The Software Life Cycle—A Management and Technological Challenge in the Department of Defense

BARRY C. DE ROZE AND THOMAS H. NYMAN, SENIOR MEMBER, IEEE

Abstract—The importance of software in defense systems continues to intensify as new systems emerge in response to increasing threats and declining force levels. The need to manage this software as a critical component of defense systems over their life cycle is becoming widely recognized. A general awareness of this need as an institutional problem requiring special attention within the Office of the Secretary of Defense has been growing as software problems have reached top level defense management visibility with increasing regularity. Software costs are continuing to multiply in step with advancing weapons systems sophistication, and opportunities for cost avoidance now are leveraged against large dollar investments. These conditions characterize the computer industry at large, but very little overall focus has been visible to improve the software development process. Consequently, the Department of Defense (DoD) has undertaken a two part effort to accelerate both near-term and long-term improvements in software development for weapons system applications. These efforts, the establishment of software life cycle management policy and practices, and the vigorous development and application of new software technology are discussed in considerable depth.

Index Terms—Computer resources, configuration management, DoD, software, software life cycle, software management, software risk analysis, standardization.

I. Background

THE Department of Defense (DoD) is presently spending over $3 billion per year on defense system software, excluding software for corporate management and logistics automatic data processing (ADP) systems. This high investment area, that we will call embedded computer applications, is the subject of our discussions. It is our opinion that DoD has been doing a poor job managing this increasingly important resource, and further we have been doing little to encourage the application of science and technology to improve it. Both of these shortcomings must change!

In this paper, we discuss the problems and their underlying causes which confront DoD in this area, summarize the corrective actions now being taken on the management front, and review some of the more promising developments being investigated and applied within the software science and technology program.

Manuscript received November 11, 1977; revised November 18, 1977 and January 19, 1978.

B. C. De Roze is with the Office of the Secretary of Defense, Washington, DC 20301.

T. H. Nyman was with the Research and Advanced Technology Directorate within the Office of the Under Secretary of Defense, The Pentagon, Washington, DC 20301. He is now with the General Research Corporation, McLean, VA 22102.

Neither the problems nor the proposed solutions that will be discussed are new. They have been studied at great length in technical meetings and within industry for the last ten years or more. The new element is in the sense, at all levels of DoD, of the need to act decisively and to act now!

Over the past few years there have been a series of defense-sponsored studies—each highlighting areas where defense-systems software requires improvement for reasons of cost and reliability [1]-[4]. During the past year, DoD has put together (with considerable help from industry and the academic communities) a plan of management and technical action—this action plan is the topic of this paper.

The main thrust of the remarks in this paper will be directed at embedded computer applications and not at the corporate management and logistics ADP environment. The discussion will focus on those issues which provide the incentives for, and barriers to, good software engineering and management practices in the defense system acquisition process. Recognizing, however, the need for consistency with the ADP community, close ties are being maintained, and they must continue to be maintained to assure maximum transferability of ideas, tools, techniques, and products.

In general, because of their R&D nature and close tie-in with hardware components of defense systems, there is general acceptance that the software practices most applicable to embedded computers should parallel closely those management practices which have been developed for hardware acquisition. This view will be expanded upon later; first we will quickly review, in Section II, the scope of the problems. Section III describes current actions being taken by the DoD to improve the management of computer resources and software in both the long and the short term; Section IV examines the research and development thrusts, and the priorities that the DoD has set to reinforce these management steps. Finally, Section V offers concluding remarks about where we stand today, and what still remains to be done.

II. Software Issues in Perspective

Software is big business within the DoD. As noted earlier, the current annual expenditure in software for embedded systems (weapons, platforms, command and control, and intelligence) is now estimated in excess of $3 billion; yet even this substantial sum is only the tip of the iceberg. It includes direct costs only and represents a conservative estimate based on incomplete and nonuniform data. This uncertainty, as a matter of fact, is indicative of a clear problem in itself. The distribution of software costs for all military systems for a given fiscal year shows that 68 percent of known costs are consumed in development of new systems (R&D), while the remaining 32 percent of the known cost is categorized as operation and maintenance (O&M) of systems already in the field. The majority of complex software systems are new and still in the development cycle. As these new systems are deployed, this cost distribution will reverse to emphasize the increased O&M burden. This, coupled with greater system longevity, may ultimately result in a five or ten to one ratio of O&M cost to R&D cost when viewed over the total life cycle of a typical system. With these projections, we will need an army of software maintainers.

Over and above the cost picture, software is finding its way into the critical path of many more defense systems. Major defense systems currently exhibiting a critical software dependency number approximately 115, with 50 percent of these in the R&D phase and the remainder in the O&M phase, and with the degree of software critical dependency increasing among the newer systems.

The functional applications of software within the DoD pervade almost every program. Performance critical software can be found in all extremes—from the large Worldwide Military Command and Control System (WWMCCS) and Trident Fleet Ballistic Missile (FBM) down to miniaturized flight control packages for missiles and aircraft. Ancilliary support equipment such as trainers, simulators, automatic test equipment, test range instrumentation and special test vehicles also contain software functions of increasing criticality.

A. Nature of the Problem

Several clearly observable manifestations, such as excessive software development and maintenance costs, schedule slippages and delays, and excessive errors and faults, can be seen repeatedly. Underlying these superficial manifestations—lack of transferability, incomplete hardware/software tradeoffs, and the treatment of software as data rather than as deliverable and testable products—are some of the causes behind the cost and quality deficiencies.

This special issue includes other papers which deal with some of these basic problems. Let us briefly explore some of the issues which are of particular relevance to top management, within DoD as well as the civil sector.

B. Inadequate Cost and Schedule Estimates

Studies by industry and DoD have concluded that today there are no simple universal rules for predicting software costs accurately, and that to do so requires an understanding of the nature of the individual program and the individual routines within the program [4], [5]. This will likely remain the situation for some time to come, but we must begin now to take action to reduce the amount of "individuality" in cost estimating. In this regard, it must be said that we currently suffer from a poor historical data base of software costs. Not only is the DoD unaware of what is being spent on software in the development, production, and operational/maintenance phases of a defense systems' life, but there is considerable uncertainty as to the proportions of dollars which should be spent. As it currently stands, DoD in general does not budget dollars or plan time for the predictable problems and changes which we should prudently anticipate in each of these phases.

This situation is aggravated by the lack of common definitions, procedures, and organization in planning and managing software development. While most management and technical approaches may have merit, and some are excellent, it is not practical for government review officials to be well acquainted with the merits and shortcomings of all the approaches presented to them. In addition, it is difficult to develop useful universal yardsticks. If DoD reviewers cannot really understand what is proposed for each program because of definitional problems, then they cannot apply sound judgment in their management responsibilities. In general, DoD must accept the estimates which are provided—no matter how optimistic—and budget and schedule to them. That's how the problem gets started!

C. People as a Cost Factor

Each contractor has the problem of retaining highly trained individuals to design, develop, test, and maintain the software. At a recent software conference, statistics were produced to show that the turnover time for an average programmer was about three years [6]. There is a motivation problem for the software "production" worker, just as with the hardware "assembly line" worker. To illustrate how highly labor-intensive software production is, it has been estimated that increasing programmer productivity by just one instruction per man-day could save $45 million per year in defense system development and maintenance alone.

D. Tracking User Requirements

The lack of means to produce clear, concise, and unambiguous statements of user requirements is one of the biggest contributors to the high cost of software. The problem here again has to do with the absence of a clear understanding on the part of both software users and developers as to what can be accomplished with software. Frequently, users either underestimate or overestimate the state of the art. At the same time it is important that software specialists be able to anticipate the likely directions of change, and design software and software tools so that it is fairly easy to accommodate changes when they come.

E. People in Hardware/Software Tradeoffs

Another part of the software environment impacting on the personnel issue is the lack of ability to make the necessary design tradeoffs between hardware, firmware, and software implementation. The classical institutional separation of these functions in design organizations acts to reinforce this barrier. To assess the strengths, weaknesses, and ensuing implications of these tradeoffs on a "systems-wide" basis for life-cycle cost and reliability, systems engineers with in-depth understanding of all these disciplines, and who can objectively perform and integrate tradeoff analyses to produce a balanced system, are needed. These people are in extremely short supply and their cultivation in the future remains a major educational and career incentives problem.

F. *Insufficient Software Management and Control*

Another area of overall concern has to do with the subject of software management in the DoD, and specifically as it pertains to embedded systems. DoD has become somewhat expert at knowing how to divide the responsibilities of the "requirements," "development," and the "procurement" people in the hardware acquisition process; and there is a fairly clear line of demarcation between what is hardware and what constitutes data. Software, however, creates something of a problem, for up until recently most managers and contracting personnel were content to treat it simply as data. As costs began to soar, it became obvious that some management changes were in order. To manage software in the same manner as hardware, we must begin to think of software in the same manner as hardware; we must begin to think of software as "property" and not solely as "data" (fully recognizing the legal implications of the term "property"). Cost information which is submitted for data items in contracts are usually only estimates and do not provide for detailed cost breakdowns for each data item. Frequently, it is difficult to get a clear and distinct separation of data costs from engineering efforts tied to a deliverable contract schedule item. Steps must be taken to clear up this matter for software estimates.

More to the point, however, DoD personnel must recognize that from a functional standpoint, computer software is equivalent to hardware, and must be delivered as an active system component. This means that technical and management control is required to insure a quality engineered and tested product. Management instruments and disciplines influencing computer software engineering, prototyping, configuration control, quality assurance, production control, reliability, maintainability, standardization, modular partitioning, design reviews, and life-cycle costing must be applied.

G. *Insufficient R&D for Enhanced Software Productivity*

The next issue concerns the need for more directed research and development on software. We refer here to the need to convert software development from an art into an applied science. This can only be accomplished by giving increased attention to the research and development of software tools and formal methods, such as those that characterize the engineering fields. We must get away from the notion that software advancements are the sole domain of "individualists" or "artisans." The best practitioners, individual superstars, can produce high quality and efficient software on schedule at low cost. However, it is not very useful to decree an approach to "hire just the good programmers." In fact, there is no scientific measure of either the quality of software or the performance of practitioners. It is all "unknown"–and the fact that software is "invisible" makes it that much harder. In this regard, great encouragement can be found in the efforts which have recently been observed in both the government and business sectors to expand upon the concept of software production facilities. The "software factory" concept is fast becoming an important means for imparting the necessary disciplines that are needed for software "creation." It involves the adoption of a system of integrated aids and tools to provide a disciplined and repeatable approach to software development; the replacement of ad hoc agglomerations of individualistic techniques and tricks with a standardized and integrated methodology. One of the key objectives of this approach has been to introduce manufacturing methods and engineering principles into those software production processes which satisfy design, implementation, and management requirements of control.

H. *Duplication in Software Development*

There is currently no way of knowing exactly how much DoD spends on "support" and "applications" software which had previously been produced for some other program or programs. Military Department spokesmen state that they believe it is extensive and that the first effort to control costs should begin in this area. Without clear software development standards, such as programming languages and adequate software management policies for defense systems acquisition, the increase in costs owing to the duplication of software can only be expected to grow.

I. *High Cost of "Maintenance"*

A significant cost factor, not to mention the operational readiness implications, has been software errors or problems discovered well after acquisition. One recent DoD study showed that development costs for Air Force avionics software averaged about $75 per instruction while the cost of "maintenance" corrections of deployed software has experienced costs in the range of $4000 per instruction [7]. The purpose of quoting these figures is not to offer them as representative numbers but to demonstrate that the costs of software maintenance are many times those of development. Note here that software maintenance, in addition to correction of problems (and the perpetual cycle of injecting new errors with each correction), includes the updating and revision of applications programs caused by changes to, or expansion of, the operational mission. In the future the DoD will need to take a comprehensive look at the life-cycle approach to acquiring software. This will include the formulation of design and management principles, the development and validation of software life-cycle costs models, and evaluation of design methodologies for ease of maintenance and program update.

III. Current Policy Actions

The problems and issues raised are real, and they require immediate attention. DoD has begun to take action aimed at improving the management situation for both the short and the long term.

The proper organizational focii both at the Office of the Secretary of Defense (OSD) and Service levels have been created. A DoD-wide Software Management Plan which addresses all of the problems identified earlier has been drafted and widely coordinated. This plan has been released and is available through the Defense Documentation Center [8]. A second management action, DoD Directive 5000.29, establishes policy for software management and control by DoD components of embedded computer resources and software during development, acquisition, deployment, and support, and was issued in April, 1976. A third action, DoD Instruction 5000.31, adopted seven higher order programming languages as interim standards for use in developing new defense systems.

The theme pervading all of these steps is to elevate software policy, practices, procedure, and technology from an artistic enterprise to a true engineering discipline. Or to put it another way—to treat software more like hardware—throughout its complete life cycle.

We would like next to elaborate further on specific actions and policies now being implemented within the DoD. These initiatives apply to all major defense system acquisition programs under the review authority of the Defense System Acquisition Review Council (DSARC) [9], as well as all weapons programs of lesser magnitude.

A. Software Requirements and Risk Analysis

The first area of emphasis under this newly formed policy concerns the requirements validation and risk analysis attendant to computer resources and software. Briefly stated, computer resource requirements, with particular emphasis on software cost and risk and on hardware/software/firmware design tradeoffs, must be reviewed, analyzed, and validated during the early concept formulation and program validation phases of new defense system development, prior to the full scale development decision point [9]. This analysis must assure conformance of planned computer resources with stated operational requirements. Risk analysis, preliminary design, hardware/software integration methodology, use of existing software modules, standardization, external interface control, security features, and life cycle system planning are included in the review. Testability of software, reliability, integrity, maintainability, ease of modification, and transferability will be major considerations in the initial design. The risk areas and a plan for their resolution must be included in a Decision Coordinating Paper, an official document which defines program goals, establishes cost/schedule/performance thresholds, and demonstrates implementation of Secretary of Defense guidance [9].

In addition, computer resource requirements will be continuously coordinated and reconciled with system operational requirements throughout system development after the decision to enter full scale development.

The effect of this policy will be to emphasize examination of software related systems issues very early in the development cycle prior to a major management commitment, and to insure that software is given the same visibility as hardware during these early phases of system evolution. In addition, it will provide top level Service and OSD management visibility into cost, schedule, option, and risk parameters at a time when subsequent development can be meaningfully influenced.

B. Computer Resource Life-Cycle Planning

A computer resource life-cycle plan will be developed prior to the decision to enter full scale development, and will be maintained throughout the life of the new system. The purpose of the plan is to identify important acquisition and life-cycle planning factors, both direct and indirect, and to establish specific guidelines to ensure that these factors are adequately considered in the acquisition planning process [10]. Resource planning is to include hardware and software design, cost, and schedule factors; documentation; and operating and maintenance personnel needs.

This policy will place economic trade-offs, acquisition strategy, and maintenance and modification decisions on a life-cycle basis, and eliminate the tendency to optimize development costs and schedules at the expense of the subsequent operations and support costs. DoD must stop mortgaging its future in exchange for fleeting benefits during development.

C. Configuration Management of Computer Resources

This policy action concerns the configuration management of computer resources in major defense systems. DoD Program Managers can no longer permit software to be treated as a data element to be acquired by a one-line entry on a list of deliverable documents. Instead, software will be treated as a full-fledged configuration item, with all the required disciplines, controls, and testing included. The emphasis will be on product definition, requirements traceability, interface definition and control, cost, quality traceability, and the corollary control disciplines. In particular, software performance will be specified and tested with progressive design review audits similar to hardware.

D. Support Software Deliverables

When it is cost-effective to do so, unique support items required to develop, test, and maintain the delivered computer resources over the system's life-cycle will be specified as deliverable, with DoD acquiring rights to their design and/or use. Examples of such support items are compilers, linkers, environmental simulators, optimizers, documentation aids, test case generators and analyzers, and training aids. The provisions of the Armed Services Procurement Regulations will govern the implementation of this policy.

This policy again emphasizes the life-cycle cost-effectiveness aspects of the acquisition decision. It is in part intended to remove the long term dependency on a single development contractor (thereby preserving DoD's maintenance and support options for the longest possible time), and it represents a necessary, although not sufficient step toward achieving true transferability of support software across mission and application lines.

E. Milestone Definition and Attainment Criteria

Specific milestones to manage the development of computer resources, including computer system and support software, will be used to ensure the proper examination of analysis, design, implementation, integration, test documentation, operation, maintenance, and modification phases. These milestones will include specific criteria that will permit a measure of their attainment. This policy relates to the product definition and work accomplishment aspects of configuration management, but the additional stress on quantitative demonstration criteria is significant to note.

Also of particular significance is the rigorous treatment which must be accorded to test and evaluation, beginning in the earliest phases of system development, and culminating in a complete operational test and evaluation by the ultimate military users.

F. Software Language Standardization and Control

Higher order programming languages, as opposed to assembly or machine languages, will be used during software design and development, primarily to enhance life-cycle serviceability of the software. Only DoD approved high order programming languages (HOL's), as specified in DoD Instruction 5000.31, "Interim List of DoD Approved High Order Programming Languages," will be used for new defense system software, unless it is conclusively demonstrated that none of the approved HOL's are cost effective over the system life cycle, and this will not be easy. Each DoD approved HOL will be assigned to a designated control agent who will be responsible for insuring the stability of the language, validating compliance of compiler implementations with the standard language specifications, gathering data as to the use of the language, maintaining a repository and disseminating information on compilers, tools, and application modules, and improving the language over time.

This policy impacts both the language selection and proliferation problems noted earlier. In general, high-order languages do afford considerable life-cycle benefits (particularly in the operation and support phases) even though some inefficiencies may be experienced in development. These inefficiencies will be reduced with new modern languages and increased compiler capabilities. Exceptions to the use of high-order languages must be justified over the life cycle, and not just for development advantages.

DoD's long range objective is to reduce to a minimum the number of approved high-order programming languages—starting from seven interim standards. But the objective for achieving this objective is cost reduction, so DoD must be flexible in how this standardization objective is applied over time. User and application languages for console and operator interaction, simulators, and automated test equipment are not yet included in the current policy initiatives.

G. Policy Initiatives Summary

These policy initiatives, incorporated in DoD Directive 5000.29 and DoD Instruction 5000.31, will have immediate impact on the way the Defense Department does business in all new weapon program starts, and when possible in ongoing programs as well. In a very real sense, the directions and intentions are clearly established and now attention must focus on the specifics of how to best reach these goals. This brings us to the second cornerstone of the defense software initiatives, the software Science and Technology (S&T) Program [11].

IV. Software Research and Technology Thrusts

In the long range, the DoD must depend upon the research and technology directions of the nation at large to fully achieve the objectives of converting software development from an art to an engineering discipline. However, the Office of the Under Secretary of Defense for Research and Engineering is focusing on several approaches to speed this process along, such as encouraging software engineering workshops among university and military research offices, and generation of textual curriculum material for in-house courses of instruction in software engineering. But the near term priorities must be (and are) on refining and transferring existing technical know-how to all portions of the defense program development community. This transfer will emphasize the specifics of how to best accomplish the policy initiatives described earlier. To achieve these technology transfer goals at the earliest time, strong management attention at the OSD level and within the Services, with adequate funding and priorities, is aggressively being pursued [12]. The Defense Department budget for fiscal year 1978 (FY78) requested $30 million for the software science and technology (S&T) program in support of the management initiatives, an increase of $14 million over FY77. Subsequent congressional action on the 1977 Defense Budget reduced this amount to $18 million, but actions are underway to gain Congressional approval to reprogram at least some of the deleted funding. The FY79 Defense Budget is less aggressive, but seeks considerable real growth to $30 million to support the continuation of these software S&T initiatives.

This increase in funding is principally targeted to coalescing and demonstrating new software tools, methods, and techniques to improve software quality and reliability, thus providing an early opportunity to avoid future cost growth. By using demonstrations, the value of various software advances can be verified and the application risks reduced to a point acceptable to program managers. These S&T activities are grouped into the three general areas described below.

A. Software Management Technology

To achieve the desired payoff from the newly established policy and procedures, specific guidelines to assist new program managers are a necessary first step. This is where the software S&T base plays a critical role. The policy directives described earlier are broad and general, and many specific questions still remain on how to implement these policies for each new weapons system development program. A particularly difficult problem is the choosing from among a variety of demonstrated alternative methods those software tools and controls best suited for a particular program manager's needs. Thus, a key role of the S&T program will be to survey the successful software methods, and place them into perspective from the program manager's viewpoint. One approach has been to develop guidebooks. The guidebooks which have been produced to date are intended to aid government and industry program managers and practicing programmers in areas such as documentation standards, data collection, monitoring and reporting software development status, statement-of-work preparation, life-cycle planning, program library specifications, chief programmer team operations, structured programming, and training materials [13], [14]. Additional guidebooks are in preparation on subjects such as reviews and audits; configuration management; requirements specification; verification, validation and certification; computer program maintenance; cost estimating and measuring; and software quality assurance. These "how to" documents should significantly assist in the implementing of the new software management thrusts, although good judgment by software developers will continue to be the most important ingredient. The guidebooks cannot be "cookbook" solutions, since there is still much to be learned in the science of software management. Continuing attention will be given to evaluating the utility of these many guidebooks. Once their utility is demonstrated, consideration will be given to incorporating them as contract provisions.

The program manager's choice of effective software management and control techniques is further complicated by lack of measurement criteria and experience data. The experience data that do exist lack consistency and prevent meaningful comparisons from being made. Thus, a key role of the S&T program is to catalog the merits of the various methods for aiding software management and production from requirements generation through operational test and evaluation. Comprehensive software data are being gathered from several recent projects that have used modern methods of programming, including the Aegis, Trident, DLG(N)-38, SSN-688, Viking Lander, Apollo, Safeguard, and Cobra Dane Programs. These data will be placed in a new Data Analysis Center for Software at the Rome Air Development Center (RADC), Rome, NY. The specific software management and production methods represented will be analyzed and compared, with good and bad points identified, lessons learned cited, and guidelines drafted for use by defense program decision makers. In addition, the most meaningful quality and progress metrics will be selected and proposed for future use in monitoring weapon system software development.

Other S&T thrusts in the Software Management Technology area are requirements analysis techniques and life-cycle management planning technology. The Air Force will continue to refine the Computer-Aided Requirements Analysis tool, while the basic tool is evaluated in Air Force, National Security Agency, Navy, and National Aeronautics and Space Administration (NASA) programs. Meanwhile, the Army will investigate the general utility of a requirement analysis concept developed under sponsorship of the Ballistic Missile Defense Program. Under the life-cycle management technology subarea, the Army has documented and will refine the key elements required of a computer resource (including software) life-cycle plan [10].

The efforts in the Software Management Technology thrust area will be focused initially at consolidating the current technology, and later on advancing the state of the art. In order to assist in this goal, the Defense System Software R&D Technology Plan [12] has established the role of a Principal Technology Agent (PTA). The PTA is responsible for monitoring all projects within the area of responsibility, keeping abreast of the state of the art, and advising Defense Department technologists, developers, and users on software technology matters. The PTA's for the Software Management Technology area are: requirements analysis—Air Force; life cycle management planning—Army; cost/quality data collection and analysis—Air Force; management control technology—Air Force; and policy and procedure guidance—Air Force.

Nineteen percent of the overall software science and technology program effort in FY77 was directed to the Software Management Technology area. This will remain at nineteen percent in FY78 and increase to twenty three percent in FY79.

B. System Architecture Technology

As the degree of automation in defense systems continues to grow, the needs for interoperability, distributed capabilities, and resource (data base) sharing also grow. As interconnections of software supported systems are examined, network (or system) control problems surface and needs for better architectural understanding are recognized. The technology investments in system architecture are focused on evolving techniques and criteria for evaluating potential merits of new processing architectures, and methodologies to optimize design characteristics such as among software/hardware/firmware. For example, in defense applications where processing functions are highly structured and not subject to frequent change (such as a flight controller), it may be more appropriate to replace software with specialized microprocessor hardware. The tradeoff methodology to support such decisions on a life-cycle cost basis is not currently available, but ongoing investigations are probing for insights into this deficiency area within all Services.

A second dimension of the architectural technology area is fault tolerance. In most defense applications, we cannot accept an approach of reloading and rerunning the program when a system fault interrupts or crashes the processing. Spacecraft, air defense, fire control, and aircraft navigation processing tasks are characteristics of such high reliability applications. Processing architecture designs must accommodate occasional hardware failures, and software must protect against hardware failures when possible. Parallel hardware redundancy investigations coupled with experiments in software failure detection monitoring and control are primary thrusts under Air Force sponsorship [15].

Another system architecture area of investigation concerns the emulation capability of modern microprogrammable processors. A number of deployed systems contain embedded computers which are rapidly becoming unsupportable and consequently obsolete. By emulating the instruction set and interfaces of the obsolete computers using modern higher speed lower cost minicomputers, swap-outs are possible without the expense and disruption of software conversion. Emulations of the AN/UYK-20 and AN/GYK-12 have been successfully demonstrated [16], thus making possible a reopening of competition for applications which have effectively been the exclusive domain of one contractor for many years. A new initiative will evaluate the feasibility of several standard military computer family architectures. Generation of vender independent specifications for current inventory embedded computers is a critical element of this project. Emphasis is to define form, fit, and function specifications at the embedded computer (black box) level. A more futuristic part of the program is exploring the technical, economic, and competitive feasibility of a standard computer architecture (instruction set, register composition, and I/O protocols) which could serve a variety of DoD applications [17]. The form, fit, and function concept is being investigated at both computer, as a packaged product, and internal module levels. Consideration is presently being given to establishing a jointly funded multiservice effort for this futuristic part of the architecture program, coupling the major application areas of space, avionics, command and control, telecommunications, and surveillance. The long range goal is to reduce the number of different machine architectures the DoD must support (both software and hardware support) while simultaneously preserving the ability to inject new technology over the life cycle within a highly competitive environment.

A fourth area of investigation involves system level interoperability. Many opportunities are available for the future, and requirements exist today to interconnect some of the computer systems which support more or less autonomous weapons functions. Such interoperability promises increased capabilities, but at a significant technical and operational risk today. The joint Navy/Defense Advanced Research Projects Agency (DARPA) Advanced Command and Control Architectural Testbed (ACCAT) at Naval Ocean Systems Center, San Diego, CA, will be used to explore intraship and intership computer system connectivity issues. Avionics computer system interoperability will be explored in the Air Force Avionics Laboratory's Digital Avionics Information System (DAIS) demonstrator, the Naval Air Development Center's Basic Avionic Subsystem Integration Concept (BASIC) testbed, and under the Army's Avionics Laboratory's Digital Modular Avionics Program (DIMAP). Bus Multiplex Standard 1553A forms the foundation for these avionics systems connectivity experiments and permits hardware to be interchanged among these demonstrators. Alternative command center interoperability protocols and interfaces for Army applications will be investigated using the Telecommunications Design Center facility at the Army's Center for Tactical Computer Sciences (CENTACS), Fort Monmouth, NJ.

A final thrust involves computer network security, particularly technology to permit protection of different levels of classified information being processed or stored concurrently. As the technology for merging distributed data bases matures, the need for concurrent (time shared) protection among various classified levels will rapidly become a major barrier to fully integrated command, control, communications, and intelligence operations. The need for resource sharing and the connectivity demanded in crisis management situations will dictate interoperability among defense system computers. Guaranteed separation among various national security classifications is not possible with today's computer architectures operating in time-shared modes. In addition, need-to-know protection is not now available. The primary thrust of the planned computer security technology activities is to document consolidated requirements, develop threat versus cost models, develop security metrics and security certification procedures, and explore technical approaches with the hope of influencing the next generation of commercial processor architectures to provide the desired level of security protection. The security kernel approach appears the most promising for the short term. It has been demonstrated on a PDP-11 minicomputer [18], the premise upon which it is based has been proven theoretically valid, and efforts are underway to implement the concept into the Bell Laboratories UNIX minicomputer operating system and the IBM 370 "Virtual Machine" operating system.

The Defense System Software R&D Technology Plan assigns Principal Technology Agent responsibilities for the System Architecture Technology area as follows: reliability and survivability–Air Force; interoperability–Army; advanced architecture concepts and tradeoffs–Air Force; computer architecture standardization and commonality–Army; testbeds and demonstrators–all Services.

Of the \$16M Software S&T program, 26 percent was invested in technology for the Systems Architecture area in FY77. This will increase to 28 percent at the FY78 program and grow to 40 percent of the FY79 program as the desire to advance technology is emphasized over consolidating technology already available.

C. Technology to Improve the Programming Environment

The two thrusts in this area are motivated by the desire to increase the productivity of software system designers, programmers, and testers. Since the software development process is a labor intensive operation, productivity increases translate directly into savings. Over the past decade, technology improvements have increased programmer productivity by nearly one order of magnitude. However, in the same time frame, hardware costs have dropped by over two orders of magnitude, and consequently software costs (design, coding, and testing) now constitute 70 percent of the automation costs for the average new embedded computer application. The two thrusts being pursued are standardization and advanced tools for software developers.

Actions have been taken to reduce the proliferation of HOL implementations. To improve the future supportability of software systems and increase the transfer of available software among new systems, seven HOL's have been selected under DoD Instruction 5000.31 and all new weapons systems development programs must use one of these HOL's: TACPOL, CMS-2, SPL-1, Jovial J3, Jovial J73LI, Fortran, or Cobol. Whichever language is selected, 100 percent of the programming must be done in the HOL, where constructs permit. If optimization is needed, then subroutines may be coded in assembly or machine language, but documentation will be provided at both HOL (functional) and implementation language levels. Each Service will be responsible for establishing configuration control for at least one of the languages. The initial formation and establishment of the control facility and the continuing advancement of the language and its tools will be provided by the software S&T program. The exact functions to be performed in language control are still under study, but a key role will be to verify and distribute compilers and maintain a repository of programmer aids and tools. The center may also serve as a repository for high leveraged software applications modules.

One experiment, the National Software Works demonstrator, will determine the feasibility of integrating a distributed network of dissimilar software tools to create a repository which is interactive and accessible via the ARPANET [19]. Control and accountability of the use of assets within the distributed prototype "software factory" are key technical issues being explored. Other related activities such as the Navy's System Design Laboratory and the Army's Integrated Software Development System, will investigate software development concepts previously demonstrated for particular systems and determine their value to Service-wide embedded computer applications. A variant of the Integrated Software Development System is also being applied under the Army Position, Location, and Reporting System Program.

A final effort under the standardization thrust is the investigation of the technical and economic feasibility of a modern common DoD HOL. A fully integrated set of language requirements has been painstakingly coordinated which is responsive to all DoD application areas (command and control, avionics, space systems, telecommunications, intelligence, surveillance, simulation and test, and evaluation) [20]. Existing HOL's have been evaluated relative to the requirements, and four competitive efforts are now underway to design a modern HOL based on the Pascal language. The risk areas are utility across broad DoD embedded computer applications and ease of use, or practicality. These concerns will be resolved in FY79, and if successful, the plans call for adding this modern language to the list of approved DoD HOL's, while simultaneously deemphasizing future use of several of the older HOL's. This HOL effort is also important because of the large number of deficiencies of current interim standard HOL's, particularly for real time I/O intensive applications.

A second major thrust involves the evaluation of advanced tools for software testing and validation. As described earlier, a data base capturing recent experiences using modern methods of programming is being established. Numerous lessons learned and trends should result from analysis of these data. Various metrics and test milestone criteria are under investigation to determine utility. Experiments with automatic generation of test case programs focused at fault detection of mission critical functions are in progress in order that these critical software faults can be discovered, isolated, and corrected early in the development cycle. Much work remains to be done in perfecting the many dimensions of the software test process.

As part of the second thrust, a major opportunity is automatic generation of software documentation. The more advanced management control techniques utilize a computer to monitor and record the status of software development. This same computer can be used by the programmer for diagnostic checks, traceability, tool application, or emulation of the system being developed. Thus, the programmer enters all his work into the computer via a remote console and the computer tracks the development progress, issues reports for project management, and assimilates the system documentation. If 10 percent more automatic generation of documentation were accomplished than at present, at least $10 million annually could be saved by the DoD. The results of the analyses conducted on software data deposited in the RADC Data Analysis Center for Software should also help resolve many of the procurement issues of "what data should be provided" and "how useful are the data." In the opinion of many DoD software maintainers, we procure too much data and what we get is not very useful.

Finally DARPA is conducting research into formal methods for software verification and maintenance. Today, there is no supporting body of knowledge which permits us to guarantee that a software package is error free. Typically, we test it until we run out of funds or time, and certify that it is correct for the inputs we applied during test. The ARPA investigations are seeking new foundations of knowledge which will lead to assertive procedures of correctness. Thus far, techniques have been demonstrated which permit software modules of less than 1000 lines of code to be "proven" correct. But much more work is needed.

The Defense System Software R&D Technology Plan assigns Principal Technology Agents for the Implementation and Maintenance Environment area as follows: formal methods of software verification and maintenance–DARPA; TACPOL language environment–Army; CMS-2 and SPL-1 language environments–Navy; Jovial language environments–Air Force; Cobol and Fortran language environments–Office of the Assistant Secretary of Defense (Comptroller).

In FY77, 55 percent of the software S&T program was directed to the Implementation and Maintenance Environment area. This will remain at 53 percent through FY78 and decrease to 37 percent in FY79 as the language control facilities are transferred to O&M accounts.

D. Testbeds

The fundamental obstacle to the transfer of new software technology to the developer is the establishment of confidence and credibility of new techniques or methods in a system context. For software, this involves not only showing that the new technology solves a problem or improves the product, but that it is better or cheaper than alternatives currently in use. The principal approach chosen to transfer software technology is through systems context demonstrators. Software "building blocks," as components of a system, can be evaluated in such testbeds as the Navy's Advanced Command and Control Architectural Testbed [21], the Air Force Digital Avionics Information System Testbed [15], the Army Telecommunications Design Center, the Defense Advanced Research Project Agency's National Software Works [19] demonstrator, or the Army's Tactical Management Information System Program. These testbeds also permit the military user to experiment with capabilities and they assist him in formulation of requirements. Of course, the final proof of the technology transfer process is the application of the new building blocks in weapon system development programs.

V. Conclusions

In this paper we have reviewed some of the more important policy actions that have been taken within the DoD. We have also described the essence of our technological thrusts which underpin the management practices. In varying degrees, these techniques are being applied to all current and new defense system programs. DoD is now getting to the point where meaningful impact can, and is, being made on the way it does business within the defense software community. DoD has great need for help in fulfilling our aspirations to "beat" the software problem, and there is no doubt that we will receive it.

Definitions of Terms Used

ACCAT	Advanced Command and Control Architectural Testbed.
ADP	Automatic Data Processing (synonymous with Electronic Data Processing as used here).
Aegis	An advanced shipboard weapon system.
AFAL	Air Force Avionics Lab.
AN/GYK	Army-Navy designation for ground based computer.
AN/UYK	Army-Navy designation for general utility computer.
ARPANET	A computer-to-computer communications network using packet message techniques.
BASIC	Basic Avionic Subsystem Integration Concept.
CENTACS	Center for Tactical Computer Science.
CMS-2	Computer programming language circa 1966.
Cobol	A business oriented computer programming language circa 1959.
Cobra Dane	Radar Surveillance System.
DAIS	Digital Avionics Information System.
DARPA	Defense Advanced Research Projects Agency.
DIMAP	Digital Modular Avionics Program.
DoD	Department of Defense.
DSARC	Defense System Acquisition Review Council.
FBM	Fleet Ballistic Missile.
Fortran	Scientifically oriented computer programming language circa 1955.
FY	Fiscal year.
HOL	High Order Computer Programming Language.
I/O	Input/Output.
Jovial J3	Jules own Version of the International Algorithmic Language, type 3.
Jovial J73LI	Jules own Version of the International Algorithmic Language, type 73 level I.
NADC	Naval Air Development Center.
NASA	National Aeronautics and Space Administration.
NOSC	Naval Ocean Systems Center.
OSD	Office of the Secretary of Defense.
O&M	Operation and Maintenance.
Pascal	Modern computer programming language circa 1968.
PDP	Computer marketed by Digital Equipment Corporation.
PLRS	Position Location and Reporting System.
PTA	Principal Technology Agent.
RADC	Rome Air Development Center.
R&D	Research and Development.
Safeguard	Ballistic Missile Defense Program.
SPL-1	Modern computer programming language circa 1974.
S&T	Science & Technology (early part of R&D).
TACPOL	Tactical Procedure Oriented Computer Programming Language.
UNIX	A software operating system for PDP-11 computer.
WWMCCS	Worldwide Military Command and Control System.

References

[1] "DoD weapon system software acquisition and management study," the MITRE Corp., Bedford, MA, Rep. MTR-6908, vol. I and II, June 1975.

[2] A. Kossiakoff *et al.*, "DoD weapon systems software management study," Applied Physics Laboratory, The Johns Hopkins Univ., Laurel, MD, Rep. SR-75-3, June 1975.

[3] J. H. Manley, "Findings and recommendations of the joint logistics commanders software reliability work group (SRWG report)," Headquarters, Air Force Systems Command (XRF), Andrews AFB, MD, Final Rep., vol. I and II, Nov. 1, 1975.

[4] J. Goldberg, "Proceedings of a Symposium on the high cost of software," held at the Naval Postgraduate School, Monterey, CA, DDC Accession Number AD-770 121, Sept. 17-19, 1973.

[5] "Proceedings: Software cost and sizing workshop," USAF Electronic Systems Division, Hanscom Field, MA, Oct. 1974.

[6] H. A. Richardson, "Abridged proceedings from the software management conference–First series 1976," sponsored by the Los Angeles Section of the American Institute of Aeronautics and Astronautics in cooperation with the Association for Computing Machinery, Washington, DC, Mar. 22-23, 1977, and Anaheim, CA, Apr. 5-6, 1977.

[7] W. L. Trainer, "Software–From satan to savior," in *Proc. NAECON Conf.*, May 1973.

[8] B. C. De Roze, "Defense systems software management plan," Office of the Assistant Secretary of Defense (Installations and Logistics), DDC Accession Number AD-A022 558, Mar. 19, 1976.

[9] DoD Directive 5000.1, "Major system acquisition," revision, Jan. 18, 1977.

[10] "Life cycle events," in *Software Acquisition Management Guidebook Series*, USAF Electronic Systems Division, Hanscom Field, MA, Tech. Rep. 77-22, DDC Accession Number AD-A037 115, Feb. 1977.

[11] T. H. Nyman, "Information processing technology," *RDT&E Technical Area Descriptions*, DDC Accession Number AD-C011 372, Apr. 1, 1977, pp. 279-298.

[12] T. H. Nyman *et al.*, "Defense system software R&D technology plan," R&D Technology Panel to the Management Steering Committee for Embedded Computer Resources, Office of the Under Secretary of Defense for Research and Engineering, Department of Defense, Rep., Nov. 1977.

[13] *Software Acquisition Management Guidebook Series*, USAF Electronic Systems Division, Hanscom AFB, MA:

"Monitoring and reporting software development status," DDC Accession Number AD-A016 488, Sept. 1975;

"Regulations, specifications and standards," DDC Accession Number AD-A016 401, Oct. 1975;

"Contracting for software acquisition," DDC Accession Number AD-A020 444, Jan. 1976;

"Statement of work preparation," DDC Accession Number AD-A035 924, Jan. 1977;

"Software development and maintenance facilities," DDC Accession Number AD-A038 234, Apr. 1977;

"Software documentation requirements," DDC Accession Number AD-A027 051, Apr. 1977;

"Computer programming configuration management," ESD Rep. TR-77-254, Aug. 1977;

"Software acquisition management guidebook: Software quality assurance," ESD Rep. TR-77-255, Aug. 1977.

Aeronautical Systems Division (AES), Wright-Patterson AFB, OH:

"Overview of software development and management," DDC Accession Number AD-A030 591;
"Software acquisition process," DDC Accession Number AD-A030 592;
"Summary of related standards and regulations," DDC Accession Number AD-A430 593;
"Technical aspects relating to software acquisition," DDC Accession Number AD-A030 594.

[14] *Structured Programming Series*, USAF Rome Air Development Center, Griffis AFB, NY, vol. 1–15, July 1975, DDC Accession Numbers follow:

"Programming language standards," DDC Accession Number AD-A016 771;
"Pre-compiler specifications," DDC Accession Number AD-A018 046;
"Pre-compiler program documentation," DDC Accession Number AD-A013 255;
"Data structuring," DDC Accession Number AD-A015 794;
"Program support library requirements," DDC Accession Number AD-A003 339;
"Program support library program specifications," DDC Accession Number AD-A007 796;
"Documentation standards," DDC Accession Numbers AD-A008 639 and AD-A016 414;
"Program design study," DDC Accession Number AD-A016 415;
"Management data collection and reporting," DDC Accession Number AD-A008 640;
"Chief programmer team operations," DDC Accession Number AD-A008 861;
"Estimating software resource requirements," DDC Accession Number AD-A016 416;
"Training materials," DDC Accession Number AD-A026 947;
"Software tool impact," DDC Accession Number AD-A015 795;
"Validation and verification," DDC Accession Number AD-A016 668;
"Final report," DDC Accession Number AD-A020 858.

[15] B. List, "A major crossroads in the development of avionics systems," *Astronaut. Aeronaut*., Jan. 1973.

[16] W. J. Kenny and C. D. May, "The CDC 480-AN/AYK-14(V) computer system," in *Proc. COMPCON Fall 1977, IEEE Comput. Soc. Int. Conf*., Washington, DC, IEEE Catalog 77CH 1258-3C, Sept. 6–9, 1977.

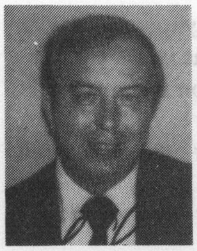
[17] W. E. Burr and W. R. Smith, "Comparing architectures," *Datamation*, Feb. 1977, pp. 48–52.

[18] W. L. Schiller, "The design and specification of a security kernel for the PDP 11-45," the MITRE Corp., Bedford, MA, DDC Accession Number AD-A011 712, May 1975.

[19] W. Carlson, "National software works," presented at the AIIE Conf. Distributed Syst., July 1977.

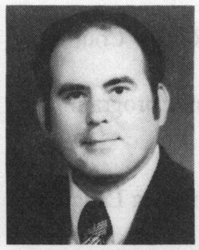
[20] "Higher order DoD requirements for computer programming languages–revised IRONMAN," DoD Higher Order Language Working Group to the Management Steering Committee for Embedded Computer Resources, Office of the Under Secretary of Defense for Research and Engineering, July 1977.

[21] F. Hollister, "Advanced command and control architectural testbed (ACCAT)," Defense Advanced Research Projects Agency/IPT, Arlington, VA, Sept. 20, 1977.



Barry C. De Roze received the B.S. degree in physics from Miami University, Oxford, OH, and the M.S. degree in aerospace engineering from Rensselaer Polytechnic Institute, Troy, NY.

He is currently the Manager of Defense System Software within the Office of the Secretary of Defense, Washington, DC. In this capacity he is responsible for DoD's policy, practice, procedure, and technology initiatives in software management, and for their implementation in weapon, communications, command and control, and intelligence systems. Prior to joining DoD, he was the Program Director for Software Reliability at Vitro Laboratories Division of Automation Industries. He was responsible for technology development and application, and for software verification and validation on a number of DoD systems. He was previously with the Corporate Systems Center of United Aircraft Corporation where he was concerned with the guidance and control of missile and space vehicles.



Thomas H. Nyman (S'61–M'75–SM'78) received the B.S. degree in electrical engineering from the University of Washington, Seattle, and the M.S. degree from the Massachusetts Institute of Technology, Cambridge.

He was Staff Specialist for Electronic Systems Technology with the Research and Advanced Technology Directorate within the Office of the Under Secretary of Defense, The Pentagon, Washington, DC, (Research and Engineering) (formerly DDR&E). His responsibilities involved Defense Department policy formulation, technical review and program evaluation of basic research, exploratory development and early advanced development activities in command, control, avionics, communications, and computer technology. Prior to joining DDR&E in 1975, he was with the MITRE Corporation, the U.S. Navy, and Bell Laboratories in programs involving sensors, navigation, communications, and weapons delivery. He is presently with the General Research Corporation, McLean, VA.