Core Resource Management for Large Real-Time Computer Program Development

Pravin L. Kadakia* and David J. Herman†
RCA Government Systems Division, Missile and Surface Radar, Moorestown, New Jersey

Large-scale real-time computer programs designed to drive major military systems regularly invoke astronomical requirements for core memory in their embedded computers. On the other hand, space and weight constraints in ships, aircraft, and land-based mobile units exert a strong counterforce to restrict the number and size of the computers. These contrasting demands underscore the critical need for aggressive control of core resources throughout the computer program development cycle for major weapon systems. This paper describes an approach to core resource management that has been applied successfully on the Navy's AEGIS Ship Combat System. The core management process begins in the system definition stage with estimates of computer program size based on system performance requirements, program architecture, and computer capability. The estimating process iterates throughout development and includes a budget of core reserves for anticipated growth. Stringent monitoring and control, invoked during the design and test stages, include both formal milestone reviews and frequent internal status reviews.

Introduction

SINCE the introduction of general-purpose computers into large-scale systems, the availability of computer core memory and processing time has been a major constraint in system development. In the commercial world, disks and magnetic-tape peripherals can usually be used to augment CPU memory capacity as needed, and distributed processing techniques and multiprocessor architectures provide other avenues to accommodate unforeseen processing requirements.

In contrast, the space and weight constraints on systems for use in ships, aircraft, and land-based mobile units prohibit the use of flexible but resource-consuming approaches available as add-ons in commercial systems. Once the basic mainframe system configuration is established, core and time resources for these large-scale real-time computer programs must be firmly established and controlled from the beginning of the computer program development process.[1-4]

This paper describes the derivation, uses, and benefits of a core resource management approach used successfully by RCA in the development of the AEGIS Ship Combat System computer programs.[5] This approach was applied for prediction and control of computer program size and development costs, and for accommodation of future computer program growth to support additional capability.

System Description

Seven major computer programs form the heart of the AEGIS Ship Combat System for the Ticonderoga, the first of a new class of automated, high-firepower, guided-missile warships being built by the U.S. Navy. Twenty-five individual systems performing detection, control, and engagement functions form an integrated combat system capable of responding to either a single threat or a coordinated multiple attack in adverse environments. The combat system simultaneously and automatically processes data from these systems, manages tactical doctrine, coordinates assigned missions, determines modes of operation, and controls target engagements with the appropriate weapons. The seven tactical computer programs require 774,000 words of core, the support programs require 1.8 million words, and an operational readiness self-test system that is integral to the system requires 1.6 million words.

Computer Program Description

The complexity of the seven tactical computer programs ranges from the real-time program for control of the phased-array radar (where computer execution time is critical) to the command and control, command display, and operational readiness self-test programs (where functional requirements result in critical core-allocation problems). The weapons control system and fire control system computer programs also have critical core and real-time requirements; the training program uses the combat system equipment and computer programs for training shipboard operators in system use.

The central computer suite consists of three standardized four-bay AN/UYK-7 computers supporting the radar control, command and control, and weapons control subsystems.[6] Each four-bay computer contains four central processing units, eight standard single-density memories, four double-density memories, and four input-output controllers, providing a total core of 262,144 32-bit words for each four-bay computer. In addition, a one-bay AN/UYK-7 computer supports the command display subsystem, one AN/UYK-20 (a minicomputer) supports the operational readiness self-test subsystem, and four AN/UYK-20s support the fire control subsystem.[7]

Table 1 lists the core associated with each computer-controlled operational subsystem. The methodology described in this paper to project and control core utilization was applied to both the AN/UYK-7 and AN/UYK-20 computers.

Core Budget Allocations and Reserves

The early design effort for the AEGIS Ship Combat System computer programs included preliminary estimates of the final sizes of the computer programs and allocations of program modules to particular computers. These allocation estimates were upgraded throughout the development period in an iterative process and were based on four criteria:

1) Specified functional performance of the subsystem (i.e., the computer program performance requirements).

Submitted Oct. 21, 1981; presented as Paper 81-2222 at the AIAA Computers in Aerospace III Conference, San Diego, Calif., Oct. 26-28, 1981; revision received Sept. 13, 1982.

*Principal Member Engineering Staff, Computer Programs Development, Naval Systems Department.
†Manager, Computer Programs Development, Naval Systems Department.

Table 1 Core requirements for computer program test configuration

Subsystem program	Computer	Resident core requirements, kilowords
Command display	One-bay AN/UYK-7	60[a]
Phased-array radar control (including training control)	Four-bay AN/UYK-7	221[a]
Command and control	Four-bay AN/UYK-7	205[a]
Weapons control	Four-bay AN/UYK-7	195[a]
Fire control	AN/UYK-20 (4)	40[a]
Operational readiness	AN/UYK-20	53[a]
Gun control	One-bay AN/UYK-7	50
Underwater weapon control	One-bay AN/UYK-7	46
Sonar simulator	One-bay AN/UYK-7	44
Interface simulator	Two-bay, 3-bay AN/UYK-7	213
Tactical data links	AN/UYK-20 (2)	15.5
Electronic warfare	AN/UYK-19	96

[a]Tactical computer programs (total 774,000 words).

2) A standard four-bay AN/UYK-7 configuration of CPUs, memories, and input-output controllers and backplane wiring that provides both memory-sharing among the various CPUs and allocation of computer resources.

3) A casualty configuration to ensure system survivability through a cable connectivity that provides both a primary and a backup communication channel for message paths from computer to computer and from computer to sensor/weapon system interface.

4) Provision in every computer for computer program growth to accommodate additional capability during the life cycle of the operational ship.

The accuracy of the initial core estimates varied directly with the estimator's level of knowledge of each computer program module, but all estimates were based on five reference points: 1) definition of the functional requirements as specified in the first issue of the computer program performance specification; 2) data taken from previous engineering development models; 3) estimates from previous, similar computer programs; 4) system analysis of critical functions; and 5) system-level simulation of reaction time, missile guidance, and weapons scheduling.

Initial projections of core reserves were based on contractual requirements and on previous engineering experience and knowledge. First was the contractual requirement established by the Navy for a 20% core reserve of total installed core delivered with the final operational computer program; this is called the Tactical Digital Systems Office (TADSO) limit.[8] The second estimate, based on management experience derived from the previous engineering model, provided a design growth reserve of 20%, composed of 15% growth reserve available for use through the development and qualification and acceptance test phases, plus an additional 5% growth reserve through multi-subsystem integration. Third, again based on previous management experience, was a 5% growth reserve to accommodate system integration during the production test phase. Figure 1 shows the application of these reserves to the task of managing core resources.

Core Resource Management During Development

Without careful management and use of special control techniques, the computer program configuration described in Table 1 would probably have exceeded the core capabilities long before the system became operational. Accordingly, a rigorous resource management program was instituted for control throughout the development process. The basic management and control process begins with a hierarchy of structured reviews invoked formally by the Navy on all major computer program development projects.[9] These reviews include a preliminary design review, a critical design review, qualification and acceptance testing, and system testing. These are described here in summary form to indicate both the purpose and scope of the formal reviews:

1) Preliminary design review—The purpose of this review is to assess and agree on preliminary functional requirements to be imposed on the computer programs, to establish broad performance requirements for the computer programs, and to delineate the basic design architecture. This review also establishes core estimates for each program module, based on a critique of the core estimation process, and identifies potential core-growth problem areas for special monitoring.

2) Critical design review—At this point the computer program requirements are solidified to the point that a comprehensive program design specification is available, along with a data base document. The design review panel reviews core counts for each module, using functional design flow charts and a special core estimate algorithm for estimating core. Core estimates and budgets are updated, with recommendations for design optimization to avoid excessive core growth. After successful review, coding is begun.

3) Qualification and acceptance testing—In addition to the actual testing, this milestone also provides for review of the code and the program module structure as well as the firm program description document and data base design. Once again, core estimates are refined to reflect system changes, and an actual core size is established. Further design simplifications are frequently undertaken to assure ample core reserve for unexpected growth when the programs are tested as part of the full weapon system.

4) System testing—Computer programs are firm at this point, with only minor modifications required to meet system requirements. If the resource management process has been implemented properly, there now remains a full 20% reserve of core for use as necessary.

These four checkpoints constitute the basic framework not only for core management, but for the entire computer program development process. Since these formal reviews are widely separated in time, they were supplemented by more frequent internal reviews that concentrated directly on core requirements. Because of the dynamism of core requirement fluctuations, reviews were scheduled biweekly throughout the early part of the development process, becoming monthly reviews only after solidification of the design at the critical design review. Internal reviews included representatives from the system engineering, project management, and programming activities. Thus, peer review was used successfully to smooth the vagaries of human estimation and to verify the analysis. The process provided monthly updates in a published document distributed to show the allocations, both real and estimated, on a module-by-module basis.

Figure 2 illustrates the repetitive nature of the review process used for resource management. These reviews make it possible to detect trends and potential problems in time to take corrective action that is meaningful and well considered.

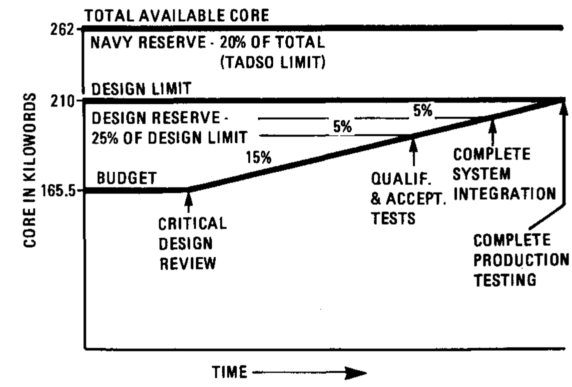

Fig. 1 Distribution of core resource budget and reserves.

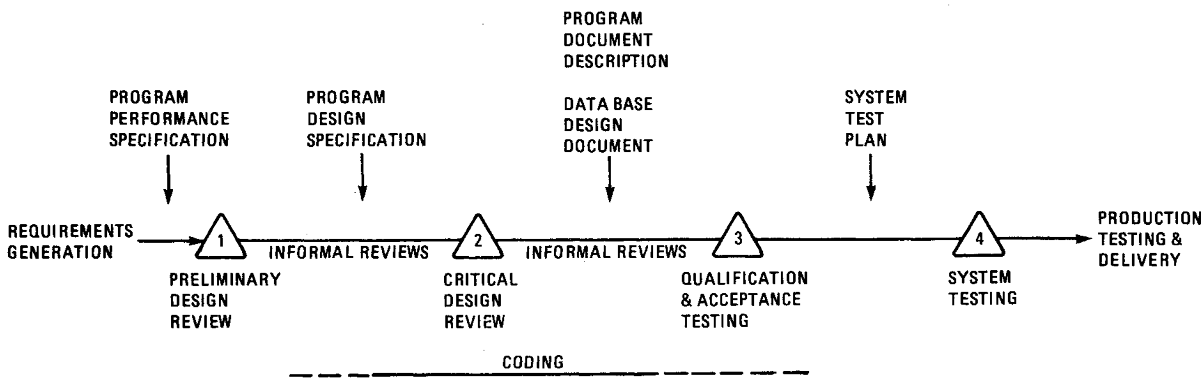

Fig. 2 Computer program developed cycle, showing major milestones and documentation requirements.

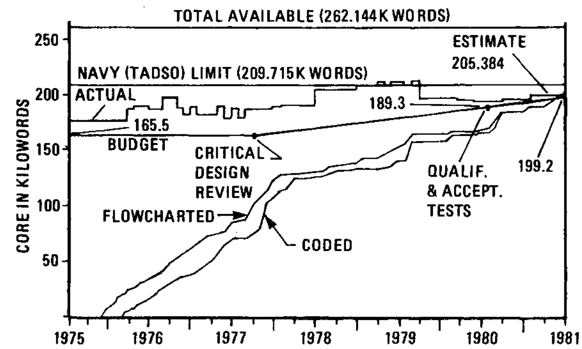

Fig. 3 Typical system element core management performance history (through June 1981).

Among the options used to control core-growth problems are the following: 1) redistribution of functions from one computer to another, 2) recording of modules for increased core efficiency, 3) reduction or modification of functional requirements imposed, and 4) addition of core memory.

The reviews also provide the computer program project managers with valuable experience, allowing them to predict problems through inferences drawn from past performance. Some typical observations:

1) No change in core requirements over a given review interval probably means that no thought has been devoted to core management.

2) A drastic change in core requirements usually indicates that disaster is imminent.

3) Any specification change should signal the need for an immediate review of core budget.

This repetitive process of resource management was documented and charted to provide management visibility at every point of the program.[10] A typical core profile for a major weapon system element during program development is shown in Fig. 3.

Development Cost Analysis

In an effort to predict the cost of developing or changing computer programs, a study was undertaken to derive an estimate of the number of equivalent source statements required to support real-time processing operations using CMS-2 compiler source language. Because of the varying degrees of complexity of the subsystems, a survey was made of the subsystem coding rates during the computer program production phase (from preliminary design review through qualification and acceptance testing). These data showed that coding production rates ranged from 85 to 120 equivalent source statements per man-month, depending on the complexity of the subsystem being coded. Modules typical of each subsystem were reviewed to ascertain the numerical relationship between equivalent source statements and object code composed of instructions, local data, and global data. The following algorithm was derived using weighted averaging techniques to reflect the coding complexity of the entire system:

$$\begin{array}{c}\text{No. of}\\ \text{equivalent source}\\ \text{statements}\end{array} = \frac{\begin{array}{c}\text{No. words of}\\ \text{instructions at}\\ \text{assembly level}\end{array}}{1.46} + \frac{\begin{array}{c}\text{No. words}\\ \text{of}\\ \text{local data}\end{array}}{3.28} + \frac{\begin{array}{c}\text{No. words}\\ \text{of}\\ \text{global data}\end{array}}{4.67}$$

Experience has proved this algorithm to be sufficiently accurate for day-to-day use. Using similar analytical techniques, the relationships applicable to other compilers can also be determined.

The algorithm is used in two ways. First, during the definition phase of computer program development, the algorithm is applied to estimate the size (in term of equivalent source statements) of the production computer programs, using functional flowcharts (i.e., counting the number of instructions at assembly level, and local as well as global data) as a basis for estimating module size. Second, during the design and code phases of computer program development, the number of equivalent source statements of the algorithm is used to size the impact (i.e., the relationship between equivalent source statements and man-months) of proposed changes to the functional requirements. In either case, development costs can be estimated by using the production rates mentioned above.

This tool has been valuable during the more than 10-year development cycle of the AEGIS Ship Combat System. As an example, a significant corrective action was required in 1979 to offset excessive core growth. (This problem and the action taken are shown in Fig. 3, where the actual core usage may be seen reaching and then exceeding the established limit, followed by a sharp drop late in the year to well below the limit.) The corrective action taken was basically a simplification of design requirements. The algorithm was applied to good effect in estimating both the cost of this effort and its impact on each computer program module.

Conclusion

Development of large-scale military real-time computer programs is an extraordinarily complex process that demands continuous and rigorous monitoring. From the system definition phase through qualification and acceptance testing, there is a constant danger of an explosion of core requirements that could render the evolving software system impracticable.

As a consequence, the core management process is a series of virtually unending reviews, with stringent monitoring of program modules, not only at formal reviews but also on a monthly or even biweekly basis. These frequent reviews afford project managers the visibility necessary to recognize the threat of core expansion, to control it, and to combat it.

The program described herein for core resource management is by no means the only available approach to the problem. It has, however, proved to be an effective software development tool, providing appropriate controls and excellent visibility of status and trends. These or any other tools for management and control of core and time resources are straightforward and available to any project manager. Their application, however, is a matter of team discipline and constant vigilance.

References

[1] "Computer Resources and the DSARC Process—a Guidebook," Dept. of Defense, Washington, D.C., April 1981.

[2] "Management of Computer Resources in Major Defense Systems," Department of Defense, Instruction 5000.29, April 1976.

[3] Trausworthe, R.C., *Standardized Development of Computer Software, Part 1 Methods,* Prentice Hall, Englewood Cliffs, N.J., 1977, pp. 355-360.

[4] Larman, B.T., "Spacecraft Computer Resource Margin Management," *AIAA Computers in Aerospace III Conference,* AIAA, New York, Oct. 1981, pp. 97-103.

[5] Howery, R.W., "Developing and Managing Large Computer Programs," *RCA Engineer,* Vol. 24, No. 1, June-July 1978, pp. 52-58.

[6] "AN/UYK-7 Computer, Technical Description," Sperry-Univac, Defense Systems," PX 5652E, FEb. 1973.

[7] "AN/UYK-20 Computer, Technical Description," Sperry-Univac, Defense Systems, PX10431B, March 1975.

[8] "Tactical Digital Standard (TADSTAND) D," Dept. of the Navy, 08Y/RJH T-9, Ser. 239, July 2, 1980.

[9] "Defense System Software Management Plan," Dept. of Defense, Aug. 1975.

[10] "Tactical Digital Systems Documentation Standards, Dept. of the Navy; SECNAVINST 3560.1, Aug. 1974.