USAISEC

US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES
(AIRMICS)

TECHNOLOGY ASSESSMENT OF SOFTWARE ENGINEERING

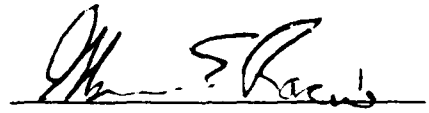
(ASQBG-I-89-011)

February, 1989

AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

This research was performed as an in-house project at the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC). This effort was performed under the AIRMICS Technology Insertion Program to support the U.S. Army Information Systems Command (USAISC) in the development of a report titled "Long Range Planning Guidance - Objective Configuration." An initial meeting was held in early December in Atlanta to coordinate the task. Twenty-six topics were selected for consideration, with AIRMICS agreeing to conduct technology assessments on fifteen of the topics. Planning Research Corporation (PRC) was assigned responsibility for conducting the remaining assessments and consolidating all the assessments for use in the planning document. In a two-week period, AIRMICS completed the assessments and provided the results to ISC-DCSPLANS and ISEC-SID. This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

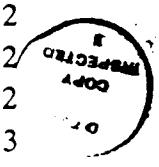
s/ 

Glenn E. Racine
Chief, CISD
AIRMICS

s/ 

John R. Mitchell
Director
AIRMICS

Table of Contents



I Historical Review

A. Software Engineering - the Discipline: History shows that engineering evolves from craft and commercial exploitation and that it requires scientific foundations. Science is especially important when natural intuitions about phenomena are weak. For software engineering, the technical basis is computer science. Characteristics of an engineering practice include iteration between formal analysis and design, heavy use of earlier designs, tradeoffs between alternatives, handbooks and manuals, pragmatic approaches to being cost-effective, and significant economic concerns.

The history of software engineering practices precedes the coining of the term "Software Engineering" in that software was being developed as soon as there were stored program computers. Commercial software development can be described in stages from "programming-any-which-way" (early 60's) through "programming-in-the-small" (early 70's) to "programming-in-the-large" (early 80's). Programming-any-which-way focused on mnemonics, precise use of prose, emphasis on small programs, representing structure - both symbolic information and numeric information - and gaining an elementary understanding of control flow. System state was not understood apart from flow of control. Programming-in-the-small focused on simple input-output specifications, emphasis on algorithms, data structures and types. Programs typically executed once and then terminated. There was a small state space that was symbolic or numeric. Programming-in-the-large focused on systems with complex specifications, there was emphasis on interfaces, management, and system structures, long-lived databases came into existence and program assemblies executed continually (real-time). There is a large structured state space that is symbolic or numeric.

B. Report of the Software Engineering Institute (SEI) on Assessment Findings: In June 1988, the SEI reported to the affiliates at the annual affiliates symposium the findings and consequences derived from the self assessments done in a workshop environment. They identified three major categories of issues:

1. Senior management
2. Organizational
3. Control and management

The senior management issues were divided into problems with management policies and with management tracking-control.

The second group of issues was identified as organizational issues and includes:

- Project management
- Software engineering discipline
- Training
- Systems engineering
- Software quality assurance

The control and measurement issues are the third major group of issues and include:

- Software Configuration Management
- Design and code reviews
- Software Engineering Process Group
- Metrics
- Standards
- Testing

C. Report of the AMC Software Task Force: In February of 1989, AMC issued a report titled "Report of the AMC Software Task Force. (HESS, et. al., 1989)" Their statement of the overall problem is:

> "As Army systems come to increase their reliance on computers, software complexity increases and its critically grows. Although progress has been made in recent years, the Army still does not have a cogent software acquisition and support policy nor does it have a strategy for insertion of new technology in a controlled, purposeful manner. This report builds on numerous previous studies to identify issues which confront the Army, documents underlying problems which the Army must face through this and into the next century, and identifies actions which must be taken to correct the problems."

The executive summary of that report reads:

> "Computer software is a necessary element in Army weapon systems but that software is contributing an inordinate amount of cost and risk to system acquisition. This report identifies the problems which the Army is facing and will continue to face unless corrective actions are taken. The report builds on and is consistent with a number of previous studies, investigations, and reports. It identifies recommended actions and calls on the Army to establish an orderly, task-oriented assault on the problems.
>
> The report suggest there are three primary challenges to the Army: Reduce the growth in the cost of software the Army is acquiring, bring the process to develop and maintain software under control, and maintain and extend the technological superiority of Army weapon systems through a focused investment in software. A taxonomy of software problems is presented which groups today's software problems into five areas: capability of people, absence of a clear and cogent software policy, lack of process controls, an absence of adequate procedures, and failure to capitalize on and plan for technology.
>
> The report recommends an integrated course of action to try to gain control of the of the software in the Army's software systems. The recommendations call for the Army to: develop a strategy for software acquisition, organize to

better manage the acquisition and support of software intensive systems, establish controls to guide programs and manage system acquisitions, and allocate sufficient resources for mission critical software. An implementation strategy is suggested which provides for intensive, dedicated management until significant progress is demonstrated."

The study identified at the detail level in the area of people, 24 separate problems; in the area of policy, 26 separate problems; in the area of process, 24 separate problems; in the area of procedures, 80 separate problems; and in the area of planning, 28 separate problems.

This study is exclusively focused on the problems of embedded systems software. Despite this focus and despite some differences in priorities, most of the problems are common to the MIS community as well.

There is a bibliography of government studies which was used by AMC to arrive at the problem statements and recommended solutions.

D. Joint Service Task Force on Software Problems: In July of 1982, a Joint Service Task Force on Software Problems found that:

- Computer software has become an important component of modern weapon systems.

- There is a danger that the U.S. military mission could be adversely affected if DOD software development and support were inefficient.

- Positive action had to be taken to improve the state of the software practice in the DOD.

The problems noted by the Joint Task Force include:

- Difficulty in establishing system requirements.

- Management did not understand software problems.

- No clean mechanism to transition technology into the development process.

- Lack of a software process.

- Lack of consistent standards for software.

- Tools were unfriendly, not integrated, not standardized, not designed for flexibility or profitability and inadequately supported.

- Lack of good analytical methods and hard empirical data for software.

- Documentation was found to be the wrong type, poorly written, incorrect, and inadequate.

E. The Challenge of Software Engineering Project Management: In August of 1980 an article titled "The Challenge of Software Engineering Project Management" (SEPM) appeared in the IEEE Computer Magazine [THAYER, et. al]. They were trying to define the major issues of software engineering at that time and challenged professionals in the field to refute any of the twenty major SEPM issues discussed in the article. The contributing professional was to do this by preparing and publishing a technical paper which presented a usable solution that virtually eliminated or significantly reduced the importance of the hypothesized problem. The identified problems in SEPM were divided into the categories of planning, organizing, staffing, directing, and controlling. Most of the identified problems are still relevant. A reader desiring further information on these problems should refer to the cited article.

F. Involved Organizations: The need for a more disciplined approach to move "Software Engineering" to a true engineering methodology is well recognized. A number of groups and organizations have been formed to address the issues. The most notable are the IEEE and ACM. There are many other organizations that influence the discipline of software engineering. These include the National Science Foundation (NSF), American Institute of Aeronautics and Astronautics (AIAA), International Federation for Information Processing (IFIP), American Federation of Information Processing Societies (AFIPS), American Society for Information Science, Instrument Society of America, International Association for Computing in Education, Society for Industrial and Applied Mathematics, Society for Information Display, Data Processing Management Association (DPMA), American Association for the Advancement of Science (AAAS), and The Society for Computer Simulation (SCS). Research is sponsored by numerous government agencies, including the U.S. Army (AIRMICS, ARO, CECOM, SDC), Air Force (AFOSR, RADC), U.S. Navy (NRL, ONR), and NASA. Computer Science undergraduate programs in colleges and universities are considered for accreditation by the Computer Science Accreditation Board (CSAB), sponsored by ACM and IEEE.

G. Initiatives: There are numerous initiatives which have been organized to do work in software engineering topics. Included among these are the Software Technology for Adaptable Reliable Systems (STARS) program and the Software Engineering Institute (SEI), both of which are funded at levels of fifteen to twenty million dollars per year. The SEI, located at Carnegie Mellon University, is focused on technology transition of Software Engineering concepts and tools for the embedded systems of the DOD. Although focused on the embedded systems, the tools and concepts the SEI is identifying are largely applicable (perhaps with modification) to the IMA needs. Agencies within the Army (AIRMICS, CECOM,

ARO, SDC) are also doing work on software engineering topics as are agencies in the other services which have not yet been mentioned. The National Science Foundation is sponsoring more than forty industry-university cooperative research centers including the Software Engineering Research Center at Purdue University and the University of Florida (AIRMICS is a member). Another non-DOD initiative is the AdaNet that promotes technology transfer of software engineering products from the Ada community and is sponsored jointly by the National Institute for Standards and Technology (NIST - formerly the National Bureau of Standards) and the National Aeronautics and Space Administration (NASA).

H. Problem Summaries: The problems noted by the Software Engineering Institute, the Joint Service Task Force on Software Problems and in the article "The Challenge of Software Engineering Project Management" along with problems cited by Information Systems Command personnel are summarized below.

Software Engineering Process: The software engineering community has not yet arrived at a consensus definition of the software engineering process and this process is not consistently defined across separate development projects. The lack of a consensus definition of the software engineering process leads to a lack of management comprehension of the problems of software development. Because the process is not consistently defined, critical software engineering process steps are often viewed by management as non-essential. Lack of a consensus definition of the software engineering process also inhibits the development of friendly, integrated, standardized, flexible, well supported software engineering tools. Lack of a consensus definition of the software engineering process also inhibits the introduction of new technology. Inconsistent development processes lead to incompatible systems and systems which are not delivered on time to a specified quality. This lack of process definition also leads to the assignment of engineers with insufficient software design background or training to handle complex software sub-systems. This lack of process definition makes it difficult and expensive to capitalize on the best available practices, difficult for people to move between projects, and difficult for the quality organization to review progress.

Software Requirements Development: The development of software requirements is an intensive, error-prone, and imprecise process. The requirements development process is made more difficult than it need be due to the lack of reuse of existing components and the lack of a domain analysis to show the similarity of functions across many systems within the same functional domain. Such functions as federal tax withholding are well defined but are redefined for each application system which uses it. Functions such as replenishment and excess determination within the logistics community are redefined for each application system which is developed. With appropriate designs this redefinition is redundant. Several tool sets exist today which aid in the process of requirements development. Among them are Problem

Statement Language/Problem Statement Analyzer (PSL/PSA), Software Requirements Engineering Methodology (SREM), TAGS (a name, not an acronym), and the Vienna Development Method (VDM). The automated tools require significant amounts of training to use and are heavy resource consumers. VDM is a formal specification technique and requires even more training for its effective use.

Software Project Management: Effective project management is found lacking. Organizational focus is lacking or nonexistent in providing a common approach to the software engineering process across all projects, establishing a technical peer communication network, establishing measurable goals for product quality, allocating resources for interproject software engineering issues such as metrics, software quality assurance, technology, and software training/education, focusing on critical system risk areas. Management tracking and control of the software engineering process is lacking or nonexistent. There is little evidence of formal procedures being used uniformly throughout the organizations for estimating software size, effort, and development schedules for projects. The consequences of these shortcomings are that project managers select from the most familiar processes and support facilities and the software process used or followed is largely driven by contracts. There is uneven and generally inadequate control of the software organization. The quality of management reviews varies and requirements changes do not receive consistent and disciplined control. There is no defined and understood commitment process, the lack of product sizing renders estimates of cost and schedule next to useless. The lack of good cost estimates means unbounded projects that often result in gross cost overruns, and uncontrolled time and resource consumption. The lack of effective schedules often leads to last minute crises.

Software Control and Measurement: There is no orderly focus on the software development process. This includes process definition, metrics, test and review methods, standards, and technology planning. The consequences include ad hoc software development techniques becoming the rule and not the exception, successful software engineering processes may not be repeatable, the development process is difficult to improve in an orderly way, there is no collection of process data, there is no focal point for the evaluation and insertion of new tools and methodologies. Effective change control for requirements, specifications, designs, code, and tests was not generally practiced. The consequences include that it is difficult to develop quality software on a predictable schedule, lack of change control results in ineffective testing, resources may be expended without management knowledge, source code is not controlled. There is little effective use of formal design and code reviews on software projects. This means that the major burden of error detection falls to testing, lower quality systems are delivered to the users, and maintenance costs are excessive.

Software Organizational Staffing: System engineering skills needed to define and guide large and complex systems are often not present or not consistently maintained throughout a project. The consequences of this are that system requirements and requirements changes are not adequately defined or are not defined in time to meet project schedules. Additionally, customer changes are not adequately considered from a system point of view, hardware/software tradeoffs are not consistently analyzed, and effects of changes on the subsystems (hardware or software) are unknown.

Software Quality Assurance (SQA): Software quality assurance organizations, in general, do not focus on compliance with development product and process standards. The consequences include inconsistent use of software quality assurance in the development process. SQA lacks the teeth to enforce the meeting of requirements or development standards and is also badly understaffed, resulting in lack of uniform adequate coverage across projects. Reduced product quality is likely.

Software Testing: Testing methods are inconsistent, insufficient time and resources are allocated to testing, and modern test techniques are not in general use. As a result, testing is frequently suspended when time runs out, meaning inadequate test coverage. A lack of adequate unit testing defers problems to the systems integration test and operational phases. A lack of regression testing has serious negative effect on the integrity of the system and a lack of stress testing increases the likelihood that the user will encounter program failures when the system is heavily loaded.

Software Metrics: There is limited data gathering and analysis to support software process improvement. The result is that the weakest process steps are difficult to identify, process improvement is difficult to verify, and the cost benefits of change are difficult to justify. Opportunities for process improvements are difficult to identify, and maintaining existing computer programs is more difficult than it need be.

Software Maintenance: Market research at International Data Corporation shows that enhancement maintenance accounts for 50 percent of federal software maintenance spending. Enhancement means adding new features to programs in response to changing requirements. Adapting software to new hardware or operating system environments consumes 26 percent of federal software maintenance dollars. Error fixing consumes the remaining 24 percent of federal spending on software maintenance. There is great need for programmer education even in basic principles such as modular design. The estimate is that only 20 to 25 percent of practicing programmers know about these techniques and an even smaller percentage actually use them. Software maintenance is not well planned for at any of the early stages of the software development process. A small

improvement in this process can lead to a significantly larger set of systems which can be supported. Software maintenance is a complex process which involves all the processes used in software development and has the additional problem of honoring all prior constraints from the original solution and of understanding the existing design in terms of the requirements.

Software Reuse: Software reuse is a necessary additional process to add to the software engineering development and support process. As yet it involves little understood techniques and requires support tools which do not yet exist.

Software Portability: Software portability is an aspect of software reuse which has been around for many years but which must be solved each time new equipment is added to the existing inventory of hardware.

Software Engineering Training: Inadequate professional software training and education is being provided to engineers and managers involved with software. Consequences include professional training and education provided primarily through contract funding rather than through standard organizationally supported classes. This has a negative effect on worker productivity and mobility across projects and adds to the shortage of skilled people to do software development tasks effectively. This results in reduced efficiency, productivity, and quality during software development.

Legal Issues: Legal issues also need to be addressed for the display of information since, so far, automated systems output is not accepted as legal evidence while in electronic form. Data from automated systems must be printed and archived in some physical form to be accepted as legal evidence. This requires much more printing and preparation of other forms of physical copy than will otherwise be required for the use of the information.

II. Currently Available.

A. SEI Assessment Process: The SEI currently has available a self-assessment process to identify the state of the practice in Software Engineering within an organization. It helps rank order requirements for modernization of the software engineering development environment. To date, the self-assessment process has been used in a workshop environment and the general experience has been that most of the organizations are still in the initial stages of maturity for a software engineering support culture. The initial stage of maturity is characterized by a software engineering process that is dominated by highly capable individuals, is not necessarily repeatable or well defined, and one in which measurements are not routinely made. Such processes are characteristically difficult to manage and to improve in a predictable way.

There also exists a contractor assessment process that parallels the self- assessment process. It is being proposed for use on DOD software development contractors.

B. Project Management Tools: There are a number of project management tools available to support management of software development projects. Examples include Estiplan; Interactive System; System Development Method (SDM/70); SDM/Structured; Method/1; Middleware; Apecs/8000; BSO Planner; Software Development Processor; Estimacs; Planmacs; 3D/One System Dev.; Information Planner; RMS; Projectmanager; Project Management, Command and Control; Stradis Information Systems Development Method; N5500 Project Management System; N1100 Project Management System; Information Systems Planning; Software Life Cycle Management (SLIM); Rand Development Center; MicroPert Family; Provac; Aide-de-Camp Software Engineering Environment; Spectrum Methodology; Spectrum-2, -3, etc; EZPert; Vis1on; and Vis1onmicro. These range from manual methods through automated tools on PC's, to tools on IBM mainframes. The performance and cost of these tools should be evaluated for the Army environment.

C. Text Retrieval Tools: One of the IMA problems is the finding of textual material within an intimidating supply of papers. This is not a unique problem to software engineers but it is a particular problem to software engineers involved in maintenance of existing systems. There exist tools for the scanning of hard copy to create an electronic version of textual materials. Once corrected to match precisely the input, the electronic version can be scanned using information retrieval systems which operate rapidly. One such system was demonstrated at AIRMICS over a year ago. The claim (which seemed supported by the demonstration) was that information could be identified in seconds from a database of text as large as one year's worth of the Wall Street Journal. An example text retrieval system is the one currently in use in the legal profession. This retrieval system is a full text retrieval

system which can selectively retrieve on any word within a document. This system has an extensive thesaurus capability that could be put to good use within the Army. Applications of the text retrieval technology would require an increase in the amount of disk storage available by an order of magnitude – even if the text were compressed. Use of this technology without the scanning capability could be used to make information currently in the files of the Standard Army Management Information Systems (STAMIS) and other databases more accessible. The cost of these tools to the Army should be identified as should the specific cost savings of using such a retrieval technology. Automated techniques are available and must eventually be used.

D. Display of Information: Another of the IMA problems is the use of material from automated systems. It is a particular software engineering problem, however, in that software engineers are typically working with multiple forms of software from the requirements document to the design document or from the design document to source code or from test cases to results. There are other examples in which these relationships require the multiple display of documents.

Currently, printouts used in support of the programming task are excessively common. This results in delays in getting the work done. The technology identified in the paragraph on text retrieval allows the finding of the information. It does not allow the easy use of the information on-line because limited display capabilities of most terminals limit the information which can be displayed at one time in too severe a way. Using terminals with larger screen capabilities like those of the Sun workstation would reduce the need for printouts of material.

E. Software Reuse: Software Reuse is a technology opportunity in which significant research is being performed both at AIRMICS and in a number of other organizations. The Reusable Ada Packages for Information systems Development (RAPID) Center at Software Development Center-Washington began operations in January 1989. This is an initial step only and much work remains to make reuse an effective concept within the IMA for the development of replacement systems. In this area, the Army is a leader. Reuse is not a panacea and should not be overpromised. With significant additional work and a continuing effort that is appropriately funded from R&D funds, this effort can provide benefits within the next five years.

F. Prototyping: Prototyping is another useful approach to software development. This approach to software development has promise to reduce the time for new systems to move from concept stage to operational status. It also has promise to reduce total life-cycle costs since some of the use characteristics and enhancements which later become changes during the maintenance phase can be addressed during

the development phase. Prototyping will be enhanced by the reuse and management concepts discussed above.

G. Computer Aided Software Engineering (CASE) and Integrated Environments: CASE tools will enhance both prototyping of new systems and reuse of existing software components. There are several CASE and software development Integrated Environments available for use. Vendors include, among others, Advanced Technology International, Atherton Technology, Cadre Technologies, Cincom Systems, Cortex, Digital Equipment, Index Technology, Integrated Systems, Interactive Development Environment, Interactive Software Engineering, Meta Systems, Nastec Corporation, Netron, Ontologic, Oracle, Promod, Relational Technology, Software AG of North America, The Stepstone Corporation, Tektronix, and Texas Instruments. Some of them appear quite good but all are incomplete. One of the problems with CASE tools today is there is no existing interface standard so that the output from one company's CASE tools can be conveniently used as input to another company's CASE tool.

H. Object-Oriented Design: Object-oriented design is a relatively new approach to software design. Object-oriented design is an approach to design in which objects within the software system typically have a one-to-one relationship with objects in the real world which are being modeled by the system. Further, each object in the software system has associated with it the procedures which would be associated with that same object in the real world. It is one of the hottest buzzwords in software engineering today. It provides promise for organizing both data and procedures in a modular self-contained way that can enhance reuse. The language Smalltalk is the best known of the object oriented languages. Ada provides some support for object-oriented design but is missing some attributes which would make implementation of software objects easier. The biggest current difficulty is in getting specifications to be written as objects so that the translation from requirements to software implementation objects would be easier. The technique is as yet not fully proven and may be more applicable in some functional domains than in others.

III. 1995 (Near Term).

A. Management: Software Engineering in 1995 will be much like it is now since most of the people who will be software engineers then are already working in the field. The most pressing problem between now and 1995 will be the need for improvement in the productivity of software development and support personnel. One of the most significant areas for change is in the management and improvement of software development. The SEI assessment processes will force competitive organizations to define the process they will follow for software development and support in order to remain competitive. The best of the organizations will move beyond defining their software engineering process to managing, on a continuing basis, the improvement and optimization of their software engineering development and maintenance environment. Measurements of product amounts, product quality, and project resource consumption will be recognized as essential to the management of the software development and support environment. Minimum supportive elements of a software engineering environment will have standards for software construction, inspection and review mechanisms, practical measurement techniques for both quality and productivity, and seamless communications.

B. Technical: Technical improvements to enhance productivity will include increased storage and display capacities. These will be critical to the improvement of productivity during software development and support. Optical disks of the Compact Disk (CD) Write Once, Read Many (WORM) variety will be the most prevalent during this time period. On the horizon will be optical disks with rewrite capabilities. Capacities of DASD will be in the hundreds of terrabytes range to support the massive information retrieval requirements to manage whole families of reusable components. The managers of software developers will also require data about the software development process so that they can better manage the software development process. Cooperative decision making will become a reality because of the increased availability of this information.

Text Retrieval: One of the problems identified earlier was text retrieval. Promised text retrieval systems include those that will handle text and graphics concurrently within the same file and do retrievals on both text and graphics.

Information Display: Another of the problems identified earlier was the limited capabilities for display of information. By 1995, display of information in electronic form will be much enhanced with the capability of displaying simultaneously, through windowing techniques, three to five legal size pages of information legibly with the capability to expand or contract pages as necessary to maintain context.

Software Reuse: Software reuse within a well managed environment has the potential for significant increase in development productivity. Software reuse within

an organization for the development of replacement or enhanced systems will become common. Software reuse which must cross the boundaries of organizations (as in contractors vs. the federal government) will still be limited because of the legal issues involved. Reusable components will include functional requirements, alternative designs, design decisions and the supporting rationale for those decisions, source and object code, manuals, test cases including test results, and use history both in terms of number of systems in which each component has been used and the use history of the systems themselves. Additionally, the relationships of each component with its associated documentation and the versions of each component and the relationships of each component with other components will be in this database of reusable components. In addition to the products created by the software development and support process will be the reference products which are used by software developers and maintainers to follow the software engineering procedures as they are defined.

Integrated Environments: Integrated environments for software development and support will have orders of magnitude greater capacity than current environments and will be more capable of full life cycle support than are today's versions of integrated environments and CASE tools.

Software Maintenance: Much has been written about the need for spending more money up front during the requirements definition/specification stages in the software development cycle in order to reduce total life-cycle costs. However, short-term budgetary concerns have resulted in little being done to achieve this goal. As a result, software currently being developed will have large maintenance costs. Further, software maintenance will be an even greater part of the total resource consumption than it is today since little attention is focused now within either the research community or within the academic community on the special problems associated with software maintenance. Increased numbers of systems to support, along with a continuing demand for error free operation, will increase the need for analysis tools to support software enhancement, adaptation, and error correction.

C. Microcomputers: Software engineering procedures for integrating microcomputers effectively into a total computing network are required so that information created by individuals for managing significant tasks will be made available to others as needed. Microcomputer based commercial off the shelf software will be acquired and used in increasing numbers and will be used by more and more individuals for accomplishing tasks as yet not imagined. Integration of separately acquired packages will be needed. A part of the integration procedure will be the migration from current standard custom developed management information systems to commercial off the shelf products which perform essentially the same function. This will be a necessary consequence of the demand for increased productivity. Management and acquisition of this software inventory on

favorable terms will be a significant problem. The microcomputer hardware inventory supporting these commercial packages will be volatile and will require management.

Costs of maintaining the software inventory through contractual arrangements will be a significant item of the software support budget. Monitoring the use of these tools so that only those packages that are used are kept in the inventory will be a requirement for managing the training needs associated with this software as people are hired and for keeping the continuing support budget in line. In-house government staff to support these commercial packages will be required to enhance the productivity of product users. Commercial off the shelf products designed for multiple organizations, including non-government organizations will be cheaper to support than the current custom systems. Economics will rule.

D. New Technologies: There are industry projections that by 1992 erasable optical disk drives will have sales of $1,323 million from a 1988 level of $18 million. Neural network tools for model builders (simulators only) will grow from $18 million in 1988 to $150 million in 1992. Laser based IC processing equipment will grow from a 1988 level of $200 million to a 1992 level of $722 million. Most surprising is that superconducting products are projected to grow from a 1988 level of essentially zero to a $2.3 billion level by 1992. The implications of these technology projections on software engineering are that these increased capacities for storage and processing will strongly enhance the user expectations for computing support. Productivity levels for software engineers will be under even greater pressure for both functional growth and for improved reliability of more complex systems.

IV. 2010 (Long Term).

In 2010, Software Engineering will be more of an engineering discipline. The life cycle will be better defined, more unified, and will have more automated support. Decisions on which tool to pick for a particular problem or application will be more objective than they are today. There will be a large inventory of Ada software to maintain. Functional programming through functionally specific programming languages will be more common.

A. End User Software Development: Due to increasing demand for information, software development and support will require increasing levels of productivity. In 2010 the soldiers using the Army's automated systems will have grown up using editors, spread-sheets, and database applications on computers in school and at home. They will thus be ready to use the computer creatively to solve the assignments they receive. In a very real sense, computer use will parallel telephone use and as every person who uses a phone now performs the operations that a telephone operator once performed, users in 2010 will develop many of their own applications and thus have increased productivity. The systems available to them for this will have had the benefit of software engineering principles applied to them so that they will be easy to use, effective in the support of user requirements, responsive, reliable, and cost effective.

B. Professional Software Development: There will still be full time development of automated systems. Management of software development will have made significant improvements in the software engineering process. Leading companies will be able to define requirements and establish reliable project schedules with reliable levels of software quality delivered on-time and within budget. Legal issues will continue to plague the industry with respect to the reuse of existing components, since, as with security issues, when one legal issue is resolved, other legal issues become significant. Despite this, reuse as a way of developing new systems will be the dominant form of new development projects. CASE tools and Integrated Software Development Environments will be the rule rather than the exception although they may be under some other name. As software development becomes more of an assembly process through reuse, much of what is today considered software maintenance will be redefined as software development. This will be true of both enhancement and adaptive maintenance which today are estimated to consume 76 percent of the software maintenance budget which in turn takes 40 to 60 percent of the software budget within the Federal Government. Unfortunately this improvement in procedures will be for systems under development at this time. Applications being developed in 1989 through 1995 will not have the benefit of these improved procedures and software maintenance in 2010 will still be a significant portion of total software costs.

For applications being developed in 2010 by the professional staff, parallel and distributed computers will be dominant and programming for them will be as well supported then as current computers are supported today. Optical disks for permanent on-line storage will have rewrite capabilities and will be in sizes now unimaginable. The volumes of information available in on-line libraries will be extensive enough that new methods of education will be seen as appropriate but will still be under design and test.

V. Glossary.

1. AdaNet: A network sponsored by the Department of Commerce and NASA for the transition of Ada technology into private industry. Cooperative support is given by agencies of the DOD.
2. AFOSR: Air Force Office of Scientific Research.
3. AIAA: American Institute for Aeronautics and Astronautics.
4. AIRMICS: U.S. Army Institute for Research in Management Information, Communications, and Computer Sciences.
5. ACM: The Association for Computing Machinery.
6. Apecs/8000: One of many project management systems.
7. ARO: Army Research Office.
8. Atherton: One of many companies providing a set of CASE tools.
9. BSO: One of many project management systems.
10. CASE: Computer Aided Software Engineering.
11. CD: Compact Disk.
12. CECOM: Communications - Electronics Command, a subordinate command to the Army Materiel Command.
13. Cincom: One of many companies providing a set of CASE tools.
14. DPMA: Data Processing Management Association.
15. Estimacs: One of many project management systems.
16. Estiplan: One of many project management systems.
17. EZPert: One of many project management systems.
18. IEEE: The Institute for Electrical and Electronics Engineers.
19. IMA: Information Mission Area.
20. Meta: A company which provides one of many CASE tool sets.
21. MicroPert: One of many project management tools.
22. Middleware: One of many project management tools.

23. NASA: National Aeronautics and Space Administration.

24. Nastec: One of many companies supplying a set of CASE tools.

25. Netron: One of many companies supplying a set of CASE tools.

26. NIST: National Institute of Standards and Technology. The new name for the National Bureau of Standards.

27. NRL: Naval Research Laboratory.

28. NSF: National Science Foundation.

29. ONR: Office of Naval Research.

30. PC's: Personal Computers - also a reference to a specific personal computer - the IBM PC.

31. Planmacs: One of many project management systems.

32. Projectmanager: One of many project management systems.

33. Promod: One of many project management systems.

34. Provac: One of many project management systems.

35. RADC: Rome Air Development Center - Air Force Research in Software Engineering.

36. SCS: The Society for Computer Simulation.

37. SEI: Software Engineering Institute.

38. SEPM: Software Engineering Project Management.

39. SQA: Software Quality Assurance.

40. STAMIS: Standard Army Management Information Systems.

41. Stepstone: One of many companies providing a set of CASE tools.

42. Stradis: One of many companies providing a set of CASE tools.

43. terrabytes: A term for 1,000 billion bytes of storage.

44. Vis1on, Vis1onmicro: Two project management tools from the same company.

BIBLIOGRAPHY

BALZER, Robert, et.al, "Software Technology in the 1990's: Using a New Paradigm," IEEE Computer, November 1983.

BASILI, Victor, et.al, "Implementing Quantitative SQA: A Practical Model," IEEE Software, September 1987.

BEINHORN, George, "In Search of Forgotten Text," PC World, November 1988.

BOEHM, Barry, et.al, "Software Technology in the 1990's: Using an Evolutionary Pradigm," IEEE Computer, November 1983.

BROOKS, Frederick, "No Silver Bullets," UNIX Review, November 1987.

CAVANO, Joseph, "Quality Assurance in Future Development Environments," IEEE Software, September 1987.

CERVENY, Robert, "Why Software Prototyping Works," Datamation, August 15, 1987.

CHIKOFSKY, Elliot, "Software Technology People Can Really Use," IEEE Software, March 1988.

COLLOFELLO, James, et.al, "Software Quality Assurance for Maintenance," IEEE Software, September 1987.

DART, Susan, et.al, "Software Development Environments," IEEE Computer, November 1987.

FAIRLEY, Richard, "Software Engineering Concepts," McGraw-Hill, 1985.

GLASS, Robert, "Software Design: It's All In Your Mind," Computerworld, November 7, 1988.

GRADY, Robert, "Measuring and Managing Software Maintenance," IEEE Software, September 1987.

HANNER, Mark, "CASE Tools; Productivity for the Masses," DEC Professional, December 1988.

HENDERSON, Peter, et.al, "Guest Editors' Introduction: Integrated Design and Programming Environments," IEEE Computer, November 1987.

HESS, James A., et. al., "Report of the AMC Software Task Force," U. S. Army, February, 1989.

HUMPHREY, Watts, "Software Engineering Process," Software Engineering Institute Affiliates Briefing, June 1988.

KISHIDA, Kouichi, et.al, "Quality-Assurance Technology in Japan," IEEE Software, September 1987.

KNOWLES, Anne, "Taking the Wraps Off the Technologies of the 1990's," Managing Technology, December 1988.

LEAVITT, Don, "A Common Ground in Managing Software Projects," Software News, January 1986.

LISWOOD, Woody, "Project Management for Professionals Only," PC World, November 1988.

LOY, Patrick H., "Enlarging the Scope of SQA," ACM Software Engineering Notes, January 1989.

MARTIN, Edith, "The Context of STARS," IEEE Computer, November 1983.

MILLS, Harlan, et.al, "Cleanroom Software Engineering," IEEE Software, September 1987.

OFFUTT, A., "Automatic Test Data Generation," Software Engineering Research Center report SERC-TR-25-P, August 1988.

POSTON, Robert, et.al, "Counting Down to Zero Software Failures," IEEE Software, September 1987.

PRIETO-DIAZ, Ruben, et.al, "Classifying Software for Reusability," IEEE Software, January 1987.

RAMAMOORTHY, C., et.al, "Software Engineering: Problems and Perspectives," IEEE Computer, October 1984.

REILLY, Angela, "Roots of reuse," IEEE Software, January 1987.

SCACCHI, Walt, "The USC System Factory Project," ACM Software Engineering Notes, January 1989.

SHAW, Mary, "Software and Some Lessons from Engineering," Software Engineering Institute Affiliates Briefing, June 1988.

SHRIVER, Bruce, "Reuse Revisited," IEEE Software, January 1987.

TAFT, Darryl, "People Still Seen as No. 1 Challenge for Technology," Government Computer News, April 1, 1988.

THAYER, Richard, et.al, "The Challenge of Software Engineering Project Management," IEEE Computer Magazine, August 1980, pgs 51-59.

TOMIJIMA, Althea, "How Japan's Recently Amended Copyright Law Affects Software," IEEE Software, January 1987.

WALDO, Jim, "A New Generation," UNIX Review, August 1988.

WEIL, Ulric, "How Information Technology Will Change the World," Government Computer News, November 21, 1988.

WILSON, Ron, "Object-oriented Languages Reorient Programming Techniques," Computer Design, November 1, 1987.

--. "1987 ACM Guide to Computing Literature," 1988

--, "RISKS: Cumulative Index of Software Engineering Notes," January 1989.

--, "TABS, A report from the Technical Activities Board of the IEEE Computer Society," Computer Magazine Supplement, October 1988.