63-3-3

SP-1092

SOFTWARE DESIGN AND IMPLEMENTATION

J W. Singleton

4 March 1963

(SP Series)

SOFTWARE DESIGN AND IMPLEMENTATION

SP-1092

March 4, 1963

J.W. Singleton

SYSTEM DEVELOPMENT CORPORATION, SANTA MONICA, CALIFORNIA

FOREWORD

This document reports the technical content of a project conducted in the summer of 1962 by

Richard A. Marciano
Robert R. Marshall
Stanley Sadofsky

in which Mr. Singleton participated.

Preparation of the report was materially assisted by Marvin Adelson and Lee S. Christie. Suggestions for illustrations were made by Herbert D. Benington.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The System Development Corporation has participated in the design and implementation of some sixteen command and control systems for varied agencies of government. Through this participation, we have achieved 10,000 man-years of experience in the field. We believe that this experience affords insight into the management as well as the technical aspects of the system design process. In the Spring of 1962, SDC established a project to examine the history and synthesize this process. This report represents the result of the project.

To put the process of software design and implementation in context, we should review the recent past. Over the course of the last two to three years, the government has sponsored a series of concept studies whose purpose has been to identify and define distinguishing characteristics of the military command and control system. Quite a number of these studies have been conducted--our own count totalling fourteen. Table 1 lists four of these studies, selected because they are relatively well known and have been influential; e.g., the Winter Study Group Report was quite widely briefed; the IDA study on Computers in Command and Control, often called the Kroger Report, has been widely distributed in document form. SDC participated in these four studies, as well as in many of the others not referenced.

Two points can be made about these concept studies. First, they have all been conducted recently. Those of us working close to the field tend to have the impression that the concept of the military command system has always existed. It is instructive to observe that such concept studies have been available only within the last three years. Too little time has elapsed and too little experience has been accumulated to satisfy our requirement for adequate understanding of the military command system. There has been, correspondingly, little experience from which to develop a body of knowledge and a set of dependable management principles to assure effective control of the military command system development process.

WINTER STUDY GROUP "Final Report"	15 September 1960
AF SCIENTIFIC ADVISORY BOARD "Human Factors in Computerized Systems"	24 February 1961
IDA/INS SUMMER STUDY "Summary Report on Command and Control"	15 September 1961
IDA/RESD "Computers in Command and Control"	November 1961

TABLE 1. Four Recent Concept Studies of Military Command and Control Systems

Second, the four studies shown in Table 1 exemplify the remarkable agreement that all of the relevant studies exhibit regarding concepts, conclusions, and recommendations about military command systems. All agree on basic principles of management and technical design. This consensus strongly suggests the identity of fundamental characteristics in command systems from which conditions necessary for their effective development can be determined. These characteristics can be summarized as criteria that distinguish the command system from the military weapons system.

DISTINGUISHING CHARACTERISTICS OF COMMAND AND CONTROL SYSTEMS

Table 2 shows the characteristics of both the weapons system and the command system as these relate to the criteria selected.

	Weapons System	Command System
HARDWARE ART	Critical	Incidental
COST/EFFECTIVENESS	High	Low
FUNCTIONAL INTEGRATION	Low	High
EMPLOYMENT	Mixed Temporary	Single Continuing
USER	Incidental	Critical
EVOLUTION	Low	High

TABLE 2. Comparison of Six Criteria Characterizing Command and Control Systems

It is generally agreed that continued new developments in the hardware state of the art are important to future weapons systems, e.g., in propulsion, guidance, communications, etc. This is not the case for command systems. All of the concept studies agree that present capability in computing techniques and communication devices is adequate to support effective application of present technology to the problems of military command. Additional technological development does appear to be warranted in the display field, but even here the hardware art does not constitute an obstacle to providing information processing assistance to military command organizations. The state of the art is far ahead of the finesse with which we apply existing technology.

Another criterion is the ratio of cost to potential effectiveness of the weapons system and command system. A number of the studies have demonstrated that command systems have a lower or smaller cost/effectiveness ratio than weapons systems. This is due to the fact that command systems cost less and have a potentially longer useful life than special purpose weapons systems.

An important finding of the concept studies is that functional integration is the major challenge in applying command system technology to current problems of military command. By functional integration we refer to things--such as the concepts, doctrine, and language of command--necessary to assure that the several operational military headquarters do, in fact, contribute to a unified and effective national military command capability. This is in contrast to the weapons system integration problem which is primarily technical in character.

With respect to employment characteristics of the two classes of systems, quite a sharp distinction can be noted. It makes little difference to an interceptor aircraft or a bomber or a missile squadron whether operational control is assigned to SAC or NORAD or EUCOM; the weapon will work equally well. It makes a great deal of difference to a NORAD COC to know and incorporate the characteristics of the operational military organization. In recognition of this point, we describe the command system as providing information processing assistance to the using military headquarters. The command system supports a single military user and supports the continuing mission responsibilities of the user. It is not the objective of the system to substitute for the exercise of military responsibilities by the commander and his staff.

A corollary to this employment characteristic of the command system is that the using organization itself must exercise a strong and continuing influence in the design and implementation of the system. It can do this through deliberate and controlled application of the concept of evolution. If any one word can be chosen to exemplify the recommendations of the concept studies on command systems, it is evolution. This is the fundamental principle of technical design that applies to the command system development process. Clearly, evolution refers to software, not to hardware. We have tried to develop a complete and meaningful definition of software in order to demonstrate what evolution encompasses and how it takes place.

Software is made up of three elements, listed in Table 3 on the following page. The set of software products, which are developed, released, and employed operationally in day-to-day use of the command system, includes

a. design documentation such as the Operational System Description and Operational Specifications;

b. computer programs and their supporting descriptive documentation for both operational, utility, and support functions;

c. operator methods and procedures handbooks;

d. orientation materials; and

e. system exercising and training materials that enable the commander to conduct exercises under stress conditions and loads that the system must be prepared to meet.

Software products are the output of an orderly set of design phases or activities, performed through the techniques of analysis, experiment, and test. Descriptions of the design phases and design processes constitute much of the balance of this document.

PHASES	Advanced System Planning and Analysis	
	Software System Requirements	
	Software System Design	
	Software Production	
	Orientation and Installation	
	Operations	
PROCESSES	Design Analysis	
		Operations Analyses
		Configuration Studies
		Feasibility Studies
		Trade-off Analyses
		Current System Constraints
	Operational Experiments	
	System Test	
PRODUCTS	Operational System Description	
	Operational Specifications	
	Computer Programs and Documentation	
		Operational
		Utility
	Methods and Procedures Handbooks	
	Exercising and Training Materials	
	Orientation Materials	

TABLE 3. Software

What is the purpose of software? It is to turn a general purpose capability into a specially tailored instrument to support the decision responsibilities of the using military organization. The general purpose capability typically includes a computer, a communications network for the receipt and transmission of data, and an operations personnel crew on duty station. The information processing functions performed within this capability are controlled by the software in some fixed or variable sequence that can itself be modified to support the changing responsibilities that constitute the mission of the command.

We do not intend that the terminology used in Table 3 be considered sacred; however, it has proved to be applicable and meaningful in a wide variety of systems. We do believe that the design phases identified here, and the software products resulting therefrom, represent the minimum essential set to assure effective software development and adequate operating capability in the completed system. **Management responsibility must be established for each of the design phases and software products.**

DEVELOPMENT OF COMMAND AND CONTROL SYSTEMS

Let us now place the software design and implementation process in perspective of the total development framework for a command and control system. Figure 1 is a simplified schematic of this framework, emphasizing the relationship that the using military command holds to the development process and the concurrent design that must go on within the development process between the software design agency and the hardware design agency or agencies. The hardware design process is shown as "configuration design" to suggest that, ordinarily, little

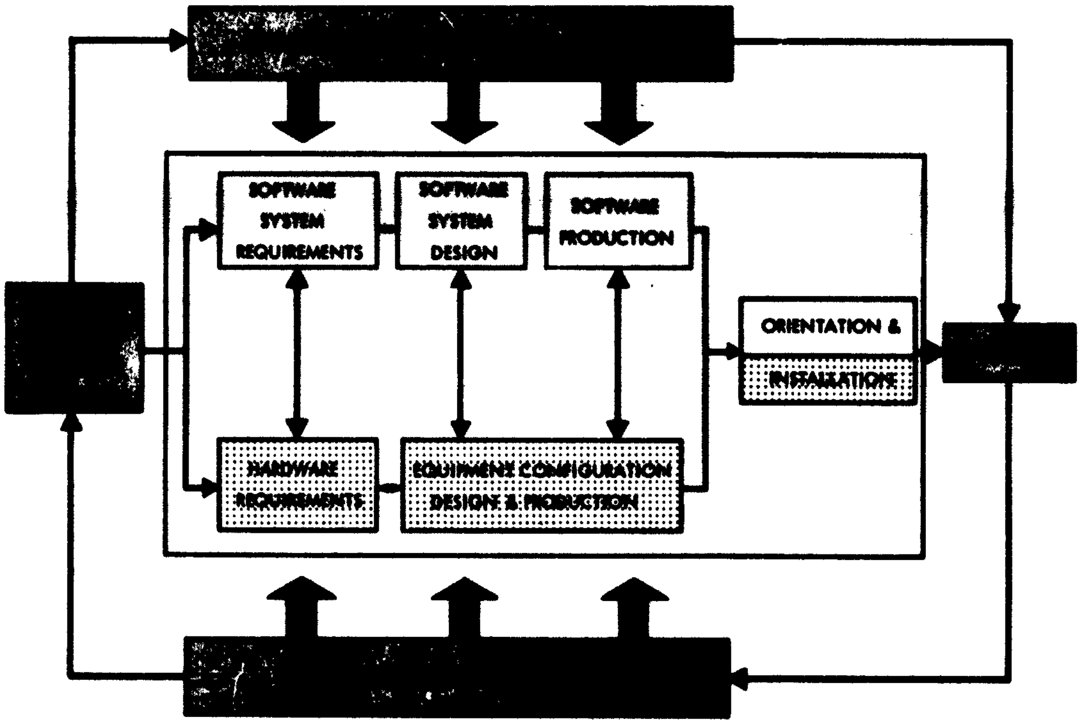
SOFTWARE SYSTEM REQUIREMENTS
SOFTWARE SYSTEM DESIGN
SOFTWARE PRODUCTION
ORIENTATION & INSTALLATION
HARDWARE REQUIREMENTS
EQUIPMENT CONFIGURATION DESIGN & PRODUCTION

FIGURE 1. Development of a Command and Control System, Static Representation

or no R and D is required on equipment components and techniques. Rather, the major design task is to select, from among existing components and techniques, that set which most nearly matches the user's needs for information processing capability.

RESPONSIBILITIES OF THE USER

Notice that the using military command completely bounds the system development process. Execution of advanced system planning and design, and technical management and design control, are not always considered the user's responsibility. The military command may not possess the in-service talent to perform these functions. It is often desirable to create this talent by augmenting in-service personnel with trained professionals who will constitute the user's technical representatives. However, it is vital that the user maintain the responsibility, and exercise such authority as necessary, to assure that the development process is controlled by his operational needs. Clearly, it is the user's responsibility to integrate the system into on-line operations and thereafter to conduct continuing exercise and evaluation, which will enable existing capability to serve as the operational base for growth and evolution.

DYNAMIC ASPECTS OF SYSTEM DEVELOPMENT

The schematic form of representing the system development process used in Figure 1 has been called the "wiring diagram" approach. This is useful to highlight characteristic design phases of the process and to identify the points of inflection at which the user's technical representatives for design control can conduct review and concurrence. At the same time, this representation implies that the process is static, whereas in fact it is highly dynamic. To portray the dynamic character of the development, Figure 2 presents essentially the same information in phasing chart form. This form displays the ripple effect that takes place in command system development and better represents the continuity and integral character of successive stages of software design and implementation.

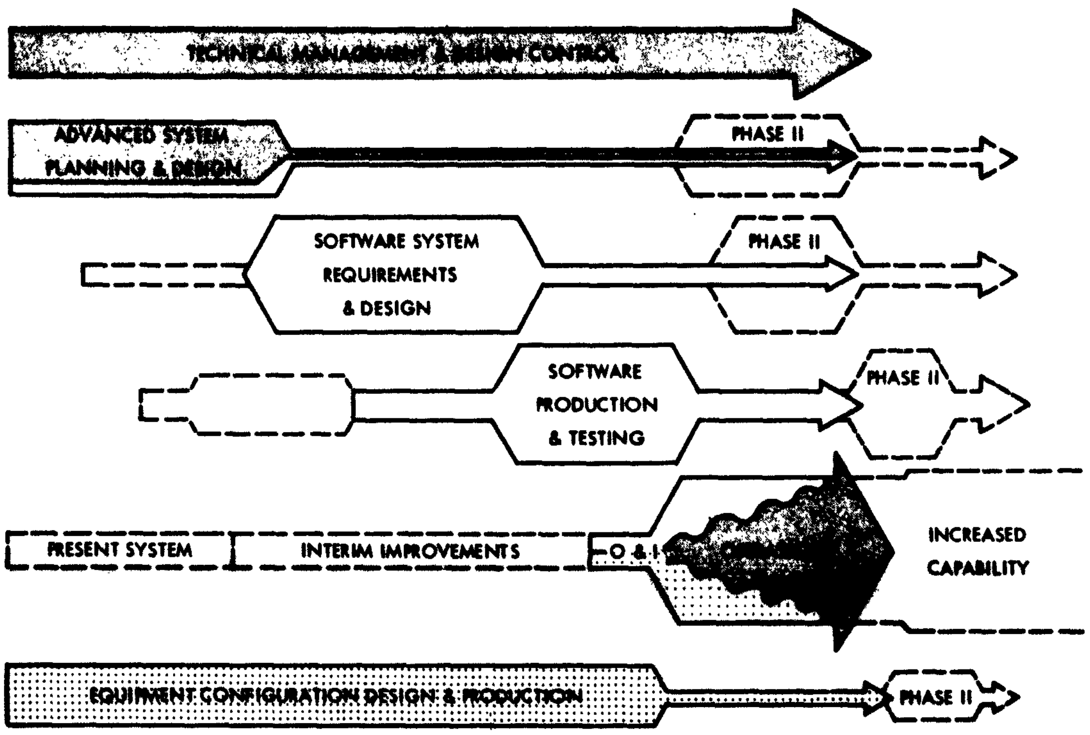
TECHNICAL MANAGEMENT & DESIGN CONTROL
ADVANCED SYSTEM PLANNING & DESIGN
PHASE II
SOFTWARE SYSTEM REQUIREMENTS & DESIGN
PHASE II
SOFTWARE PRODUCTION & TESTING
PHASE II
PRESENT SYSTEM
INTERIM IMPROVEMENTS
O & I
INCREASED CAPABILITY
EQUIPMENT CONFIGURATION DESIGN & PRODUCTION
PHASE II

FIGURE 2. Development of a Command and Control System, Phasing Chart

Participation by Software Designers. We have found it important for the software design agency to participate in the user's advanced system planning and design. Typically, three percent of the total professional manpower that SDC has applied to command system software development has gone into that activity. Although this has been only a small proportion of the SDC effort, and only a small proportion of the total effort applied to that phase, we have found it to be extremely important. It provides an understanding on the part of the software designer of the operational problems faced by the using command and, at least as important, such participation establishes identification and communication between the user and those responsible for continuing phases of the software design.

Development Continuity. The phasing chart also enables us to emphasize the importance of continuing a capability (in nucleus form) through each design phase. Such continuity provides a tangible basis for evolutionary growth. Development of a military command system does not cease once a single development cycle has been accomplished, but leads naturally and desirably into a second phase resulting in increased capability to the using command. The chart deliberately shows a compressed time span within which the phase II design and development takes place. Our experience has indicated that once an operational capability exists, additional requirements are easily and quickly identified and implemented in the command system technology.

Spin-Off Capability. In addition to showing initial and succeeding phases of the typical software development process, the phasing chart also permits us to point out the desirability of installing early "spin-off" capability in the present system. The purpose of deliberately identifying such spin-off is two-fold: to improve present command operations in critical areas, and to lead to a smoother transition from the present system to the more fully automated operations capability.

We will not discuss the characteristics of Advanced System Planning and Design in this presentation. Rather, let us turn to the process of software system development in order to identify the major milestones in the development process: requirements, design and production.

Software system requirements are established through analysis of the operational objectives for the system. The purpose of this analysis is to translate from the operational language in the military statement of requirements into implementation language, that is, to translate from objectives to information processing tasks. The translation process is performed by the technique of software system design analysis. Required command operations capability is analyzed to identify the information processing tasks required to support it. The resulting information processing design is compared with the contemplated or proposed equipment configuration being worked on concurrently. Alternative software designs are then subjected to feasibility and trade-off analyses to assure an effective match between the total system configuration and the operational objectives that the command system is to fulfill.

Before a proposed software design is accepted, it must be reviewed by personnel familiar with the current system. In this connection, we look upon the current system as a constraint. Every military organization has certain characteristics, certain traditions, certain directions of growth, a certain tolerance for change. It is essential to assure that the proposed software system design fits within these constraints of the user's organization. At this point, it is feasible and desirable to identify early spin-off features that can be installed in the present system. Some of these, like suggested modifications in SOP's, may come free. Others may cost something, but will be worth the effort to provide early capability.

We have found it necessary to emphasize the importance of design analysis because, occasionally, we still meet what we call the "throw it over the fence" philosophy. This philosophy holds that it is possible to begin detailed implementation--in this case computer program production--solely on the basis of an initial statement of operational requirements, with the expectation that an effective system will result. We try not to judge whether the philosophy of "throw it over the fence" is appropriate for weapons or sensor systems, or even for equipment portions of a command system. It is perfectly clear, however, that such a philosophy doesn't work for software. Operational language is different from software implementation language, and no one speaks both languages equally well--a continuing process of translation is essential. Besides, when operational objectives are established, all of the information required for implementation either may not exist or may not be recognized. Rather, it is developed--by discovery or invention--as necessity arises during the software system requirements phase.

The design phase that we have described results in preparation of the Operational System Description. The OSD contains the translation from operational language to implementation language and identifies information processing tasks that lead to more detailed design. Contents of the OSD are illustrated in Table 4.

- IDENTIFY COMMON INFORMATION PROCESSING CHARACTERISTICS
- LOGICAL DATA-FLOW PLAN
- DEFINITION OF COMMAND-LEVEL TASKS
- EVOLUTIONARY PLAN
- ADDITIONAL DESIGN GUIDANCE REQUIREMENTS

TABLE 4. Operational System Description

Note that the first requirement is to spell out those information processing tasks that have been identified, especially those that are common to several of the system operational objectives. We might illustrate this point with an example. Most readers will be familiar with air defense systems in which an important information processing task is that of tracking aircraft. It is necessary to track two major classes of aircraft: target aircraft (i.e., potential unknowns or hostiles) and interceptors that are committed to identify the targets. In every air defense system we're acquainted with, the logic and techniques of tracking are common both to processing target aircraft radar returns and to processing interceptor aircraft radar returns--even though the operational objective is quite different in the two cases.

The next major requirements of the Operational System Description (Table 4) are:

a. developing the data-flow plan, and
b. defining command-level tasks.

Information processing tasks should be flow-charted and traced logically through all processing steps to assure that the design passes the test of completeness. Then the tasks should be defined in accordance with the command organization that represents the user's plan for his system operations crew.

Table 4 next lists an evolutionary plan as a requirement of the OSD. By reference to an evolutionary plan, we mean that the Operational System Description should contain not only a software design for the present development cycle, but also a tentative plan, for accommodating probable growth requirements, that will constitute the design of successive evolutionary phases. The advantage of providing for such a plan is to call attention to anticipated future requirements, thus facilitating creation of meaningful schedules for continued evolutionary system improvement.

Finally, the OSD should include additional design guidance requirements (Table 4). We have always found that performance of the software system design analysis reveals both omissions and ambiguities in the user's Functional System Design. Provision should be made for continued authoritative interpretation of the initial design and also for correction of the loopholes that will have appeared as the Operational System Description is prepared.

Completion and publication of the Operational System Description constitute one of the significant milestones in the software development process and should serve as a review and concurrence item to the user's representatives for technical management and design control. It is extremely important that this review and concurrence be well planned, be well attended, and be concluded with the feedback of useful direction to the software design agency. This is precisely the sense in which Figure 4 was drawn, showing an input from the technical management and design control function to the Operational System Description, even though there will be no results at this time from the conduct of operational experiments.

The OSD initiates work on the next phase of the software development--software system design--whose document product is the Operational Specifications. The characteristic of this phase is to allocate information processing tasks to the several elements or components of the software system. These elements include the operational computer programs, the operations personnel crew, the displays and intervention controls that interface between these two, and the supporting data base from which both the computer programs and the operations crew derive the information and decision alternatives with which to perform their tasks.

The allocation of information processing tasks to these elements of the software system is again performed through the technique of design analysis. In this case analysis is required to establish and verify a preferred plan for task allocation. While the analytic techniques are comparable to those discussed in conjunction with the Operational System Description, they are concerned at this stage with software elements only and therefore operate at a reduced scope and criticality.

Software design analysis is substantially augmented and assisted by the conduct of operational experiments. The objective of operational experiments is to assure participation of the user in a responsible design capacity. There are several techniques by which operational experiments for design can be conducted. For example, we have successfully administered system exercises for design purposes rather than for training, thereby achieving the participation of present operations personnel within a command headquarters. Another technique is to have using command personnel participate in paper exercises, referred to as "scenarios." A third and highly effective technique is to establish an experimental facility in which members of the using command serve as subjects in functional simulations designed to test the critical task allocation alternatives that have been identified in the preceding analysis. As this statement implies, it is necessary that the facility be accessible to the user--ideally, adjacent to or near the present command headquarters.

Product of the software system design is the Operational Specifications, a design document. Table 5 illustrates characteristic contents of the Operational Specifications.

- ALLOCATION OF TASKS
- RULES OF INFORMATION PROCESSING AND COMPUTATION
- EXERCISE AND TRAINING DESIGN SPECIFICATIONS
- SYSTEM TEST SPECIFICATIONS

TABLE 5. Operational Specifications

The contents include a description of the tasks allocated to each element of the software system, together with a detailed specification of the rules of processing and computation that are to be followed for each task. By rules of processing and computation we refer to such things as:

a. capacities, rates, and routing of information;

b. specification of the data base, including data formats and parameters;

c. display formats and operator intervention controls;

d. the job design and manning specifications around which the operations crew can be organized and trained.

We have found it very important to include in the Operational Specifications the design of the exercise and training capability that is to be produced as an integral part of the operational system. The purpose of such a capability is to enable the using command to exercise, train, and evaluate the readiness of the operational command system. Design of exercise and training features is specifically directed toward high load and stress situations which are not ordinarily encountered in day-to-day peacetime activities. Regular system exercises also identify weaknesses in present operations, which require continued improvement in the process of evolutionary growth. We have been impressed by the acceptance achieved by the concept of integral system exercise and training in most of the command and control systems under development.

Finally, Operational Specifications (Table 5) should specify criteria and conditions under which tests of the completed software system will be conducted. Tests of the operational computer programs and the display/operator controls interface are particularly crucial. We put the system test specification here to assure that the people who have specified the design characteristics of the software also go on to set out test criteria and conditions under which performance of the software system will be evaluated. All too often test criteria are employed that measure a set of performance characteristics completely different from those intended in the design, because different people at a later time set up the tests. This point is well illustrated by the experience of the British in the early years of World War II.

Convoys of British merchant ships sent into the Mediterranean were being subjected to heavy attack from German and Italian aircraft based in Italy and North Africa. The British War Cabinet was quite concerned about the mounting loss of ships and cargo. The recommendation was made and accepted to install anti-aircraft guns on the merchant ships themselves. After a time, an investigation of the effectiveness of the installations was prompted by the increasing scarcity of trained personnel and anti-aircraft equipment with competing priorities for the resources committed to the merchant ships. The criterion used at the outset of the investigation was: How many enemy aircraft have been shot down by merchant ships? It was learned that hardly any enemy aircraft had been shot down, and work was started on retrieval of the equipment and personnel from the merchant fleet for other applications. Before this was done, the <u>right</u> question was asked: What had been the attrition experience of merchant convoys equipped with anti-aircraft guns, compared to those without them? Evaluation of the answers to this question revealed that merchant convoys with anti-aircraft installations had experienced an attrition rate of 10 percent; merchant convoys without anti-aircraft guns had experienced an attrition rate of 25 percent. In short, according to the right criterion, the installations were well worth the cost. This fact was obscured by the test criterion first proposed.

Operational Specifications, often known as Op Specs, constitute another significant milestone in the software development, at which review and concurrence are required by the user's representatives for technical management and design control. After review and revision, the Op Specs become the basis for the final design phase of the software process, the software component design. The characteristic of this phase is detailed internal design of each of the elements of the software system that we have described.

With respect to operator methods and procedures handbooks, component design is equivalent to production. Once the detailed job analysis and job design necessary for specification of operator procedures have been accomplished, the only remaining production task is that of writing it all down in handbook form. In contrast, the operational computer programs require substantial production coding after the component design. The same is true for exercise and training features of the system, and for some of the orientation materials. Because the operational computer program system looms large in this phase, we can use it for purposes of this discussion. Typically, 50 percent of the professional manpower required for the complete software job in a command and control system is applied to computer program component design and production.

Preparation of the component design of the computer programs is again supported by design analysis.

An example of the kind of design alternative appropriate for analysis at this stage can be drawn from the early days of SAGE. The design decision was made to produce the first model of the SAGE computer program system in four packages--an air surveillance package, an identification package, a weapons direction package, and a duplex or switchover package that would instantaneously transfer to the standby computer if the active computer failed. The purpose of this approach was to facilitate checkout and installation of significant portions of the program system.

One of the consequences of this approach was to require a separate computer program capability in each package for display and console switch interpretation. Once all four packages were integrated, installed, and operational in the early SAGE sectors, there was no requirement to continue this concept of separate display and switch interpretation for each program package. In fact, there was substantial concern that this design characteristic was inefficient in its use of internal computer storage. Therefore, one of the early design proposals in evolution of the SAGE System was to integrate display and switch interpretation routines into a common processing routine. Mathematical analysis was performed, supported by simulation, demonstrating that technical efficiency would be obtained by the proposed integration and that no degradation would be incurred in other performance characteristics of the system. Scheduling considerations, influenced by the user's representatives for technical management and design control, resulted in the change being implemented in Model 6 of the SAGE computer program system.

Product of the software component design will be a set of design documentation. In this case, we refer to the computer program design as the Program System Description. Typical contents of the Program System Description are shown in Table 6.

- COMPUTER PROGRAM FLOW DIAGRAMS
- PROGRAM DESIGN STRUCTURE
- DATA ORGANIZATION
- STORAGE ALLOCATION
- TRANSFER FUNCTION

TABLE 6. Program System Description

This document should start with the detailed flow diagram sequence of processes to be performed by the computer programs, leading to the detailed organization and internal structuring of the program system. Again, we face the problem of introducing new terminology at this point. Instead of talking about information processing tasks we will now be referring to internal computer program routines such as bookkeeping, table make-up, simulation, display generation, data recording, executive control, etc. The need to understand this translation into new terminology must be met by the user's representatives for technical management and design control, and calls for continued close interaction with software implementation personnel.

Storage allocation of the computer program in the machine must be specified. Consequent trade-off implications must be analyzed between storing both data and control programs in internal core memory, in contrast to storage in auxiliary memory such as tape or disc file. Computer program data organization is determined and influences storage allocation design. Finally, the transfer function is prescribed. The transfer function constitutes the coding specification from which production coding of the programs is accomplished.

At, and after, the point of coding specifications, we find application for the so-called higher order computer programming languages, often referred to as procedure oriented languages. Development of procedure oriented languages is one of the most promising technical breakthroughs achieved in software state of the art. Procedure oriented languages offer powerful benefits in the production coding of large-scale computer program systems and also in subsequent maintenance and retrofit of additional design features to programs written in procedure oriented language.

The Program System Description constitutes another review and concurrence item requiring participation and feedback by the user's representatives for technical management and design control. Production coding of computer programs follows this concurrence. Testing of the computer programs and related software features takes place according to the conditions and criteria set forth in the previously described Operational Specifications.

Let us summarize the software development process (Figure 1). First, we have described the design and implementation of software as an orderly process, depending heavily on continuity of professional talent and the integrity of successive design phases. Second, we have shown that software development can be planned and managed by the user to assure satisfaction of command operational objectives. This will not happen by accident. To achieve this result, the user must have a technical staff competent to represent him in the functions of technical management and design control. The user's technical representatives must concentrate efforts on the milestone points in software design and implementation, which are appropriate for design review and concurrence.

The reader will have recognized that our discussion thus far has been idealized. Software design and implementation always take place in the real world, and we have never yet had the experience of completing a development cycle without perturbations being introduced from the real world. Such perturbations are a natural part of command system development and, while some may represent only noise, others will represent signal and should be accommodated. We make provision for evolution during development just as we make provision for evolution during operations. Examples of perturbations that represent signal rather than noise include changes in equipment capability or availability being planned in the concurrent hardware process, changes in funding and scheduling of the system, and changes in the character of the using command (either in mission or in command organization). The latter point is often overlooked. For example, the commander of one of the major combatant commands has the reputation of possessing an omnivorous appetite for technical detail. Another has the reputation of delegating much of the day-to-day detail work of the command to his staff. Clearly these characteristics of the commander are important and influence command expectations of an information processing system. When command succession takes place, the expectations of the new commander must be determined and implications to the command system evaluated. This is not to say that technical principles of design should be subverted to suit individual taste; but neither can the commander's requirements be ignored.

Changes in command force capability, strategic concepts, or intelligence regarding the threat must be continuously evaluated and reflected in the development process. If a sufficiently severe perturbation occurs, it may be necessary to move the development cycle six months downstream, review the Operational System Description and the Operational Specifications, and assure that significant implications of perturbations in the relevant real world are appropriately reflected.

A word about the phases of system orientation, installation and operations. First, installation and orientation constitute a crucial part of the development process, one that can make or break the effective use of the command system. This phase includes the transition from development to operations, and requires the introduction to the system of new personnel within the using command. These personnel, including but not limited to the operations crew, not only lack the background of prior experience with system design objectives, but also may lack the motivation that the developers have had to make the system work. Therefore, as much effort and planning as can be devoted to installation and orientation activities will prove beneficial. In particular, one of the things we have found useful in assuring a successful installation phase is the conduct of pre-operations training exercises; one of the software products made available for use in this period is the exercise and training system. Preparation and administration of a series of simulation exercises permit system operations to experiment, to learn, and to establish confidence in the information processing system without the press of critical operations decisions upon them.

The second point to be made concerning installation and orientation is that the characteristic of effective application of a command system is the establishment of confidence on the part of the using command in the information processing techniques and technology that are now available to them. One does not establish confidence in a new capability overnight, yet this is precisely what the conventional military concept of an operations date implies. The operations date concept may be necessary for administrative purposes, i.e., it signifies transfer of management responsibility from developers to operators, and is reasonable with respect to hardware. It is not reasonable with respect to software. Software development evolution and change goes on beyond any arbitrary operations date. Recognition of this fact is spreading; one hears such terms as "ultimate system," "final system," "complete operational capability" less and less. Perhaps the best way to describe the integration of a command system into the operational inventory of the using military organization is as a process of transition from an off-line capability into an on-line function.

Even this transition involves a residue of off-line capacity on a continuing basis for maintenance, design, and development of new capabilities, and for operations data reduction and analysis.

Finally, through operation of the command system, the using military organization has available a pertinent basis for continued growth and evolution. Using techniques of exercise and evaluation, the user can identify deficiencies and discover areas in which additional information processing assistance is necessary. The same system exercise and training capability described earlier is the vehicle for this. Simulation is the key by which stress situations can be presented to the system to identify development requirements. New operational objectives are fed back to the functional system design and to ongoing development.

Manpower Allocations in System Design. It is now important and appropriate to support some of the generalizations that we have made in this presentation with quantitative data representing experience that SDC has accumulated in its work on command and control systems. We have mentioned at several places the proportion of professional effort that we have applied to various phases of the software design and implementation process. Table 7 reflects this average experience. These data are drawn exclusively from SDC's work, and are not intended to reflect the experience of other organizations; neither do they represent the total effort from all organizations at each development phase.

ADVANCED PLANNING AND DESIGN	3
OPERATIONAL SYSTEM DESCRIPTION	6
OPERATIONAL SPECIFICATIONS	35
SOFTWARE PRODUCTION	50
ORIENTATION AND INSTALLATION	6

TABLE 7. Typical Distribution of Effort in Per Cent

Three percent of SDC professional manpower has been applied to the advanced system planning and design phase in participation with the using command organization; six percent to preparation of the Operational System Description; thirty-five percent to preparation of the Operational Specifications; fifty percent to software component design (of which the Program System Description is one part) and software production; finally, six percent is applied to installation and orientation support activities. In some cases, SDC has continued to support the maintenance of the information processing system after delivery to the user--these figures are not considered here since our emphasis has been on design and implementation.

PITFALLS IN SYSTEM DESIGN AND IMPLEMENTATION

To describe some of the hazards of system design we have selected a set of problem areas encountered in our own experience. Our objective in identifying these pitfalls is to be constructive. We believe that these problems are not obvious, and that the readers of this document can avoid them if alerted to their imminence and significance. Our objective is to be able to report in three years that this particular set of pitfalls is no longer with us.

- FAILURE TO INITIATE SOFTWARE DESIGN EARLY
- UNDERESTIMATING MAGNITUDE AND COMPLEXITY
- LACK OF EXPERIENCED TECHNICAL RESOURCES
- ABSENCE OF SOFTWARE DESIGN GUIDANCE
  - Procedure Oriented Language System
  - Documentation Requirements
  - System Software Capability
- INADEQUATE CONCURRENCE AND CHANGE PRINCIPLES AND PROCEDURES

TABLE 8. Pitfalls

The first problem is failure to initiate software design sufficiently early. This is a common problem. In a recent experience, both technical representatives from the using agency, and SDC in the software design role, recognized that the statement of operational objectives was incomplete and would not result in an entirely effective system, yet neither time nor talent was available to perform the analysis, evaluation, and revision of these operational objectives, which were necessary for better guidance of detailed design and implementation. As a consequence, the resulting information processing capability was one with which we, as well as the user, were less than pleased.

The next hazard is the underestimation of time and dollars required to perform an effective software design. The classic case of this pitfall is the SAGE system. Early predictions for the first operational SAGE computer program system called for one year's time and sixty professional people. It turned out to require two years and 600 people.

There are still cases in which we encounter gross underestimation of the magnitude and complexity of the software design and implementation. A current example can be drawn from one of the theater commands, which has submitted a requirement to the Pentagon for development of an automated combat operations center to control the atomic strike force. This is a significant and major undertaking for this particular command. Yet in the submission a requirement is noted for only fourteen professional personnel to design and implement the entire software system: six designers and eight programmers.

The next point--lack of experienced technical resources--is intended to include two factors: personnel and working environment. Undeniably, an adequate professional team must be developed to perform the Operational System Description and Operational Specifications phases. Yet the numbers of well-trained professionals in the country from whom to create a software design team are severely limited, and are already stretched thin over the current generation of development programs.

Second, there is an often-overlooked requirement to provide environmental conditions that will encourage and maintain a dynamic, vigorous, creative capability in a software design team. These environmental conditions include the availability of varied information processing design applications for professional personnel to work on. If professionals have only one job over successive years, they tire of that job and either atrophy or leave for the apparently greener pastures that other applications offer. Further, a research relationship for those people engaged in a system application project is necessary to enable people working on the project to receive the benefit of new research developments and to permit them to contribute ideas about research problems to the research group. Finally, it is vital to provide opportunity for career continuity and growth to professionals in a software design team. Like professionals anywhere, information processing experts know about and require opportunities for career achievement and will go where these exist.

The next point listed in Table 8, drawn from our recent experience, is the areas in which software design guidance is required to assure effective command system development. We have referred above to the application of procedure oriented languages in computer program production coding. Although these languages are not fully developed, we have confidence that their value is well established in the software state of the art. Nevertheless, at a recent major source selection proceeding for equipments for a military command system, no consideration was given to whether a procedure oriented language was available, or required for use, in development of associated software. This omission may yet be repaired, but need not and should not have occurred.

With reference to the requirements for documentation, the needs are obvious. If one tries to thread his way through the technical documentation associated with the current generation of command and control systems under development in the military services and DOD, he finds neither uniformity nor consistency. In fact, he finds some systems almost completely lacking in technical design documentation simply because time and effort have not been devoted to creating a document base adequately describing plans and capabilities being incorporated in the software.

We have observed a recent case in which an important capability for the system software was omitted in early design and is only now being retrofitted. This is the requirement for a data recording and reduction capability--a record keeping capability enabling the field commander to analyze the performance of his information processing system. The data recording and reduction capability will be available for scheduled operations use, but only at the cost of efficiency and time in the total development process.

Let us conclude with the requirement for timely review and concurrence procedures. As much as anything else, this is the point we have tried to stress throughout this report. It is possible to plan for technical review and concurrence, and necessary to do so if the user wants to assure faithful development of command system software to meet his operational objectives. Here again, we can draw an illustration from recent experience. During the course of software design for a major command system, we prescribed display format and variability requirements in the System Operational Specifications. The proposed design was submitted to the user's representatives for technical management and design control for review and concurrence. No feedback guidance was received on the basis of the documentation, and implementation of the displays was initiated. Some six months later the user generated and provided to us a vastly expanded set of display requirements. We are now working to conform to those requirements, but the cost and delay occasioned by this rework could have been eliminated by appropriate and timely concurrence.

These are the pitfalls drawn from recent experience. Our purpose in enumerating them is to enable those facing problems today to avoid errors of the past.

CONCLUSION

Nothing that we have said here is new in a scientific sense, or even definitive. Nonetheless, it is useful to illuminate the field of command and control software development by a synthesis of our accumulated experience. We have tried to show that software design and implementation is important and responsible--deserving of the best professional resources. While the software development process is complex, it has yielded to orderly description and management. Relevant knowledge and skills exist and can be used to increase our understanding, competence, and control over this critical portion of the job of creating military command and control systems.

UNCLASSIFIED

System Development Corporation,
Santa Monica, California
SOFTWARE DESIGN AND IMPLEMENTATION.
Scientific rept., SP-1092, by
J. W. Singleton. 4 March 1963, 29p.
Unclassified report

DESCRIPTORS: Command & Control Systems.
Programming (Computers).

Presents a synthesis of System
Development Corporation's
accumulated experience in the field
of command and control software
development. Shows that software UNCLASSIFIED

---

design and implementation is
is deserving of the best
professional resources. Concludes
that while the software development
process is complex, it has yielded
to orderly description and management
and that relevant knowledge and skills
exist and can be used to increase our
understanding, competence, and control
over this critical portion of the job
of creating military command and
control systems.

UNCLASSIFIED

UNCLASSIFIED