closely with the Joint Integrated Avionics Working Group (JIAWG) which is tasked by DoD to implement the reuse of avionics and communications software between three major platforms; the Navy A-12, the Air Force Advanced Technology Fighter (ATF) and the Army's Light Helicopter Experimental (LHX). JIAWG and the Ada Joint User's Group (AdaJUG) are on the leading edge of contractual and incentive issues. Their task is to find ways to obligate and incentivize a multitude of contractors to provide reusable components across these three development efforts. They are investigating ways to allow each contractor to maintain restricted rights to certain developed components, provide those components to a competitor (for a price), and have the competitor use the component in their application (and receive an incentive award from the government for doing so). The expected outcome is:

1. The commercial incentive to build quality reusable components is there.

2. The incentive for a competitor to use the component is also there because:

a. It is more profitable for them to use the competitor's component than to develop it from scratch.

b. They receive an incentive award from the government for doing so.

3. The government wins because:

a. Contractors are developing higher quality reusable components.

b. Components can be reused by other contractors at a reduced cost.

c. Systems maintainability is improved.

d. Life cycle costs are reduced through improved reliability.

Most incentive questions remain yet unresolved. This is an area that will require significant emphasis in the future. Many of the incentive issues are tied closely with the legal issues because of requirements in the Federal Acquisition Regulation (FAR) and the Defense Federal Acquisition Regulation (DFAR). As RAPID works closely with our Beta sites and Army customers, we must create a win-win environment where reuse is good for the customer, the contractor, the government, and the system.

Legal

Legal problems associated with reusing software and technical data are brought up constantly as a major obstacle to reuse. These can significantly impact on our ability to integrate reuse into our systems support methodology of the future. RAPID has developed an interim non-disclosure/restricted rights policy to continue with our proactive approach to reuse. The policy includes the following provisions:

1. The legal policy must be read by all RAPID users. The policy includes the following information:

a. Definitions. Few people understand that the rights given to software do not apply to the documentation that supports it (documentation is considered technical data and falls under different provisions of the FAR/DFAR).

b. Commercial Off-the-Shelf (COTS) Software, Technical Data and all Restricted/Limited Rights Data. This requires that all software and technical data placed in the RAPID library must have the originator's copyright, trade secret notices, and/or restrictive legends imbedded at the beginning of each Ada specification, Ada body, and associated technical data (documentation).

c. Responsibilities. What is required of the user, project manager, RAPID Center manager, quality assurance, and configuration management.

2. A non-disclosure statement must be signed before becoming an authorized RAPID user. The non-disclosure statement requires the user to abide by all restrictive markings as well as provides specific guidance on how to handle the incorporation of a component with restricted rights into other software or technical data. It requires the user to adopt reasonable operating and physical security measures to protect extracted data from unauthorized disclosure to third parties. Finally, it states that components are provided without warrantee and the government will not be held liable as a result of use or misuse of components.

Legal issues that impact on reuse will continue to be problematic for several years. It is incumbent upon the industry and government to take interim steps now. If not, the fear of legal unknowns may obscure the opportunity to bring true engineering to systems development in the future.

CONCLUSION

The excitement of new technology will continue to grow as industry provides us with a never ending array of sophisticated tools and environments. Standards will allow greater inter-operability and communications capability than ever before. Reuse of knowledge and technology in systems development will propel the quality, reliability, and maintainability of future systems to new heights. Reuse of that same available knowledge and technology will allow us to build systems that meet the requirements of the user the first time. Technology will be ready for the 21st century.

Technology is not the answer. Management must ensure that their most valued resource is prepared to effectively use 21st century technology. People are that resource. RAPID and others are moving forward to promote a solid engineering approach to systems development and reuse. Management in academia, industry and government must make a commitment to support the training and education required to keep pace with technological advancement. We cannot expect technology to build systems for us. Smart people can do extraordinary things provided they are given the tools and resources. We need to make people smart, provide them with a methodology to do things smart, create an environment where they want to do things smart, and make sure legal and contractual issues won't prevent them from doing it.

REFERENCES

1. Basili, Victor R. Viewing maintenance as reuse-oriented software development. IEEE Software, 1990, v7n1, 19(7).

2. Clapp, J. Software reusability: a management view. Proceedings of IEEE Computer Society's Eighth International Computer Software and Applications Conference, 1984, 479-480.

3. Department of Defense. Software Master Plan (Preliminary Draft), February 9, 1990, Volume I.

4. D'Ippolito, Richard S. Using Models in Software Engineering. Proceedings of Tri-Ada '89, 1989, 256-265.

5. Hooper, James W; Chester, Rowena O. Software Reuse: Managerial and Technical Guidelines. Proceeding of the Eighth Annual National Conference on Ada Technology, 1990, 424-435.

6. Jones, G. Software reusability: approaches and issues. Proceedings of IEEE Computer Society's Eighth International Computer Software and Applications Conference, 1984, 476-478.

7. Lanergan, Robert G.; Grasso, Charles A. Software Engineering with Reusable Designs and Code. IEEE Transactions on Software Engineering, 1984, vSE-10n5, 498-501.

8. Matsumoto, Yoshihiro. Some Experiences in Promoting Reusable Software: Presentation in Higher Abstract Levels. IEEE Transactions on Software Engineering, 1984, vSE-10n5, 502-513.

9. McCain, R. A software development methodology for reusable components. Proceedings of the Eighteenth Hawaii International Conference on System Sciences, 1985, Volume 2, 319-324.

10. Moad, Jeff. Maintaining the Competitive Edge. Datamation, February 15, 1990, 61-66.

11. Paul, Lois. Reusable Code Prescribed for Software Productivity. Computerworld, 1981, v15n19, 15.

12. Perry, J.M. Perspective on Software Reuse. SEI Technical Report CMU/SEI-88-TR-22, ESD-TR-88-023, 1988.

13. Redwine, S.T., Jr.; Riddle, W.E. Representing and Enacting the Software Process. Proceedings of the 4th International Software Process Workshop, 1989, 133-135.

14. Scaling Up: A Research Agenda for Software Engineering. Communications of the ACM, 1990, v33n3, 281-293.

15. Woodward, Herbert P. A Better Approach to Software Engineering. Proceedings of the Eighth Annual National Conference on Ada Technology, 1990, 343-348.

ABOUT THE AUTHOR

Jack Rothrock is a captain in the United States Army assigned to the RAPID project at Software Development Center Washington. He is actively involved in the development of the RAPID program and is responsible for establishing and administering all RAPID Beta Test Sites. Captain Rothrock received his Master of Science degree in Information Systems Management from Eastern Washington University and is a member of the ACM. He currently chairs the ACM SIGAda Reuse Subgroup on management and legal issues.