Volume 5 Number 4 October 1983

Annals of the History of Computing

SPECIAL ISSUE: SAGE (Semi-Automatic Ground Environment)

Contents

Departments

About this Issue

This Special Issue on SAGE, the pioneering computer-based air-defense system, has had, like SAGE itself, a complicated beginning. Two old SAGE hands, Mort Astrahan and Jack Jacobs, wrote an article on the design of the SAGE computer and submitted it to the *Annals*. Herb Benington, another old SAGE hand, was looking, at the time, into the possibility of republishing his 1956 paper on the development of the software for SAGE. Bernie Galler suggested that an entire issue of the *Annals* be devoted to SAGE, and in a weak moment, I agreed to act as editor. I recruited Jack Jacobs and Louise Meyer to help me, and we set to work.

SAGE was a very large enterprise involving dozens of organizations and thousands of people. A list of just the major contributors would be quite long. It was obvious that we could not hope to cover all the major aspects of SAGE—its conception, design, production, operation, test, funding, politics, management, organizational relationships, and so on. Instead, I took the editor's privilege of looking at SAGE from my own limited perspective of design and test. I hope those of you who read this issue will gain some feel for what SAGE was, for the technical environment in which it was created, the kind of people who designed it, and a little about how they felt about it. Speaking as one of them, I feel that SAGE was a great experience, socially as well as technically. The opportunity to be a part of a truly important enterprise—to be in at the beginning of a new and revolutionary art, to do things for the first time and see them built and work as they were supposed to—such an opportunity comes to only a few lucky ones. We were *very* lucky, and I hope that through the barrier of words comes a little of the excitement and enthusiasm that gripped us all and that we still remember.

The issue is more of a sampler than a unified description. There are two papers from the 1950s and three new ones on the computer, on radar data transmission, and on test and experiment. There is an overview and—perhaps the heart of the issue—a discussion by some of the participants representing some of the major organizations. Reliability was the fundamental driver of the design of SAGE, and we are fortunate to have some remarks from Jay Forrester on that subject. Each paper was written to stand alone and therefore has some introductory material that may be redundant. I hope you will forgive us for any duplication. Finally, there is a report of a recent trip to North Bay, Ontario, to see a SAGE center in operation.

This issue appears in a significant year for SAGE. There are, as I write, six SAGE centers still in operation, and running very well, I might add: doing their job, meeting their reliability specifications (downtime less than 4 hours per year), all 55,000 vacuum tubes in each center glowing softly. SAGE is now 25 years old; the first center went operational on July 1, 1958. The remaining ones contain, as far as I know, the oldest operating computers in the world. But this year is the last. By the end of 1983, all the remaining SAGE centers will be shut down, and the task of air defense for the United States and Canada will be carried out by new centers equipped with modern hardware but logically and operationally the lineal descendants of SAGE.

Only a few of the major designers of SAGE are represented in this issue. I will not attempt to name the others. There are many, most but not all of whom I knew and whose faces and accomplishments I remember, although I can no longer remember names. In my view, there were only two who were absolutely necessary to SAGE, without whom there would have been no SAGE: George Valley and Jay Forrester. Jay was the leader of the Whirlwind group at MIT, the leader of the SAGE division at Lincoln Laboratory, the inventor of core memory, and the strong intelligence at the heart of the SAGE design. George Valley was the chairman of the Air Force Scientific Advisory Board's Air Defense Systems Engineering Committee, assistant director and later associate director of Lincoln. Without George there would have been no Lincoln Laboratory and no SAGE. I am sorry he was not able to participate in our discussion, and I hope that someday he will publish his own memoirs of his key role in the nation's defenses.

The role of the U.S. Air Force in the design of SAGE is given too little attention in this issue. Obviously the Air Force had the need and provided the funding that created SAGE. Beyond that, however, many air force officers played roles of fundamental importance. Senior officers, such as General Earle E. Partridge, contributed their understanding and support in good times and bad. Other officers managed project offices, participated in the operational design, planned and ran tests and exercises, trained and provided crews, and did (and did well) a thousand things that had to be done. I hope their story will be told in some other place.

I would like to express my appreciation for the dedicated efforts of all the authors and participants and for their patience with the many telephone calls. The assistance and encouragement of Bernie Galler and Nancy Stern have been invaluable. Ed Galvin, MITRE archivist, did his usual superb job of collecting information and pictures. Without Mondy Dana there would be no *Annals*; without Louise Meyer there would be no issue on SAGE.

Robert R. Everett
Editor, Special Issue

Contributors

Morton M. Astrahan (Ph.D. Northwestern University 1949) joined the IBM Corporation in Endicott, N.Y., in 1949. He participated in the design of the IBM 701 in 1950, and from 1952–1956 he managed system planning and part of the development work for the AN/FSQ-7 of the SAGE system. From 1956–1970 he was at IBM's Advanced Systems Development Division in San Jose (with two years in France as advisor to the IBM European Laboratories). In 1970 he joined the Research Division, where he is currently engaged in the development of relational database systems. He organized and was first chairman of what is now the IEEE Computer Society. Since 1952 he has helped manage the Joint Computer Conferences and is now on the AFIPS National Computer Conference Committee. He is a Fellow of IEEE and received the AFIPS Distinguished Service Award in 1975.

Herbert D. Benington (B.S.E.E. MIT; B.A. Oxford University) worked for MIT and the System Development Corporation on developing the prototype software for the SAGE system. In 1963 he joined the Office of the Secretary of Defense. From 1973–1981 he was vice-president of the MITRE Corporation, serving as general manager of its Metrek Division and Washington Center. He went back to SDC in 1981 and currently manages a project to modernize data processing for the Naval Intelligence Command.

Robert Bright attended the University of Pennsylvania and worked for the Bell Telephone Company of Pennsylvania. He was later transferred to the Western Electric Company as a radar systems engineer and mobile radio and microwave specialist. In 1949 he went to American Telephone & Telegraph Company as a radio engineer. He worked with Western Electric as superintendent of systems engineering on the SAGE project and then rejoined AT&T in 1958. In 1959 he became executive communications administrator of the Washington Office, where he was in charge of worldwide communications services for the U.S.

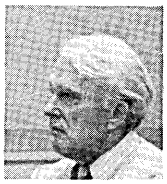
president. He was transferred to AT&T Long Lines Department in 1963. He was appointed director of Government Communications Projects at Western Electric in 1966 and general manager of Government Projects and International Systems in 1969. He retired from Western Electric in 1976 and now lives in North Carolina.

Robert P. Crago (B.S.E.E. Carnegie-Mellon University 1949; M.S.E.E. California Institute of Technology 1949) joined the IBM Corporation in Poughkeepsie, N.Y., in 1949. He held several engineering management positions on Project High, IBM's activity in support of the SAGE system. Since 1956 he has been with the Federal Systems Division, working especially on applying IBM's commercial developments to military needs. He is currently a technical specialist in the Systems Architecture organization at FSD headquarters in Bethesda.

Robert R. Everett (B.S.E.E. Duke University 1942; M.S.E.E. MIT 1943) joined the MIT Servomechanisms Laboratory in 1942 as a graduate student and in 1943 as a staff member. He was Jay W. Forrester's assistant on the Whirlwind project and became associate director of the Digital Computer Laboratory. When the Lincoln Laboratory was formed by MIT in 1951, he became associate head of Division 6, of which he became head in 1956. Division 6 was responsible for overall systems design and testing of the SAGE system and its direction centers; it developed the first magnetic-core memories invented by Forrester. The SAGE-design parts of Lincoln were spun off into the nonprofit MITRE Corporation in 1958, and Everett was technical director. In 1959 he was appointed vice-president, Technical Operations. In 1969 he was appointed president and chief executive officer, a position he still holds. He is a Fellow of IEEE and is an advisor to several federal defense organizations.

Jay W. Forrester (B.S. University of Nebraska 1939; S.M. MIT 1945) was director of the MIT Digital Computer Laboratory from 1946 to 1951 and head of Division 6 of the Lincoln Laboratory until 1956. He was in charge of the design and construction of Whirlwind I, and he guided the planning and design of the SAGE system. In this work he invented—and holds the patent for—random-access, coincident-current magnetic storage. In 1956 he became professor of management at MIT's Alfred P. Sloan School of Management (where he was named Germeshausen Professor in 1972) and currently directs the System Dynamics Program. He has received the Medal of Honor from IEEE (1972), the Harry Goode Award from AFIPS (1977), and honorary doctoral degrees from several universities. He is a Fellow of IEEE, the Academy of Management, the American Academy of Arts and Sciences, and AAAS.

John V. Harrington (B.E.E. Cooper Union Institute of Technology 1940; M.E.E. Polytechnic Institute of Brooklyn 1948; Sc.D. MIT 1957) served with the U.S. Navy during World War II, and then worked with the U.S. Air Force Cambridge Research Laboratory for five years. In 1950 he joined the Lincoln Laboratory as leader of the Data Transmission Group of Division 2 and was responsible for developing radar data-processing and transmission equipment for the SAGE system, as well as the first telephone-line modems. He served as head of Lincoln Laboratory's Radio Physics Division 3 from 1958 to 1963, when he joined MIT's faculty and became the first director of MIT's Center for Space Research. In 1973 he went to the Communications Satellite Corporation, where he is now senior vice-president of research and development and director of COMSAT Laboratories. He is a Fellow of IEEE, AIAA, and AAAS and is the recipient of the Air Force Medal for exceptional civilian service and the Gano Dunn award of the Cooper Union.



John F. Jacobs (B.S.E.E. Illinois Institute of Technology 1950; M.S.E.E. MIT 1952) joined the Whirlwind project at the Digital Computer Laboratory as an MIT graduate student and then as a staff member. From 1952–1958 at Lincoln Laboratory he worked on logical design for Whirlwind II/FSQ-7, established the Systems Office for design control, and held responsibility for the SAGE computer program and weapons system integration. He was associate head of Lincoln's Division 6 at the time he joined the MITRE Corporation in 1958 as associate technical director. He was senior vice-president for corporate planning and development when he retired in 1977. He is currently special consultant to MITRE.

Albert R. Shiely, Jr. (B.S. U.S. Military Academy 1943; M.S.E.E. University of Illinois 1947) was a pilot with the U.S. Air Force during World War II and served in several electronics research and development positions from 1947 to 1954. As chief of the Air Defense Systems Operations Division of the Air Research and Development Command from 1954–1957, he was in charge of engineering for the USAF on the SAGE system. From 1957–1967 he was assigned to USAF headquarters in Washington, D.C., and to the Electronic Systems Division at Hanscom Field, Mass., in electronics staff and program management. Promoted to brigadier general in 1967, he commanded the USAF European communications area until he was appointed vice-commander of the Air Force Communications Service in 1969. In 1971, promoted to major general, he commanded the Electronic Systems Division until he retired in 1974. He now lives in Barrington, N.H., where he has served as a selectman and on local community committees.

Norman H. Taylor (B.A. Bates College 1937; M.S.E.E. MIT 1939) worked on the Whirlwind project at the MIT Digital Computer Laboratory from 1947 to 1952 and was associate head of Lincoln Laboratory's Computer Division, where he was in charge of the MTC, the FSQ-7, and the TX-0 and TX-2 computers. By 1958 he was manager of SAGE weapons integration. From 1958 to 1969 he worked for Itek Corporation, Control Data Corporation, and Arthur D. Little. In 1969 he founded and to 1982 was president of Corporate-Tech Planning. He is now an independent consultant. He helped run the first Joint Computer Conferences in 1950–1951. He is a Fellow of IEEE, and received its electronic reliability award.

Henry S. Tropp is a professor of mathematics at Humboldt State University and a member of the Editorial Board of the *Annals of the History of Computing*. In 1973 he conducted a discussion similar to the one in this issue with the participants in Project Whirlwind for the AFIPS/Smithsonian Computer History Project. In October 1982 he toured the SAGE facility in North Bay, Canada, with a group organized by the Computer Museum.

C. Robert Wieser (B.S.E.E., M.S.E.E. MIT 1940) worked with the Boston Edison Company from 1940–1942, when he joined the MIT Servomechanisms Laboratory, developing fire-control systems. He went to the Digital Computer Laboratory in 1949 and applied Whirlwind I to air-traffic control and then air defense. In 1951 he joined the Lincoln Laboratory and was leader of the group designing and testing the Cape Cod Air Defense Direction Center and preparing the operational and mathematical specifications for the SAGE system. At Lincoln he became head of the Systems Division (changed to the Data Systems Division in 1963), assistant director, and deputy director. In 1968 he joined the Office of the Secretary of Defense and in 1971 went to the McDonnell Douglas Astronautics Company as director of Advanced Weapons Programs. Since 1982 he has been vice-president and general manager of the Western Division of Physical Dynamics, Inc./RES Operations.

Charles A. Zraket (B.S.E.E. Northeastern University 1951; M.S.E.E. MIT 1953) joined MIT's Digital Computer Laboratory in 1951 and was group leader in the Digital Computer Division at Lincoln Laboratory from 1952–1958. He went to the MITRE Corporation in 1958 as head of the Advanced Systems Department and has been a technical director and vice-president in MITRE's Bedford, Mass., and McLean, Va., divisions. Since 1978 he has been executive vice-president, chief operating officer, and trustee of MITRE. He is a Fellow of IEEE and a member of several policy groups and committees concerned with national security and energy resources.

SAGE Overview

JOHN F. JACOBS

Editor's Note

I would like to thank Jack Jacobs for this overview, which gives a brief review of the history of SAGE and of some of the organizations involved. I cannot think of anyone better able to discuss SAGE than Jake who, along with his many other contributions, created and ran the Systems Office that coordinated the design efforts of the numerous organizations having subsystem design responsibilities. We at Lincoln had nominal overall design authority, but we were not foolish enough to insist on it very often. Almost all the time, the Systems Office would handle problems by investigating them, getting everyone's input, and then coming up with a solution with plenty of backup and justification. Everyone's agreement was then sought, usually in formal coordination meetings, which frequently had as many as a hundred participants. Agreement was almost always achieved, not because nobody wanted to object, but because the sheer hopelessness of trying to upset the carefully worked out and documented Systems Office solution was obvious to everyone. Besides, Lincoln was responsible, and no one wished to usurp that important but highly risky position. Since no design decision was considered valid within the community without Lincoln concurrence, one was faced with agreeing or else taking the onus for holding up the whole schedule and bringing down the wrath of the entire community on himself. To make this work, of course, the Systems Office had to be prompt, accurate, and thorough. At this task, as at all others, Jake was superb.

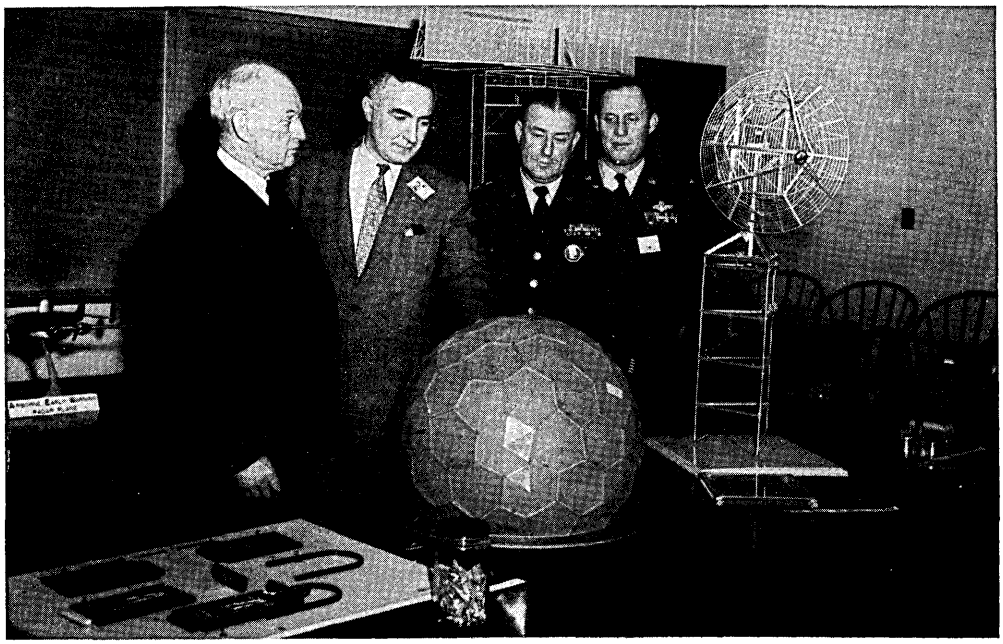
Even though there had been no warning of the Japanese attack on Pearl Harbor, the American public maintained a complacent attitude toward the lack of adequate air defense in the years right after the end of World War II. Much of this complacency may have been due to the fact that the United States had developed the atomic bomb and had demonstrated its deathly potential. Although there was an uneasy fear that Russia, the estranged ally, would someday possess this technology, America was thought to be secure in her arms supremacy.

The August 1949 disclosure to the Truman administration by U.S. intelligence that the Russians had exploded a nuclear bomb—and had developed bombers capable of carrying such a device over the North Pole and into the United States—jolted America out of her complacency and into the Cold War. One of the groups in the Department of Defense acting on this information (which was not made public until late in September) was the newly formed Air Force Scientific Advisory Board. One of the board members, George E. Valley from MIT, proposed that a group of experts be assembled to address themselves to the suddenly paramount issue of U.S. air defense. As a result of that recommendation, the Air Defense Systems Engineering Committee (ADSEC)—also known as the Valley Committee—was formed in December 1949 with Valley as its chairman.

The Valley Committee's first meeting marked the genesis of a novel concept for an air-defense system

Author's Address: MITRE Corporation, Burlington Road, Bedford, MA 01730.

Categories and Subject Descriptors: K.2 **[History of Computing]** —*hardware, people, SAGE, software, systems.* General Terms: Design, Management. Additional Key Words and Phrases: defense, J. W. Forrester, Lincoln Laboratory, G. E. Valley.
Photographs courtesy MITRE Archives.



January 1956 press conference announcing the SAGE system for continental air defense. *Left to right:* Admiral Edward Cochran, George E. Valley, Major General Raymond G. Maude, Colonel Dorr Newton.

that would become known several years later as the SAGE system. SAGE—Semi-Automatic Ground Environment—was deployed eight years after Valley's committee was first assembled. As it evolved, SAGE spawned new technologies, businesses, agencies, and careers. The history of SAGE has many threads running through it that weave a pattern or legacy; some of the principal elements that comprise SAGE are discussed later in this issue. SAGE became the first major command and control system; Whirlwind, MIT's pioneering digital computer, played a crucial role as the system's heart.

In a matter of months members of the Valley Committee concluded that the existing air-defense network, which was left over from the war and consisted of a few large radars and manual methods of processing and relaying radar data, was almost wholly inadequate. They proposed a larger number of smaller radars for greater coverage nationwide, with communication lines between the areas of coverage, and centralized, automated systems to handle the information. They recommended that the existing system be upgraded as quickly as possible by a competent technical organization, and that the longer-range solution should include extensive use of computers to handle surveillance, control, and bookkeeping functions.

The ADSEC recommendations were set in motion at MIT through Project Charles, an air-defense study group headed by F. Wheeler Loomis, on leave from the University of Illinois. The group consisted of a number of experts (including Valley) from the scientific community. After a thorough review of the existing situation, Project Charles recommended, first, that the existing manual system be upgraded (Bell Laboratories and Western Electric were later chosen to undertake this upgrading, which came to be known as the Continental Air Defense System (CADS) project), and second, that a research laboratory be established to undertake the long-range development of a more capable ultimate system. Project Lincoln, later known as Lincoln Laboratory, was established in 1951.

While credited with the development of SAGE, Lincoln benefited from work done in other research laboratories. Lincoln drew both ideas and personnel from these laboratories and adapted their products or technologies to the air-defense problem. One of the most dramatic innovations from another laboratory was the Whirlwind computer, originally developed in the late 1940s by MIT's Digital Computer Laboratory as a computer for a navy flight trainer and airplane stability analyzer.

Whirlwind was the first real-time control computer. Lincoln considered it a good candidate for the air-defense control machine because it had been designed to meet two real-time control needs that were critical to the air-defense problem: fast processing speed and maximum reliability. The research and development involved in attaining speed and reliability for Whirlwind laid the groundwork for the design of the SAGE computer.

In 1950 Jay W. Forrester invented and led the development of the random-access core memory as a replacement for the then-current but limiting technology of cathode-ray-tube (CRT) storage. Compared to the cathode-ray memory in Whirlwind, the core

memory doubled the operating speed, quadrupled the input data rate, increased the mean time to failure from two hours to two weeks, and reduced the maintenance time from four hours a day to two hours a week. Whirlwind personnel also emphasized tube reliability in order to overcome the problem of frequent tube failures that plagued the early generation of computers. Whirlwind staff, working with tube manufacturers, developed special tubes that were less prone to failure than other tubes. Whirlwind staff also developed marginal checking, a procedure for continually monitoring the deterioration of vacuum tubes. This procedure allowed deteriorating tubes to be identified before actual failure.

Lincoln drew from work done at the Air Force Cambridge Research Laboratory (CRL) on data communication. Among CRL's communication innovations that were adapted for SAGE was a technique called slowed-down video, which provided, by digital transmission over phone lines, a continuous picture of what was in the range of the radar. Another CRL technique put to use in SAGE was a radar processor that included a beam-splitting device capable of determining the center of the beam after the beam swept across a target, thus giving increased angular accuracy of the target location. CRL also initiated a scheme for sending generalized digital data over a standard phone line.

As soon as the component parts of the system were developed, Lincoln Laboratory produced an experimental air-defense system called the Cape Cod System, which first coordinated the various components of the system and realized its capabilities. In 1952, those working on the Cape Cod project demonstrated the system's abilities to track and control aircraft and its capabilities for surveillance and weapons control. The Cape Cod System was the first to use computer time-sharing and to use extensively CRT display consoles and light guns to transfer information from the screens to the computer.

The Cape Cod System used the Whirlwind computer, but a more reproducible, maintainable machine was required for the deployed system. The successor to Whirlwind was the FSQ-7, a computer jointly developed by Lincoln and IBM and specifically designed to meet the air-defense needs.

The final SAGE plan called for duplex computers located at direction centers throughout the country. The FSQ-7 was the first system to use this duplex computer scheme. It was designed to have one computer in active control and one to serve as a test machine capable of assuming the operational load should a breakdown occur. The FSQ-7 was one of the earliest production machines to incorporate random-access core memory. (This technology serves as an example of how a military development can spin off commercially: core memory, initiated for the FSQ-7, actually made its first appearance in a production machine in 1955 in IBM's 704.) The FSQ-7 also incorporated a dual arithmetic element that by simultaneous processing of both the X and Y positions of the data, made possible even greater speed than the Whirlwind computer. Finally, the experience of joint design by Lincoln and IBM laid the groundwork for later coordination among the many organizations that were to become involved in the SAGE production.

The software for the FSQ-7 broke ground in programming. Such a large operating program was never required before the FSQ-7. Furthermore, the program had to be developed with few qualified programmers and few of the programming aids available today. A result of programming the FSQ-7 was the experience gained in developing large-scale programs of the kind SAGE would require. Hundreds of inexperienced programmers received formal and hands-on training that would enable them to program for SAGE, and the first sophisticated utility system containing the compiler, the checker, and the communication pool was developed.

SAGE NOMENCLATURE

Whirlwind Developed as a high-speed, parallel, synchronous digital computer for a variety of applications.

Whirlwind II Second-generation Whirlwind computer, developed for use in the air-defense system. Initial name of the SAGE computer prototype.

AN/FSQ-7 (also **FSQ-7, Q-7**) Air Force nomenclature for the production version of Whirlwind II. This computer served as the active element at the sector level in the SAGE direction centers.

AN/FSQ-8 (also **FSQ-8, Q-8**) Modified FSQ-7 computer with specialized display system used at the division (multisector) level in the SAGE combat centers.

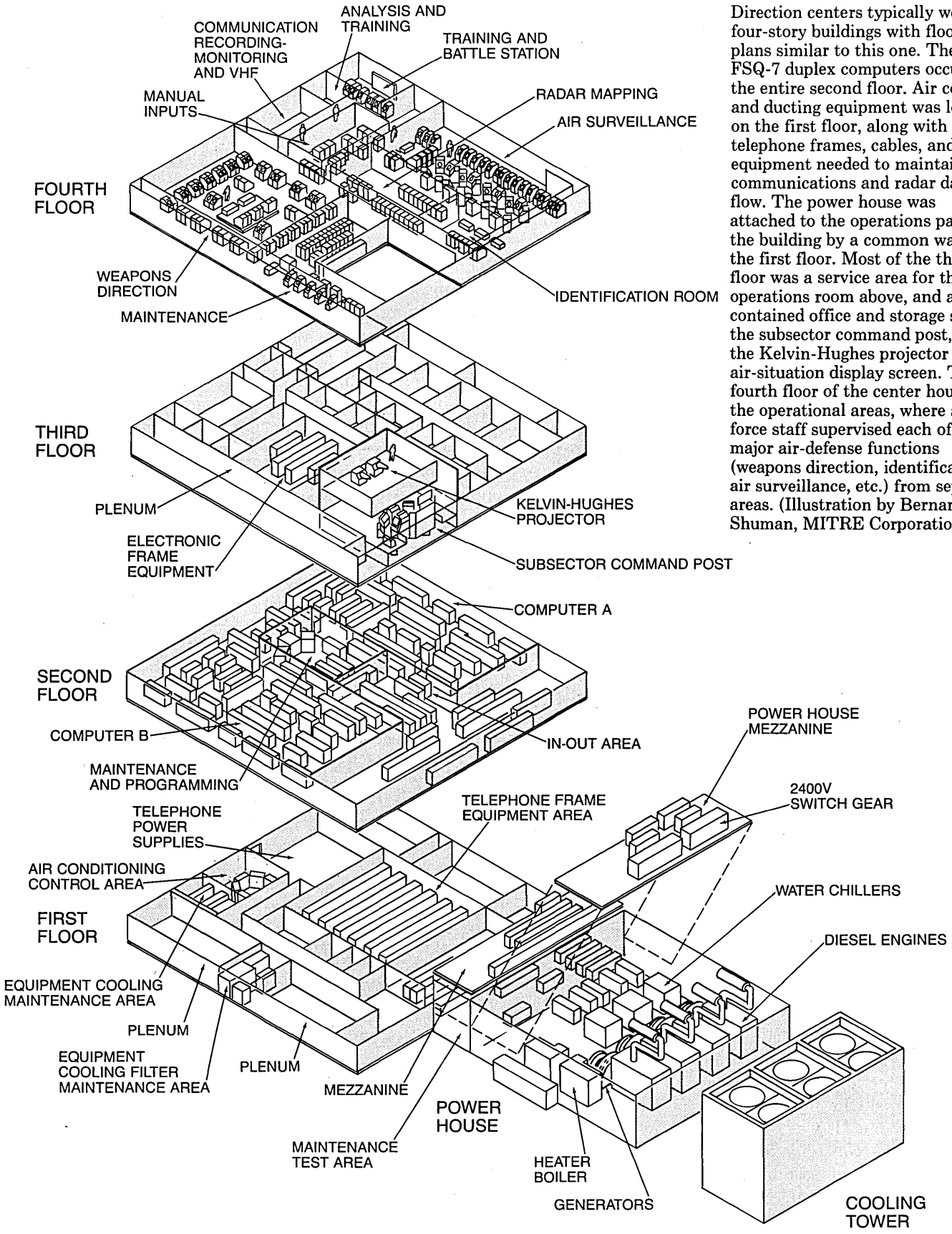
XD-1, XD-2 Single-computer prototypes of the AN/FSQ-7. One was installed at IBM's Poughkeepsie location, the other at Lincoln Laboratory.

TX-0 Experimental, transistorized next-generation computer system used to develop new techniques to replace AN/FSQ-7 vacuum-tube technology.

AN/FSQ-32 Proposed transistorized replacement for the AN/FSQ-7. A single model was built and installed at Strategic Air Command Headquarters.

AN/FST-1 Radar data-processing and transmitting equipment employing so-called slowed-down video technique. Designed by Lincoln Division 2 and built by Lewyt Corporation for the gap-filler radars.

AN/FST-2 Radar data-processing and transmitting equipment which converted analog radar signals to a digital format. Also reduced clutter and performed beam splitting. Designed by Lincoln Division 2 and built by Burroughs Corporation for the SAGE system.



Direction centers typically were four-story buildings with floor plans similar to this one. The SAGE FSQ-7 duplex computers occupied the entire second floor. Air cooling and ducting equipment was located on the first floor, along with telephone frames, cables, and equipment needed to maintain communications and radar data flow. The power house was attached to the operations part of the building by a common wall at the first floor. Most of the third floor was a service area for the operations room above, and also contained office and storage space, the subsector command post, and the Kelvin-Hughes projector and air-situation display screen. The fourth floor of the center housed the operational areas, where air force staff supervised each of the major air-defense functions (weapons direction, identification, air surveillance, etc.) from separate areas. (Illustration by Bernard Shuman, MITRE Corporation.)

Lincoln played the primary role in SAGE development until research proved the conceptual system was feasible. The focus then turned to producing the system. Since Lincoln was chartered to do only the research and development work, production fell outside its realm of responsibility. Various commercial manufacturers were sought to produce the actual SAGE components. Because of its work on CADS, Western Electric was chosen to provide the administrative support, engineering services, acceptance testing, and evaluation for the project. Lincoln retained responsibility for initial systems engineering, central design, and the master operating program. IBM was to follow through on the production computer, and a split-off from the Rand Corporation, the System Development Corporation, whose staff was familiar with programming and the air-defense problem, assumed responsibility for the evolution of the operational program. The Burroughs Corporation was chosen to produce the FST-2, which included the beam splitter, and AT&T was given responsibility for digital ground communications. The coordinating agency was the Air Defense Engineering Services (ADES) project office in New York, directed by Colonel Richard M. Osgood from the Air Materiel Command (AMC) and assisted by Colonel Albert R. Shiely from the Air Research and Development Command (ARDC). Establishing the ADES office represented the first attempt to apply a systems approach to the development of an electronic system.

Once SAGE was considered operational, the emphasis turned toward integrating existing and new weapons into the system. In the late 1950s the air-defense system was relatively fragmented; many new weapons were being developed, but the authority for weapons development was dispersed. For the new weapons to have maximum utility, the operation of all systems had to be coordinated. This meant, especially to the people at Lincoln, integration with the SAGE system.

The original SAGE program was designed to control manned interceptors. A new series of interceptors, the F-102 and F-106, had to be included. By the late 1950s, control of weapons such as Nike, the ground-to-air missile developed by Bell Telephone Laboratories and Western Electric, and BOMARC (Boeing-Michigan Aeronautical Research Center) A and B, the Air Force primary defensive missile, had to be integrated with SAGE. New systems, such as the Airborne Long Range Input (ALRI) system, which used the first airborne radar platforms; the Texas Towers, a string of early-warning radars off the coast of New England; and the frequency diversity radars, a family of radars operated on different frequencies to reduce the threat of jamming, also had to be tied in.

Lincoln provided the initial systems engineering work; however, the continued modification and adaptation of the system for further integration was not the laboratory's responsibility. As a first attempt to coordinate weapons integration, the Air Force created the SAGE Weapons Integration Group (SWIG), composed of air force and weapons manufacturers personnel. This organization had little authority, little technical expertise, and little consensus of purpose; it ultimately lasted less than a year. The need for an organization with a broader base of power and with participants whose primary concern was integration became clear. In 1957 the Air Force established a higher-level Air Defense Systems Management Office (ADSMO), staffed by ARDC, AMC, and Air Defense Command (ADC), to attend to the integration prob-

By the time SAGE was fully deployed in 1963, U.S. air-defense coverage was the responsibility of 23 geographically determined sectors. The heart of each air-defense sector was its direction center, where air-surveillance information from radars within the sector was received, interpreted, and displayed by the twin AN/FSQ-7 computers to the sector's commander and staff.

The New York air-defense sector was the first to be declared operational in July 1958, during ceremonies held at McGuire Air Force Base near Trenton, N.J. The sector's direction center at McGuire, shown in this 1958 photo, is today headquarters for the 21st Air Force.

SAGE CHRONOLOGY

1949

Aug.	Russians detonate atomic device.
Nov.	George E. Valley, MIT, proposes to Theodore von Karman, chairman, Air Force Scientific Advisory Board, that a study of air-defense requirements be undertaken.
Dec.	Air Defense Systems Engineering Committee (ADSEC) is established, with Valley as chairman.

1950

Sep.	First MIT experiments transmitting digitized data from Microwave Early Warning (MEW) radar at Hanscom Field (Bedford, Mass.) to Whirlwind computer in Cambridge, Mass., over commercial telephone lines.
Oct.	ADSEC's final report is issued, defining the air-defense system that will become known as SAGE.
Dec.	Gen. Hoyt S. Vandenberg, Air Force Chief of Staff, asks MIT to establish and administer an air-defense laboratory, and to perform an intensive investigation of the air-defense problem.

1951

Jan.	Air Force contracts with Bell Telephone Laboratories to improve existing ground-radar-based air-defense system.
Jan.	Air Force contracts with University of Michigan to expand ballistic missile program into a system for air defense.
Feb.	"Project Charles" established at MIT for short-term investigation of air-defense problem.
Apr.	First live demonstration of automatic aircraft interception using Whirlwind computer and MEW radar.
Jul.	"Project Lincoln" established at MIT as laboratory for air defense—original charter for MIT Lincoln Laboratory.
Aug.	Air Force Air Research and Development Command (ARDC) assumes responsibility for administration of Project Lincoln.
Oct.	MIT's Whirlwind staff at the Digital Computer Laboratory joins Project Lincoln as Division 6.

1952

Feb.	Secretary of the Air Force T. K. Finletter assigns top priority to air-defense matters; promises MIT whatever funding required.
Apr.	Name "Project Lincoln" changed to "Lincoln Laboratory."
May	Memory Test Computer (MTC) under design.
Jun.	Plans for "Cape Cod System" published—scaled-down prototype of nationwide SAGE system.
Jul.	Lincoln considering several manufacturers for production of air-defense computer.
Oct.	IBM awarded subcontract by Lincoln to study computer project; Division 6–IBM engineering collaboration under way.

1953

Jan.	Lincoln publishes Technical Memorandum No. 20—a proposed air-defense system called "Lincoln Transition System."
Jan.	First Division 6–IBM technical meeting, Hartford, Conn.
Mar.	Lincoln publishes report, "Cape Cod System and Demonstration."
May	ARDC decides to pursue Lincoln Transition System and phase out University of Michigan system.
Jun.–Jul.	Division 6–IBM "Project Grind" meetings.
Summer	Division 6 staff moves from MIT in Cambridge to Lincoln Laboratory in Lexington.
Aug.	First bank of core storage wired into Whirlwind after MTC tests succeed.
Sep.	IBM receives contract to produce two single-computer prototypes: the XD-1 and XD-2.
Sep.	Cape Cod System fully operational.
Nov.	Decision made to have duplex computer system.
Dec.	Cape Cod System tracks 48 aircraft.

1954

Feb.	First production contract for SAGE computer—called the AN/FSQ-7—awarded to IBM.
May	Air Materiel Command establishes Air Defense Engineering Services (ADES) at Wright-Patterson Air Force Base for acquisition of the Lincoln Transition System. Western Electric becomes involved in ADES management.
Jul.	Lincoln Transition System is renamed SAGE—Semi-Automatic Ground Environment.
Sept.	ADES moves to New York City and acquires representatives from ARDC, ADC, and AMC.

1955

Mar.	"Red Book" operational plan is published—complete definition of SAGE.
Apr.	ADES becomes part of newly formed Electronic Defense Systems Division.
Jun.–Jul.	Simplex version of AN/FSQ-7 (XD-1) installed at Lincoln by IBM.
Dec.	System Development Division emerges from Rand Corporation.

1956

Feb.	Development of TX-0 announced—experimental transistorized computer.
Apr.	Lincoln urges Air Force to find agency to manage integration of weapons with SAGE system.

lem. ADSMO suffered failings similar to SWIG's: it was still at too low a level and had too little technical support to have any clout.

By way of strengthening ADSMO, the Air Defense Systems Integration Division (ADSID) was formed, with a general officer, Major General Kenneth P. Bergquist, in charge. Still, technical support was required. MIT, unwilling to let Lincoln get more deeply involved in SAGE deployment by continuing to supply the technical support, refused a role as technical adviser but agreed to help establish an organization separate from Lincoln and MIT to provide the necessary technical support. The organization, which came to be known as the MITRE Corporation, was formally established in 1958. One of its first responsibilities was to serve as technical adviser to ADSID on integrating weapons under SAGE.

Further coordination difficulties occurred in the late 1950s and early 1960s when new command and control systems such as ballistic missile warning, air communication, and satellite surveillance were developed. Like the original weapons systems, these systems were all developed independently, and there was no guarantee of their coordinated operation. To meet the need to coordinate the new command and control systems, the Air Force Command and Control Development Division (C^2D^2) was formed in 1963. Soon C^2D^2 was subsumed under a high-level organization for weapons and systems integration, the Electronic Systems Division (ESD), which included the development divisions from ARDC, and AMC, and C^2D^2.

The first SAGE direction center went operational in July 1958 at McGuire Air Force Base. SAGE was fully deployed by 1963. In total, 23 direction centers, three combat centers, and one programming center were built. Because each center was duplexed, there were 54 CPUs in all. In the intervening years, the original SAGE plan underwent modification and expansion. In 1957, when Russia launched Sputnik, the United States became increasingly concerned over the special vulnerability of the SAGE system, whose computers were located at Strategic Air Command (SAC) bases—bonus targets in a ballistic missile attack. The idea of Super Combat Centers (SCCs) evolved: deep underground direction centers housing the new AN/FSQ-32 computer. The AN/FSQ-32 computer replaced the outdated vacuum-tube technology of the FSQ-7/8 with transistor technology. The idea of the SCC was scrapped in favor of the less costly Back-Up Interceptor Control (BUIC) system, which provided automation at the radar sites as well as at the direction centers. The backup automation ensured air-defense capabilities, even if the SAGE direction centers fell under attack.

As of this writing, in mid-1983, six SAGE direction centers are still in operation: Hancock Field, New York; Fort Lee, Virginia; McChord Air Force Base, Washington; Malmstrom Air Force Base, Montana; Luke Air Force Base, Arizona; North Bay, Ontario, Canada. Some of the FSQ-7s have been operating for more than 20 years. SAGE is now being phased out in favor of a new system that emphasizes FAA radars and new Regional Operational Control Centers (ROCCs). Because SAGE never saw actual combat, it is difficult to evaluate its effectiveness, but the system merits appreciation simply for its contributions to both the computer-communications field and air defense.

Jun.	IBM's first production FSQ-7 system accepted in manufacturing test cell.
Sep.	Air Force asks Lincoln to manage weapons integration task; Lincoln declines.
Nov.	ARDC holds conference on weapons integration problem.
Dec.	Experimental SAGE Sector (ESS) begins shakedown tests.
Dec.	System Development Division of Rand begins independent operation as System Development Corporation.
Dec.	ARDC recommends establishment of Air Defense Systems Management Office (ADSMO) to oversee integration.
1957	
May	SAGE Weapons Integration Group (SWIG) assembles at Hanscom Field.
Jun.	Lincoln urges that Division 6 take over weapons integration responsibility.
1958	
Mar.	Secretary of the Air Force proposes to MIT that a new organization be formed to provide systems engineering support to ADSMO.
Jul.	First of 24 SAGE direction centers operational at McGuire Air Force Base, New Jersey.
Jul.	Division 6 becomes basis of new systems engineering organization, incorporated as the MITRE Corporation.
1963	
	The SAGE system is fully deployed in 23 air-defense sectors: 22 in the United States and one in Canada.
1983	
Jan.	Six SAGE systems still running.
1984	
Jan.	All SAGE systems shut down.

SAGE—A Data-Processing System for Air Defense

ROBERT R. EVERETT, CHARLES A. ZRAKET, AND HERBERT D. BENINGTON

The paper is adapted from a presentation at the 1957 Eastern Joint Computer Conference. The authors give details of the Semi-Automatic Ground Environment (SAGE) system and how it developed.

*Categories and Subject Descriptors: K.2 [**History of Computing**]—hardware, SAGE, software, systems*
General Terms: Design, Management
Additional Key Words and Phrases: defense, Lincoln Laboratory, U.S. Air Force, real-time control, AN/FSQ-7, FST-2

Editor's Note

The definition of the SAGE system evolved from the Air Defense Systems Engineering Committee (Valley Committee) concept through many modifications as Lincoln Laboratory, the other contractors, and the U.S. Air Force faced fiscal, technical, and operational realities. By 1956, the definition of the design of SAGE was substantially fixed; most of the critical subsystems had been tested in either the Cape Cod System or the Experimental SAGE Sector. Adequate money was available. The prime contractors were able to predict how long it would take to do their jobs. Instead of writing a new paper on the definition of the design of SAGE, we have chosen to reprint a paper written in 1957, the year before the SAGE system became operational. The paper describes SAGE and all its subsystems as it was understood at the time.

By 1957, some of the SAGE direction-center buildings had been built and some of the subsystems had been installed. The System Program Office was functioning effectively, and all the participants had planned their actions according to a master schedule prepared by the Air Defense Engineering Services Project Office. The following paper was presented at the Eastern Joint Computer Conference in December 1957 in Washington. Changes made in the system after that time were generally those required to adjust (cut back) the system to match the available monies and to correct for the overestimates made by the designers. The changes also reflected the declining priority of air defense, the growing awareness of the need for integration, and the mechanisms set up to control the evolution of the system. Nevertheless, the paper is an excellent description of the system that was initially deployed.

Authors' Addresses: R. R. Everett and C. A. Zraket, MITRE Corporation, Burlington Road, Bedford, MA 01730. H. D. Benington, System Development Corporation, 7929 Westpark Drive, McLean, VA 22101.
Illustrations courtesy MITRE Archives.

The Requirement of SAGE

The past decade has shown an increase in the air threat to this country to an extent that has outdated manually coordinated traffic-handling techniques and manual data processing. General Earle E. Partridge, Commander-in-Chief, North American Air Defense Command, stated (*U.S. News & World Report*, 6 September 1957) the need for a defense system that is prepared to work instantly and that will blanket the entire United States. Until recently, we have relied on an air-defense processing system whose traffic-handling techniques were almost identical to those used during the Second World War. Fortunately, there has been substantial improvement in our inventory of automated air-defense *components*. These include improved radar systems, automatic fire-control devices, navigational systems, and both missiles and manned aircraft of high performance. But successful air de-

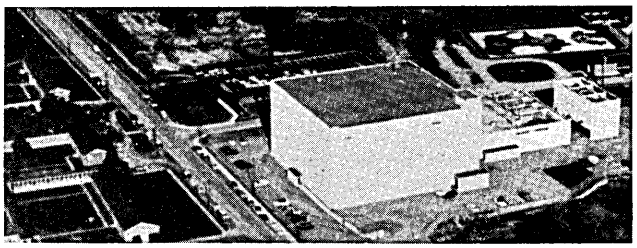


Figure 1. A SAGE direction center building contains power-generation and computing equipment, operational areas for directing sector operation, and office and maintenance facilities. Data are transmitted to this center both automatically and by voice phone. The center communicates with adjacent SAGE centers and transmits guidance data to weapons under its control.

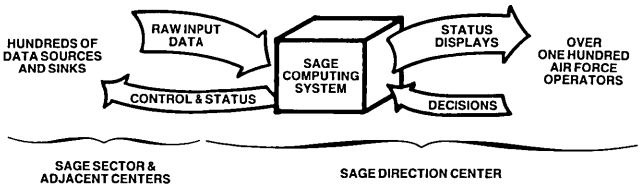


Figure 2. SAGE data processing. The direction center continuously receives input data from hundreds of locations within and without the sector. Some of these data are transmitted digitally over telephone lines and read directly into the computer; some are transmitted by teletype or voice phone and transcribed onto punched cards before input to the computer. In 1 second, over 10,000 bits of data representing hundreds of different types of information can be received at the direction center.

fense requires both good components and intelligent utilization of these components. More important, intelligent commitment of new weapons requires up-to-date knowledge of the complete enemy threat and of the success of weapons already committed.

The air-defense data-processing problem is one of nationwide data-handling capability: facilities for communication, filtering, storage, control, and display. A system is required that can maintain a complete, up-to-date picture of the air and ground situations over wide areas of the country; that can control modern weapons rapidly and accurately; and that can present filtered pictures of the air and weapons situations to the air force personnel who conduct the air battle.

The Semi-Automatic Ground Environment System—SAGE—was developed to satisfy these requirements. SAGE is a large-scale, electronic air-surveillance and weapons-control system and is composed of three groups of facilities: those required to process and transmit surveillance data from data-gathering sources to data-processing centers; data-processing centers where data are evaluated and developed into an air situation and where weapons-guidance orders are generated; and communications facilities to transmit data to weapons, to command levels, to adjacent centers, and to other users such as the Civil Aeronautics Adminstration (CAA) and federal civil defense agencies. SAGE uses very large digital computing systems to process nationwide air-defense data. SAGE is a real-time control system, a real-time communication system, and a real-time management-information system. The basic ideas of this system resulted from the efforts of George E. Valley and Jay W. Forrester, both of MIT.

A large number of organizations have contributed to the development of SAGE since its conception in the Air Force and at MIT's Lincoln Laboratory. The International Business Machines Corporation (IBM) designed, manufactured, and installed the AN/FSQ-7 Combat Direction Central and the AN/FSQ-8 Combat Control Central including the necessary special tools and test equipment. The Western Electric Company, Inc. provided management services and the design and construction of the direction center and combat center buildings. These services were performed with the assistance of the subcontractor, the Bell Telephone Laboratories. The Burroughs Corporation manufac-

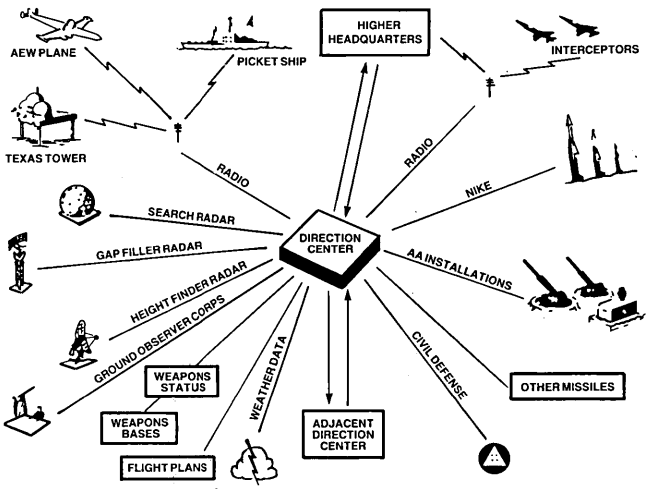


Figure 3. A direction center receives digitally coded data automatically and continuously from search radars and height finders over voice-bandwidth communications circuits. Data on flight plans, weapons status, weather, and aircraft tracks are received, respectively, from the Air Movements Identification Service (AMIS), weapons bases, USAF Weather Service, Ground Observer Corps, and airborne early-warning and picket ships over teletype and voice telephone circuits. Similarly, data from the direction center are transmitted in digitally coded form over voice-bandwidth communications circuits to ground-air data-link systems, to weapons bases, to adjacent direction centers, and to command levels; data to other users are transmitted over automatic teletype circuits.

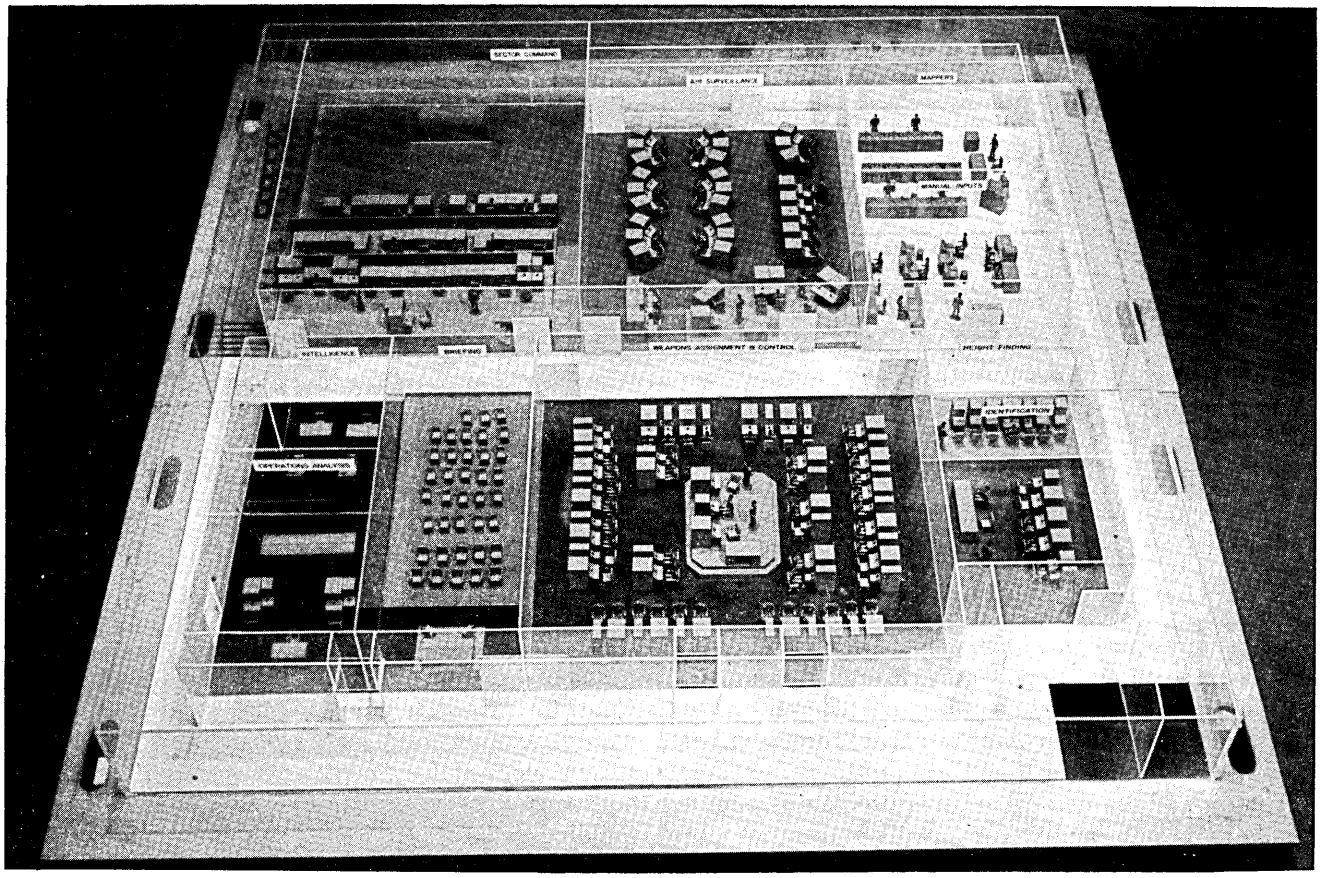


Figure 4. The fourth floor of the direction center contains separate operational rooms for air surveillance, identification, status input, weapons assignments and control, and command functions. Up to 50 operators are required in one room to man the consoles, which are directly connected to the computer.

tured, installed, and provided logistic support for AN/FST-2 coordination data-transmitting sets. The System Development Corporation (until recently a division of the Rand Corporation) assisted Lincoln Laboratory in the preparation of the master computer program and the adaptation of this program to production combat direction centers.

Sectors and Direction Centers

With SAGE, air defense is conducted from about 30 direction centers located throughout the United States (Figure 1). A center is responsible for air surveillance and weapons employment over an area called a sector. Each center contains a digital computing system—the AN/FSQ-7—containing almost 60,000 vacuum tubes. Over 100 air force officers and airmen within the center control the air defense of the sector. Most of these men sit at consoles directly connected to the computer where they receive filtered displays of the computer's storage of system-status data; they direct the computer through manual keyboards at each console. The Boston sector is typical; its direction center is located at Stewart Air Force Base in New York. Its area of responsibility extends from Maine on the north to Connecticut on the south and from New York on the west to a point hundreds of miles off the seacoast on the east.

The computer in the direction center can store over 1 million bits of information representing weapons and surveillance status of the sector at one time (Figure 2). These bits represent thousands of different types of information. For example, the computer generates and stores positions and velocities of all aircraft, or it stores wind velocity at various locations and altitudes. Within the computer, a program of 75,000 instructions controls all automatic operations; input data are processed, aircraft are tracked, weapons are guided, outputs are generated. Each second, the computer can generate over 100,000 bits of digital information for display to air force operator consoles. Each operator receives cathode-ray-tube displays that are tailored to his needs, and he may request additional information or send instructions to the computer by means of keyboard inputs on his console. Each second, the computer can generate thousands of bits of information for automatic digital transmission via telephone or teletype to weapons and missiles, to adjacent centers or higher headquarters, and to other installations within the sector.

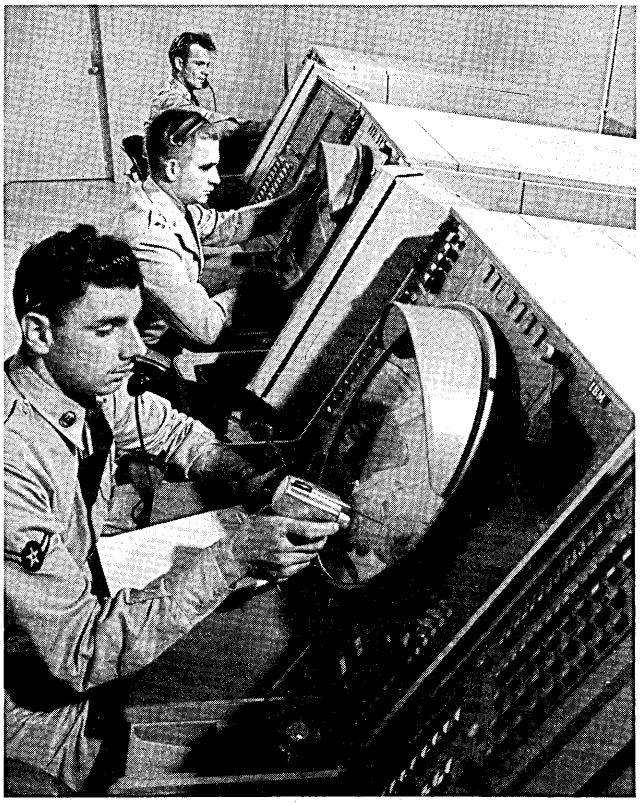


Figure 5. Each operator sits at a console that contains display and input facilities tailored to his responsibilities.

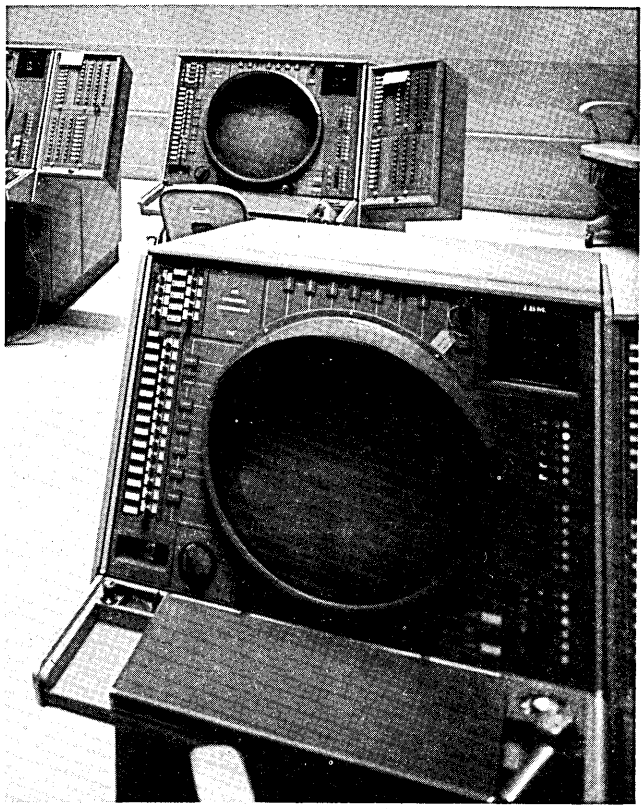


Figure 6. The operators insert data into the computer through keyboard actions.

How fast is this system? Obviously, response times from input to output vary with the task performed. Fastest response is required by automatic control functions (such as weapons guidance) and for man-machine communication (such as displays of requested information). For many of these functions, only several seconds are required from stimulus to response. For others, several minutes may elapse before the effects of new data are reflected throughout the system. We will now consider, in more detail, the three major areas that comprise SAGE data processing. First, the *sector* or *environment* that contains the data sources or sinks coordinated by the direction center. Next, the *man-machine component*—how the operators within the direction center are informed of the air situation and how they affect its progress. Finally, the *computing system* that performs the automatic component of the direction-center function.

The SAGE Sector

The direction center communicates with over 100 adjacent installations (Figure 3). Air-surveillance data are received from several types of radars. Long-range search and gap-filler radars located throughout the sector provide multiple coverage of the air volume within the sector; picket ships, airborne early warning (AEW), and Texas Towers extend this coverage well beyond the coastline; height finders supply altitude data. Within the direction center, these data are converted by the computer to a single positional frame of reference and are used to generate an up-to-date picture of the air situation. Other inputs to the direction center include missile, weapons, and airbase status; weather data; and flight plans of expected friendly air activity. Such data, which are received from many installations within and without the sector, are automatically processed by the computer and used by direction-center operational personnel to assist identification of aircraft, employment of weapons, or selection of tactics.

The direction-center computer communicates automatically and continuously with adjacent direction centers and command-level headquarters in order to ensure that air defense is coordinated smoothly between sectors and conducted intelligently over larger areas than a single sector. For example, an aircraft flies out of a sector; surveillance data from the center are automatically transmitted to the proper adjacent center in order to guarantee continuous tracking and

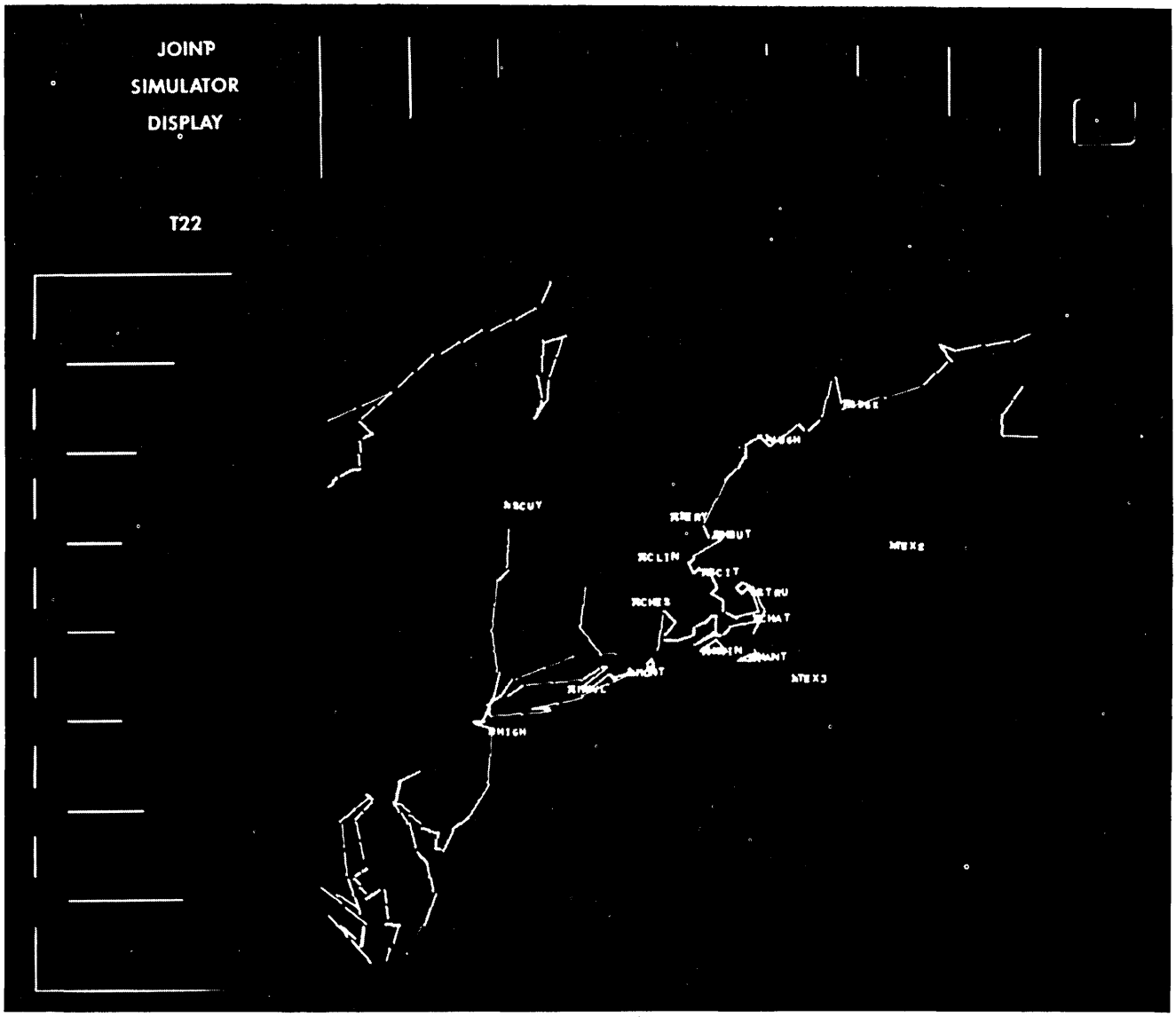


Figure 7. Situation display (on Charactron tube developed by Hughes Products Co.) of New England coastline and adjacent installations.

interception. In this way, adjacent centers are continuously warning, informing, and acknowledging. The final function of the direction center is to continuously transmit status, command, or guidance data to airborne interceptors and missiles or to related ground installations.

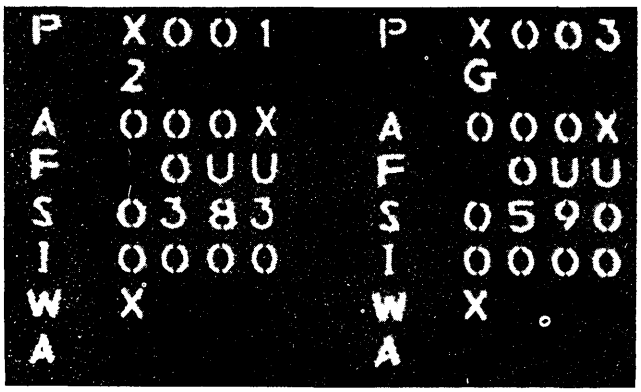


Figure 8. Typotron digital display (developed by San Diego Division of Stromberg-Carlson; formerly Convair Division of General Dynamics).

Three types of data transmission are used for both inputs and outputs. First, data sources or sinks that require high transmission rates communicate directly with the SAGE computer by means of digitally coded data transmitted at 1300 pulses per second over voice-bandwidth telephone lines and radio channels. Typical applications of this type of channel are inputs from search radars and intercommunication between adjacent centers. Teletype provides a second channel that is slower but equally automatic. Input flight plans are transmitted from Air Movement Identification Services. Finally, voice telephone communications are used in cases where high automaticity is either unnecessary, too expensive, or not feasible. If such information must be entered into the computer, either punched cards or operator keyboard inputs are used.

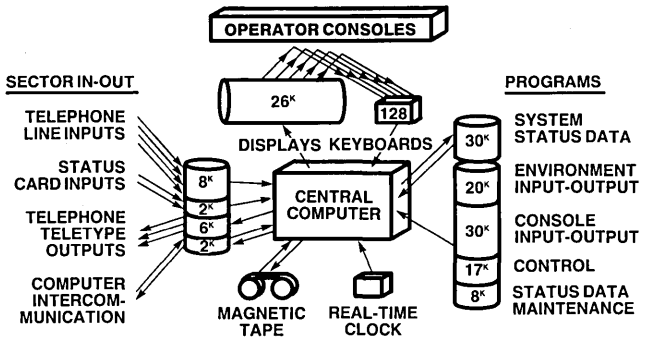


Figure 9. Each of two identical computers includes the central computer that performs all calculations, the 75,000-instruction air-defense program, and the millions of bits of system-status data. Both of the latter are stored on auxiliary magnetic drums.

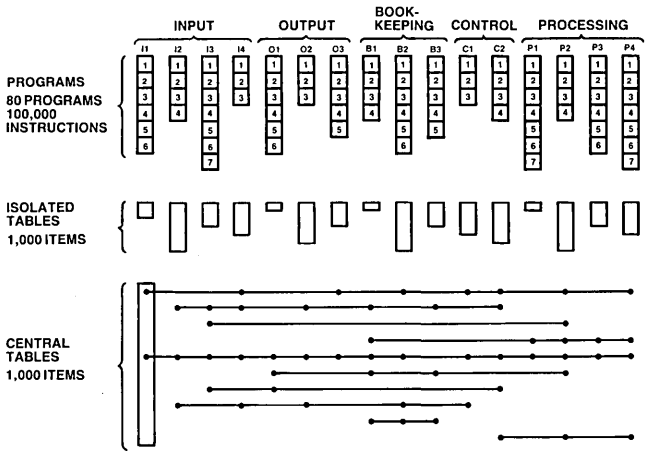


Figure 11. Static program organization.

All data sources and sinks in the sector operate asynchronously. Inputs from each source arrive at the direction center with very different average and peak rates. Each source is processed by the computer with a priority and sampling rate consistent with the role of the particular data in the overall air-defense function. Likewise, the computer generates output messages with a frequency and timing that will ensure adequate transmission of guidance and status data and yet will make maximum use of finite phone-line and teletype capacity. *One of the major functions of the SAGE computer is coordination and scheduling in real time of sector inputs and outputs with the manual and automatic functions performed in the direction center.*

The Man in the System

Although SAGE has made many of the data-processing functions in a direction center automatic, many tasks remain that are better performed by the man. Operators can relay computer outputs by phone or radio to adjacent installations and weapons; they can recognize certain patterns more rapidly and meaningfully than any of our present computers and take appropriate action. Most important, operators are required for tactical judgments such as aircraft identification or weapons deployment and commitment. If a major advantage of the FSQ-7 computer is its ability to maintain and store a complete picture of the sector situation, an equally important advantage is that the *same* computer can rapidly summarize and filter these data for individual presentation to the more than 100 air force personnel who both assist and direct air-defense operations.

The fourth floor of the center contains operational areas from which air force personnel supervise the computer and the sector. Each of the major air-defense functions—radar inputs, air surveillance, identification, weapons control, operations analysis, training, simulation, and sector command (Figure 4)—is supervised from a separate room.

Each operator sits at a console that contains display and input facilities tailored to his responsibilities (Figure 5). The operators insert data into the computer by pushing keyboard buttons (Figure 6). Each console is

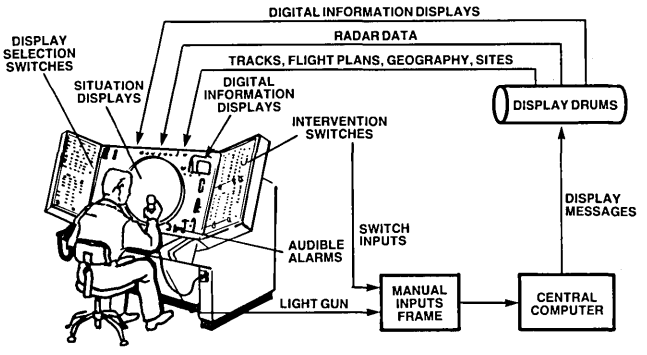


Figure 10. Major means of communication between automatic equipment and operating personnel.

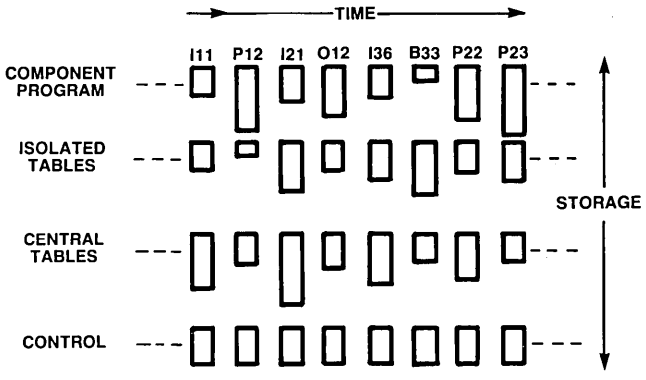


Figure 12. Dynamic program operation.

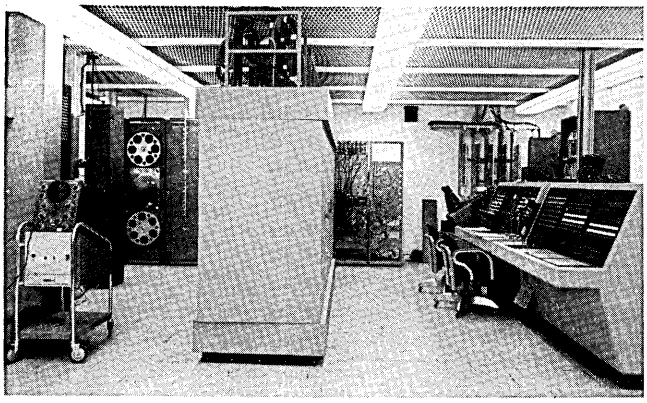


Figure 13. Digital data transmitted automatically to the direction center via telephone lines can be selected for insertion into the computer at an input patch panel.

provided with an input capacity to the computer of 25 to 100 bits of information at one time. The total keyboard input capacity for all consoles is over 4000 bits, which are sampled by the computer every several seconds.

A 19-inch Charactron cathode-ray tube displays geographically oriented data covering the whole or part of the sector (Figure 7). On this *air-situation display scope*, the operator can view different categories of tracks or radar data, geographic boundaries, predicted interception points, or special displays generated by the computer to assist his decision.

Every 2½ seconds, the computer generates about 200 different types of displays, requiring up to 20,000 characters, 18,000 points, and 5000 lines. Some of these are always present on an operator's situation display. Others he may select. Some he may request the computer to prepare especially for his viewing. Finally, the computer can force very high-priority displays for his attention.

The operator's console can also contain a 4-inch Typotron digital-display tube that is used to present status data such as weather conditions at several airbases or attention data that, for example, show the operator why the computer rejected his action (Figure 8). Sixty-three different characters are available in the Typotron. The FSQ-7 display system can display these characters at the rate of 10,000 characters every few seconds to all the digital display scopes.

SAGE Computing System

The SAGE FSQ-7 computer occupies the entire second floor of the direction center. About 70 frames containing almost 60,000 vacuum tubes are required to handle all input-output data, to perform air-defense calculations, and to store system-status data. In order to ensure round-the-clock operation, two identical computers are required. These are located on opposite sides of the floor with unduplicated input-output equipment and maintenance consoles situated in between.

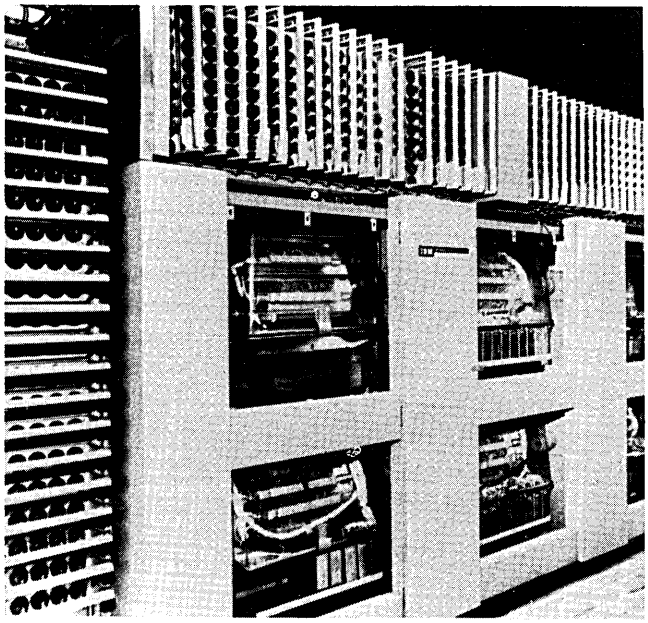


Figure 14. Magnetic drums are used for buffer storage of I/O data and storage of system-status data and computer programs. Twelve physical drums (six shown) have the capacity for almost 150,000 32-bit words. Half of this capacity is required for storage of the real-time program.

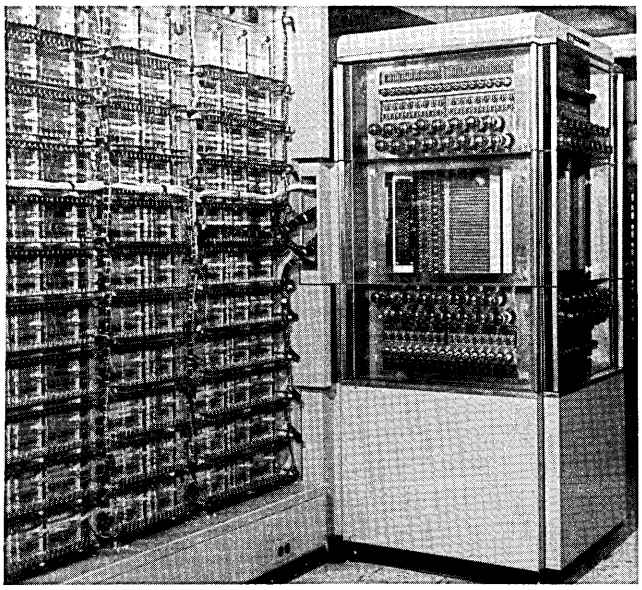


Figure 15. Magnetic core memory. The central computer is a binary, parallel machine with an 8192-word core memory and a speed of roughly 75,000 single-address instructions per second. Numbers representing positional data are stored and processed as vectors with two 16-bit components in order to facilitate processing.

Figure 16. Control console. Separate control consoles (including standard IBM punched-card equipment) and magnetic-tape units are provided for each of the duplexed computers.

Figure 9 shows the logical organization of one of the two identical computers. Since only one of these computers performs the real-time air-defense function at any one time, we can discuss simplex processing before considering the problems of duplex operation.

The computer system consists of the following major components: a central computer, the air-defense computer programs, and the system status data stored on auxiliary magnetic drums. The central computer is buffered from all sector and console in-out equipment by magnetic drums (except for the console keyboard inputs, which use a 4096-bit buffer core memory). Finally, a real-time clock and four magnetic tape units (used for simulated inputs and summary recorded outputs) complete the FSQ-7 system.

The *central computer* is a general-purpose, binary, parallel, single-address machine with 32-bit word length and a magnetic core memory of 8192 words. The memory cycle time is 6 microseconds. Each instruction uses one 32-bit word, and the effective operating rate is about 75,000 instructions per second. Four index registers are available for address modification. One unique feature of the central computer is the storage and manipulation of numerical quantities as two-dimensional vectors with two 16-bit components. In this way, a single sequence of instructions can simultaneously process both components of positional data, effectively doubling computing speed for this type of processing. Twelve magnetic drums, each with a capacity of 12,288 words of 32 bits, are used for storage of system-status data, system-control programs, and buffer in-out data. Under control of the central computer, data can be transferred in variable-length blocks between these drums and core memory. The total drum storage capacity is about 150,000 words of 32 bits.

During an average 1-second period, the central computer transfers from 20 to 50 blocks of data, each containing 50 to 5000 words, between the central computer's core memory and the terminal devices. In order to ensure maximum utilization of the central computer for air-defense processing and control, an in-out break feature is used. With this feature, calculations in the central computer continue during input-output operations; they are only interrupted for the one core-memory cycle required to transfer a word between the core memory and the terminal device. The in-out break has proved very valuable since considerably more than 50 percent of real time is required for input-output searching, waiting, and transferring.

The input-output buffering devices process in-out data independently of the central computer and so free the computer to do more complex air-defense processing. (Separate read-write heads are provided for the buffering equipment and for the central computer.) In their buffering role, these devices can receive or transmit data while the computer is performing some unrelated function.

Consider, for example, the general manner in which input data from voice-bandwidth phone lines are received. The serial 1300-pulse-per-second message is demodulated and stored in a shift register of appro-

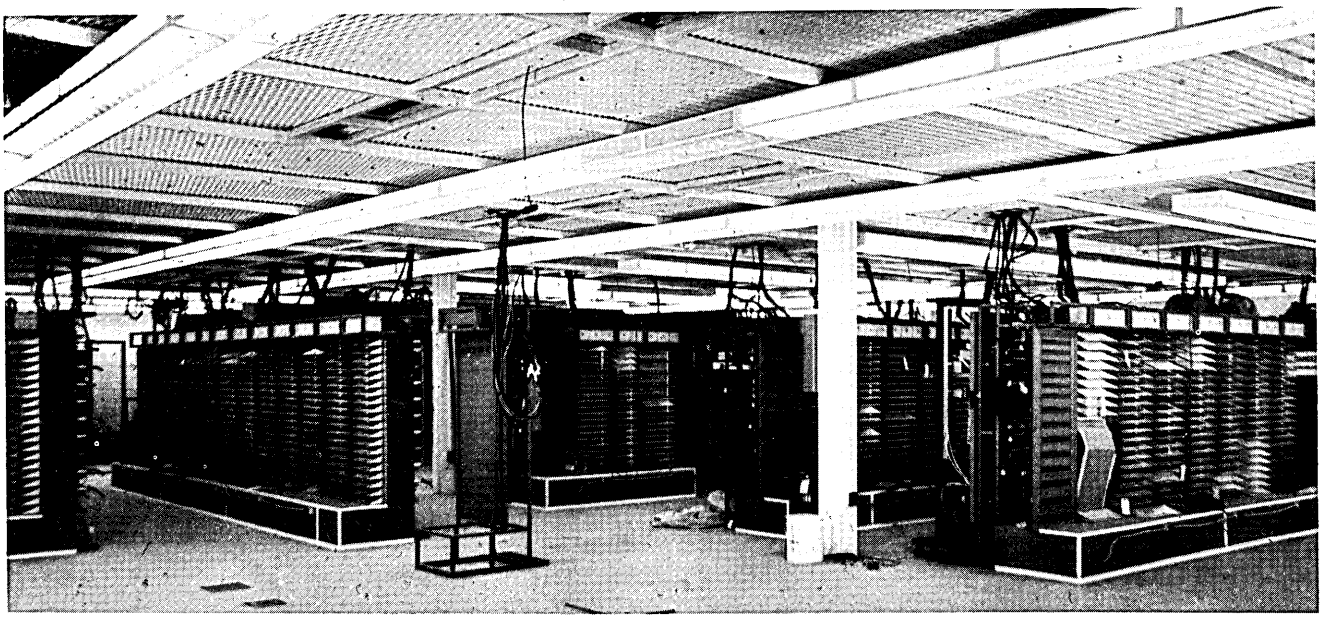


Figure 17. Central computer frames. There are about 70 frames containing nearly 60,000 vacuum tubes in the system.

priate length. When the complete message has been received, the message is shifted at a higher rate into a second shift register (whose length is a multiple of 32 bits), thus freeing the first register to receive another message. When the first empty register is located on the input buffer drum, parallel writing stores the word in 10 microseconds. A relative timing indicator is also stored on the drum with the message since the computer may not process the message for several seconds and since time of receipt at the direction center is often critical. The central computer can read these randomly stored data by requesting a block transfer of occupied slots only. Output messages are processed conversely. In a few milliseconds, the central computer can deposit (on the output buffer drum) a series of messages that will keep several phone lines busy for 10 seconds.

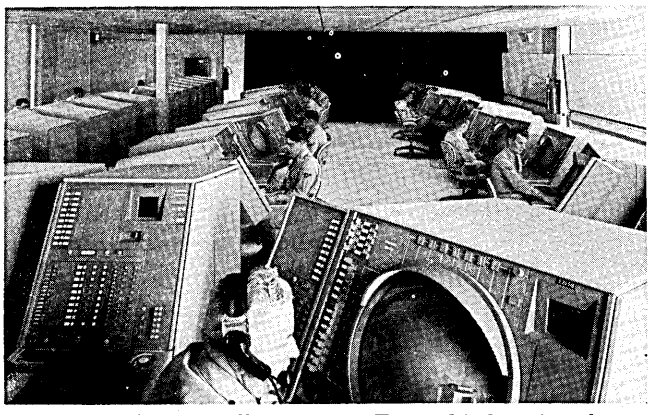


Figure 18. Air surveillance room. From this location the air force operators direct aircraft detection and tracking and communicate with adjacent direction centers. The operator in the foreground is instructing the computer to assign one of the tracks shown on his 19-inch cathode-ray display (Charactron) to another operator for special monitoring. Situation displays on this tube can be forced by the computer or requested by the operator. The small 4-inch tube (Typotron) is used for display of tabular status data. In addition, the console contains keyboard facilities for inserting data into the computer and telephone facilities to provide appropriate priority communications with other stations within and without the direction center.

The processing ability of the buffer devices is fully exploited in the display system (Figure 10). In this case, the central computer maintains a coded table on the buffer display drum. This table is interpreted and displayed by special-purpose equipment every 2½ seconds at the appropriate console. The central computer can change any part of the display at any time by rewriting only appropriate words on the drum.

The central computer performs air-defense processing in the following manner (see Figures 11 and 12). The buffer storage tables, the system-status data, and the system computer program are organized in hundreds of blocks—each block consisting of from 25 to 4000 computer words. A short sequence-control program in the central computer's core memory transfers appropriate program or data blocks into core memory, initiates processing, and then returns appropriate table blocks (but never programs) back to the drum. To take advantage of the in-out break feature, operation of each air-defense routine is closely coordinated with operation of the sequence-control program so that programs and data are transferred during data processing.

By time-sharing the central computer, each of the air-defense routines is operated at least once every minute—many are operated every several seconds. One interesting feature is that the frequency of program operation is locked with real time rather than allowed to vary as a function of load; during light load conditions the sequence-control program will often "mark time" until the real-time clock indicates that the next operation should be repeated. Such synchronization with real time simplifies many of the control and input-output functions without causing any degradation in system performance. Figures 13–19 show the SAGE system in operation.

Reliability

One last aspect of the computing system remains to be discussed: reliability. As mentioned earlier, 24-hour-per-day uninterrupted operation of the computing system was a requirement that could not be compromised. The FSQ-7 is a crucial link in the air-defense chain. If the computing system stops, the surveillance and control functions are interrupted, men and machines throughout the sector lose vital communications, and the sector is without air defense.

In order to ensure continuous system operation, any component whose failure would cripple the system has been duplexed whenever possible. As a result, two complete, independent computers are provided—each with separate drums, central computers, input-output buffering devices, and magnetic tapes. Equipment associated with individual input-output channels is generally not duplicated: consoles, phone-line demodulators, shift registers, etc. Loss of one of these pieces of equipment would merely cause loss of some data and minor system degradation, rather than complete shutdown of the direction center.

At any one time, one computer performs the air-defense job—this is the active computer. The standby machine may be operating in one of several modes: it may be down for repair (unscheduled maintenance time); it may be undergoing routine preventive maintenance (marginal checking), or even assisting in the maintenance of other equipment within the sector.

The switchover process interchanges the roles of the computers—the standby machine goes active, the active machine goes to standby. Simplexed devices connected to one machine are automatically transferred to the other, and the air-defense program begins operation in the newly active machine. From an equipment point of view, switchover requires only a few seconds. However, all of the system-status data that were available before switchover must be available to the newly active computer. Otherwise, the entire air-situation picture would need to be regenerated; this would cripple sector operations as effectively as if both computers had stopped. Accordingly, the active machine transmits changes in the air-situation data to the standby machine several times per minute via an intercommunication drum. Computer switchover is hardly noticeable to operating personnel.

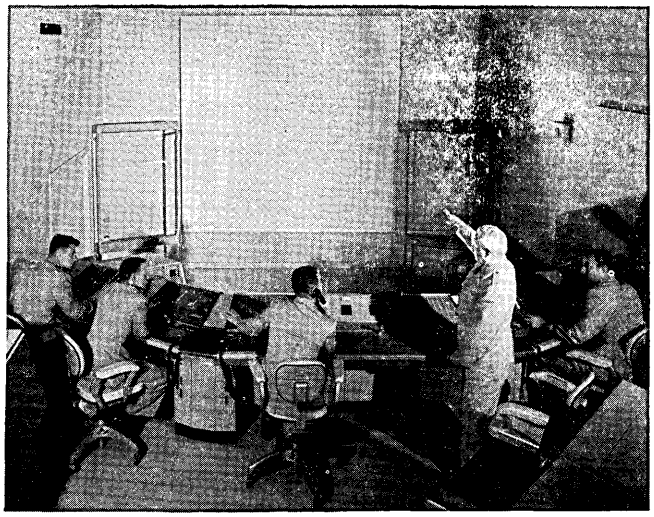


Figure 19. Command post (experimental SAGE sector). Operation of the direction center and the sector is supervised in the command post by the sector commander and his staff. A summary of the current air situation in the sector and adjoining areas is projected on a large screen.

Although the requirement for continuous operation is a stringent one, SAGE is less vulnerable than many other digital computer applications to transient errors in the FSQ-7. For most operations, the computer operates iteratively in a feedback loop. In these applications, the system is self-correcting for all but a few improbable errors. Parity-checking circuits in the input and output buffer equipment and in the computer-memory system eliminate some data subject to transient errors.

REFERENCES

Astrahan, M. M., B. Housman, J. F. Jacobs, R. P. Mayer, and W. H. Thomas. January 1957. Logical design of the digital computer for the SAGE system. *IBM Journal of Research and Development 1,* 1, 76–83.

Benington, H. D. June 1956. Production of large computer programs. *Proc. Symposium on Advanced Programming Methods,* Washington, D.C. ONR Symposium Report ACR-15, 15–27.

Israel, D. R. April 1956. Simulation in large digital-control systems. *Proc. National Simulation Conference,* Houston.

Ogletree, W. A., H. W. Taylor, E. W. Veitch, and J. Wylen. December 1957. AN/FST-2 radar processing equipment for SAGE. *Proc. Eastern Joint Computer Conference,* Washington, D.C. New York, IRE, pp. 156–160.

Vance, P. R., L. C. Dooley, and C. E. Diss. December 1957. Operation of the SAGE duplex computers. *Proc. Eastern Joint Computer Conference,* Washington, D.C. New York, IRE, pp. 160–172.

History of the Design of the SAGE Computer— The AN/FSQ-7

MORTON M. ASTRAHAN and JOHN F. JACOBS

This paper tells the story of the development of the SAGE *(Semi-Automatic Ground Environment) air-defense computer, the AN/FSQ-7. At the time of its operational deployment in 1958, the AN/FSQ-7 was the first large-scale, real-time digital control computer supporting a major military mission. The AN/FSQ-7 design, including its architecture, components, and computer programs, drew on research and development programs throughout the United States, but mostly on work done at MIT's Project Whirlwind and at IBM.*

*Categories and Subject Descriptors: K.2 [**History of Computing**]—hardware,* SAGE*, software, systems*
General Terms: Design, Management
Additional Key Words and Phrases: defense, Lincoln Laboratory, U.S. Air Force, Whirlwind, IBM Corporation, AN/FSQ-7 computer

Editor's Note

In 1952, we at Lincoln realized that the production of the SAGE computers would be a major undertaking and that it was none too soon to get a first-class manufacturer aboard. IBM was soon selected for the job, a decision no one ever regretted.

The Lincoln people, filled with the hubris of young engineers and fresh from Whirlwind, had the idea that they would design the machine and that IBM would do the production engineering, whatever that was, and build the necessary quantity. The IBM people, also proud and capable, fresh from the 701, and much more knowledgeable about what it took to produce equipment, had the idea that a page or two of specifications was all that Lincoln need supply.

The first meetings of these two groups were loud and rancorous. As I look back on them, they were social rather than technical. We argued about everything. IBM used square steel tubing for racks; MIT used L-shaped aluminum. The amount of time spent on this subject was remarkable unless one saw it (as I do now, but didn't then) as a process of getting acquainted. After a while, as the two groups began to know and respect each other, the arguments became more cogent and took place between individuals instead of between organizations. As the job grew, the Lincoln people found they had more and more to do in other areas, and the IBM group increased in size and strength and took over more and more of the job until they essentially had it all.

From my point of view it was a fine relationship. IBM did a superb job. I learned that if you want something difficult done, get people to do it who will fight with you, stand up for what they believe, and take over the job at least as fast as they are able.

I cannot think of two better individuals to describe the activity than these two principal architects of the FSQ-7.

Authors' Addresses: M. M. Astrahan, IBM Research Laboratory, K55-281, 5600 Cottle Road, San Jose, CA 95193. J. F. Jacobs, MITRE Corporation, Burlington Road, Bedford, MA 01730.
Illustrations courtesy MITRE Archives.
© 1983 AFIPS 0164-1239/83/040340-349$01.00/00

Introduction

The SAGE (Semi-Automatic Ground Environment) air-defense computer, the AN/FSQ-7, was developed at a time when Department of Defense (DOD) officials perceived that Soviet bombers carrying nuclear bombs were a primary threat to the United States. The generally held belief in the validity of this threat gave the SAGE program the highest DOD priority. The AN/FSQ-7 design, including its architecture, components, and computer programs, drew mostly on work done at MIT's Project Whirlwind and at IBM. How all this came about is the subject of this paper.

SAGE system programming is an interesting story in its own right, but is outside the scope of this paper. Similarly, the system for management of deployment worked out among the Air Defense Engineering Services project office, Lincoln Laboratory, the Air Defense Command, and the contractors deserves a more thorough treatment.

Prologue

The need for air defense was driven home in the United States by the Japanese attack on Pearl Harbor in 1941. Pearl Harbor demonstrated the need for air surveillance, warning, and real-time control. Shaken by Pearl Harbor, the United States became serious about air defense within its continental limits. By the end of World War II, there were more than 70 ground control of intercept (GCI) sites.

Each GCI site consisted of one or two search radars, a height-finder radar, and ground-to-air and air-to-ground communications. The operators sat in front of plan-position indicators (PPIs), which presented the air situation on a scope with long-persistence phosphors. Aircraft appeared as "blips" of light on the face of the tube. Information on targets from adjacent sites was cross-told by voice telephone. The control centers were usually built around a large, edge-lit, plexiglass board that showed the local geographic features. Aircraft of interest were marked on the board by operators standing on scaffolding behind the board using grease pencils. The big board also showed status information, which was written backward by the operators. The network of GCI sites became known later as the *Manual System.*

Following the Allied victory, the most powerful air forces were in the hands of the Allies, including Russia. There seemed no justification for the expense of maintaining the radar sites established during the war, and support eroded.

In 1947 the Army Air Corps was organized as the U. S. Air Force, a separate service. The Air Force was given the air-defense mission and proceeded to plan the revival of the Manual System. The importance of this mission was increased with the subsequent Russian production of nuclear bombs, and was further strengthened by events in Korea. Meanwhile, the Air Force Chief of Staff, General Hoyt S. Vandenberg, became more and more concerned about the United States' vulnerability to airborne attack. The Air Force Scientific Advisory Board was exposed to the problem, and in 1949, the board set up an Air Defense Systems Engineering Committee (ADSEC) under George E. Valley, a physics professor at MIT. The committee became known as the Valley Committee.

The Valley Committee began by looking at the newly reactivated air-defense system. This system had been authorized by Congress through the Air Force, and consisted of about 70 GCI radar sites. Except for improved radars and height finders, it was quite similar to the Manual System air-defense setup established during World War II. The committee quickly concluded that the air-defense system as reshaped by the Air Force had very low capability. It recommended that a competent technical organization look into what could be done to improve the system in the short run. As a result, the Western Electric Company and the Bell Telephone Laboratories were given the task of upgrading the existing system; this was to become the Continental Air Defense System (CADS) project. The Valley Committee also suggested that a longer-range look be taken at the problem. It recommended the extensive use of automation, particularly computers, to handle the bookkeeping, surveillance, and control problems in the implementation of the next generation of air-defense systems. This conclusion was supported by the development of the Whirlwind computer at MIT. Whirlwind promised to provide real-time control over a large number of aircraft. It was also noted that the ability to pass digital information over phone lines had been demonstrated at Bell Telephone Laboratories and at the Air Force Cambridge Research Laboratory. To deal with one of the major problems, low-altitude surveillance, the committee recommended the establishment of a large number of short-range, low-maintenance radars placed close together to fill gaps in coverage.

The Valley Committee report led Vandenberg in December 1950 to ask MIT to establish a laboratory for air-defense research and development. The Air Force Scientific Advisory Board endorsed this request and asked MIT to undertake an interim study of the air-defense problem. The study, called Project Charles, ran from February to August 1951. It gave further support to the concept of a computer-based system. The laboratory was established within MIT

in 1951 as Project Lincoln, and in 1952 became the MIT Lincoln Laboratory. The SAGE system evolved from the work of this laboratory (MITRE 1979; Redmond and Smith 1980).

Project Whirlwind

The Whirlwind computer project at MIT's Digital Computer Laboratory (DCL) was of crucial importance to the development of the AN/FSQ-7 for several reasons. First, it provided a demonstration of real-time control by a digital computer, without which the SAGE project could not have been approved. Second, it provided a reservoir of people with the skills and experience needed to participate in the SAGE system design and development. Third, it provided an experimental testbed for the system design. The story of the Whirlwind project and the role of key people like Jay W. Forrester and Robert R. Everett has been described by Redmond and Smith (1977; 1980).

The Cape Cod System

In the spring of 1952, DCL operations and people concerned with air defense were merged into Lincoln Laboratory as Division 6. Whirlwind was working well enough to be used as part of Lincoln's experimental air-defense system, called the Cape Cod System. It consisted of a control center at the Barta Building in Cambridge, Mass., where Whirlwind was housed, an experimental long-range radar on Cape Cod at South Truro, Mass., and a number of short-range radars called "gap fillers." The control center contained computer-controlled operating stations for interaction with human operators. It was equipped with ultrahigh-frequency communications to aircraft supplied by the Air Research and Development Command (ARDC) and the Air Defense Command (ADC), for the purpose of creating a realistic test of the system.

The Valley Committee and Project Charles had indicated that a preferred solution for dealing with the low-altitude detection problem was to connect together many radars (preferably short-range, low-maintenance ones) and make a composite picture of the air situation out of the data taken from these radars. It was largely the need to deal with so much data that prompted the Valley Committee to favor the use of the computer aids in processing the data in real time. Just as Whirlwind had the potential for filling the needs for this additional data load, work at the Air Force Cambridge Research Laboratory (CRL) under Jack Harrington on digital transmission of radar data had the potential for filling another need: communicating the data. Harrington's group had developed a technique (actually, several techniques) for transmitting these data. One technique, called slowed-down video, divided the coverage area of short-range radars into a large number of wedge-shaped boxes, the number bounded by range resolution required and the angular resolution that one could achieve with the radar. The boxes were mapped onto a stream of bits sent on a phone line. The stream was synchronized with the radar pulses and the angular position of the radar. Each bit was a 1 if the corresponding box contained a signal return above a certain magnitude; otherwise it was 0. This technique showed promise for short-range radars, but it was far too inaccurate for the long-range radars.

CRL was also working on methods of providing more angular precision than could be achieved by means of beam forming. One scheme that eventually resulted in another SAGE development, called the AN/FST-2 (Ogletree et al. 1957), derived from beam-splitting experiments carried on at CRL. It depended on the fact that as a radar beam rotates, the pulse rate is high enough that several returns are received from a single aircraft. Harrington's group invented a device that determined the center of the target after the beam had swept over it. This device made it possible to increase the angular accuracy by an order of magnitude. Harrington's team also developed a scheme for sending generalized digital data over a standard phone line that had been adapted to the Cape Cod System. Harrington and many of his team from CRL joined Lincoln Laboratory when it was instituted. As soon as Whirlwind was able to perform, an experimental MEW radar at Hanscom Field was connected by phone line to Whirlwind, and the first tracking programs were developed. By 1952, the Cape Cod team had demonstrated the ability of the computer to track and control aircraft in small numbers. The Cape Cod System was intended to demonstrate the operations that were to be executed for field use—in particular, the surveillance function and weapons-control function. Both functions required information on the position of hostile, friendly, and neutral aircraft. A scheme where all of the operators in the control center worked from the same positional database became a requirement.

In the scheme that was adopted, target data were transmitted to the center in angular coordinates. The computer translated the data into Cartesian coordinates and combined them with the position of the radar that picked up the data, so that each piece of data had an X–Y position in a common coordinate system and could be compared with stored track data (successive positions of an object being tracked). Each operating station was equipped with a console with a

cathode-ray-tube (CRT) situation display that combined track and map data. During the course of the operating cycle, the computer presented successive data locations to an X–Y deflection register that simultaneously positioned the beam on each of the operating stations. The operator used the so-called light gun to tell the computer to associate a track with other keyed information, such as track number, identification, altitude, speed, and armament. The operator placed the light gun over the display screen at the position of interest and pressed a trigger switch (Figure 1). When the screen was illuminated at that position, a signal was sent to the computer saying in effect that the deflection register contents identified the data item selected by the operator.

In order to reduce the load on the tracking programs, radar returns from fixed objects were filtered from the gap-filler data by a device called a video mapper. The mapper was a standard plan-position display for a single radar with a photocell viewing the whole display. Returns from fixed objects were covered with opaque material so that these returns did not activate the photocell and thus were rejected.

By the time the Cape Cod System was finished, it had about 30 operational stations with appropriate displays. The data required by an operator could not all be accommodated on the graphic situation display, so the Whirlwind group created an auxiliary display for text data associated with a particular track.

The Cape Cod System was used in exercises that included SAC bombers playing the role of hostiles, and the ADC and ARDC interceptors playing a friendly role. Before the experimental SAGE sector that grew out of the Cape Cod System was finished, 5000 or so sorties had been flown against the system to test the system and its component parts.

Whirlwind II

It was clear to those who had participated in the Valley Committee and in Project Charles that Whirlwind was more of a breadboard than a prototype of the computer that would be used in the air-defense system. To turn the ideas and inventions developed in Whirlwind into a reproducible, maintainable operating device required the participation of an industrial contractor. The conceptual production computer became known as Whirlwind II.

The Whirlwind II group was set up in 1952 to deal with all design questions, including whether transistors were ready for large-scale employment (they were not) and whether the magnetic-core memory was ready for exploitation as a system component (it was). The Whirlwind II group also spent much of its time

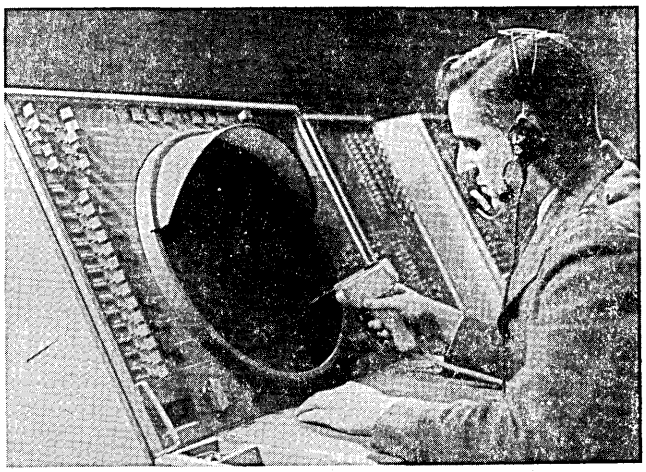


Figure 1. Light gun in use.

in negotiation with ADC and ARDC headquarters personnel in structuring the overall air-defense system, including the definition of areas of control, cross-telling among sectors, need for weapons allocation, manning requirements, and air-defense doctrine.

The most important goal established for Whirlwind II was that there should be only a few hours a year of unavailability of the operational system. The Whirlwind II team thought this was possible, extrapolating from the experience on the Cape Cod System. Most of the design choices faced by the Whirlwind II group involved the trade-off among the number of tracks that could be processed, the number of interceptors that could be employed simultaneously, and the system availability criteria.

Selection of a Computer Contractor

The idea of engaging a manufacturer to help with the design engineering and manufacturing of the field computer was implicit in the nature of the research-and-development mission of Lincoln Laboratory. A team was set up consisting of: Jay W. Forrester, head of Lincoln Division 6 and director of DCL; Robert R. Everett, associate director of Division 6 and associate director of DCL; C. Robert Wieser, leader of the Cape Cod System design; and Norman H. Taylor, chief engineer of the division. They were responsible for finding the most appropriate computer manufacturer and designer to translate the progress made so far in the Cape Cod System into a design for the next-generation air-defense system. This system was to become known as the Lincoln Transition System. In 1954 it was renamed SAGE.

Early in 1952, the team made a survey of the possible engineering and manufacturing candidates and

chose four for further evaluation: IBM, Remington Rand (two different divisions), and Raytheon. The team visited all three companies, reviewed their capabilities, and graded them on the basis of personnel, facilities, and experience.

The team looked at the technical contributions of the companies in terms of reliable tubes and other components, circuits, hardware, packaging, storage systems, and magnetic tape units. The companies were graded on their probable capability of bringing the Whirlwind II from development to production, including their experience in setting up production of high-quality electronics, their understanding of tests required, and the availability of their trained people. The team evaluated the production organization, the quality of assembly work, size of organization, similarity of the proposed work to the company's standard product, present availability of production capacity, service organization, and training ability. Finally, the team considered the proximity to MIT and the train travel time to the various headquarters. Each of the four men on the team made his own assessment, using the weights decided on before the trip. IBM received the highest score and was selected.

The IBM decision to accept the contract was made at the highest management levels. It involved evaluation of the risks versus the benefits. Some of the risks considered were technical feasibility, monetary risk, effect on commercial programs of losing people to the project, and potential liability for mishaps posed by the operation of a real-time system. Advantages included direct involvement in technical advances plus an opportunity to respond to a national defense need.

The IBM Contract

IBM set up its SAGE effort as a separate project independent of the usual constraints of commercial development. It was expected to set its own design, procurement, test, and documentation procedures commensurate with the stringent requirements of the contract.

"Project High" began in September 1952 in anticipation of a study subcontract from the MIT Lincoln Laboratory. A six-month subcontract was issued in October. Office space was rented on the third floor of a necktie factory on High Street in Poughkeepsie, N.Y.—the project got its name from this location. John Coombs, who had recently joined IBM from Engineering Research Associates, was the first project manager.

During the next few months, the expanding IBM group learned the current status of air-defense studies. The group visited the Boston area frequently in order to study the Cape Cod System and to become acquainted with the overall design strategy of the Lincoln Labs people as well as their specific proposals for central processor design. A visit was made to a competing system at the University of Michigan—the Air Defense Integrated System (ADIS), which grew out of Project MIRO, a ground-control system for the BOMARC ground-to-air missile.

In January 1953 the system design began in earnest. IBM bought the High Street building and assigned 26 people to the project. The Lincoln Whirlwind II team organized itself along major subsystem lines: an arithmetic-element section, a memory section, drum-design section, and so forth. The IBM team organized itself in a similar pattern. These counterpart groups began trying to design the system on a joint basis. The Lincoln group, fresh from its experiences of making Whirlwind I operate and designing the Cape Cod System, viewed the IBM task as that of packaging Whirlwind devices so the system could be reproduced easily and quickly. On the other hand, the IBM people expected to participate in all levels of central computer system design and favored the technology familiar to them.

The AN/FSQ-7 was designed by joint MIT–IBM committees that managed to merge the best elements of their members' diverse backgrounds to produce a result that advanced the state of the art in many directions. The committees presented their proposals at joint meetings that often involved 20 to 40 participants. Miraculously, these groups were able to arrive at a consensus and make progress. The MIT people had the final word on design specifications, but most decisions really were based on joint agreement.

During 1953 the meetings involved a lot of traffic between Poughkeepsie and the Boston area. Because of bad roads, driving was difficult. Some of the early meetings were held in Hartford, Conn., which was the halfway point between Poughkeepsie and Bedford, Mass. Another way of going from Poughkeepsie to Boston was to take an evening train to New York and a berth on the Midnight Owl to Boston. Small groups began chartering aircraft for a one-hour direct flight. On several occasions a large group would charter a DC-3. This helped to justify IBM's first corporate aircraft.

The first Hartford meeting was held January 20, 1953. John Coombs, the senior IBM man at the meeting, said that the purpose of the meeting was to allow the people working on the system, at both MIT and IBM, to exchange descriptions of what was being done. Jay Forrester, the first Lincoln speaker, went into some detail about the background of the program and his perception of the roles of the Lincoln and IBM

people. He characterized the program as urgent, with a prototype system required by 1954. He referred to memorandum TM-20 which contained a description of what was then known as the transition system. He stated that none of the existing computers, including Whirlwind I, the IBM 701, and others, were suitable. Because of the nature of the problem, specialized peripherals would be required, and existing machines had nothing like the reliability required for the job. Forrester suggested that IBM place a representative at the Cape Cod facilities. He gave a fairly complete description of the status of Whirlwind II thinking at MIT.

J. F. Jacobs of Lincoln presented the arguments for choosing vacuum tubes for the arithmetic and control units. It was too early for transistors, and magnetic-core circuits were too slow. H. D. Ross of IBM reported some tentative arithmetic-element decisions, including the use of one's-complement arithmetic and the use of flip-flops instead of the pulse regenerator used in the IBM 701. M. M. Astrahan of IBM described proposals for logical design innovations. These involved index registers, dual arithmetic elements for simultaneous processing of X and Y coordinates of tracking data, and an interrupt scheme for operating in-out equipment simultaneously with program-instruction execution.

Other Lincoln speakers included R. L. Best on basic circuits, W. N. Papian on magnetic-core memory, J. H. McCusker on magnetic core production, and K. H. Olsen on the Memory Test Computer. Other IBM speakers were N. P. Edwards on nonmemory magnetic-core applications, E. H. Goldman on buffer storage and display, and J. A. Goetz on component reliability and standardization.

Lincoln's N. H. Taylor discussed the schedule. He told the group that Lincoln had set an objective of having a prototype computer with its associated equipment installed and operating by January 1, 1955. Installation, testing, and integration of the equipment in the air-defense system had to be started on July 1, 1954. The nine months preceding this, October 1, 1953, to July 1, 1954, would be required for procurement of materials and construction of the model. That left about nine months for engineering work in connection with the preparation of specifications, block-diagram work, development of basic circuit units, special equipment design, and all the other things necessary to permit actual construction to begin. The schedule for this work was very tight. Taylor estimated that IBM would require about 235 development engineering professionals at the peak.

The meeting was concluded by T. A. Burke of IBM who described IBM's progress on the subcontract, which would end in three months. He was concerned that the follow-on Air Force prime contract be issued in time to avoid interruption of work.

A second joint meeting was held in Hartford on April 21, 1953. The first meeting had resulted in formation of a number of committees made up of IBM and MIT engineers who were to prepare design specifications. The second meeting consisted mostly of status reports from these committees.

In April IBM received a prime contract for computer design specifications. On May 21 another Hartford meeting was held, this time to deal with packaging of Whirlwind II. Much of the meeting was spent on standardization of pluggable units. It was agreed that the mechanical design group should proceed with the design of a six-tube pluggable unit, with backup designs for four-tube and nine-tube units. Another meeting on packaging was held June 1, 1953, at which a final decision was made to have both six-tube and nine-tube units. A breakdown of the central machine (arithmetic, control, and memory) into seven main frames was described.

Robert P. Crago joined Project High in June 1953. He became manager of engineering design in July 1954, and manager of Project High in February 1955.

Project Grind

The Hartford meetings acted as an information exchange, a catalyst for initiating action, an opportunity to identify overlooked aspects of the machine, and a forum in which people could interact on a personal level. By the time of the last Hartford meeting, a modus operandi had been established between the IBM and the MIT staffs, who had basically agreed on the central machine. It would have a single-address order code in a 32-bit word. The memory would have a read-write cycle in the range of 5.5–7.5 microseconds for 8192 words of 33 bits, including a check bit. Data words required only 16 bits, so each retrieval involved two data words.

The central machine turned out to be the easy part of the job. In the rest of the system, decisions were not being made fast enough to meet the schedule. There was not enough time for detailed study of all the alternatives available, so choices had to be made primarily on the basis of the experience the individuals had with the subject area under consideration. To expedite this decision making, it was agreed that a series of meetings would be held in which as many of the necessary decisions as possible would be made in a short period of time. These meetings were called Project Grind because the participants were to grind away at each topic until a decision was reached. There

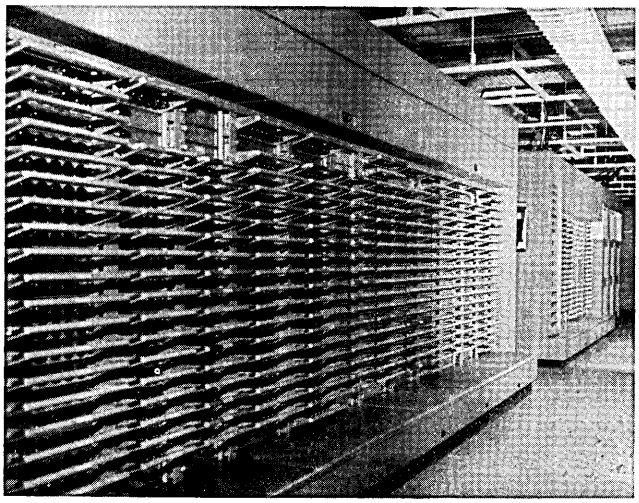


Figure 2. Typical computer frame (front).

were seven days of these meetings between June 24 and July 15, 1953. In order to identify the machine under design within IBM as well as MIT, the Whirlwind II name was dropped in favor of air force nomenclature, and the system was given an air force number, AN/FSQ-7. An AN/FSQ-7 planning group was identified, consisting of about 20 members drawn from both IBM and MIT. The procedure that was followed consisted of taking subsystems one at a time and forging whatever decisions could be made with the existing background and knowledge. Minutes of the Project Grind meetings were taken to record some of the decisions and some of the reasons for those decisions. Any problem could be brought into the open so that decisions could be made as soon as possible. It was also agreed that everyone should present even tentative plans for various parts of the system, as long as everyone knew that they were tentative.

The first Project Grind meeting, on June 24, 1953, was devoted to the radar inputs. Slowed-down video inputs, video mappers, and slowed-down video-input registers were discussed, and participants agreed on a general description of the input registers.

At the second meeting the subjects were marginal checking, power supplies, and magnetic-core memory. The third meeting dealt with magnetic drums. It was tentatively agreed that there would be six parallel fields of 34 bits each (two bits for status) per physical drum, with two heads per bit for input-output buffering, and probably five physical drums in the computer. The fourth meeting was concerned with output display systems. A 2-second display cycle was tentatively accepted; all display data streamed by the display consoles every 2 seconds, and each console displayed the items requested by the operator. It was agreed that there would be 16 words available per displayed track, allowing for display of history of all tracks. The fifth meeting was concerned with cross-telling, output drums, output links for digital information, a display-maintenance console, and mechanical design.

At the sixth meeting the concern was standard circuits and the action of the standards committee. Four tube types were definitely approved. It was decided that 0.1-microsecond pulses would be used wherever possible in the system. It was generally agreed that a project meeting should be held at least once every other week.

The seventh and last meeting, on July 15, covered mapper subcontracts, cross-telling, review of the drums, paper-tape machines, input counters, manual inputs, and power supplies. It was generally agreed that paper tape would not be used in the FSQ-7.

Development and Production

Project Grind resulted in fewer decisions than considered necessary to meet the schedule, but it had a remarkably good effect on the working relations of the people involved. It also demonstrated the need for some ongoing method for reaching a consensus on high-level specifications.

This need eventually prompted Lincoln to set up a Systems Office, under the direction of J. F. Jacobs, to establish what was then called design control. It was necessary for IBM and MIT to come to terms on the design of the FSQ-7. It was also necessary that a description, in specification terms, be written of what the Air Force was buying. The Systems Office took inputs from IBM, MIT, ADC, and Lincoln Project Office of the Air Force—and later inputs from the 4620th Air Defense Wing—and created a forum in which consensus about the main features of the design in all aspects of the system could be obtained. When this consensus was reached on the various parts of the system, a document would be prepared for the purpose of recommending to the Air Force that it approve or disapprove all or part of a proposed procurement of the pieces of the system.

IBM created a three-man Engineering Design Office to control system design—the IBM interface to Jacobs's Systems Office. The three individuals shared a common office, promoting close communication and cooperation. Design, procurement, implementation, and test principles and practices were initiated and controlled from this central point. The commercial design practices and components then available were not adequate to meet the stringent reliability requirements of the air-defense mission. Few military specifications were applicable, so new component specifi-

cations and design practices were required. The design practices and disciplines developed for the SAGE computer later helped IBM to standardize the hardware of its commercial product line.

In September 1953, IBM received a contract for two single-computer prototype systems, XD-1 and XD-2. XD-1 replaced Whirlwind in the Cape Cod system during 1955. The arithmetic, control, and memory units were shipped in January to the Lincoln site in Lexington, Mass. (Figures 2 and 3). Final testing was done there, along with integration of other frames shipped during the year. The modified system was renamed the Experimental SAGE System. The XD-2 was produced to support programming system development and to provide a hardware testbed in Poughkeepsie.

The broad outline of the SAGE network was delineated in 1954. The first serious plan visualized 46 computerized direction centers. It became evident to the Air Force that it would be desirable to automate the Air Defense Division headquarters. These headquarters, called combat centers, had the responsibility for directing the operations and allocating weapons on a large-scale basis, involving several direction centers. This called for a computer like the FSQ-7 with a specialized display system.

The system was named the FSQ-8. The locations for Q-7s and Q-8s were chosen and a delivery schedule was worked out calling for production of three systems the first year (1957) and 10 to 12 in each of the subsequent four years. As the program continued, periodic revisions were made of the number of automated sectors and the installation schedule.

The first production contract was awarded to IBM in February 1954. The first production system was accepted in its manufacturing test cell on June 30, 1956, and was declared operational at McGuire Air Force Base on July 1, 1958. To implement the deployment schedule, IBM built a manufacturing plant in Kingston, N.Y. IBM manufactured a total of 24 FSQ-7s and three FSQ-8s. These were deployed along the northern perimeter and the east and west coasts of the United States.

Innovations

The SAGE system provides a demonstration of the kind of innovation that can be achieved when cost is secondary to performance. This kind of environment is difficult to create in a commercially oriented company, but SAGE provided the environment. Ambitious performance goals were met by the operational systems. Furthermore, as hardware costs dropped, most of the SAGE innovations became cost effective for the commercial market. The following items are highlights of some of these innovations.

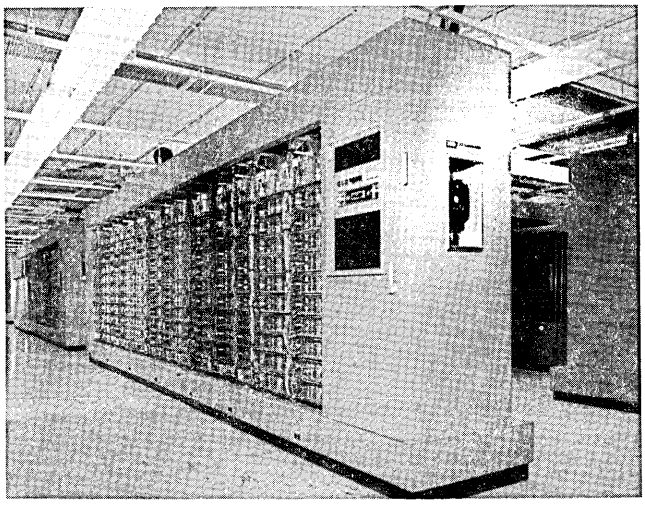


Figure 3. Typical computer frame (back).

1. *Core memory in a production machine.* This is probably the single most important innovation in SAGE. The size and reliability required could not have been achieved by any other memory technology existing or proposed in 1953. The core memory used in SAGE evolved directly from the pioneering work of Forrester and the MIT groups that developed the feasibility model and built the Memory Test Computer (MTC). "By May 1953, the MTC was demonstrating the swift, highly reliable operation of arrays of cores 32-by-32, stacked 16 high" (Redmond and Smith 1977). The original system design called for 8192 words of 33 bits, including a check bit, arranged in two banks of 33 planes. Each plane was a 64 × 64 matrix. When the requirements of the application program became apparent, a 256 × 256 unit (65,536 words) was designed to replace one of the smaller banks. In cooperation with the MIT group, IBM developed the methods of manufacturing and testing uniform, reliable, and inexpensive core memory in production quantities. This involved an automatic core tester and a core-plane stringing process that used hypodermic needles to guide the fine wires through the tiny cores.

2. *Active-standby duplex system.* The AN/FSQ-7 was the first computer system to use two computers in active-standby roles for reliability. In previous dual-computer systems, both computers did the same thing and compared output. In SAGE, the standby computer could run test programs or other work while the active computer ran the air-defense programs. The active computer maintained situation-status information on

an intercommunication drum accessible by both computers (Everett et al. 1957; Vance et al. 1957).

The original concept was three computers located at different sites within a geographic area called a sector. The radar inputs were to be connected to two of the three with sufficient displays that any two of the three could run the system at full capacity. This mode was rejected because of the high costs of communication and replicated personnel support facilities.

The duplex decision was not made until November 1953. Because it involved design changes in the input-output components, a separate group was formed to do the redesign without affecting the schedule for construction and test of the XD-1 and XD-2 prototypes. The design philosophy was to duplicate every unit that could shut down the whole system. Thus the central computer and input drums were duplicated, but display consoles and modems were not (Everett et al. 1957). Great care was taken to ensure that the switchover facilities did not introduce single failure modes affecting both of the duplexed systems.

3. *Digital communication over standard phone lines.* The transmission of digital data over voice-grade phone lines at 1300 bits per second was pioneered by the Lincoln people. Harrington's group (Division 2) designed the first modems to convert digital data to and from analog waveforms that could be accommodated by voice-band channels. The channels required special conditioning to minimize noise pickup and eliminate unequal phase shifts across the frequency spectrum. The phase shifts were not noticeable in voice transmission but distorted the data waveforms.

4. *Time-sharing.* Time-sharing a computer for real-time tracking of hundreds of airplanes, real-time control of weapons, and interaction with human controllers was a bold concept. It required invention of programming techniques to ensure timely sequencing through all the tasks (Everett et al. 1957). Programs and data tables were paged in from drums, and only the tables were rewritten. Data input-output and display data were fully buffered by drums.

5. *Input-output (I/O) control with memory cycle stealing.* SAGE marked the introduction of the I/O break, also called memory cycle stealing. This forerunner of modern channels allowed computation to continue during I/O operations, interrupted only for the core-memory cycle required to transfer a word between the core memory and the I/O device (Everett et al. 1957). It involved a register to count the number of words transferred and a memory address register, incremented for each word transferred, to specify the location of the next word (Astrahan et al. 1957).

6. *Associative input system with drum buffer.* The input buffer drums contained radar data intermixed from several sites. Each data item was tagged with the identity of its source radar. The central processing unit (CPU) could request all the data from a particular radar. This constituted an associative memory access.

7. *Branch and index instruction.* The AN/FSQ-7 index registers were an adaptation to a parallel machine organization of the Williams B-tube (Williams and Kilburn 1952). The branch and index instruction allowed a single instruction to decrement an index register, test for the end of a loop, and branch back to the beginning of the loop (Astrahan et al. 1957).

8. *Computer control of marginal checking.* Marginal checking by varying supply voltages was proved effective for vacuum-tube circuits by the Whirlwind experience. The AN/FSQ-7 extended the capability by allowing program control of the voltage excursion magnitude and its point of application (Astrahan and Walters 1956).

9. *Display, light gun, and keyboard input in a production machine.* The Cape Cod System demonstrated the functions needed in a cathode-ray-tube display console, including the use of light guns. The AN/FSQ-7 display system constituted the first use of such consoles in a production computer system. The graphic situation displays used the Convair 19-inch Charactron tubes in which the electron beam was passed through a mask in order to shape the beam into the form of one of 64 characters. The shaped beam was then deflected to the desired position on the screen. A textual display used the 5-inch Hughes Typotron, which also had a character mask but had a storage screen instead of the standard phosphor. IBM designed the display consoles but subcontracted production to Hazeltine.

10. *Circuit standards.* A central-circuit design group was responsible for design or approval of all CPU circuits. The group followed a set of design standards based on component tolerances and compatibility with marginal checking (Nienburg 1956).

11. *Component specifications and vendor control.* Special contracts were made with manufacturers of vacuum tubes, capacitors, diodes, and resistors to ensure the uniformity and reliability of the products. IBM required these vendors to institute strict controls over the design, manufacture, and testing of the components and actually monitored the manufacturing and testing processes at these vendors' plants (Heath 1956).

12. *Circuit packaging.* In the pluggable units, all components except vacuum tubes were mounted on etched circuit boards. IBM's Manufacturing Engineering Department worked with General Mills to develop the Autofab machine, which assembled and soldered the circuit boards. These automatic soldering

techniques greatly increased the reliability of the circuit boards, as did the development of double-sided boards with plated-through holes.

Postscript

An AN/FSQ-7 system weighs 250 tons and has a 3000-kilowatt power supply. Twenty-four FSQ-7 systems were installed. The first began operating in 1958 at the McGuire Air Force Base direction center in New Jersey. Performance data on the seven remaining systems were compiled for the 24-month period from March 1978 to February 1980. Each system used 49,000 vacuum tubes. The tubes had a mean time to failure of 50,000 to 100,000 hours. The average percentage of time that both machines of a system were down for maintenance was 0.043 percent, or 3.77 hours per year. The average percentage of time both machines were down for all causes, including air conditioning and other situations not attributable to the computers, was 0.272 percent, or 24 hours per year.

REFERENCES

Astrahan, M. M., and L. R. Walters. December 1956. Reliability of an air defense computing system: Marginal checking and maintenance programming. *IRE Transactions on Electronic Computers EC-5*, 4, 233–237.

Astrahan, M. M., B. Housman, J. F. Jacobs, R. P. Mayer, and W. H. Thomas. January 1957. Logical design of the digital computer for the SAGE system. *IBM Journal of Research and Development 1*, 1, 76–83.

Everett, R. R., C. A. Zraket, and H. D. Benington. December 1957. SAGE—A data processing system for air defense. *Proc. Eastern Joint Computer Conference*, Washington, D.C. (see the preceding article in this issue of the *Annals*).

Heath, H. F., Jr. December 1956. Reliability of an air defense computing system: Component development. *IRE Transactions on Electronic Computers EC-5*, 4, 224–226.

MITRE Corporation. 1979. "MITRE: The First Twenty Years." Bedford, Mass.

Nienburg, R. E. December 1957. Reliability of an air defense computing system: Circuit design. *IRE Transactions on Electronic Computers EC-5*, 4, 227–233.

Ogletree, W. A., H. W. Taylor, E. W. Veitch, and J. Wylen. December 1957. AN/FST-2 radar processing equipment for SAGE. *Proc. Eastern Joint Computer Conference.* Washington, D.C. New York, AIEE, pp. 156–160.

Redmond, K. C., and T. M. Smith. October 1977. Lessons from Project Whirlwind. *IEEE Spectrum 14*, 10, 50–59.

Redmond, K. C., and T. M. Smith. 1980. *Project Whirlwind, The History of a Pioneer Computer.* Bedford, Mass., Digital Press.

Vance, P. R., L. G. Dooley, and C. E. Diss. December 1957. Operation of the SAGE duplex computers. *Proc. Eastern Joint Computer Conference*, Washington, D.C. New York, AIEE, pp. 160–172.

Williams, F. C., and T. Kilburn. February 1952. The University of Manchester computing machine. *Review of Electronic Digital Computers*, 57–61. (*Joint AIEE–IRE Computer Conference*, Philadelphia, December 10–12, 1951.)

Production of Large Computer Programs

HERBERT D. BENINGTON

The paper is adapted from a presentation at a symposium on advanced programming methods for digital computers sponsored by the Navy Mathematical Computing Advisory Panel and the Office of Naval Research in June 1956. The author describes the techniques used to produce the programs for the Semi-Automatic Ground Environment (SAGE) system.

*Categories and Subject Descriptors: K.2 [**History of Computing**]—SAGE, software, systems*
General Terms: Design, Management
Additional Key Words and Phrases: Lincoln Laboratory

Editor's Note

When we all began to work on SAGE, we believed our own myths about software—that one can do anything with software on a general-purpose computer; that software is easy to write, test, and maintain; that it is easily replicated, doesn't wear out, and is not subject to transient errors. We had a lot to learn.

As Herb Benington discusses in the following paper, we had already successfully written quite a lot of software for experimental purposes. We were misled by the success we had had with capable engineers writing programs that were small enough for an individual to understand fully. With SAGE, we were faced with programs that were too large for one person to grasp entirely and also with the need to hire and train large numbers of people to become programmers—after all, there were only a handful of trained programmers in the whole world. We were faced with organizing and managing a whole new art.

Author's Address: System Development Corporation, 7929 Westpark Drive, McLean, VA 22101.
Adapted with permission from *Proceedings, Symposium on Advanced Programming Methods for Digital Computers*, Washington, D.C., June 28–29, 1956. ONR Symposium Report ACR-15, Office of Naval Research.
Illustrations courtesy MITRE Corporation.

Bob Wieser (who led the software design and production effort at Lincoln) and his group decided with great wisdom to build the tools needed for such an endeavor instead of trying to do the whole job with the limited resources at hand. We paid a price—the schedule slipped by a year—but the organization that was established really got on top of the job and stayed on top.

Much of what Herb and others created for the SAGE job was forgotten and had to be relearned later by others when they faced similar problems. I confess to having a certain amount of purely human pleasure at watching other organizations suffer through the problems of building large programs—organizations that had been so critical of our own difficulties.

One thing not to forget is the challenge of putting so large and complex a program into a limited computer capacity. The FSQ-7 was the largest machine we felt able to build in the early 1950s; its capacity is trivial by today's standards. One might think that with today's technology, SAGE-like software would be easier to build. Unfortunately, this seems not to be so. There is a kind of Parkinson's Law for software: it is infinitely expandable and swells up to exceed whatever capacity is provided for it.

Foreword

The following paper is a description of the organization and techniques we used at MIT's Lincoln Laboratory in the mid-1950s to produce

programs for the SAGE air-defense system. The paper appeared a year before the announcement of SAGE; no mention was made of the specific application other than to indicate that the program was used in a large control system. The programming effort was very large—eventually, close to half a million computer instructions. About one-quarter of these instructions supported actual operational air-defense missions. The remainder were used to help generate programs, to test systems, to document the entire process, and to support those other managerial and analytic chores so essential to producing a good computer program.

As far as I know, there was no comparable effort under way in the United States at the time, and none was started for several years. Highly complex programs were being written for a variety of mathematical, military, and intelligence applications, but these did not represent the concerted efforts of hundreds of people attempting to produce an integrated program with hundreds of thousands of instructions and highly related functionality. In a letter to me on April 23, 1981, Barry W. Boehm, director of software research and technology at TRW, says of the paper, "I wish I had known of it a couple of years ago when I wrote [a] paper indicating how many of today's software engineering hot topics had already been understood in 1961 in Bill Hosier's IRE article. Your paper predates much of that understanding by another five years."

By chance, the paper was presented in Washington, D.C., in June 1956 at a symposium on advanced programming methods for digital computers, sponsored by the Navy Mathematical Computing Advisory Panel and the Office of Naval Research. The paper was given there because Wes Melahn (soon to become president of System Development Corporation, and now at the MITRE Corporation) was deeply concerned with the programming of an air-defense system, as well as with the theory and mathematics of advanced digital computing at universities. All the other papers at the symposium were presented from the perspective of either universities or the nascent computing industry. The hot topics were machine organization, development of algorithms, and the development of higher-order languages. The common goal was to produce instructions that cost less than $1 per line. The audience was somewhat chilled to hear that we could not do better than $50 per instruction in our particular effort—and that we were talking about tens of thousands of pages of documentation.

I lost interest in the subject until several years ago, when I joined the MITRE Corporation and became interested in what had happened to data processing in the ensuing 20–25 years. I showed the paper to a number of colleagues, some of whom knew nothing of the SAGE development and some of whom had been deeply involved with it. Generally speaking, they were surprised that we had developed or used techniques with SAGE that today are considered essential to the effective production of large computer programs. (We did omit a number of important approaches, which I will say a little more about below.)

It is easy for me to single out the one factor that I think led to our relative success: we were all engineers and had been trained to organize our efforts along engineering lines. We had a need to rationalize the job; to define a system of documentation so that others would know what was being done; to define interfaces and police them carefully; to recognize that things would not work well the first, second, or third time, and therefore that much independent testing was needed in successive phases; to create development tools that would help build products and test tools and to make sure they worked; to keep a record of everything that really went wrong and to see whether it really got fixed; and, most important, to have a chief engineer who was cognizant of these activities and responsible for orchestrating their interplay. In other words, as engineers, anything other than structured programming or a top-down approach would have been foreign to us.

Between the early 1950s and the mid-1960s, thousands of computer programmers participated in the design, testing, installation, or maintenance of SAGE. They learned the system well, and as a result, the chances are reasonably high that on a large data-processing job in the 1970s you would find at least one person who had worked with the SAGE system. The initial SAGE prototype program slipped its initial schedule by about one year. After that, dozens of major modifications were installed at dozens of sites with slips of at most several weeks. The disciplined approach, which had started at MIT's Lincoln Laboratory, persisted for over 15 years at SDC. Why is it, then, that there are so many tales of computer-program projects whose schedule slippages were much greater than SAGE's and whose overruns are often horrendous? There are three major reasons.

First, the industry went through a phase where we decided that computer programming and the computer programmer were "different." They could not work and would not prosper under the rigid climate of engineering management. Just a few years ago, I heard with amazement the executive vice-president of one of our very largest information-system firms say, "Herb, you have to realize the

programmers are different; they have got to get special treatment." I almost ran out to sell his stock short, but then I discovered that his more realistic middle management had realized the failure of this nostalgic view of the computer programmer.

Second, if anything, the pendulum has swung too far in the other direction. Many of our government-procurement documents act as if one produces software in the same way that one manufactures spacecraft or boots. When I got back into the computer programming business several years ago, I read a number of descriptions of top-down programming. The great majority seemed to espouse the following approach: we must write the initial top-down specification (for example, the A Spec), then the next one (typically, the B Spec), so we will know precisely what our objectives are before we produce one line of code. This attitude can be terribly misleading and dangerous. To stretch an analogy slightly, it is like saying that we must specify the characteristics of a rocket engine before measuring the burning properties of liquid hydrogen. Generally, software is the most complex component of a system. Twice as much software can improve the performance of a system by 1 percent or by 500 percent. The percentage can only be determined if a great deal of detailed analysis (including coding) is undertaken to understand the "burning properties" of software. I do not mention it in the attached paper, but we undertook the programming only after we had assembled an experimental prototype of 35,000 instructions of code that performed all of the bare-bone functions of air defense. Twenty people understood in detail the performance of those 35,000 instructions; they knew what each module would do, they understood the interfaces, and they understood the performance requirements. People should be very cautious about writing top-down specs without having this detailed knowledge, so that the decision-maker who has the "requirement" can make the proper trade-offs between performance, cost, and risk.

To underscore this point, the biggest mistake we made in producing the SAGE computer program was that we attempted to make too large a jump from the 35,000 instructions we had operating on the much simpler Whirlwind I computer to the more than 100,000 instructions on the much more powerful IBM SAGE computer. If I had it to do over again, I would have built a framework that would have enabled us to handle 250,000 instructions, but I would have transliterated almost directly only the 35,000 instructions we had in hand on this framework. Then I would have worked to test and evolve a system. I estimate that this evolving approach would have reduced our overall software development costs by 50 percent.

The third reason that we keep seeing missed schedules was pointed out to me by the editor of one of our best computing journals, who says he has concluded that producing large computer programs is like raising a family. You can observe your neighbors and see all of the successes and failures in their children. You can reflect on the experiences you had as one member of a large family. You can observe all the proper maxims of life and society. You can even study at length the experiences of many others who have raised families. In the final analysis, however, you have to start out and do it on your own, learn the unique options you have, see what unexpected problems arise, and, with reasonable luck, perform about as well as those who have been doing it forever.

The latter observation may be reassuring to the new program manager, but there have been numerous significant advances in the techniques for producing large computer programs since we did the SAGE job over 25 years ago. A few that strike me as most important are:

- We now use higher-order languages in virtually all situations.
- Almost all software development and unit testing are done interactively at consoles in a time-sharing mode.
- We have developed a large family of tools that allow us to do much precise design and flow analysis before coding. (I still say that we should use these techniques before we start finalizing our top-down requirements.)
- We have developed organizational approaches that improve or at least guarantee the quality of the systems much earlier in the game. These include some of the structured languages, code reviews, walk-throughs, etc.

For further progress, I would stress the following.

- Since the SAGE effort, we have talked about the need to invest in tools that help produce programs—that is, in tools for coding, editing, testing and debugging, configuration management, consistency checking, structural analysis, etc. I believe too little effort has been spent on thinking through such tools and standardizing them so that they can become analogous to the relatively few higher-order languages that we use with great facility.
- Finally, there remains a tremendous range and ability among computer programmers to do

different jobs. Some are good gem-cutters for any kind of stone. Some can play very special roles—for example, where fastidious approaches are needed. Some are brilliant and articulate conceptualizers and leaders. Some should not be allowed near a computer. We must learn to recognize these types, to use them in their right place, and to set higher standards for not using people even though the market seems insatiable.

—*Herbert D. Benington*

Introduction

At the 1955 Eastern Joint Computer Conference, Jay W. Forrester suggested that the evolution of electronic digital computers might be roughly divided into five-year periods, each period with its paramount significance.

> 1945–1950 was the period of electronic design. From 1950–1955, attention has been focused on the solution of scientific and engineering problems. 1955–1960 will encompass the upswing in the commercial data-processing applications. . . . 1960–1965 will probably mark the shift of major attention to the use of digital computers as the central elements in real-time control systems.

With respect to this last period, Forrester continues:

> General purpose digital computers, as outlined in [recent news] releases, are to be the nerve centers for tying together the flow of information in our forthcoming new air defense system. This type of control system, we can assume, will develop further into a high-speed automatic control and regulation of future civilian air traffic. . . .
>
> [Or,] consider the chemical plants and oil refineries. . . . In the last 30 years the automatic controls in an oil refinery have risen . . . to some 15 percent of the investment in a refinery [or often about] $15,000 worth of automatic controls. I believe we will see digital computers as controllers and monitors of operation in these plants to permit closer control, higher-speed chemical reactions, larger outputs, and a better product.

During the past five years, we have seen developments in automatic programming where the emphasis has paralleled Forrester's first three periods. We can compare the electronic-design phase with the development of basic programming techniques of translation, compilation, and interpretive routines. Scientific and engineering calculations have been assisted by the PACT and A-2 compiling systems, and commercial data processing by BIOR and B-0 (to name but a few). More important, our colleagues who build computers have come to realize that a computer is not useful until it has been programmed, and that programming is an expensive job that requires both machine assistance and human sympathy.

This paper looks ahead at some programming problems that are likely to arise during Forrester's 1960–1965 period of real-time control applications. At first glance, these are problems that will result from the need for very large, very efficient programs, where one program (consisting of over 100,000 machine instructions) may be used in several machines during periods of months or years. On closer inspection, we realize that these are problems that must be faced whenever the need arises for the systematic preparation and operation of large, integrated programs, whether these programs are used for commercial processing, scientific calculation, or program preparation itself.

During the past several years at the Lincoln Laboratory, several system programs containing over 30,000 machine instructions each have been prepared. These programs are used for data processing and control in real-time systems. Production of these programs is briefly described here, particularly in terms of cost and organization. Four problem areas are stressed.

The first problem is *computer operation.* Computer time is at a premium when a large program is being prepared by relatively inexperienced programmers, when the machine and its terminal equipment are being shaken down, and when the machine-program system requires inordinate testing and debugging. The only answer is highly systematic, highly mechanized program preparation and computer operation. A Lincoln Utility System of service routines containing 40,000 instructions has been prepared to ease this problem.

The second problem is *program or system reliability.* Needless to say, a large program is distressingly prone to all types of design and coding errors, including some very subtle ones. In spite of this tendency, it must be extremely reliable if it is to control effectively a system involving extensive equipment or manpower. This is true not only in a real-time system, but also in commercial applications unless equipment engineers can outvote lawyers. Reliability is also a major factor in the preparation of ambitious automatic programming systems—how many unreliable programs have been produced with supposedly well-tested compilers?

Next, there is the problem of *supporting programs.* It has been the experience of the Lincoln Laboratory that a system of service programs equal in size to the main system program must be maintained to support preparation, testing, and maintenance of the latter.

Finally, there is the problem of *documentation.* In the early days of programming, you could call up the programmer if the machine stopped. You seldom mod-

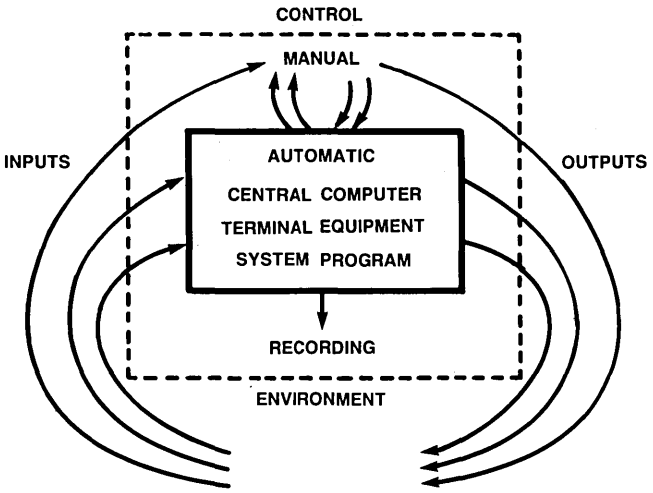


Figure 1. Typical control system. In general, a typical control system uses automatic and manual elements. The automatic portion consists of a centralized digital computer, terminal equipment communicating with the environment, and a computer program incorporating system memory and standard operational procedures.

ified another person's program—you wrote your own. Although present automatic programming technology has done much to make programs more communicable among programmers, there is a long way to go before we can take an integrated program of 100,000 instructions and make it "public property" for the user, the coder, the tester, the evaluator, and the on-site maintenance programmer. The only answer seems to be the documentation of the system on every level from sales brochures for management to instruction listings for maintenance engineers. Such documentation will require the development of new methods and new languages; more significantly, it will require a much more extensive use of the computer to assist in program production, documentation, and maintenance.

At the last ONR symposium on automatic programming held two years ago, the most popular theme was simplifying program input through the use of symbolic inputs, machine compilation and generation, algebraic translation, etc. Very little was said about checkout or debugging, training, or operation. I suspect that for many the past two years have been a period of realizing that automatic programming concepts must go beyond the input process into these other areas.

Large Programs for Control and Processing

Before considering these problems in more detail, consider some rudiments of large systems and large programs. Figure 1 represents a broad flowchart of a typical control and processing system such as might be used for air-traffic control, industrial-plant control, or commercial applications. The area inside the dashed line represents the control system; the area outside is the environment to be controlled. In general, control consists of a manual and an automatic component. Manual in-out data could use voice phones or radios, teletypes, meters, etc. Typical automatic inputs and outputs might be teletype data or high-bandwidth digital data from or to analog-to-digital converters.

The central control is a high-speed, general-purpose, digital machine that includes in-out terminal equipment and is controlled itself by the system program. Depending on the degree of system automation, manual control and processing might range anywhere from one half-awake computer operator (who will be awakened by an alarm) to a staff of several hundred operators and supervisors, each of whom must communicate directly with the computer. The machine can signal the man through indicator lights and alarms, cathode-ray displays, or printed data; the man can respond with digital keyboard inputs or a variety of analog-to-digital devices. Periodically, the computer records data for later analysis of system performance.

From the computer's point of view, then, the system consists of a wide variety of inputs and outputs, each with different data characteristics—peak rate, average rate, reliability, coding, etc. The system program must perform a wide variety of tasks.

1. It must remember the state of environment. Depending on the application, this may require from 100,000 to many billions of bits of information stored on drums, tapes, or photographic plates.

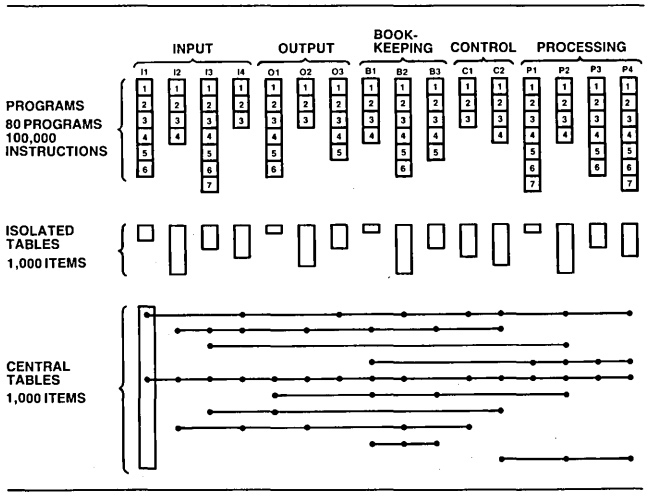


Figure 2. Static program organization. A system program of 100,000 instructions is organized into programming groups for input, output, etc. Each group contains several subprograms and requires both isolated and central tables.

2. It must sample each input either periodically or on demand, translate the data, test for reasonableness (usually in terms of the present state of the environment), and either revise its memory content accordingly or transmit the data for further processing.

3. It must, either periodically or on demand, calculate, monitor, correlate, predict, control, summarize, record, and decide.

4. It must encode and transmit outputs to all terminal devices.

5. Finally, the program must control the frequency and sequence with which it performs each input, output, processing, or bookkeeping task.

In order to give these features some physical meaning, let us attach rough numbers to a typical control problem. Figure 2 shows the organization of a typical 100,000-instruction program that contains 80 component subprograms. In other words, each subfunction requires a logically distinct subprogram containing an average of 1250 instructions. In the figure, each box (e.g., I12) represents a subprogram; they are grouped as follows.

1. There are four major *input* channels (e.g., punched cards, teletype, audio-bandwidth data link, and manual keyboards) designated by program groups I1 to I4. For each channel, several different types or sources of data are received by the control element. For example, I3 requires seven subprograms, I31 to I37.

2. There are four major *processing* functions, which require a total of 24 component subprograms. In an air-traffic-control application, a typical process might be: first, review all aircraft landing at all airports; next, monitor these with respect to airspace assignment and sudden trouble situations; finally, prepare a revised space assignment.

3. A third group of 15 subprograms are required for program *bookkeeping*. These programs coordinate communications between all other programs, monitor ystem load, and prepare summary data for output.

4. The *output* makeup programs use three channels—for example, cathode-ray display, audio-bandwidth data link, and teletype. Fourteen subprograms are required to scan the system memory and make up properly coded output messages.

5. Finally, seven *control* subprograms are required to control the timing, sequencing, and operation of all other subprograms.

The 100,000 instructions represent standing operational procedures for the system; they do not change as the system operates. The system memory, which is stored separately in system tables, can be broken down into two blocks: *isolated tables*, which store information required by one program group only (e.g., I2), and *central tables*, which store data shared by two or more program groups. In measuring the complexity of the table structure, the total table memory required by tables is not nearly so important as the number of items. In this sense, an item is defined as one unique type of information. A single item may be represented once in the tables (e.g., "process I42 is being performed"), or the item may be represented 1 million times (e.g., "customer account number").

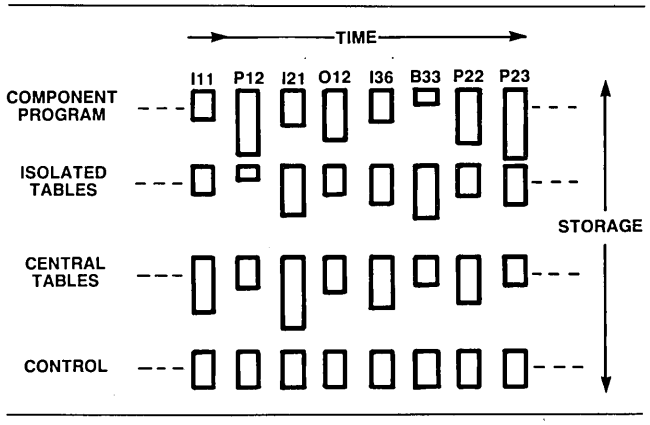


Figure 3. Dynamic program operation. Component subprograms (Figure 2) time-share the control computer. Each component program requires isolated and central tables; a control program, which remains permanently in storage, directs sequence and frequency of operation of component subprograms.

In the example given, 1000 items each are required for the isolated and central tables. For 10 of the central items, the program groups which set or use the item are shown; for example, the first item is used by I1, I4, O3, B2, C1, C2, P2, and P4. If 1000 such lines were drawn, the dot matrix would measure the communications (and complexity) within the program.

Figure 3 shows how the component subprograms time-share the machine to perform the control and processing functions (only a small portion of the complete program sequence is shown). Each component subprogram requires its isolated tables, pertinent portions of the control tables, and certain control subprograms. Eighty programs must time-share the machine. In general, some subprograms will operate unconditionally in a fixed sequence but at different frequencies; other programs will operate on demand.

Large-Program Systems—Centralized versus Decentralized

At this stage, we can consider the effect of program size and integration on the design, testing, and operation of the program. To date, there have been several programming systems of over 50,000 machine instruc-

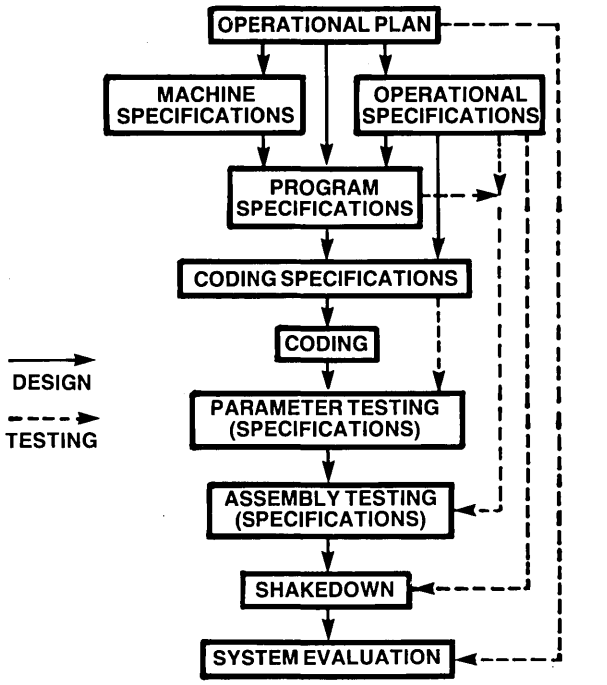


Figure 4. Program production. Production of a large-program system proceeds from a general operational plan through system evaluation; for example, assembly testing verifies operational and program specifications.

tions prepared for business and scientific applications. For the most part, however, these programs have been what might be called large *decentralized* programs; that is, the data-processing function has been divided into a dozen or so parts, and the communication between these parts has used blocks of data stored on magnetic tape or punched cards.

Usually, the format and coding (i.e., the structure) of these blocks can be unequivocally defined with relative ease. This considerably simplifies the design problem; after the blocks have been documented, groups of programmers can be assigned to each part with the assurance that little communication between these programmers will be necessary. If the fullest decentralization is desired, the component programs will not share machine storage or machine time. (In some applications, even different machines are used.)

Control of data processing in a decentralized system is primarily manual. Tape reels and programs are changed by computer operators (and even shipped to remote locations). If an unexpected result develops, an engineer or accountant or supervisor can print out intermediate data and decide after the fact what course should be taken. Efficient use of computer time need not be closely monitored, since there are no real-time constraints.

In testing or debugging one part of the system, data produced by other parts are not required until the very last moment that the system is put into operation. (Probably many of the decentralized systems currently in operation still contain many minor errors which are being compensated for daily by users who have become accustomed to these minor idiosyncrasies.)

The important point is that one can write a large programming system and still maintain a high degree of decentralization. Like most decentralizations, this course produces a system that contains semantic inconsistencies, ambiguities, and errors; operating inefficiencies result from duplication and wasted motion.

Real-time control systems have presented the first computer application where a very large program is required to perform all assigned functions, and yet where the disadvantages of decentralization cannot be tolerated. Success or failure of the system usually depends on efficient use of computer operating time. Internal control of the real-time program must be highly organized if efficient time and storage allocation are to be achieved, if the many in-out devices are to be adequately sampled, and if automatic decisions are to be made when unusual conditions develop within the program or from the external environment.

The control program must be *centralized.* This complicates design and coding since communication between component subprograms must have a high bandwidth. The use of each of the thousands of central table items must be coordinated between 100 or so component subprograms. Organized, readable specifications for the design and coding phase accomplish part of this task. Even then, only the most thorough testing of the entire program ensures that system threads have been carefully worked out, that incompatibilities are discovered, and that all contingencies are accounted for.

Preparation of a System Program

Figure 4 indicates the nine phases used at the Lincoln Laboratory in preparing a large system program. First, an *operational plan* defines broad design requirements for the complete control system consisting of the machine, the operator, and the system program. This plan must be prepared jointly by the computer systems engineers and the eventual user of the system.

From this plan, detailed *operational specifications* are prepared that precisely define the "transfer function" of the control system. In this representation, the computer, its terminal equipment, and the system program are treated as a black box. On the other hand, this description is sufficiently detailed that programmers can later prepare the system program using only

machine and operational specifications. The operational specifications correspond to the equations the scientist gives a programmer; numerical analysis has yet to be performed.

Program specifications outline implementation of the operational black box by the system program. These specifications organize the program into component subprograms and tables, indicate main channels of program intracommunication, and specify time- and storage-sharing of the machine by each subprogram. Continuing the analogy, program specifications correspond to a broad flowchart of the solution.

After the operational and program specifications have been completed, detailed *coding specifications* are prepared that define the transfer function of each component subprogram in terms of the processing of central and isolated items. From these specifications, it is possible to predict precisely the output of the subprogram for any configuration of input items. The coding specifications also describe all storage tables.

Each component subprogram is *coded* using the coding specifications. Ideally, this phase would be a simple mechanical translation; actually, detailed coding uncovers inconsistencies that require revisions in the coding specifications (and occasionally in the operational specifications).

After coding, each component subprogram is *parameter tested* on the machine by itself. This testing phase uses an environment that simulates pertinent portions of the system program. Each test performed during this phase is documented in a set of *test specifications* that detail the environment used and the outputs obtained. In the figure, the dashed line indicates that parameter testing is guided by the coding specifications instead of by the coded program; in other words, a programmer must prove that he satisfied his specifications, not that his program will perform as coded. (Actually, test specifications for one subprogram can be prepared in parallel with the coding.)

As parameter testing of component subprograms is completed, the system program is gradually *assembled and tested* using first simulated inputs and then live data. For each test performed during this period, *assembly test specifications* are prepared that indicate test inputs and recorded outputs. Assembly testing indicates that a system program satisfies the operational and program specifications.

When the completed program has been assembled, it is tested in its operational environment during *shakedown*. At the completion of this phase, the program is ready for operation and evaluation.

Figure 5 indicates reasonable production costs that might be expected in preparing a system program of 100,000 instructions. Considering the present technology of program preparation, our experience does not indicate that these are at all overly pessimistic estimates. The estimates shown do not include training of programmers, preparation of ancillary programs, development of control-systems techniques, or overhead supporting activity. They include only engineering manpower required to produce the system program. Let us assume an overhead factor of 100 percent (for supporting programs, management, etc.), a cost of $15,000 per engineering man-year (including overhead), and a cost of $500 per hour of computer time (this is probably low since a control computer contains considerable terminal equipment). Assuming these factors, the cost of producing a 100,000-instruction system program comes to about $5,500,000 or $55 per machine instruction. *In other words, the time and cost required to prepare a system program are comparable with the time and cost of building the computer itself.*

PHASE	ENGINEERING MANPOWER (MAN-YEARS)	COMPUTER TIME (HR)	PAPER OUTPUT (PG)
Operational Plan	?	0	500
Operational Specs	30	0	2,500
Program Specs	10	0	500
Coding Specs	30	0	5,000
Coding	10	0	3,000
Parameter Testing	20	1,000	2,000
Assembly Testing	30	2,000	1,500
Shakedown	?	?	?
Evaluation	?	?	?
	130	3,000	15,000

Minimum Production Time = 18 Months

Figure 5. Production cost. Using present techniques, the production cost for a 100,000-instruction program can easily require $55 per instruction.

The Lincoln Utility System

In order to simplify the preparation and operation of all programs, the Lincoln Laboratory has prepared a set of service routines called the Lincoln Utility System. This system was designed to assist all programmers in using the machine; its present size—40,000 machine instructions—is indicative of the importance attached to its role. The Lincoln system does not provide automatic-coding facilities in the conventional sense. Compared with systems that have been developed at computing centers where scientific and engineering calculations predominate, the Lincoln system has concentrated more on systematizing computer operation and program debugging than on developing automatic translation of programmer language into machine language. Design of the system followed these ground rules.

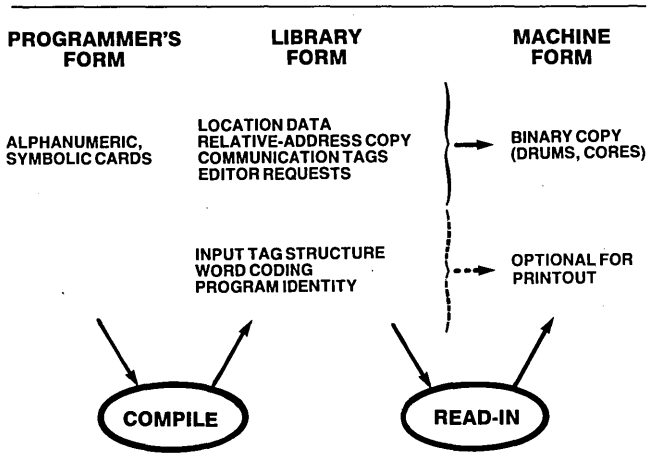


Figure 6. Program input process. With the Lincoln Utility System, compiled programs are stored with the programmer's full input structure; at read-in time, the program is finally converted to machine binary language. Even at this time the symbolic input structure is available to other service routines.

1. At the Lincoln Laboratory, most programs are prepared by relatively inexperienced programmers. As many features as possible were included to help them, yet no features were used that were so complicated that only experienced programmers could use them with facility. Also, programmers do not operate the machine during debugging; they are required to plan and document their operations beforehand.

2. Computer time for parameter testing, assembly testing, and system shakedown is scarce. A large effort has been devoted to systematizing and mechanizing computer operations in order to use minimum computer time.

3. The Lincoln Utility System includes several features that assist programmers in communication and documentation problems encountered during the design and testing phases of system program production.

4. The Lincoln Utility System contains extensive debugging features including facilities for remote, flexible card control of the computer and programs to be tested.

5. Programs are prepared in machine language because automatic coding techniques developed to date do not guarantee the efficient programming required for a real-time system. (In retrospect, this ground rule seems very shaky.)

6. The Lincoln Utility System, which is quite large, has not been so centralized that its initial production was delayed or that its revision and improvement are difficult.

With the Lincoln Utility System, programmers code in floating address using some subroutine requests, particularly for card input and printed outputs. When programs are compiled, they are stored on a magnetic-tape library with their full input structure; that is, the library copy contains program identity, a relative-address binary copy, assigned memory locations, a floating-address tag table, subroutine requests, etc. Storage in this form has several advantages. First, modifications to a program can be expressed in the floating-address input structure; for recompilation, the compiler does not require a complete program copy. Second, all postmortems during and after program operation are retranslated into input language; programmers do not write programs in symbolic form and receive fixed-address outputs. Third, major modifications in storage addresses and locations can be made to a checked-out program at the time the program is read into the machine because system design parameters are stored in a central communication pool (see Figure 6).

In order to debug programs, a "checker" facility is used. This is a service program of 10,000 instructions that allows the program to be tested—the checkee—to be operated either interpretively or noninterpretively under control of a pseudoprogram of executive instructions. When the checkee is operated in the interpretive mode, the checker automatically detects loops, arithmetic alarms, illegal in-out sequences, and illegal instructions. It stores a history of program operation including branches, change-registers, and in-out transfers. In the interpretive mode, the checkee cannot cause a machine halt; when alarm conditions are detected, the checker automatically generates special outputs and moves on to another job. The checker provides a wide variety of outputs including instruction-by-instruction printouts, dynamic change-register printouts, and alarm printouts. Using the executive instructions, a programmer can set machine registers or memory registers to test values; he can start and stop the checkee at selected locations; he can request different outputs for different regions of the program; he can request alarm outputs if the checkee transfers control outside a fixed region or if a loop of more than n cycles is performed; he can indicate the use of different executive subprograms depending on the results of checkee operation; he can indicate which portions of his program are to be performed noninterpretively. From a programmer's point of view, the checker is a special-purpose, checkout computer; it is a stored-program machine with highly flexible input, output, and control sections. (See Figure 7 for a sample executive program.)

All utility programs are controlled by utility control cards. Before a machine run, a deck of binary cards, checker executive cards, etc., is prepared. The operator places the cards in the reader, pushes one button, and the rest of the computer operation is automatic.

A final feature of the utility system is the use of a large communication pool of numerical parameters shared by all programmers. Each programmer can specify that constants or addresses in his program should be taken from the pool. Numbers in this pool are expressed symbolically by the programmer in both his coding specifications and his coded copy; the machine supplies proper numerical values at read-in time. These values may be unknown to the programmer and even changed from day to day. For example, communication tags are used for extracting information (usually table items) that is packed into a full word. The programmer need not know the exact location of the word in memory, nor the position of the information bits within the word. Communication tags are even used to indicate the location in memory of the program itself. A program-design group assigns specific numerical values to the tag pool from day to day, in some cases long after component subprograms have been debugged. Since numerical values are assigned only when the program is read into the machine, it is possible for system designers to move programs and tables within drum and core memory merely by changing constants in this pool. Only one central document needs to be revised, and minimum testing on the computer is required. Figure 8 indicates the allocation of the 40,000 instructions in the utility system.

PROGRAM	LENGTH
Compiler	10,500
Read-in	1,300
Library Merge-Output	4,700
Checker	7,500
Master Tape Load	2,000
In-Out Editors	2,400
Communication Pool	4,100
Utility Control	3,000
Numeric Subroutines	1,000
Miscellaneous	4,000
	40,500

Figure 8. Utility system. The Lincoln Utility System requires over 40,000 instructions as indicated.

Testing

It is debatable whether a program of 100,000 instructions can ever be thoroughly tested—that is, whether the program can be shown to satisfy its specifications under all operating conditions. Considering the size and complexity of a system program, it is certain that the program will never be subjected to all possible input conditions during its lifetime. For this reason, one must accept the fact that testing will be sampling only.

On the other hand, many sad experiences have shown that the program-testing effort is seldom adequate. When the program is delivered for operation, its performance must be highly reliable because the control system is a critical part of a much larger environment of men and machines. One error per 100,000 operations of the entire program can easily be intolerable.

As a result of facing this problem for some time at the Lincoln Laboratory, the following principles have evolved to govern our testing.

First, parameter testing (i.e., testing of individual component subprograms in a simulated environment) cannot be too thorough. This phase must discover all errors internal to the program and its individual coding specifications. Even if parameter testing were perfect (which it never is!), many errors in system design would remain to be discovered during subsequent assembly testing.

Second, initial assembly testing should be performed using completely simulated inputs. There are several reasons. First, only in this way can all test inputs be carefully controlled and all tests be reproducible. Second, when errors are discovered with a new program using live inputs, there will always be a question whether the program or the machine is at fault. Integration of the system program with terminal equipment should not be attempted until the assembled program has been well tested.

A third principle is that the testing facility used during the assembly test phase must contain extensive, flexible facilities for recording both system

```
CHECKER      CARDS / DELAYED
01   NI   11A   11R
02   AL   07
03   LP   25
04   LR   12   13
05   TR   12   13
06   BG   12A   13Z+6
07   LP   4
08   LR   14   15   16
09   BG   14A   16L+5
10   CC
11   QT
```

Figure 7. Sample executive program. The Lincoln checker is controlled by pseudoinstructions. The executive program shown indicates regions of the checkee to be performed noninterpretively (01 NI), alternate executive instructions in case of checkee alarm (02 AL), maximum-length loops (03 LP), legal regions of checkee operation (04 LR), checkee output mode (05 TR), etc.

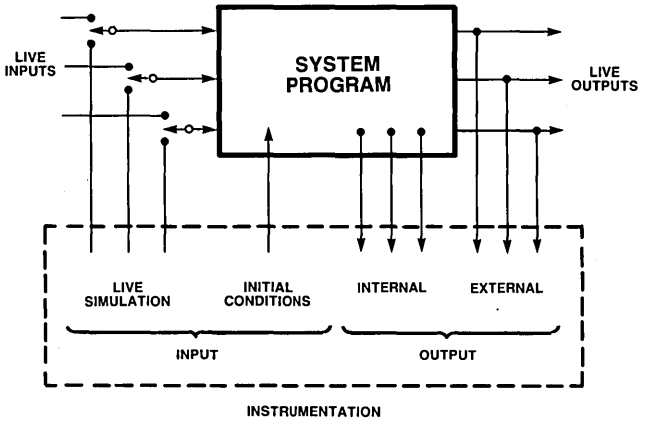


Figure 9. Test instrumentation. Proper testing of a control system requires an automatic facility for simulating inputs and monitoring outputs. With this facility, extensive testing can be performed and outputs produced for either diagnosis of system errors or verification of proper system performance.

outputs and intermediate outputs (i.e., subprogram intercommunications). Without this facility, rapid and reliable diagnosis of system errors is impossible. After a test has been conducted and errors found, it should be possible to correct the error before the program is put on the machine again.

The need for comprehensive simulated inputs and recorded outputs can be satisfied only if the basic design of the system program includes an instrumentation facility. In the same way that marginal-checking equipment has become an integral part of some large computers, test instrumentation should be considered a permanent facility in a large program.

Figure 9 illustrates the role of test instrumentation in a system program. Each of the live inputs can be individually simulated; this allows simultaneous testing with both live and simulated data. In addition, the input instrumentation allows easy setting of initial conditions for system memory; this feature is performed by a special-purpose translation program that converts alphanumeric card data into system tables.

System Program	100,000 Instructions
Utility Programs	40,000
Special Programs	10,000
Test Instrumentation	20,000
Operational Instrumentation	30,000
	200,000 Instructions

Figure 10. Production of a system program. Supporting programs whose total size equals the system program may be required to simplify production and testing of the system program.

The output instrumentation "probes" both internal data (for diagnosis) and external data (for simpler verification).

One final principle should govern system-program testing: *All successful parameter and assembly tests must be reproducible throughout the life of the system program.* These tests must be documented in test specifications that detail the reasons for the tests, required inputs, operating procedures, and expected outputs.

The original reason for this requirement stemmed from the problem of revising the program once it was operational. The slightest modification to a program can be successful under limited testing conditions and yet still cause critical errors for other operations. Since it is desirable to retest the program thoroughly after each modification, a large battery of test inputs must be available. We have discovered two other incidental advantages of detailed test documentation. First, a programmer's tests tend to be more organized and more exhaustive if he must document them. Second, if machine-versus-program reliability is ever questioned, retesting is possible. If a known program and a known test fail, the machine is at fault.

Supporting Programs

The utility and test-instrumentation programs discussed are only part of the complete set of supporting programs. In addition, special programs, which assist preparation of the system program, are used to generate routine data blocks, perform special translation of alphanumeric data into parameter tables, assemble program-sequence and timing parameters, etc.

Operational instrumentation programs are used during system shakedown and evaluation. They contain simulation and recording facilities that are far more realistic and operationally oriented than the test instrumentation. System recorded data are analyzed with a battery of data-reduction programs (Figure 10).

Documentation—Design and Revision

As indicated earlier, documentation of the system program is an immense, expensive job. The output will run to tens of thousands of pages of specifications, charts, and listings. At the Lincoln Laboratory, these currently include the following.

Operational specifications
Program specifications
Coding specifications
Detailed flowcharts
Coded program listings
Parameter test specifications

Assembly test specifications
System operating manuals
Program operating manuals

The need for this battery of documents is obvious. The system and its program must be learned and used by management, operational-design engineers, system-operating personnel, training personnel, program-design engineers, programmers, program-test engineers, evaluation personnel, and, if more than one system is maintained, on-site maintenance programmers. Each of these users has very different needs.

Consider the problem of revising the system once the program is operational in the field. A minor change in the operational specifications is proposed. First, the cost and effects of this change must be evaluated in terms of the program, the operators, and, often, the machine. In order to make the change, several hundred revisions may be required in the specifications. If the change is approved, these documents must be changed, operating manuals revised, and the program modified and thoroughly tested. The wave of changes must be coordinated smoothly.

Digital computers are often sold to management on the basis of their programmed flexibility. We have said, "If your doctrine or procedure changes, no messy, expensive, time-consuming equipment changes will be required." In reality, this is not true today. The cost of the documentation mentioned is only a symptom of the design-coordination problem in large systems.

How can we reduce this cost? Obviously, as we have done already, by more extensive use of the computer. (At the laboratory, we have partially gone in this direction through the use of punched cards for storing all central design data. Decks are easily revised, fed into the system program, or listed for the user.) We must systematize design, production, and documentation both in the small and in the large. By "in the small," I mean what is already being done in automatic programming. Instead of an algebraic translator, we need a unified "bookkeeping–logical-processing–algebraic translator." Before we get this, we will surely need much more research on coding languages and representations. Eventually, programming should become a two-way conversation between the imprecise human language and the precise, if unimaginative, machine. The programmer will say, "Do this," and the machine will answer, "OK, but what happens if . . . ?" The smallest gain of such a system would be the elimination of the coding, parameter testing, and parameter test-specification phases. Unfortunately, these phases represent only one quarter of the system cost.

Documentation "in the large" poses a bigger challenge.

1. What integrated set of documents are required to design and describe a large system?
2. What language should these documents use?
3. How should they be cross-referenced?
4. Can we eventually store them on magnetic tape and let the computer analyze, print, and code?

Summary

The techniques that have been developed for automatic programming over the past five years have mostly aimed at simplifying the part of programming that, at first glance, seems toughest—program input, or conversion from programmer language to machine code. As a result of progress in this area (and a growing number of experienced programmers), we find that large programs can now be produced; unfortunately, they are difficult to test and document. If the newest very-high-speed, large-memory computers are to be fully utilized, we must develop automatic programming procedures so that they allow cheap production of highly reliable, easily revised, well-documented system programs.

The Cape Cod System

C. ROBERT WIESER

The Cape Cod System was an advanced-development prototype for the SAGE system. The paper discusses the evolution of the Cape Cod System, which used the Whirlwind computer, from its inception in 1951 through first operation in September 1953.

*Categories and Subject Descriptors: K.2 [**History of Computing**]—hardware, SAGE, software, systems*
General Terms: Design, Management
Additional Key Words and Phrases: defense, U.S. Air Force, Cape Cod System, Whirlwind

Editor's Note

One of the important characteristics of the SAGE development was the close integration of the design with experimental tests of concepts, the discovery and correction of unforeseen difficulties in the real world, and the verification of design details. Thanks to the high priority of air defense and the complete support of the Air Force, we were able to build and operate an evolving experimental air-defense system with which to do our work. This evolving system, as Bob Wieser describes, began with a single radar and eventually became a full-scale air-defense system covering the New England area, with numerous radars, assigned air-defense interceptors, and regularly scheduled raids by SAC bombers. I believe that SAGE could not have been successfully built in such a short time without these extensive experimental facilities.

The Cape Cod System also acted as a demonstration to persuade decision makers in the Air Force, associated contractors, and the community at large that SAGE was actually doable. Today, in a world full of computers, it may be difficult to believe there was a lot of skepticism about whether SAGE could be made to work. In fact, believers were a small minority to start with, and Cape Cod played a major role in persuading important people, especially senior Air Force officials, that MIT was on the right track. Once they believed, they provided the solid backing without which SAGE would not have been possible.

In the early days, Whirlwind and its growing ensemble of peripheral equipment was only marginally reliable, yet we ran air-defense exercises every week. Whirlwind had a distinct personality—it could be fractious and difficult but it seemed to sense when something important was going on and would react appropriately. Many times I saw Whirlwind flat on its back an hour before demonstration time only to come up and run solidly when needed, then collapse again and be down for several days. For some reason that none of us, including George, could understand, Whirlwind particularly liked George Valley. Whirlwind would almost invariably run while he was in the building only to go back to sulking when he left.

Introduction

The Cape Cod System was an important and necessary step in upgrading post-World War II continental air defense. The upgrading was extensive and required improvement of just about all the functions and elements of the system, which was aptly characterized in the first report of the Valley Committee (January 1950) as "lame, purblind, and idiot-like."[1]

Author's Address: Physical Dynamics, Inc., 13 Corporate Plaza, Suite 100, Newport Beach, CA 92660.
Figures 2–5, 7–9 courtesy MITRE Archives. Figures 1 and 6 courtesy MIT Lincoln Laboratory.
© 1983 AFIPS 0164-1239/83/040362-369$01.00/00

[1] "Air Defense System," Report of the Air Defense Systems Engineering Committee, October 24, 1950, pp. 9–10.

Figure 1. Gap-filler radar.

In addition to problems of equipment reliability and maintainability, the 1950 air-defense ground environment had a number of conceptual difficulties that had to be corrected in order to achieve the high attrition rates needed to cope with an emerging nuclear threat. The Battle of Britain was won with about 5 percent attrition of attacking bombers, a number hopelessly low for defense against nuclear attack.

Briefly, the principal conceptual problems were gaps in low-altitude radar coverage, severe problems in "hand-over" (handing aircraft tracks and control of interceptors from one radar to another by voice telephone), and the inability of the system, which was entirely manual, to carry out detection, tracking, identification, and interception for more than a few targets in the coverage of any one radar. To make matters even more difficult, these problems were synergistic. The only known cure for lack of low-altitude radar cover was "gap-filler" radars (Figure 1) to cover the low-altitude "holes" between the existing long-range radars. Because low-altitude radar coverage is inherently limited to a few tens of miles, a low-altitude defense requires more frequent hand-over of aircraft tracks. The advent of faster jet aircraft required even more frequent hand-over, and further taxed the manual-control system operators by reducing the time available to intercept an attacking bomber.

Planning

The Air Defense Systems Engineering Committee, usually called the Valley Committee, envisioned a new air-defense ground environment with a number of innovations to correct the conceptual limitations of the old system. Unmanned gap-filler radars would be added to supplement low-altitude coverage, and both gap-filler and long-range radar data would be automatically encoded and transmitted over telephone lines to direction centers. Thus the netting of radars took care of the hand-over problems by providing the direction center with the composite overlapping coverage of many radars. The direction centers would also be linked by automatic digital data transmission. Because this concept included automatic detection of aircraft, it contributed to eliminating saturation in the direction center.

The direction center itself was to be semiautomatic; that is, routine tasks would be done automatically under the supervision of operators. A high-speed digital computer would collect target reports from the radar network, transform them into a common coordinate system, perform automatic track-while-scan (tracking based on periodic radar reports at 10- to 12-second intervals), and compute interceptor trajectories. Operators filtered the radar data, had override control (i.e., could initiate or drop tracks), performed friend-or-foe identification function, assigned interceptors to targets, and monitored engagements through voice communication with the interceptor pilots.

For each of these innovations there was a prior invention, but all inventions were at a very early stage of development in 1950. In operation were breadboard hardware for automatic radar target detection, digital encoding (with beam splitting), and transmission over voice-bandwidth telephone lines. This equipment, the Digital Radar Relay (DRR), was installed and con-

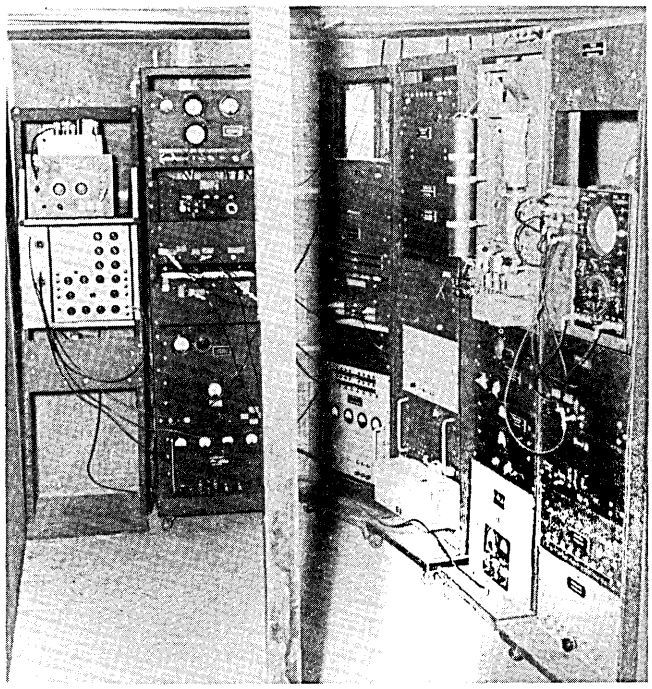


Figure 2. Digital radar relay (DRR) connected to the MEW radar.

nected to a World War II Microwave Early Warning (MEW) radar at Hanscom Field (Figure 2). Commercial telephone lines were available. The forerunner of the SAGE digital computer, Whirlwind I, had been designed, prototypes of its repetitive basic circuits and devices had been built and tested, and the computer was under construction with its central memory (electrostatic storage) yet to be integrated.

System analysis and engineering was at a very early stage. A small group was studying the application of Whirlwind to semiautomatic air-traffic control and had started work on automatic digital track-while-scan and aircraft flight-path control. This group became the nucleus of the Cape Cod design group.

Nothing in the new air-defense ground-environment concept challenged the laws of physics; however, an extensive, urgent engineering job had to be done if the concept was to be accepted. The MEW and breadboard DRR were unreliable, telephone lines were noisy, and the electrostatic storage for Whirlwind was yet to be integrated. The smattering of software in existence had been coded in machine language for scientific calculation, and some thought had been given to steering aircraft away from collision courses instead of onto them.

Given these circumstances, development of the new concept and its embodiment in the Cape Cod System relied heavily on iterative cycles of experiment–learn–improve. Since the major elements and subelements were being developed simultaneously, system engineering and integration was a parallel empirical process of exploiting what was available. Understandably, there were skeptics who viewed the concept, which was radically new and without even an analogous precedent, as more foolhardy than creative. This lent further impetus to a development approach that used what was at hand to conduct realistic experiments and used the experimental results immediately to steer further development of hardware and software. However inelegant, the approach worked very well. The ever-present realism of radar clutter, telephone-line noise, and limited computer memory drove the development pace faster than a mathematical analytical approach could ever have done.

Early Experimentation

The development and test process was evolutionary, but fell into three phases based on availability of hardware. The first phase consisted of the Project

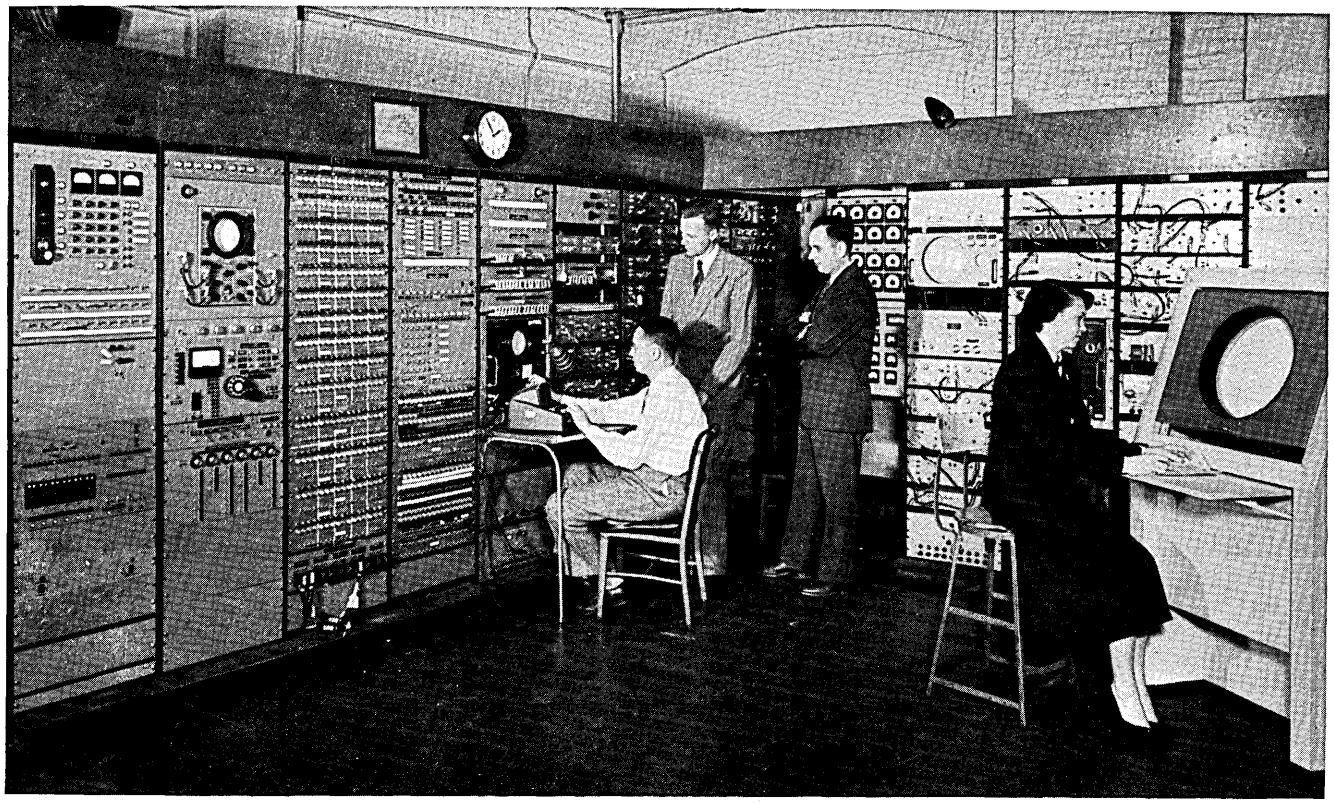


Figure 3. Whirlwind I test control. *Left to right:* Stephen J. Dodd, Jay W. Forrester, Robert R. Everett, Ramona Ferenz.

Charles demonstrations, which provided elementary proof-of-principle evidence to support continued development. Next, there was construction and test of the 1953 Cape Cod System, which demonstrated feasibility of the SAGE system. The third phase was expansion of the Cape Cod System to provide additional design data for SAGE.

The first step in the Charles demonstrations was to show that radar data, encoded and transmitted from a remote site, could be inserted into a digital computer. Phone lines were leased, and the MEW radar data were transmitted to the Barta Building in Cambridge, Mass., where Whirlwind I was under construction (Figure 3). The internal memory was not yet available, and the only usable storage consisted of five flip-flop test-storage registers. Using one register as an input buffer and another as an output register (for display), the MEW data were first inserted and displayed in September 1950, a few months prior to the formal commencement of the Project Charles investigations. However trivial an experiment this might seem in retrospect, much was learned. The reliability of the old MEW radar was poor, as was the reliability of the first breadboard DRR. Telephone noise was frequently formidable, especially when cross-talk from dialing flooded the computer with false "targets." Usable data were available less than half the time, a condition that had to be improved. As a result, phone-line noise was to a large extent cleaned up, and the MEW and DRR were shut down for a month for repair and upgrading.

In the meantime, construction of Whirlwind continued, paced by the reliability of the first bank of electrostatic storage tubes (Figure 4). The memory density was thinned to 16 × 16 (256 registers) to improve operation, and a small amount of "application time" was allocated to computer users. The applications group (later the Cape Cod design group) coded the computer to perform automatic, real-time track-while-scan of up to 10 aircraft detected by the MEW radar. The program also provided a cathode-ray-tube plan-position indicator (PPI) polar display of the radar reports and the tracks, which were initiated manually from the display tube by means of a photoelectric light gun. Thus another direction-center function was demonstrated, although on a limited scale, and with no better than a 50–50 chance of reliable operation even when "tweaked up" for a test.

The next evolutionary step, which took place early in the era of Project Charles, was to program Whirlwind to compute collision-course vectoring instructions for an interceptor aircraft automatically. Because of limited storage, the track capacity had to be reduced from ten to two aircraft (the target and interceptor). The interceptor vectoring headings were displayed by means of indicator lights on a flip-flop register and translated mentally from binary to decimal by the radio telephone "talker" for voice transmission to the interceptor pilot.

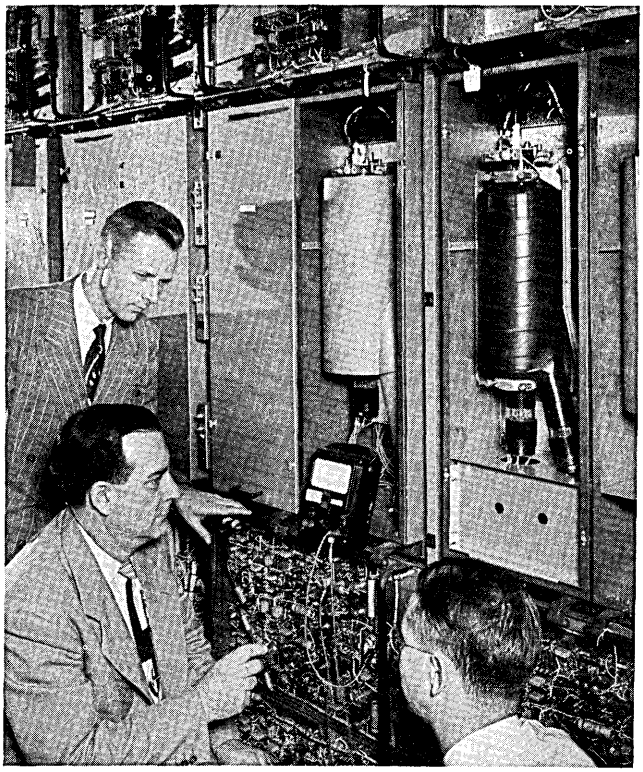


Figure 4. Whirlwind I electrostatic memory. *From top left:* Jay Forrester, Pat Youtz, and Stephen Dodd.

Having coded and tested the system, there was a strong urge to try it out quickly. The local Hanscom Air National Guard Unit was persuaded (very informally) to join the experiment as part of maintaining flight proficiency. Two officers agreed to try it out. One flew the C-45 target (a small twin-engine Beechcraft of World War II vintage), and the other flew a T-6 single-engine propeller-driven trainer. These aircraft flew at about one-quarter the speed of jets, but they were available and jets were not. On April 20, 1951, less than a year after the birth of the Valley Committee's air-defense concept, automatic computation of interceptor instructions was demonstrated live, three times, in the skies of New England. Miss distances were less than 1000 yards, quite adequate for hand-over to airborne intercept radar.

Construction

Although the air-defense functions were demonstrated on a limited scale with unreliable developmental

equipment, the system worked as expected in a realistic environment and provided a basis for proceeding with construction of the Cape Cod System. The decision to proceed came just three days after the interceptions were conducted. The timetable, which was driven by military urgency, was ambitious—the Cape Cod System was to be built and put in operation by 1953, a development time of 2½ years from limited proof-of-principle tests to demonstration of feasibility. Moreover, the "1953 Cape Cod System," as it was called, was a big step from the MEW–DRR–Whirlwind system with limited storage to a functionally complete experimental system so that all the air-defense functions could be performed experimentally, developed further, and demonstrated with acceptable reliability.

Radar netting was required. Earlier experiments showed that some form of radar data filtering was needed to remove the residual radar clutter that was not canceled by the moving target indicator (MTI). Phone-line noise had to be held within acceptable limits. Whirlwind I had to have a larger, more reliable random-access internal memory supplemented by an external memory. Buffer storage had to be added to handle the insertion of data from an asynchronous radar network. The software had to be expanded considerably. Finally, a direction center had to be designed and constructed to permit air force officers and enlisted men to operate the system—that is, to control the radar data filtering, initiate and monitor tracks, identify aircraft, and assign and monitor interceptors (Figure 5). The Cape Cod System was, in essence, a model of the SAGE system, scaled down in size but realistically embodying all the SAGE air-defense functions.

A long-range FPS-3 radar, the workhorse of the operational air-defense net, was installed at South Truro, Mass., near the tip of Cape Cod (Figure 6). It was equipped with an improved DRR. Two gap fillers, designed by Lincoln Laboratory, were installed at Scituate and Rockport, Mass., and were equipped with the new gap-filler detection and data-transmission system called *slowed-down video* (SDV). Each radar included a Mk-X IFF (identification friend or foe) system with its reports multiplexed with the radar data. Dedicated telephone circuits to the Barta Building in Cambridge were leased and tested.

At the Barta Building, work on Whirlwind continued, and the reliability of electrostatic storage was constantly improved. It was clear that the rapidly advancing research on magnetic-core memory would in all likelihood lead to a much superior memory. Development of a 1024-register core memory was initiated. A buffer drum built by Engineering Research Associates (now a part of Sperry Univac) in Minneapolis was added to Whirlwind.

A parallel effort to develop a "radar mapper" to filter data at the direction center was also initiated (Figure 7). The radar MTI of the early 1950s was all analog and provided limited subclutter visibility, es-

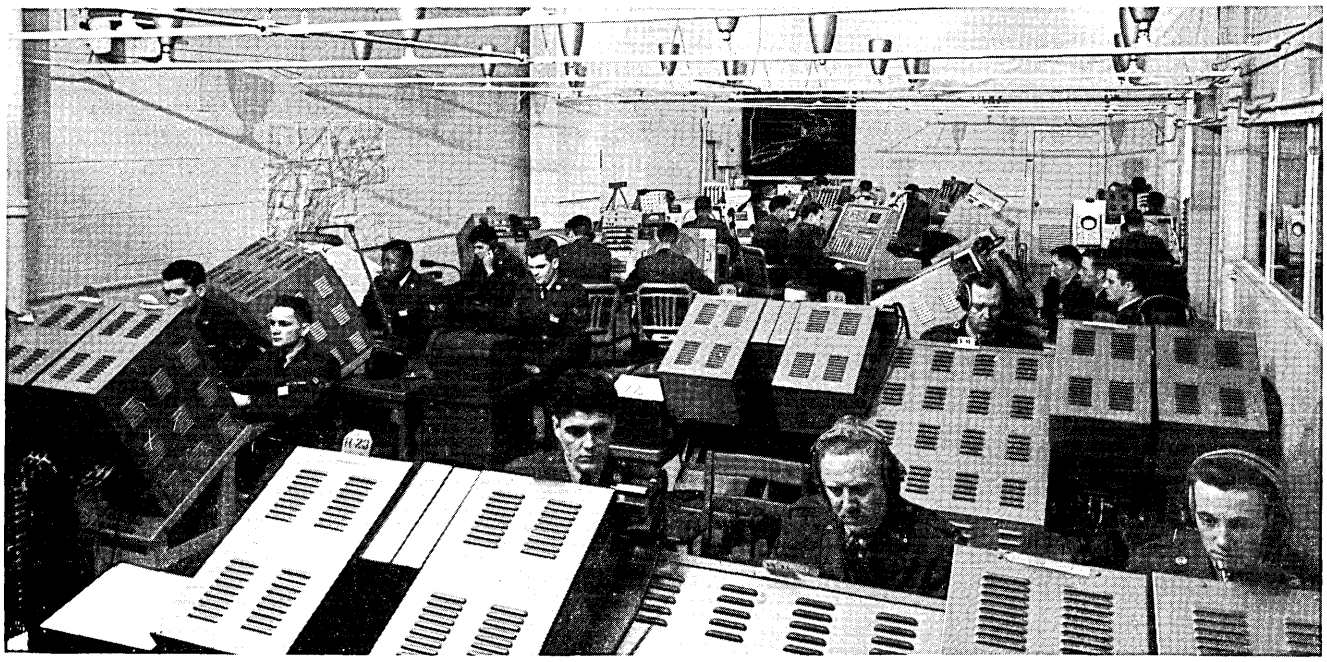


Figure 5. Cape Cod direction center (air force operators in foreground are intercept monitors).

Figure 6. FPS-3 radar at South Truro.

pecially at short ranges. Since targets could not be detected in very dense clutter, it was wasteful of computer capacity to insert dense clutter into the computer. A simple, ingenious solution was devised. It consisted of a polar PPI display of the incoming data for each radar. The cathode-ray-tube (CRT) face was horizontal, and a single photocell was mounted above it. The photocell response to the bright blue initial flash from displayed position reports controlled a gate that passed the data into the computer. Consequently, any area of the tube face that was masked (opaque to blue light) resulted in rejection of the radar data. The mask material was a paint that could be applied or removed manually and transmitted the afterglow on the tube face so that data under the mask were visible to the operator (but not to the photocell). Changes in clutter patterns were relatively slow, since they were caused by changes in weather (anomalous propagation and echoes from severe storms). Another key problem had been solved.

Construction of a realistic direction center depended heavily on the development of a versatile display console that provided the operators the information they needed to make decisions and also provided them the means to send commands to the computer. (The modern term *interactive* applies but was not in common use in the early 1950s.)

Nothing like this had ever been done before, and the technology then at hand was primitive by today's standards. What was wanted was a computer-generated PPI display that would include alphanumeric characters (for labels on aircraft tracks) and a separate electronic tote-board status display. The console operator had to be able to select display categories of information (for example, all hostile aircraft tracks) without being distracted by all of the information available.

The Cape Cod display console was developed around the Stromberg-Carlson Charactron CRT. The tube contained an alphanumeric mask in the path of the electron beam. The beam was deflected to pass through the desired character on the mask, refocused, and deflected a second time to the desired location on the tube face—electronically complex, but it worked.

The console operator had a keyboard on which he could compose a command to the computer—for example, identify a particular aircraft track as friendly. He could use either the track number (inserted via his keyboard) to designate the aircraft track or his light gun, which sensed the track-display flash and called for the computer to read the operator's keyboard command.

A great deal of engineering went into the display system electronics and the logic of what to display and what operator actions were required at each operator position. There was also the question of console layout for ease of operation. With a characteristic direct

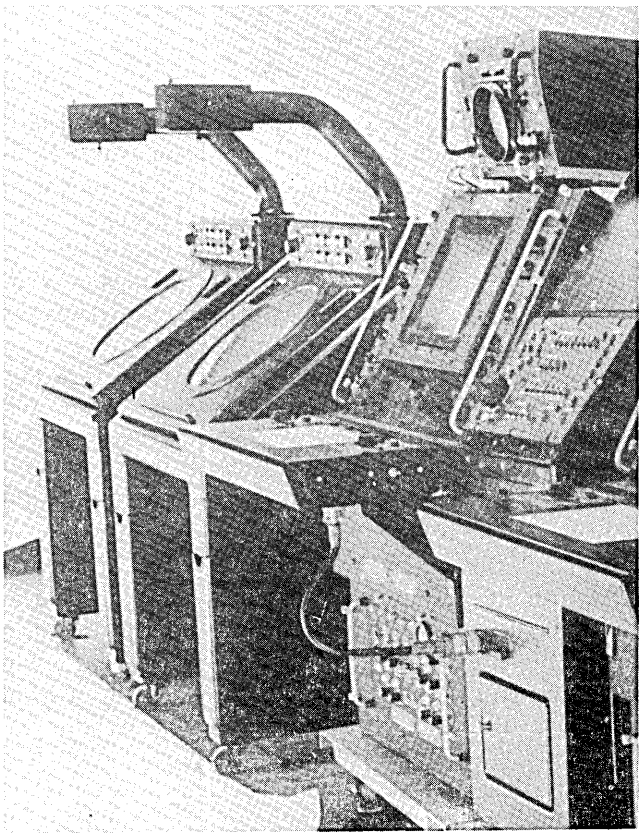


Figure 7. Two radar mappers (far left).

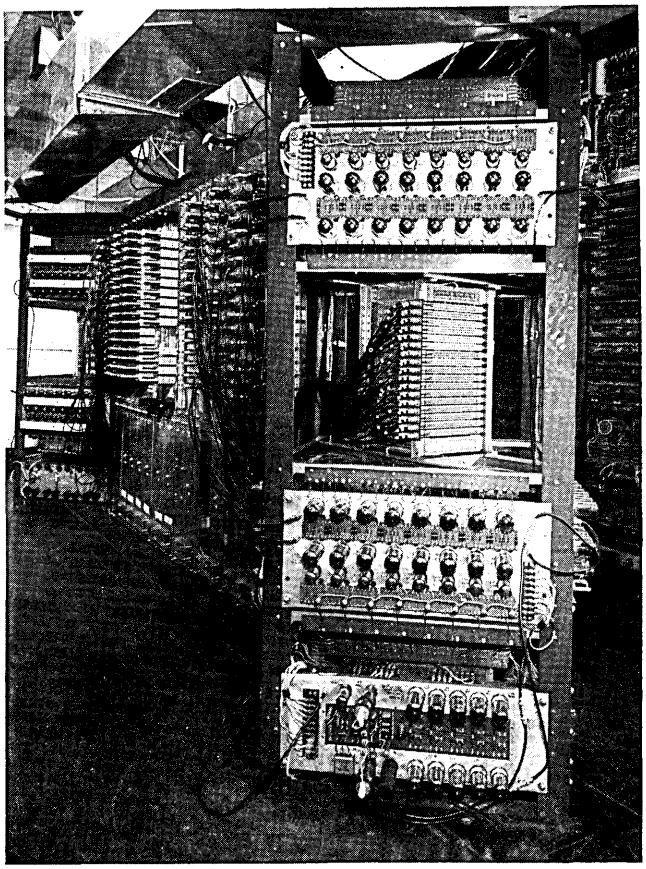


Figure 8. Whirlwind I core-memory banks.

approach, one of the engineers ran a simple experiment. He suspended a Charactron display-tube shipping carton from the ceiling of his office by means of four strings and drew the outline of the tube surface on one end of the carton. Visitors were seated before the mock-up and asked to adjust the strings to what they thought was the best height and tilt angle of the mocked-up tube face. Height of the front and rear of the carton were then measured and recorded. The measurements from many trials were averaged, and the geometry of the Cape Cod displays was determined.

Like all parts of the system, the first experimental console was built in a hurry. After the electronics were developed, the next console component sought was a custom-built metal cabinet to house the display. Delivery time was quoted as several months, so once again ingenuity came to the rescue. A household cabinetmaking shop was across the parking lot from the Barta Building. In a few days the shop built the first console cabinet of plywood—at considerably less than the cost of a metal cabinet. There was only one problem: the cabinetmaker delivered the cabinet with an attractive blond mahogany finish, which was feared to be indicative of unseemly luxury. After a brief period of admiration, the console was painted gray.

In parallel with the hardware improvements, there was major software development to be done. The integration of the external storage drum was a software problem as well as a hardware problem. The scarcity of internal memory capacity required that much of the software be stored on the drum and transferred into the central computer when needed. The radar network data, also stored on a drum, had to be read into the computer and transformed into a common coordinate system for proper registration. The operator consoles had to be integrated via software that generated a variety of displays and received and executed the operator's control commands. Since automatic ground-to-air data links were not yet in use, interceptor vectoring orders were still displayed (now in decimal) on a small "tote tube" and relayed to the pilot by voice radio.

The software task was to program quickly the largest real-time control program ever coded, and to do all the coding in machine language, since higher-order languages had not yet been developed. Furthermore, the code had to be assembled, checked out, and realistically tested on a one-of-a-kind computer that was a testbed shared for software development, hardware development, demonstrations for visiting officials (who came from near and far to monitor progress), and training the first crew of air force operators.

Operation and Expansion

All of these complex engineering tasks were carried out in parallel, on a schedule, and with remarkably little rework. By September 1953, just two years and

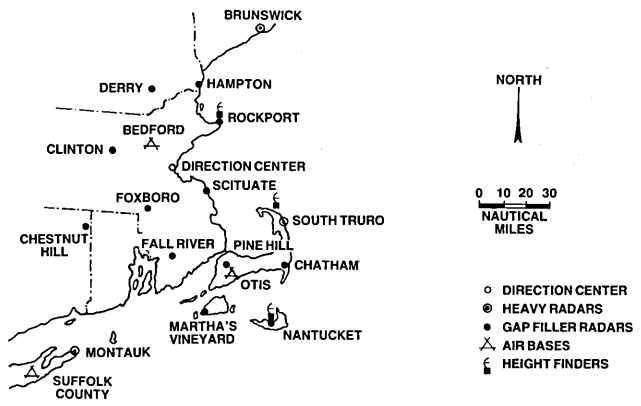


Figure 9. Map of the Cape Cod expanded radar network.

five months after "go," the Cape Cod System was fully operational, including the integration of two 1024-register random-access core-memory banks in Whirlwind (Figure 8). Reliability was excellent by the standards of vacuum-tube electronics. It was an engineering feat that in itself speaks for the imagination, enthusiasm, and skill of the people involved—probably a quarter of the number who would be assigned to a project of similar complexity today. Moreover, the management style created an environment in which rapid progress was possible; although strict in setting goals and priorities and in allocating scarce resources (skilled people and computer time), management did not engulf the workers in bureaucracy. The workers were young, bright, enthusiastic, and very much aware that they were working on the leading edge of something new and important; they were learning on-the-job skills that schools did not teach. The hours were long, the camaraderie was close, and everyone wanted to make it work.

Having demonstrated the feasibility of the radically new air-defense concept, the designers and operators of the Cape Cod System continued to expand and operate the system, principally to collect operating data for specifying the new operational air-defense system. Toward the end of 1953, the blueprint for a deployable system, the Lincoln Transition System, was published. In the summer of 1954, the system was designated the SAGE system by the Air Force.

During this period, the Cape Cod System radar network was expanded to include two more long-range FPS-3 radars located at Brunswick, Me., and Montauk Point on the eastern tip of Long Island. (Integration of the Montauk radar revealed some of the previous problems with radar registration. Air force records included three locations for the radar, two of which were in the Atlantic Ocean!) Additional gap fillers were built and integrated, bringing the entire network to a total of 14 radars by the summer of 1954 (Figure 9).

There were continuing efforts to expand and improve the system software, to reflect air force operator experience in the choice of operator displays, and to make operations more realistic. All-weather jet interceptors were assigned to support the experiments: 12 U.S. Air Force F-89Cs at Hanscom Field and a group of Navy F-3Ds at South Weymouth. Later, an operational Air Defense Command squadron of F-86Ds, based at the Suffolk County Airfield on Long Island, was integrated into the Cape Cod System, and the Air Force arranged for diversion of Strategic Air Command training flights into the Cape Cod area, so that the Cape Cod System could be used to run large-scale air-defense exercises against Strategic Air Command B-47 jet bombers.

In 1954 automatic ground-to-air data links were in development as an aid to conducting large-scale air battles and as a necessity for the forthcoming BOMARC pilotless interceptor, which would later be integrated with the SAGE system. The Cape Cod System was again extended to run an experiment. The MIT Instrumentation Laboratory (now the Charles Stark Draper Laboratory) was engaged in autopilot research and had at its flight facility a B-26 aircraft equipped with an autopilot that could be commanded by input signals from a digital data link. After agreement to a joint experiment, the ground end of the data link was connected to the Whirlwind computer, and software modifications were incorporated to transmit interceptor vectoring commands automatically over the data link and into the B-26 autopilot.

Without delay, an experimental live interception was arranged. After checking by radio with the interceptor pilot that the system seemed to be working properly, he was given the request ("let George do it") to switch to autopilot control. The interception went as planned, the pilot soon sighted the target aircraft ("tally-ho"), and let the autopilot complete a successful interception. Another important first had been accomplished. The Cape Cod System engineers, with the irreverent enthusiasm of youth, dubbed the experiment "the immaculate interception."

Conclusion

Cape Cod experiments continued at the urgent pace necessary to gather data in time to support the design of the SAGE system—hardware, software, and operating doctrine. The operation was outstandingly successful in meeting its commitments. In a more general sense, demonstration of the Cape Cod System was a much larger accomplishment because it was the initial step toward a sweeping change—a change of kind—in automation. The system was the first large-scale, real-time control system that combined remote sensing and complex control operations, all controlled by a central digital computer and supervised by human operators.

Systems of this generic type were perceived to have many civil and military applications. Digital processing technology was advancing rapidly toward cheaper, better components. The people who engineered the system and the imaginative air force officers who funded and supported it were aware of this, which is probably why conceiving, building, and operating the Cape Cod System was a source of deep satisfaction.

Radar Data Transmission

JOHN V. HARRINGTON

The paper reviews the development of radar data transmission and its role in the SAGE system. The work leading to the design of the FST-1 and FST-2 radar data-compression systems is described.

*Categories and Subject Descriptors: K.2 [**History of Computing**]—hardware, SAGE, software, systems*
General Terms: Design, Management
Additional Key Words and Phrases: radar, U.S. Air Force, FST-1, FST-2

Editor's Note

Just as important to the concept of SAGE as the digital computer was the technology for transmitting radar data over telephone lines. Fortunately, while MIT was developing a digital computer for control applications, a group under Jack Harrington at the Air Force Cambridge Research Center was developing techniques for digital signal processing and digital transmission over telephone lines. George Valley seized on these two pioneering efforts and saw that they made a centralized computer-based air-defense system possible.

We had many troubles with radar data transmission, most of which could not be foreseen without trying the equipment out in the real world. Sending digits over telephone lines sounds easy, and it is, but sending them reliably was not. The telephone system had been elegantly designed for sending analog voice, but suffered a number of distortions and noise interferences that only digits could notice—and notice them they did.

At first the telephone company was dubious about what we were doing. When the first telephone line for radar data came into the Whirlwind building to be wired into one of Jack's modems, the telephone installer insisted on wiring it into a handset. We told him we didn't want the handset, but he said it was regulations and that was that. When he left, we connected it to the modem. I don't know what happened to the handset. Later the telephone company became interested in digital transmission and designed and built the modems for SAGE.

SAGE cost a lot of money, so much that it was kind of unreal to an ordinary person. One of the few times I got some sort of feeling for what we were really up to was driving from South Truro on Cape Cod, where we were installing a radar station, back to Boston. Every so often along the road was a big wooden drum full of telephone cable that was to be installed to bring the information back to Whirlwind. Mile after mile, drum after drum—and this was just one station for the experimental system.

I also remember a visit from another organization that was working on another approach to air defense. They went away in shock when they discovered that our telephone bill was larger than their entire budget.

Introduction

The spectacular success of microwave radar in air defense during World War II, particularly for early warning and fighter direction, came largely from applying radar to point defense. There were relatively simple interconnections between points in the form of voice cross-telling. In the postwar years, the speed and range of foreseeable air actions increased, requiring a family of radars to provide adequate coverage of the field of action. The remoting or automatic relaying of radar pictures and data over relatively long distances

Author's Address: Communications Satellite Corporation, 950 L'Enfant Plaza, Washington, DC 20024.
Illustrations courtesy MITRE Corporation.
© 1983 AFIPS 0164-1239/83/040370-374$01.00/00

to some regional operation center became a development objective of great operational importance.

History

The beginnings of automatic radar data networking go back to the postwar (1946–1950) programs of the Relay Systems Laboratory in the Air Force Cambridge Research Center and the contributions of E. W. Sampson, E. B. Staples, H. Feistel, E. W. Bivans, T. F. Rogers, and myself. Our efforts centered on completing the microwave relay system initiated at the end of World War II at the MIT Radiation Laboratory, where so many of the significant steps in the development of microwave radar were taken.

Two efforts in the Relay Systems Laboratory had significant impacts on radar networking in their separate ways. The first effort concerned the development of a microwave relay system (Ames et al. 1948) that would transmit a radar video signal with the necessary range and azimuth synchronization, so that the radar plan-position indicator (PPI) picture could be reconstructed and displayed at the receiving end. The immediate application was the transmission of signals from the MEW (Microwave Early Warning) radar at Hanscom Field in Bedford, Mass., to our laboratory in Cambridge, some 20 miles and two microwave hops away. The system worked out very well. The several megahertz of bandwidth required to transmit the unprocessed video, however, and the high initial cost and maintenance of microwave relay stations at that time, led to a good deal of thinking about more efficient ways to accomplish the same objective. Those considerations evolved into the second major effort, a concept called digital radar relay (DRR). The basic idea (Ames et al. 1952) followed from the realization that the information contained in the radar picture was contained within the coordinates (range and azimuth) of relatively few radar targets. If these targets could be detected and their location transmitted in the new binary digital form, a substantial reduction in bandwidth would result, allowing the wideband microwave circuit to be replaced by a narrowband telephone line. The argument was advanced that telephone lines were abundant, available everywhere within the United States, and much less expensive than microwave transmission—therefore they should be utilized. It is ironic that three decades later the bulk of the long-line telephone traffic in this country is actually carried by microwave relay!

Development Problems

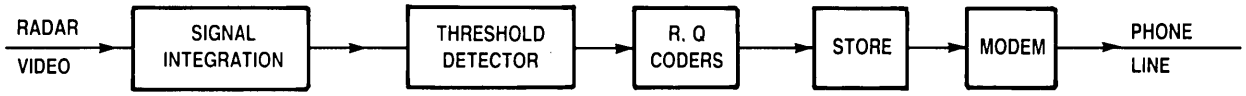
While the concept of DRR was simple and appealing, its implementation posed a number of formidable technical challenges. First, there was a need to detect radar reflections automatically from an airborne target. In a practical sense, the target was often immersed in a fairly high level of radar noise, ground clutter, or other unwanted returns. The selection process had to be very great in DRR; otherwise, the limited transmission capacity could easily be overloaded. The detection had to take into account that the target return occurred over many radar pulses—that is, that there were a large number of radar hits per beamwidth. Some form of signal integration (Harrington 1950*a*) was essential if efficient detection was to be achieved.

The principle of signal-to-noise improvement through the integration of a repetitive signal in noise was well recognized, but the high-capacity electronic storage necessary to accomplish the video addition in real time was lacking. Initially, delay lines were used in what today would be called a "comb filter arrangement"; however, these were restricted to one repetition rate and for large numbers of additions displayed marginal stability. Our group concentrated on a new storage tube developed by the RCA Laboratories at Princeton called the barrier-grid storage tube. We had some good initial success with it for video integration, and later for digital storage as well.

Another area of concentration in the development of the DRR technology was in the encoding of the target range and azimuth coordinates. The simplest technique, and the one we adopted, was to count either range or azimuth marks in a simple array counter, with the counter being reset at range and azimuth zero, and to read those out at the precise time the integrated radar signal exceeded a preset threshold. Other techniques were investigated as well, including an interesting optical disk from which azimuth could be read directly. A voltage-encoding tube (Harrington et al. 1951), adapted from a television monoscope with a special target pattern in cyclic binary code, was also developed and had multiple high-speed encoding applications.

One of the most difficult requirements in the implementation of the digital radar scheme was the provision of enough high-speed storage to store the (R, θ) code groups when they were generated—and to store them for a variable time until the slow-speed transmission channel was clear to take them. A number of choices were available, but none were really attractive; in the late 1940s the magnetic and/or integrated circuit technology that makes digital storage so cheap and plentiful today was not yet invented.

A 16-bit coordinate word had to be stored in a few microseconds, depending on the radar-range resolution desired; hence, fairly high storage speed was required. A random-access store seemed the most suit-



Digital radar relay system.

able for the nonuniform rate at which the targets occurred and for the slower but more uniform rate of read-out for transmission. The new storage-tube technology (Bivans and Harrington 1950) seemed to be the most promising for both this application and signal integration. A better understanding of the electrostatic storage mechanism (Harrington 1950*b*) in the tube led to the successful use in the late 1940s of the barrier-grid storage tube for these applications.

In the early equipment, transmission of the target coordinates over a telephone channel was accomplished by modulating a family of nine tones in the 500–2500 Hz band at about a 50–100 Hz rate to transmit eight bits plus a marker bit in parallel. This was relatively inefficient and wasteful of bandwidth; however, it easily handled many of the idiosyncrasies of the telephone lines, particularly the effects of delay distortion and the frequency changes introduced by single-sideband carrier systems.

Early Experiments

We demonstrated successful operation of the new DRR system over a phone line from the MEW radar site in Bedford to the laboratory in Cambridge sometime in 1949. It was a significant achievement that contributed directly to the work of ADSEC (Air Defense Systems Engineering Committee) and the air-defense-system concept that committee was evolving. We now had achieved automatic detection of radar targets, with sufficient sensitivity and a low enough false-alarm rate that we approached within a decibel or so the detection sensitivity of a human operator. Further, we had encoded the target coordinates, stored them, and transmitted them at low rate over a phone line for display or other processing at a remote point.

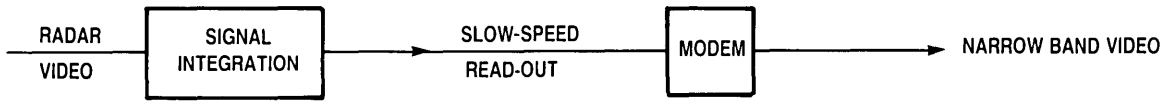
Our 1949 DRR demonstration coincided with the formation of ADSEC, chaired by MIT's George E. Valley, Jr., a radar expert and distinguished alumnus of the famed Radiation Laboratory. The Valley Committee, in examining U.S. defenses against low-flying enemy bombers, soon recognized that the only way to fill the gaps created by the earth's curvature under the long-range radar beams was to employ a much larger family or network of shorter-range "gap-filler" radars. These gap-filler radars could be remoted to a central point and would provide low-altitude coverage over a wide region. The early concept envisaged the use of small CW (continuous-wave) radars sensitive only to moving targets. They were to be mounted on telephone poles and remoted via phone lines to an operations center. The great advantage of position-determining pulse radars soon prevailed. The general importance of narrowband communications for radar data led to our group's involvement in the work of ADSEC. It was an exciting time; Valley had gathered around him a number of experienced and creative people who were rethinking the nation's entire air-defense concept. A sense of great importance and urgency was attached to their work.

One of the early steps taken by ADSEC was to recognize the need for high-speed, real-time data processing for target tracking and interception at the central point. ADSEC sponsored some early experiments in which the MEW radar at Bedford, fitted with our DRR equipment, would transmit radar-target data in real time to the Whirlwind computer in Cambridge, where digital track-while-scan operations would be carried out. These first experiments in centralized radar tracking were truly pioneering and unquestionably influenced much of the subsequent work on automatic air-defense systems.

Our group at the Air Force Cambridge Research Center joined the SAGE effort as Group 24, Data Transmission, of MIT's new Lincoln Laboratory, established in 1950 to develop a more effective air-defense system. Valley became the head of our division and eventually the associate director of the laboratory, while Jay W. Forrester and his Whirlwind team became Division 6 of the new laboratory. Thus much of the appropriate data-processing experience was in place in the laboratory for a major developmental effort on an automatic air-defense problem.

FST-1 Development

From the early ADSEC experiments, the subsequent Cape Cod System experience, and the later SAGE system came a great many ideas and a good deal of progress in the development of automatic means to accomplish the detection and transmission of radar data. Three general schemes were employed. The first of these, the Digital Radar Relay (Ames et al. 1952), primarily used in the ADSEC-sponsored MEW-Whirlwind I experiments, was followed early in the Lincoln Laboratory work by a so-called slowed-down



Slowed-down video system.

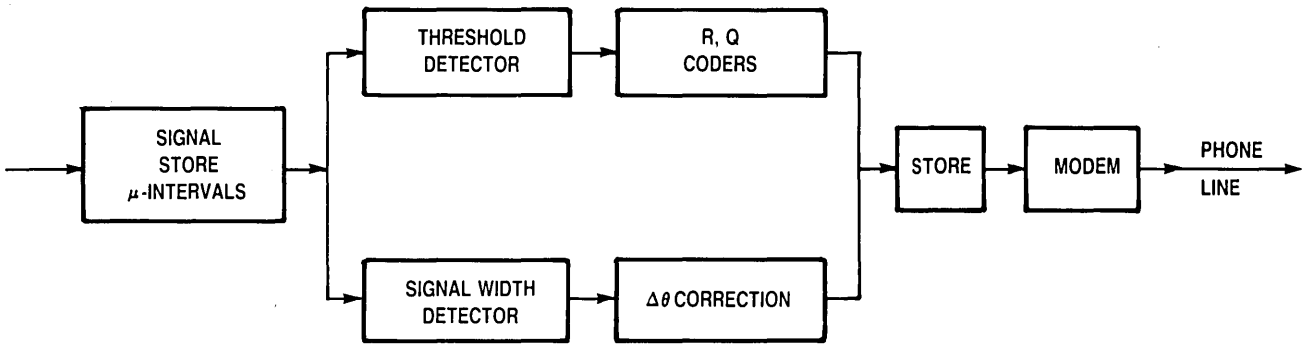
video (SDV) system (Harrington 1954), a much simpler method than the DRR scheme. It recognized that when radar signals were integrated over the repetition intervals in one radar beamwidth and subsequently read out over a longer period of time, a relatively narrowband-view signal resulted that could be directly transmitted over a telephone line. The addition of fairly simple azimuth synchronization allowed the entire picture to be reproduced essentially in real time at the remote point. SDV was cheap and effective and was built in several different forms, depending on the size (range) of the picture and the type of storage. Its big disadvantage was that it faithfully relayed all returns in the radar picture. Its accuracy was inherently poor: one beamwidth in azimuth (1 degree) and one interval (1 mile) in range. The coarse granularity was in fact the basis on which narrowbanding was accomplished, and it yielded a surprisingly useful and accurate picture from which elementary aircraft tracking could be carried out. Our group developed two principal SDV designs: one employed flip-flop storage and was used on the gap-filler radars in the Cape Cod network; the second was a storage-tube SDV system designed for the large heavy radars, and subsequently produced by the Lewyt Corporation. The production version was called FST-1.

FST-2 Development

The difficulties of trying to achieve accurate aircraft tracks at the central point from relatively coarse SDV data led to the development of the so-called fine-grain data (FGD) system (later produced as the FST-2). The FGD scheme was, in fact, a variation of the original DRR idea, but with a much more elegant detector that could identify the center of the target and code its coordinates more accurately. It required that a relatively large number of radar repetition intervals be stored such that the signals in any one range interval could be examined over the full beamwidth. Various detector schemes were used to detect the target (Harrington 1955) and its center, such as run-length detectors and sequential observers (Dineen and Reed 1956), but the simplest and most effective seemed to be the simple Neyman–Pearson observer applied to the beginning and end of the target run, with the distance between these two events determining the azimuth correction.

The form of storage used in the first FGD, after some initial attempts at using a storage-tube system, was a multiple-track magnetic drum whose rotation rate could be synchronized with radar repetition rate. This worked well and produced considerable improvement in the basic accuracy of the transmitted radar data and the tracks determined from those data. The magnetic-drum system was the prototype for the FST-2 radar data transmitting equipment, later produced by the Burroughs Corporation for the SAGE system (Ogletree et al. 1957).

Another important area addressed in the early days of the Lincoln effort was the development of modems for the transmission of radar data over land lines. The complexities of the telephone plant and the effect that various kinds of carrier equipment and delay distortion had on some of our signals was a bit of a shock at first. A scheme to transmit binary data at 1300 bits



Fine-grain data system.

per second with appropriate synchronization was soon developed, and it worked well on all of the lines we encountered (Harrington et al. 1954). (In the early 1950s, 1300 bits per second over an untreated voice line was an achievement.)

Little has been said here about the radar itself. Radar was largely a development of the previous decade, although much work continued on moving-target detection, which was essential for the proper filtering of radar data into the SAGE system. A great many radar designs were investigated that attempted in different ways to detect the Doppler envelope of the coherent sampled radar returns from any one moving target. None of these schemes, at least prior to the mid-1950s, satisfactorily provided all of the coverage, scan time, and accuracy desired, while still giving good Doppler detection. The result was a need to employ so-called video mapping to map out portions of the original radar picture where known clutter existed. If such equipment was not employed, serious overloading of both the narrowband data network and the central tracking computer could result. The fact that moving-target-indicator schemes almost worked well enough was tantalizing, but hindsight says that the initial selection of the right radar returns from the enormous population of radar returns in the average picture needed a great deal more work. We did have target filtering and selection schemes that depended not so much on the instantaneous Doppler of the target as on its motion over a longer period of many radar scans. The amounts of storage required to recognize track patterns right at the radar were considerably beyond the bounds of possibility in those days and were pretty much ruled out. Today the unbelievably low cost of storage would argue otherwise.

Summary

In general, the early years of the SAGE system development were most productive and rewarding. Considerable progress in sophisticated radar-detection methods, accurate encoding, the storage and integration of data by a variety of means, and the development of workable modems for narrowband transceivers in the radar data-transmission business was accomplished.

REFERENCES

Ames, L. A., E. W. Bivans, J. V. Harrington, and T. F. Rogers. June 1948. "The AN/CPS-1 Radar Relay System." Cambridge Research Laboratories Report No. E2003.

Ames, L. A., E. W. Bivans, J. A. Dumanian, J. V. Harrington, T. F. Rogers, and R. V. Wood, Jr. May 1952. "The Digital Radar Relay System—D.R.R., An Experiment in Data Transmission." Cambridge Research Center Report No. E5088.

Bivans, E. W., and J. V. Harrington. March 1950. "An Electronic Storage System." Paper presented at National IRE Convention, New York.

Dineen, G. P., and I. S. Reed. March 1956. An analysis of signal detection and location by digital methods. *IRE Transactions on Information Theory IT-2,* 1, 29–38.

Harrington, J. V. October 1950*a*. Signal-to-noise improvement through integration in a storage tube. *Proceedings IRE 38,* 10, 1197–1203.

Harrington, J. V. October 1950*b*. Storage of small signals on dielectric surface. *Journal Applied Physics 21,* 10, 1048–1053.

Harrington, J. V., G. R. Spencer, and K. N. Wulfsberg. March 1951. "A Five-Digit Parallel Coder Tube." Paper presented at National IRE Convention, New York.

Harrington, J. V. March 1954. "Storage-Tube Slowed-Down Video System." Lincoln Laboratory Technical Report No. 61.

Harrington, J. V. March 1955. An analysis of the detection of repeated signals in noise by binary integration. *IRE Transactions on Information Theory IT-1,* 1.

Harrington, J. V., P. Rosen, and D. A. Spaeth. April 1954. Some results on the transmission of pulses over telephone lines. *Proc. Symposium on Information Networks,* Polytechnic Institute of Brooklyn.

Ogletree, W. A., H. W. Taylor, E. W. Veitch, and J. Wylen. December 1957. AN/FST-2 radar processing equipment for SAGE. *Proc. Eastern Joint Computer Conference,* Washington, D.C. 156–160.

A Perspective on SAGE: Discussion

HENRY S. TROPP, MODERATOR
HERBERT D. BENINGTON
ROBERT BRIGHT
ROBERT P. CRAGO
ROBERT R. EVERETT
JAY W. FORRESTER
JOHN V. HARRINGTON
JOHN F. JACOBS
ALBERT R. SHIELY
NORMAN H. TAYLOR
C. ROBERT WIESER

On October 26, 1982, several people who had participated in the design and development of the SAGE system gathered at the MITRE Corporation to discuss their work and its ramifications. Henry S. Tropp was moderator. Kent C. Redmond and Thomas M. Smith, authors of "Project Whirlwind" (Digital Press, 1980), who are writing a book on the SAGE project, were also present.

Editor's Note

At Bernie Galler's suggestion, we arranged a discussion among some of the SAGE participants. I was dubious at first because the number of major contributors to SAGE is so large that I did not see how we could have a real discussion among so many, and yet I did not feel competent to leave any out. After a while, however, we were able to attract a group of people who not only contributed themselves but who represent groups of major contributors.

We were fortunate that Jay Forrester could come from MIT, Herb Benington represented SDC, Bob Bright had been at AT&T, Bob Crago at IBM, Jack Harrington at AFCRC and Lincoln Division 2, Jack Jacobs at Lincoln Division 6, Major General Al Shiely (retired) at the Air Force, Norman Taylor at Lincoln Division 6, and Bob Wieser at Lincoln Division 6.

Hank Tropp led the discussion, which was vigorous and lasted all afternoon, through dinner, and into the evening. One characteristic of old SAGE hands is that they love to talk.

I regret that more SAGE people did not have a chance to have their say. I am especially sorry that George Valley was unable to join us. We thought of all of you, however, and hope there will be other opportunities.

Henry S. Tropp: We're gathered here today to talk about an important air-defense system that has been in place for two decades: SAGE (Semi-Automatic Ground Environment). I want to thank Bob Everett for letting us use the facilities of the MITRE Corporation for our meeting.

Robert R. Everett: I'd like to say one thing right away. I had hoped George Valley would be here, but at the last moment he was not able to come. He asked me to express his regrets at not being here, and to say hello to all of you.

Jay W. Forrester: We could start our discussion by noting that we are here as a consequence of a series of small happenstance events. We should trace this enterprise back to Gordon S. Brown, who in the early part of the 1940s was director of the Servomechanisms

Moderator's Address: Department of Mathematics, Humboldt State University, Arcata, CA 95521.
Photographs by Lou Nocca, MITRE Corporation.
Categories and Subject Descriptors: K.2 [**History of Computing**] —*hardware, people, SAGE, software, systems.* General Terms: Design, Management. Additional Key Words and Phrases: defense, Lincoln Laboratory, U.S. Air Force, IBM Corporation, MITRE Corporation, System Development Corporation, AN/FSQ-7.

Laboratory at MIT. At least three of us—myself, Wieser, and Everett—grew up in that environment, which was a powerful learning experience. A great deal of responsibility was given to very young people, allowing them a chance to carry ideas from their conception and research right into the military operating field so that they understood how their mistakes would eventually come back to their own doorsteps. That experience lies behind many of the ideas about reliability that allowed the SAGE system to succeed.

At the end of World War II, I had begun to think about leaving MIT and perhaps starting a company to develop servomechanisms. Gordon Brown called me in one day and suggested that there might be more interesting possibilities. He gave me a list of about 12 projects to choose from. One was an aircraft stability and control analyzer, which represented the beginning of a sequence of events that eventually led to SAGE.

Initially the aircraft analyzer was to be an analog computer. Another happenstance directed our attention to digital computers. Perry Crawford, who now works for IBM and at that time was in the Special Devices Center of the U.S. Navy, was standing with me on the front steps of MIT at 77 Massachusetts Avenue late one afternoon. He called my attention for the first time to digital computation, to the mechanical Harvard Mark I computer, to the electronic ENIAC computer. He suggested that we move in the direction of digital computation to get out of the difficulties that analog computation was presenting. It was also Crawford who pushed the whole idea of combat information and control with digital computers, well before any high-speed, general-purpose, reliable computer had ever functioned. Working with Perry Crawford, Bob Everett and I in 1947 wrote a paper[1] on how a digital computer could be used to coordinate the activities of a naval task force—the submarines under the surface, the ships on the surface, and the aircraft overhead.

We were thus prepared to take advantage of another one of those small happenstance incidents when George Valley and his committee for the U.S. Air Force were seeking a way of coordinating radar information. Jerome G. Wiesner, then directing the MIT Research Laboratory for Electronics (and later president of MIT), suggested that Valley should talk to us about the coordination of military information. We had by that time a year of thinking about using digital computers for coordinating military combat information.

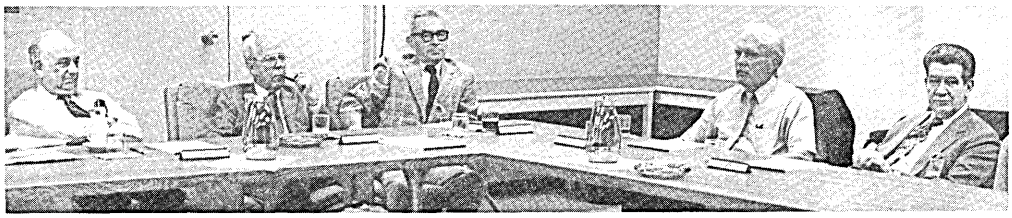
After that meeting with Valley, the program developed rapidly through Project Charles and the Lincoln Laboratory to the SAGE system that we are here to discuss.

Tropp: Your mention of happenstances saves me my next question. I want to go back in time and try to plant ourselves in 1949 or 1950. I can't recall any stored-program computer other than the EDSAC that was up and running. One side of the BINAC was operating. Whirlwind was close, and SEAC may have been close. Valley came to you with a proposal that was obviously going to require a computer that not only didn't exist, but no one had even thought about what it was going to be like. I'd like to throw out an open question to anybody who wants to respond: how did you view the problem when each of you were introduced to this incredible challenge of producing an air-defense system in a technology that by current electronic standards was limited and in an environment that was relatively unknown?

Norman H. Taylor: I didn't think it was that vague at all. We had a computer that was technically very sound and that worked very well, except for the storage-tube memory. We were confident that we could solve that problem somehow. We were transmitting radar data to the computer, and we were tracking aircraft. We had working displays, working tape units; magnetic drums were available. We were writing and using what we thought then were sizable computer programs. We knew we needed a much larger computer, but we thought we knew how to build one. There were engineering problems galore but no fundamental problems that we knew of.

John V. Harrington: I think somebody should point out that in 1949 at the time of the deliberations of the ADSEC committee—the Valley Committee—the

[1] J. W. Forrester and R. R. Everett, "Information System of Interconnected Digital Computers." Project Whirlwind, Servomechanisms Laboratory, MIT, Cambridge. Limited Distribution Memorandum L-2, October 15, 1947.



Left to right: Benington, Harrington, Crago, Taylor, Forrester (shown twice), Tropp, Everett (shown twice), Bright, Shiely, Jacobs, Wieser.

problem was really broader than just what would be used at the central point to do all the tracking and correlation. That, to be sure, was the most important and most difficult component of the whole system. Close in importance was the kind of radar network that would be necessary to provide the low-altitude coverage, which was needed to deal with what I remember as being the most serious threat: the low-altitude long-range Soviet bomber.

I remember going through an awful lot of different radar configurations. I know we had the telephone pole radar—the continuous-wave (CW) radar (which had a short range and not much discrimination, but it could measure velocity extraordinarily well). There were so many of them that we were faced with an enormous transmission problem, and data-handling problem, and central correlation problem.

Taylor: And you had the ghost problem, didn't you?

Harrington: And we had ghosts. [*Editor's Note:* Ghosts are false targets created when correlating range-only, azimuth-only, or velocity-only radar measurements.] Harry Nyquist of Bell Laboratories was first to recognize the ghost problem in the CW radar, which turned out to be its death knell.

We looked at bi-static radars. Then we looked at heavy radars. We looked at combinations of heavy radars and small radars; each of these had somewhat different data-transmission and data-handling problems. We should not forget that these radar studies were an awfully important part of the work of the Valley Committee and of the early work of the Lincoln Lab, too.

Taylor: It was also the problem that we never quite solved.

Harrington: No, we never really quite solved it. As remarkable a machine as Whirlwind I was—and for all its successes—it could easily be flooded by all of the data just from one radar. The CW radar, which was by far the best filter to detect moving targets, really never worked out. And the MTI (moving-target indicator) radar was more of a name than anything else. My apologies to those people who worked on the MTI radars, but they never were completely reliable, or completely credible. Overall, however, in the radar aspect, the communications aspect, the data-handling aspect, the command aspect, and so on, it was really a remarkably ambitious system. When you think about it happening 30 years ago, it was really remarkable.

Tropp: That's what amazed me. Maybe the question that I'm really trying to get at is, at what point did the real magnitude of what you were trying to do really sink in? Look at some of the early estimates of 8K of memory, 8K of core, a few thousand lines of code. Obviously each of you had a different reaction to the magnitude of problems involved.

Forrester: Well, it sank in slowly. Each step in realizing the magnitude was by a percentage that could be coped with one part at a time.

Herbert D. Benington: I remember two anecdotes where my problems helped SAGE to progress. First, I was having lunch with my boss, Jack Arnow. I was telling him that Whirlwind reliability was so bad, that the computer programs were so complex, that we were making very little progress in checking out the system (and having to work too many hours). Within a day or so, Jay called a staff meeting and said that we would replace the storage-tube memory by transferring the core memory from the Memory Test Computer to Whirlwind. That's when we started getting 99 percent reliability out of Whirlwind and we could check the programs out.

The second anecdote was when we had the XD-1 (the prototype of the AN/FSQ-7 computer) operating and had 8000 words of core. I started realizing then that we couldn't get the job done because there would have to be so much paging in and out from drums that we'd spend too much of our available time doing that. I was also having lunch with my boss that day, and I told him my conclusions. Jay dropped by at lunch and said, "Well, we've been developing a 65,000-word core memory, so we'll put it in." That eightfold increase made the program possible.

Everett: I think all these things are right, but several other things were important. First of all, we didn't make a design and send one bunch of people off to build the computer—another bunch of people off to do this and that—and put it all together several years later only to find out that it was wrong. We took it step by step. We were actually looking at real radar

Henry S. Tropp
"Maybe the question that I'm really trying to get at is, at what point did the real magnitude of what you were trying to do really sink in?"

data, and tracking real aircraft, long before system designs were all complete.

The second thing is that the technology was improving rapidly, and it seemed to stay about even with our recognition of the size of the problem.

The third comment I might make is that we didn't sit down and say, "We need a machine of such and such size, and if we can't make it we give up." What we did say was, "We think we can make a machine of such and such size, and given that machine, we could do the following things." As the machine got better, the job got bigger, and we were able to handle it. Even if the machine had been half as capacious, we still would have done something, although it would not have been quite the same thing.

I make these remarks because very often in today's military-development world, people try to do everything and end up doing nothing.

Taylor: You play the end game before you start.

Tropp: That's an important aspect of SAGE that has probably not really been told in the papers I've seen for the *Annals.*

Everett: I was struck at the time and have been struck since by how much a group of really smart, dedicated people with adequate resources can do toward solving problems. You put them to work on a problem, and you get a solution if you don't have too many problems, and if you don't box them in with too many restrictions. SAGE had a lot of problems, but, as Jay said, we fortunately didn't realize them all at once. As each problem came up, we were able eventually to do enough about it to get the system going. In my experience, you can do almost anything if there are only one or at most two real difficulties. But when you start to work on something that's loaded with a whole slew of very difficult problems, you can lose control and really get into trouble.

Taylor: I remember one incident along those lines when the radars were giving us a lot of trouble. I was in a study group with Al Hill in Washington, and I said, "Al, we shouldn't have ever gotten into this thing. We can't get decent data, and how can you track airplanes with no good data?" Al said, "We knew we couldn't initially get people to accept a new bunch of radars. But we could sell the idea of a computer to analyze the radar data. Now we've got the computer, let's stop and think about the radars."

Forrester: We have recalled the obscurity of the problems and the embryonic state of the technology, but I have always felt that it was much easier to build the SAGE system without having the technology available to start with than if the technology *had* been there. With the absence of the technology there was also the absence of thousands of people out there all feeling that they knew how to do it better. Therefore, we were able to move quickly with our decisions. As long as they were plausible and could be explained, we could carry other people with us. We weren't at the same time running competition with alternative suggestions that, whether they were good or not, would immobilize the process of decision making. The freedom to be decisive and to settle on things that worked even if there might be somewhere in the offing an idea that would be better, made it possible to build the SAGE system.

Taylor: I think Bob put his finger on one important thing here: the freedom to do something without approval from top management. Take the case of the 65,000-word memory we just heard about. We knew the memory was too small; we didn't have to wait for Herb to worry about it.

Benington: I'm sorry you didn't tell me.

Taylor: We could hardly run a test program on these small memories, and we knew we had to build bigger ones. Down in the basement of the Lincoln Lab, we started out with TX-0 which was really designed not only to test transistorized computers but to test that big memory. That's all it did. We built that big memory, and we didn't go to the steering committee to get approval for it. We didn't go up there and say, "Now, here's what we ought to do, it's going to cost this many million dollars, it's going to take us this long, and you must give us approval for it." We just had a pocket of money that was for advanced research. We didn't tell

anybody what it was for; we didn't have to. Take any one of those developments—whether it was that memory, the Memory Test Computer, or the cathode-ray tubes and the Charactron tubes—if we had had to go through the management stuff that we have to go through now to get $100,000 worth of freedom, we would never have done any of them. We *were* able to do it. We'd have a meeting with Bob and me and one other person—and with Jay if he were there. Occasionally these projects failed or needed more funds or more time. On these occasions, the issues did rise to higher management levels—first the Lincoln Steering Committee, next the Air Force, and as needed the New York ADES meetings. The atmosphere was one of asking for help, and usually the response was positive. As stated earlier, the problems rose to the surface, not the successes, so management addressed problems. As long as it worked we were winners.

Tropp: In the current environment we may have to reinvent organizations with freedom of action to solve problems of our present society.

Forrester: We will reinvent effective organizations in this country very quickly when we perceive as a nation that we truly have a technological crisis, such as happened, for example, in the Manhattan project, with travel to the moon, at the Radiation Laboratory in World War II, and in the SAGE system. The effective pattern in the past has been to take several people who have good ideas and give them a budget and an empty cornfield. They can build buildings, find people, and do the job faster than can most existing organizations, which have become stifled by cumbersome decision making. The missing link in the past has been absence of a way to terminate organizations when their purpose is accomplished. Most continue to exist after the job is done, become ineffective, and yet continue to absorb resources.

Tropp: I'd like to get at a little earlier aspect of the Air Force's point of view, General Shiely. In one of the documents that Bob Everett was able to get me, I ran across the fact that at least two proposals were mentioned for an air-defense system: one at the University of Michigan and one at Lincoln Laboratory. Would you care to talk about what the other one was and why the decision was made to go with the one from Lincoln Laboratory?

Albert R. Shiely: I have to take a little bit of issue with Jay that there wasn't any competition. Part of our job in New York was trying to isolate those doing the job from all the experts who were certain they knew how the job could be done by different approaches and who also were completely convinced that the SAGE system would never work. At least one competitive approach was sponsored by the University of Michigan and supported by a substantial segment of the scientific community. With these two approaches, things got to the position where the Secretary of the Air Force had to make a choice between the two approaches. I believe it was Roger Lewis who made the decision to proceed with the SAGE system. During the time it was being put into the field there continued to be substantial concern on the part of very qualified parts of the scientific community over many of the problems that have been mentioned here. Further, there was significant concern by the military operators over whether a centralized system of this type was the right way to go or whether one ought to have an improved decentralized system operating at the radar sites much as the old system operated.

As I remember, the Michigan proposal was to automate the decentralized radar system and provide the improvement by that approach. In contrast, the SAGE system was to centralize it and combine the air picture into an overall one.

So there were operational concerns about the SAGE approach, and there were technical concerns about the SAGE approach. What's missing from the general area of these papers, which are superb, is the atmosphere under which this program was pursued. One way of describing it might be called a stage of controlled panic. Jay was exactly right: we were fortunate that the big problems occurred one at a time instead of all at once. I can recall being worried every time I would see Bob Crago or Bob Everett or someone like that come to New York because I knew something was coming; and I didn't know what hand grenade some-

Jay W. Forrester
"I have always felt that it was much easier to build the SAGE system without having the technology available to start with than if the technology *had* been there."

body was going to roll out on the table, but I knew there was going to be one. I recall such things as the sudden discovery that the slowed-down video technique would not work with large radars, and we didn't know what the hell the solution was. George Valley did say, "I've got the answer on a workbench in a laboratory up here. Don't worry; all you need to do is make identical copies of that and put it into the field." I think that device on the counter had something on the order of 150 or 200 vacuum tubes. In fact, it did work, and the Burroughs Corporation took that 200-tube thing from the bench and into the field in working order in 18 months—I think it went to something like 2200 vacuum tubes. The identical copy was something of that order. Nevertheless, I think this was a minor miracle on the part of that company—one of many miracles that were worked by industry, by the technical people at Lincoln, and by many elements of the Air Force.

There were a number of problems. That was one, and the one-year programming delay was another; we had a series of panics that occurred, as I said, one at a time.

Tropp: Do you think they occurred because nobody realized what the other ones were until they showed up?

Shiely: We were building and designing and doing everything simultaneously. I agree completely with Jay that the first and most important thing was that there was a national perception of the emergency need for an improved air-defense system; there wasn't any argument. We had to do something about it, and we were told to go do it—do it as fast as we could and make it work. There was an understanding at the topmost part of the government that the need was urgent. I might add that the willingness on the part of the military side of the family to give people like ourselves in New York the authority and freedom to move and the backing to make the decisions involved, even at the price of tearing up some of the organizational structures in the process, were the keys to success as far as that side of the program was concerned. That got us the license and the freedom to do the things mentioned here.

Tropp: From this discussion, there were apparently hoards of what would today be called managerial problems. Had you known they were going to come up, you might not have tackled any of them. When you look at any aspect of it, the management was really a kind of horrendous job.

Shiely: It was a challenging job, but it was a good one in the sense that the one thing that everybody agreed on was that we had to get the job done. There wasn't anybody who was motivated to do anything other than find solutions to the problems. It was a great thing.

Forrester: There is a chapter in the SAGE history that I feel was extremely important. It was part of our background for taking the long-range view of where we were going. In 1948 Karl Compton, who was then president of MIT and also head of the Research and Development Board for the military, asked us to prepare a report on what we thought the future of digital computers would be in the military. In 1948 there were scarcely any working systems to use as a precedent. Five of us—Bob Everett, Hugh Boyd, Harris Fahnestock, Robert A. Nelson, and I—worked up a report that was a 15-year forecast of computers in the military. It culminated in a two-by-three foot foldout page, which had 15 years across the horizontal axis and 10 areas of application along the vertical axis. [*Editor's Note:* The chart is reproduced on pages 382–383.] In each intersection was described the state foreseen for the application. The applications included logistics, research, antiballistic missile defense, air-traffic control, and control of naval task forces. At each intersection was given the state of development and how much would be spent in that year for research, how much for development, and how much for production. The grand total was $2 billion. The research expenditure alone totaled over $1 billion at the end of 15 years. The report created a communications gap when we went into a meeting with the Office of Naval Research where they thought the agenda was whether or not we were going to get our next $100,000—and we came in with a forecast 10,000 times that for the next 15 years.

Tropp: Looking back on that forecast, was it conservative or pretty close?

Forrester: The air-defense part of it, which we pursued, came out about three years sooner. The cost was probably within an order of magnitude. I would say the projections for 15 years were probably as good as

Albert R. Shiely "There was an understanding at the topmost part of the government that the need was urgent."

most commercial companies do today when they estimate the cost and the schedule for the next computer in production.

Incidentally, that $2 billion was generated by looking into every step of research—the design, testing, and training of people. Those budget figures were made up out of pieces no bigger than 25 people working a calendar quarter. The 15 years were laid out by identifying small groups of people who would be working in each area. One reason the time schedule came out fairly reliably was that all of the political time delays were put in, such as a year set aside for people to agree to test and follow up on experimental equipment.

In the context of the Michigan and MIT competition, our earlier estimate of the possible technical pace gave us a frame of reference for how big and how long the job might be. We went in with an estimate and forecast that ran maybe ten times as much money and five times as much time as Michigan did. In the period of disagreement over how to build an air-defense system, our position was that if any serious group of people thought they could do the job in the time and price range that was being suggested, they should be allowed to try. But we also said that the longer-term program should be kept going until a smaller-scale system could be evaluated. Eventually the day came when it was evident to almost anyone walking through the two establishments that the long-term slow program at MIT was already further ahead. Then the $300 million of production money that was scheduled for the other system was diverted to complete the research and development for the MIT system.

Robert Bright: Reservations had been expressed by numerous learned people. There was an opposing school that said, "This is in the national interest; this is the kind of thing we ought to do. The system as proposed is not static; it could be dynamic; it could be developed along the way." At that time Bell Laboratories and Western Electric were involved in a study of continental air defense, called naturally the CADS project. In May 1955 General N. F. Twining, vice chief of staff of the Air Force, wrote to M. J. Kelly, president of Bell Laboratories, saying, "Take a look at what Lincoln's doing. We're already phasing out the University of Michigan, and we want you to undertake the management and the implementation of SAGE." With that, when it was nailed down, we built an organization comprised of Lincoln Laboratory, Bell Laboratories, IBM, Burroughs, and later the building contractors, that worked together as well as any team that I've ever seen or been exposed to. There was full and complete communication between the contractors involved in the project. Through the pink haze of 25 years, I remember that we even got to be friends with many of the people involved. Our phasing meetings, which most of you attended, were wide open. People were welcome to speak their mind, and if they were right a consensus would be developed. But that kind of project organization, that kind of management, developed on an ad hoc basis, seemed to work. One of the things that made it work was the shared location and cooperation of the Air Force Project Office and the contractors involved. We were in each other's hair almost constantly, and we could get the support we needed from the Air Force Project Office. The national interest at that time was to build a viable air-defense system, and this was foremost in the minds of everybody involved.

Robert Bright
"We built an organization comprised of Lincoln Laboratory, Bell Laboratories, IBM, Burroughs, and later the building contractors, that worked together as well as any team that I've ever seen or been exposed to."

John F. Jacobs: I agree with Bob. At the same time we were working on the system design, others were busy making the organizations involved do the things that had to be done in order to turn the air-defense concept into reality. The most important step in that direction was the Air Force's establishment and structuring of ADES (Air Defense Engineering Services). This project office was the first to deal with electronic systems doing command and control development and procurement—in fact, it was the first electronic systems program office. These systems were designed to control a number of weapons systems, rather than being tailored to the control of a single system. ADES had to deal with these elements—old and new subsystems that had to be integrated into the new air-defense components—and they also had to provide management support to command personnel.

ADES remained flexible and encouraged adaptations of their own organization to the needs of the job. The monthly phasing meetings are an example of the tools they instituted for achieving a consensus. The

	ACTIVITIES	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958
BASIC RESEARCH	COMPONENT AND CIRCUIT DEVELOPMENT STORAGE SYSTEMS MATHEMATICS AND APPLICATIONS RESEARCH COMMON TO MANY USES	R1.9 CONTINUED INVESTIGATION	R3.7	R4.8	R8.3	R11.7	R15.4	R17.0	R17.6	R17.6	R17.6
TRAINING	RESEARCH PERSONNEL APPLICATIONS PLANNING OPERATING STAFF	R0.3 INFORMAL INSTRUCTION COURSES IN 1 OR 2 COLLEGES	R0.8 50 STUDENTS WITH MORE NEEDED IF FACILITIES WERE AVAILABLE	R1.4 250 STUDENTS	R 2.2 500 STUDENTS	R 2.8 650 STUDENTS	R 4.4 850 STUDENTS	R4.9 1050 STUDENTS	R6.9 1300 STUDENTS	R7.5 1600 STUDENTS	R8.5 1950 STUDENTS
DATA CONVERSION	PHYSICAL MEASUREMENTS TO DIGITAL QUANTITIES DIGITAL TO PHYSICAL RADAR DATA TO DIGITAL FORM COMPUTER–TO–SERVO LINKS	R 0.4 CONTINUED INVESTIGATION	R0.8	R1.8	R2.0	R2.2	R2.4	R2.4	R2.4	R2.4	R2.4
* ENGINEERING AND SCIENTIFIC COMPUTERS	GUIDED MISSILE DATA REDUCTION SURFACE SHIP AND AIRCRAFT STABILITY SCIENTIFIC COMPUTATION OPERATION EVALUATION ANALYSIS DESIGN COMPUTATION MATHEMATICS AND NUCLEAR RESEARCH CRYPTOGRAPHIC APPLICATIONS	R2.3 DESIGNS NOW IN PROGRESS	R5.0 FIRST EXPERIMENTAL HIGH–SPEED MACHINES	R6.0 LABORATORY TESTING AND STUDY	R7.0 E1.0 EXPERIMENTAL TRAFFIC NETWORKS	R8.2 E2.0 OPERATING TRAFFIC NETWORKS REDESIGNS	R8.8 P3.0 CONSTRUCTION OF ADDITIONAL COMPUTING FACILITIES	R9.5 P3.0 INSTALL ADDITIONAL FACILITIES	R11.9 P5.0 INSTALL AND OPERATE	R13.0 P5.0 INSTALL AND OPERATE	R15.6 P5.0 COMPUTERS NOW MEETING SCIENTIFIC AND ENGINEERING LOAD
FIXED LOCATION CONTROL SYSTEMS	INTERCEPTION NETWORKS AIR TRAFFIC CONTROL MANUFACTURING PROCESS CONTROL	R0.2 INTERVAL OF TIME FOR EVALUATION OF COMPUTER WORK, BUDGET PLANS, CONTRACT NEGOTIATION, SETTING UP LONG RANGE INTEGRATED RESEARCH AND DEVELOPMENT PROGRAMS. THIS TIME COULD BE REDUCED. IF SO, THE ABOVE ITEMS, PARTICULARLY TRAINING, MUST BE GREATLY INCREASED.	R0.4 APPLICATIONS STUDIES PRELIMINARY DESIGNS	R2.7 BEGIN EXPERIMENTAL CONSTRUCTION DESIGN AUXILIARY EQUIPMENT	R3.2	R4.4 F2.0 FINISH CONSTRUCTION OF FIRST EXPERIMENTAL INSTALLATION	R4.9 E3.0 INSTALLATION AND PRELIMINARY OPERATION FURTHER DESIGN	R5.9 E3.0 FIELD TESTS OF EXPERIMENTAL INSTALLATIONS	R6.8 E5.0 CONTINUED FIELD TESTS AND FURTHER EXPERIMENTAL INSTALLATION PROTOTYPE DESIGN	R7.8 E6.0 PROTOTYPE CONSTRUCTION PRODUCTION DESIGN	R10.2 E10.0 PROTOTYPE FIELD TEST CONTINUED PRODUCTION DESIGN
SIMULATION SYSTEMS	DESIGN STUDY SIMULATORS SYNTHETIC TRAINING WAR COLLEGE TACTICAL SIMULATOR GROUP TRAINING–AIR AND ANTI–SUBMARINE TEAMS	R 0.1	R0.2 APPLICATIONS STUDIES	R0.5 CONTINUE STUDIES SPECIFICATIONS FOR EXPERIMENTAL USE	R0.8 E1.5 CONSTRUCT EQUIPMENT SIMILAR TO ENGINEERING COMPUTER DESIGNS	R1.3 E3.0 CONSTRUCT AND INSTALL DEMONSTRATION SYSTEMS	R3.6 P3.0 FIELD TESTS	R4.5 P3.0 EVALUATION OF TEST INSTALLATION	R5.1 P4.0 CONTRACTS FOR SIMULATOR CONSTRUCTION	R6.3 P6.0 UNITS BEING BUILT FOR TRAINING AND ENGINEERING SIMULATION
SEMI–MOBILE CONTROL COMPUTERS	SHIPBOARD COMBAT INFORMATION CENTER FIRE CONTROL RADAR DATA CORRELATION ANTI–SUBMARINE TRIANGULATION AND AUTOMATIC TASK GROUP ATTACK CONTROL	R0.2 APPLICATIONS STUDIES PRELIMINARY DESIGN	R1.4	R1.8 DESIGN EXPERIMENTAL SYSTEM	R2.2 E1.0 BEGIN CONSTRUCTION OF EXPERIMENTAL UNIT	R3.6 E3.0 FINISH CONSTRUCTION OF EXPERIMENTAL UNIT FOR SHIPBOARD	R4.2 E3.0 INSTALLATION ON SHIPBOARD	R4.8 E3.0 FIELD TESTS	R6.0 E3.0 FIELD TESTS PROTOTYPE DESIGN	R7.5 E6.0 PROTOTYPE CONSTRUCTION
MOBILE CONTROL (TRUCK AND LARGE AIRCRAFT)	MISSILE LAUNCHING CONTROL FOR ANTI–MISSILE DEFENSE AIRBORNE COMBAT INFORMATION CENTER	R 0.2 BASIC STUDIES INVESTIGATE SPECIAL COMPONENTS AND METHODS	R0.4	R0.9 CONTINUED PRELIMINARY INVESTIGATION	R1.7 EXPERIMENTAL STUDIES	R2.8 E1.0 CONTINUE EXPERIMENTAL WORK DESIGN MODEL SYSTEM	R3.3 E2.0 CONSTRUCT EXPERIMENTAL SYSTEM	R3.9 R3.0 FIELD TESTS	R4.7 R3.0 FIELD TESTS PROTOTYPE DESIGN	R6.5 R5.0 CONTINUE EXPERIMENTAL WORK COMPLETE PROTOTYPE DESIGN
AIRBORNE COMPUTERS	PRECISION MISSILE CONTROL AIRBORNE FIRE CONTROL	R0.1 APPLICATIONS STUDIES STUDY OF SIMPLIFIED COMPUTERS	R0.5	R1.0 EXPERIMENTAL RESEARCH	R2.0 BEGIN EXPERIMENTAL DESIGNS	R3.4 EXPERIMENTAL RESEARCH AND DESIGN	R4.2 CONSTRUCT EXPERIMENTAL EQUIPMENT	R4.4 E1.0 PRELIMINARY FIELD TESTS REDESIGN EXPERIMENTAL EQUIPMENT	R5.2 E2.0 FIELD TEST	R6.8 E5.0 PROTOTYPE DESIGNS
LOGISTICS	SPECIAL INPUT–OUTPUT EQUIPMENT INTERCONNECTION WITH COMMUNICATIONS EQUIPMENT HIGH CAPACITY DATA STORAGE	R0.1 R0.2	R0.3 APPLICATION STUDIES	R0.6 TRIALS WITH EXISTING COMPUTERS	R1.7 EXTEND STUDIES BEGIN WORK ON TERMINAL EQUIPMENT	R3.6 DESIGN EXPERIMENTAL EQUIPMENT FOR LARGE–SCALE USE	R6.1 CONTINUED EXPERIMENTAL WORK CONSTRUCT SOME SPECIAL EQUIPMENT	R8.0 E2.0 BUILD MODEL INSTALLATION	R10.4 E3.0 TRIAL TEST	R14.4 E6.0 CONSTRUCT PROTOTYPE EQUIPMENT
COSTS IN (MILLIONS)	TOTAL YEARLY COST T	T5.2	T11.5	T19.5	T28.5	T44.2	T63.6	T75.1	T93.2	T105.7	T138.8
	TOTAL YEARLY RESEARCH R	R5.2	R11.5	R19.5	R27.5	R37.7	R50.6	R61.1	R71.2	R 79.7	R 95.8
	TOTAL YEARLY EXPERIMENTAL E CONSTRUCTION	0	0	0	E1.0	E6.5	E10.0	E8.0	E14.0	E17.0	E32.0
	TOTAL YEARLY PRODUCTION P	0	0	0	0	0	P3.0	P6.0	P8.0	P9.0	P11.0
	NUMBER OF STAFF REQUIRED	300	600	950	1350	1800	2400	2900	3500	4100	4700

NOTES: 1. ALL COST FIGURES ARE IN MILLIONS OF DOLLARS
2. STAFF WAS ASSIGNED TO SPECIFIC ACTIVITIES IN EACH YEAR. COSTS HAVE BEEN DERIVED AS DISCUSSED IN THE TEXT.

* 50% OF THE COST IN THE ENGINEERING AND SCIENTIFIC COMPUTER ROW IS FOR ROUTINE OPERATION. OTHER ROWS INCLUDE ONLY EXPERIMENTAL AND FIELD TEST OPERATION.

COST SYMBOLS
- T TOTAL
- R RESEARCH AND DEVELOPMENT, INCLUDING OPERATION AND FIELD TESTING, PRODUCTION DESIGN, AND SALARIES OF ASSIGNED TRAINEES
- E EXPERIMENTAL CONSTRUCTION COSTS
- P PRODUCTION COSTS (NOT INCLUDING DESIGN)

UNCLASSIFIED

other organizations that should share in the credit are the Air Defense Command Planning Group under Colonel Tom Halley, the wing established by Air Defense Command under Colonel Joseph Day Lee that supplied the inputs to the operational specification, and the Lincoln Project Office, which monitored our activities at Lincoln Lab.

Tropp: Maybe this is a good time to discuss some aspects of the actual design of the hardware for the system. One thing that struck me as I looked at the Q-7 a couple of weeks ago was that I literally felt as if I were walking inside a computer, which I obviously can't do with an Apple or a TRS-80.

Taylor: You could in Whirlwind.

Tropp: You could walk inside Whirlwind, that's true. Other hardware ideas are having a duplex computer, the design of the base hardware to do that hot backup, the concept of the common drum. I hope somebody will pick up these ideas and discuss how they evolved and what some of the problems were. Jay, do you want to start?

Forrester: One thing running through the whole program was central to its success. That was an attitude of being open about recognition of mistakes and shortcomings. When a mistake was recognized, it was admitted and fixed rather than evaded or denied. An

1959	1960	1961	1962	1963	STATUS AT END OF 15 YEAR PERIOD
R18.6	R18.6	R19.8	R21.7	R23.0 CONTINUED INVESTIGATION	TOTAL COST T217.3 RESEARCH ON IMPROVED METHODS AND COMPONENTS SHOULD CONTINUE
R10.8 O STUDENTS	R11.4 2750 STUDENTS	R14.3 3250 STUDENTS	R15.1 3900 STUDENTS	R20.0 4400 STUDENTS	TOTAL COST T111.3 THE GREATEST PART OF THE TRAINING IS NOW IN OPERATING PERSONNEL
R2.4	R2.4	R 2.4	R2.4	R2.4 CONTINUED INVESTIGATION	TOTAL COST T31.2 RESEARCH SHOULD CONTINUE
R17.4 P3.0 IPUTERS REASING IN USE	R18.2 P3.0 COMPUTERS INCREASING IN USE	R20.0 P 4.0 COMPUTERS INCREASING IN USE	R21.0 P4.0 COMPUTERS INCREASING IN USE	R22.0 P4.0 COMPUTERS INCREASING IN USE	TOTAL COST T227.9 ✳ R185.9 E 3.0 P 39.0 ALL LARGE SCALE ENGINEERING AND SCIENTIFIC CALCULATION NOW DONE BY COMPUTERS
R12.2 E10.0 P5.0 T PRODUCTION EMS	R15.6 E10.0 P10.0 PRODUCTION INSTALLATIONS TRIAL AIR TRAFFIC CONTROL SYSTEM	R21.0 E10.0 P20.0 PRODUCTION	R21.4 E10.0 P50.0 PRODUCTION	R 21.4 E10.0 P100.0 PRODUCTION	TOTAL COST T402.1 R138.1 E 79.0 P185.0 PRODUCTION COMPUTERS AVAILABLE AND OPERATING IN CONTROL SYSTEMS. BULK OF PRODUCTION STILL TO COME
R8.3 P6.0 D INSTALL OPERATE	R8.3 P6.0 BUILD INSTALL AND OPERATE	R11.4 P10.0 BUILD INSTALL AND OPERATE	R12.4 P10.0 BUILD INSTALL AND OPERATE	R12.4 P15.0 BUILD INSTALL AND OPERATE	TOTAL COST T144.0 R76.5 E 4.5 P63.0 HIGHLY FLEXIBLE TRAINING AND ENGINEERING SIMULATORS IN OPERATION
R8.2 E6.0 TOTYPE D TESTS DUCTION IGN	R8.8 E6.0 P8.0 START PRODUCTION	R10.6 E6.0 P15.0 FIELD TEST PRODUCTION UNIT	R10.0 E5.0 P50.0 PRODUCTION	R10.0 E5.0 P100.0 PRODUCTION	TOTAL COST T299.3 R79.3 E47.0 P173.0 OPERATING SHIPBOARD EQUIPMENT AVAILABLE. INSTALLATION COMPLETE ON SOME MAJOR FLEET UNITS. BULK OF PRODUCTION STILL TO COME.
R7.8 R6.0 TOTYPE STRUCTION DUCTION SIGN	R9.6 E6.0 P2.0 PROTOTYPE FIELD TESTS COMPLETE PRODUCTION DESIGN	R11.5 E5.0 P5.0 START PRODUCTION	R10.6 E5.0 P15.0 PRODUCTION	R10.6 E5.0 P50.0 PRODUCTION	TOTAL COST T188.5 R74.5 E42.0 P72.0 PRODUCTION MOBILE COMPUTERS BECOMING AVAILABLE BULK OF PRODUCTION STILL TO COME.
R8.8 E6.0 TOTYPE STRUCTION	R10.2 E6.0 P1.0 PROTOTYPE FIELD TESTS PRODUCTION DESIGN	R12.0 E6.0 P3.0 START PRODUCTION	R12.6 E6.0 P6.0 PRODUCTION	R13.0 E6.0 P15.0 PRODUCTION	TOTAL COST T147.2 R84.2 E36.0 P25.0 PRODUCTION AIRBORNE COMPUTERS AVAILABLE. BULK OF PRODUCTION STILL TO COME.
R17.9 E7.0 IN WORKING TALLATIONS	R19.9 E6.0 P5.0 BEGIN EQUIPMENT PRODUCTION	R24.4 E 6.0 P10.0 PRODUCTION	R24.4 E 7.0 P20.0 PRODUCTION	R25.4 E 7.0 P35.0 PRODUCTION	TOTAL COST T271.4 R157.4 E44.0 P70.0 COMPUTERS IN EVERY-DAY USE SOLVING LOGISTICS PROBLEMS BULK OF PRODUCTION TO COME
T161.4	T193.3	T248.4	T339.6	T512.2	GRAND TOTALS T2040.2
R112.4	R124.3	R147.4	R151.6	R160.2	R1155.7
E35.0	E34.0	E34.0	E33.0	E33.0	E 257.5
P14.0	P35.0	P67.0	P155.0	P319.0	P 627.0
5400	6100	6700	7200	7550	

FIGURE 6

FORECAST OF DIGITAL INFORMATION HANDLING PROGRAM

PROJECT WHIRLWIND SEPTEMBER 14,1948
SERVOMECHANISMS LABORATORY D-32904
MASS. INST. OF TECHNOLOGY

Chart with 15-year forecast for military computers, mentioned by Jay Forrester on page 380. The grand total in 1948 was $2 billion—equivalent to $8 billion in 1983 dollars. (Figure 6 from Jay W. Forrester, Hugh R. Boyd, Robert R. Everett, Harris Fahnestock, and Robert A. Nelson, "A Plan for Digital Information-Handling Equipment in the Military Establishment." Project DIC 6345, MIT Servomechanisms Laboratory, September 14, 1948. Reproduced courtesy of Division of Mathematics, National Museum of American History, Smithsonian Institution.)

example was the second computer or the duplex computer in the SAGE centers. The decision to insist on a second computer occurred one weekend when we began to realize that there wasn't going to be the reliability in a single machine that we had been promising. By that time the Air Force had already budgeted the whole system. To double the number of computers required going back to the Air Force for the extra money. There was a lot of flak from that, but our position was that it had to be done. We wouldn't stand behind the system if they didn't. The Air Force supported such changes very effectively.

The marginal-checking system that improved reliability by a factor of ten came about as a result of forthrightly admitting a weakness and solving it. Again, it was one of those interesting happenstances. In the days of Whirlwind we had what we referred to as the "annual investigation." Somebody always began wondering again how we could spend several million dollars on one computing machine. The result would be another panel or an investigator. In about 1948, Francis Murray (a mathematician from Columbia) spent a weekend investigating on behalf of the Office of Naval Research. On Saturday in my office he said, "What are you going to do about all these components when they gradually deteriorate, and streetcars go by, and the voltage will change? You won't know you're approaching trouble until components begin to cause mistakes." We hadn't thought about that before, but we recognized it as something that had to have an answer. I told him that we would vary the screen voltages of the vacuum tubes up and down so we could tell if they still had a safe working margin, and this would detect gradual deterioration. The idea sounded so good that the next Monday morning Norm Taylor started putting such a system into Whirlwind.

Tropp: Jack Jacobs's overview paper clearly points out that the desire to get this job done was strong enough that once you had convinced people of what you could do, your mistakes weren't so important.

Taylor: I did a lot of the direct correlating of the numbers to find out what reliability we could achieve and found we needed the dual machine.

We had enough data about tube reliability and so forth to correlate one piece of data with another and make a reasonable prediction. Then we said, "Well, if we have one machine, what will be a mean time to failure? If we have two, what will it be? Indeed, is it possible to take advantage of the improvement of two because we have twice the probability of failure? After a while we made up our minds to go to two machines in a center.

We were about to make a presentation on reliability to the Air Defense Command in Colorado Springs

Norman H. Taylor
"It would take a year to get that decision nowadays."

when I said to Jay, "These are all the numbers. I've never been to Colorado Springs, but I would love to go." He said, "You can go if you make the speech."

I had never made a speech like that before, but I remember getting up, and there was what seemed like a whole room full of generals sitting in front of me—and I hadn't even met a general before that. I gave the speech almost the way I had memorized it. We simplified the equations so that I wouldn't get tangled up in it and everyone could follow it nicely. Afterward at a cocktail party, one of the generals whose name I don't remember came up to me and said, "Norm, that was a very convincing speech. We're going to go with this thing, this duplex." Just right there, that afternoon. It would take a year to get that decision nowadays.

Tropp: Not only were you building two computers instead of one, but you would also need a building twice as large as well as all the things that go with it.

Shiely: We didn't even know at that point how big a building we would need.

Taylor: A real problem was the number of people. We hadn't even found out how many operating positions we would need.

Thomas M. Smith: I was wondering about these references and this philosophy that Jay was talking about and Norm was just referring to. How did that strike you, Bob Crago, coming into their operation, working with them? Was it something new, distinct, energizing, in your experience?

Robert P. Crago: I think it was probably the first experience most of us in IBM had in working with an outside group and taking the leadership from them. We had been pretty much our own masters in previous projects. It took us a while to realize that we were dealing with some very professional people who knew what they were doing. While reading through the papers, I thought about the coordination meetings in back of the IBM branch office in Hartford. Was it your mother, Norm, who was calling in to find out if you were at the meeting or not? No one would admit to the meeting that was going on in the back room.

I think the kind of technical camaraderie that grew out of this project made it the most exciting thing that I've ever been involved in. I had the feeling that there were no hidden problems, that no one would tuck things away.

Taylor: That was part of it. We told everybody what the problems were as soon as we found them out.

Smith: And in the 20 or 30 years that have passed since, you have not encountered that kind of phenomenon in industry?

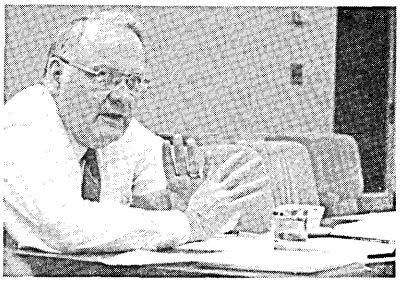
Taylor: It's usually "don't tell anybody until you hit the skids," so to speak.

Tropp: Returning to your analysis of how two computers would be better than one in terms of mean free time between failures, how about the common drum?

Taylor: The duplex arrangement wasn't designed at that time. I think it was Steve Dodd who did the details; he was in charge of that part of the design, anyway. There was to be one switch that was going to switch from one computer to the other. Dodd told me that that was the biggest oversimplifier in all the world.

Benington: We used to joke that when that switch was thrown, we had better warn Niagara Falls.

Taylor: I've been taking quite a riding all my life about that switch because it turned out to be a million-dollar switch. During its design there was a great deal of tugging and hauling on how we would use that second computer. At first we probably had the simpler idea that we would do everything on both sides simultaneously. It turned out that that was rather inefficient because we had a lot of things to do, and the other computer could do some of them while keeping a check on the computer that was carrying the operational load. Whichever machine was backup had to keep its status information up-to-date, of course, in case it had to take over. The common drum was a convenient way for the two machines to carry out the necessary information exchange.



Robert P. Crago
"I think the kind of technical camaraderie that grew out of this project made it the most exciting thing that I've ever been involved in."

Actually, duplexing the computers was the easier part of the job. Much more difficult and expensive was switching the enormous number of peripherals from one machine to the other, and searching out and duplexing all the single-point failure modes in the entire center, from the power plant to the commander's display. Nowadays there are only one or two people who have really working duplex computers. Tandem has built a tremendous business out of this, and nobody else does it.

Tropp: This is a good time to discuss some aspects of software development. The choice by the Air Force of SDC to do software was another one of those accidents. I think that Herb was going to talk about it in terms of, what was it, STP?

Benington: Yes, the System Training Program. At the same time that Whirlwind was being developed, the psychologists at Rand decided they wanted to do some experiments on team training. They were interested in how you could improve the performance of the team and how to measure it. Bill Biel (later vice-president of SDC) was the man who headed that effort, and he decided to use the air-defense function as an experiment to try it out. Rand, from which SDC spun off, was working for the Air Force. Even though they were doing most of their work with the Strategic Air Command on the offensive problem, there wasn't much large team activity in SAC, but in air defense there clearly was. So they set up team-training exercises using college students as simulated officers and enlisted men. From this work they developed some important principles. One principle was that you ought to start out with very low stress on the student operators and then gradually increase it as the team performance increased. Another was that right after the experiment you ought to debrief them on what was really going on in the world so they would look at what they had just done for three hours, what they thought was going on, and what had really happened. A final principle was that you ought to have the team analyze itself and look at ways in which it could improve its performance. They got tremendous performance from those college students. Some of the people from the Air Defense Command came and looked and said that it was just the excellent Rand training and the fact that the college students had very high IQs—that they ought to try it out on some of the actual personnel from the defense command. So they brought in some of the enlisted men and a few officers and started doing this to them. They found that they got even higher performance than they did with the college students because those people knew the business and were highly motivated. So, independent of the SAGE activity, Rand started this System Training Program for the air-defense centers, and when SAGE came along it was used in SAGE.

Having that other backup computer we discussed earlier was a great opportunity because this realistic simulation could then be run from the other computer and fed in across that drum. Also, some of the terminals could be diverted to simulate people at adjacent centers, to simulate the pilots, and the like. So that program turned out to be, I think, very successful. As long as I was familiar with the program, Rand and later SDC continued it. I guess it was also quite expensive. I would be interested in whether General Shiely has any views as an operator as to how useful that program was, and whether we still have anything to learn from it today?

Shiely: I really didn't get into that too much except secondhand. My understanding is that it was a very valuable program for keeping the operators trained. I know it was the basis for an awful lot of the work we did in system testing and getting the operational people up to snuff on the system prior to its going into service. Rand was very, very important from that standpoint.

Tropp: Many choices were made regarding who did which aspect—for example, the choice of Lincoln's proposal for the general concept of the system, the choice of IBM as the prime contractor, the choice of SDC to do the software. I'm curious about earlier thoughts as to who would do these things. For example, in doing the software, was there an original thought that it would be done at Lincoln Lab? Or that the Air Force would do it prior to finally realizing the magnitude and saying, "Hey, we'll go elsewhere"?

Forrester: For SDC it was a matter of going outside the Lincoln Laboratory. For reasons I do not recall, Lincoln did not want to build up its staff to that extent.

Everett: The MIT management did not want to. They didn't see a long-term need for a large number of software people as a part of MIT.

C. Robert Wieser: SDC wasn't chosen; it was created.

Everett: SDC was actually created here.

Forrester: The group that started working for Rand and became SDC was started here. We started hiring people, and later SDC took over its own hiring. SDC was created for the purpose of doing the programming.

Tropp: Were there earlier thoughts—because you had done your own programming on Whirlwind—that that was a project that you could do in-house?

Forrester: The programming was started inside. Bob Wieser and his group did a great deal of it before SDC came into being.

Wieser: There was serious consideration of getting IBM to pick up the programming job, too, but as I remember they didn't want it. There was serious consideration of having Bell Telephone Laboratories do it, and they didn't want it either. SDC was created to do the job; it was created inside my group. At that time, a division in Lincoln was roughly 100 people, and a group was roughly 10. When my group got to 120, I knew something had happened.

Crago: It's true that IBM was offered the opportunity of taking on the programming. We estimated that it could grow to several thousand people before we were through. I think the principal factor in deciding not to do the work was that we couldn't imagine where we could absorb 2000 programmers at IBM when this job would be over someday, which shows how well we were understanding the future at that time.

Jacobs: The computer programming was probably the most underestimated task in the entire SAGE project. The team that programmed Whirlwind and the Cape Cod System was obviously the team that should take some responsibility for developing the SAGE programming and also for carrying out air-defense programs. SAGE necessitated a system that, in addition to the verification of the air-defense concept, had to be done on a much more complicated machine—the FSQ-7. It also had to be done by inexperienced programmers, and it had to be aimed at producing a master program to be installed at a number of sites, and adapted to those sites. SAGE also required both the training for and definition of program maintenance. Finally, there was a myth that you could do all or most of the integration of weapons by modifying the computer program. This turned out to be untrue. Not only did most of these changes require hardware changes, but the cost of programming soon rivaled the cost of doing the same thing with hardware.

Forrester: I've mentioned the managerial aspects of SAGE. The managerial structure that ran through the undertaking was at least as important as the technical work. The story wouldn't be complete if we didn't mention the IBM management support and dedication. IBM management really threw their resources into the program without restraint. As an example, when it came time to schedule production, there was no air force contract yet for the machines. If IBM was to meet the schedule, there had to be a factory. IBM went ahead and started building a factory before the Air Force had signed a contract. No doubt IBM could have built typewriters in the building if the contract did not come along, but, nevertheless, they built a factory specifically for the SAGE computers on their own initiative. It was that willingness to go ahead and to take some risks that ran all the way through the program in all the participating private company operations as well as the Air Force and MIT.

C. Robert Wieser
"At that time, a division in Lincoln was roughly 100 people, and a group was roughly 10. When my group got to 120, I knew something had happened."

Tropp: When this whole thing was starting, there was no FORTRAN, no COBOL, and no ALGOL. Everything was machine language, machine code, or something special of that nature. There were very few programmers, and people were walking around saying, "There aren't enough mathematicians around, anyway; we need many more programmers." There may have been 3000–4000 of them in the United States. What was the training problem like to create the small army of programmers to do this kind of job?

Benington: It was significant. I think we did it fairly well. We had computer programming courses run by IBM, Rand, SDC, and MIT. We discovered very quickly who could program well, after training and experience. But it was almost impossible to predict at the time of hiring.

Crago: Music teachers were particularly good subjects, weren't they?

Benington: Yes, music teachers. And women turned out to be very good for the administrative programs. One reason is that these people tend to be fastidious—they worry how all the details fit together while still keeping the big picture in mind. I don't want to sound sexist, but one of our strongest groups had 80 percent women in it; they were doing the right kind of thing. The mathematicians were needed for some of the more complex applications. So we did a lot of learning of how to train people, how to decide who was good, and where to put them.

We were talking earlier about software on the Whirlwind project, which started clarifying some of my thoughts. At Whirlwind there were two software groups. One of them was under Charlie Adams, who was developing the software tools that would be needed in a university environment. He was catering to the very bright individual who wanted to get on the computer. As a result he developed things such as

symbolic addressing, which would have been heresy in Whirlwind. With symbolic addressing, you refer to your data's location not by the number of the memory cell it is in, but by a mnemonic title such as A1 or TAX or something like that. Later it was all put together in the right way. He developed an operating system so that somebody could use input/output devices or storage and not have to worry about managing all the administrative and allocation details. He developed the first higher-order language well before FORTRAN and COBOL; this was in the early 1950s. These were all tools that are useful to individuals using the computer. But in the Cape Cod System and in SAGE, the problem was one of team programming. We thought that some of those tools were dangerous because they couldn't be well disciplined. I think we were wrong in retrospect, but we developed tools of a very complex nature, and they're not always found today in some of the more advanced projects where the team can be disciplined. You could assign an individual a job, you could control the data that that individual had access to, you could control when that individual's program operated, and you could find out if that individual was playing the game wrong and punish the person. So we had a whole set of tools for design, for controlling of the team, for controlling of the data, and for testing the programs that were really quite advanced.

Tropp: Very little of that ended up in the literature.

Benington: I gave one paper in San Francisco: "Lincoln Utility Program System."[2] Charlie Adams was there; when he said, "Good show," I felt extremely pleased. Also, the paper published in this issue of the *Annals* gives some of these ideas.

Wieser: In the early days of programming for Whirlwind, I can assure you that programming was not done by mathematicians. It was done by people who in the first place understood how the machine worked, and in the second place had some reasonable grasp of the problem they were trying to solve with the machine. They thought in terms of airplanes in space, not totally in terms of symbolic logic. A lot of homework went into that. When we first went into the air-traffic-control application before we had a Whirlwind, I spent many nights over in the control tower at Logan Airport finding out how the air-traffic-control system really worked. I got up at 5 o'clock mornings, drove out to Beverly, took flying instruction, and got a pilot's license. We got involved in the problem we were working on. It was not an abstraction; it was a real engineering thing. To the extent that you could confine it to a small number of people, which was true of the Cape Cod System programming, you could get an awful lot more out of the people. That way was more efficient than having one set of people writing requirements, another set writing specs, another set translating specs into code, another set checking it out. The people did all those things as one great team. They had the knowledge of all of the steps in the process through to checkout, through the experiments with real airplanes to find out how the whole system worked. I don't know that you could apply that process to big-scale programming—what's called production programming; you probably cannot. But on the scale we were doing it then, it worked well. The programming went very, very rapidly, much more rapidly than you could do it with a structured system of training programmers and using large numbers of specialists. It was more like the analog of the model shop in industry. What we had was a model shop doing the job. It was quick; it was inexpensive.

Benington: I agree with Bob completely. The difference between Cape Cod and SAGE is really the difference between the model shop and production. By the time we got to SAGE we had to specialize. One group did the requirements, one produced programs, one tested them, and one put them in the field. We had to do it, and we also learned that you could use less-experienced people to implement and test those requirements. Today it's still the most effective way of doing it.

Crago: It's a theme that's come up here a number of times. Jay said it, and Bob said it. The commonality of effort, the dedication of people, the fact that so many of us spent quite a bit of time in the particular application so that we had a chance to see our own mistakes and had a chance to help work them out. I see so many people moving quickly between jobs today—one year at the most. They miss getting a chance to see their mistakes and straighten them out.

Forrester: In the Digital Computer Laboratory that came out of the Servomechanisms Laboratory, at the center of the operation was a core team of people who had been through several complete, successful projects from the beginning of the idea through development, the experimental work, the production in factories, and out into the field where that equipment was either working or not working. They had the knowledge of every step that was yet to come up from having been there before. They had gone through the research-to-field cycle two or three times before they came to SAGE. Such a group of people made a powerful team in dealing with how management and technology are integrated together.

[2] H. D. Benington, "Lincoln Utility Program System." *Proc. Western Joint Computer Conference*, AIEE, San Francisco, February 1956, Vol. 9.

Tropp: I think the fact that there was a nucleus is often overlooked. A group of you who started with the aircraft simulator can be traced all the way through the evolution to SAGE.

Forrester: The group had gone through two or three projects in the Servomechanisms Laboratory even before the aircraft simulator.

Tropp: That continuity covers more than two decades.

Forrester: Starting in 1940 and running through 1962.

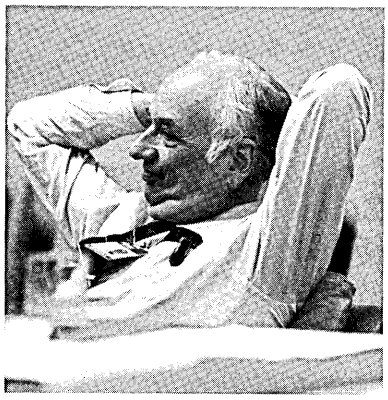
Everett: I'd like to respond to a question you asked earlier about whether people were confident that they could do the job. It seems to me that in this process we were just talking about—of going through the whole task, and of finding our mistakes and fixing them along the way—we came out at the other end not only with a much better feeling of how to do the thing, but also with a feeling of confidence that we could do it, which I think is important. I don't know how the rest of you felt, but I have no recollection of ever having been worried that we might not do it. I really believed that we could do anything if we were given enough money and left alone.

Tropp: Let's go back to what we were talking about: this attitude about having to get the job done. That was the atmosphere in this country during World War II. No matter which project I talk to somebody about, it was, "There was a war on, and we had to get the job done." That's the end of the answer; that's all there was.

Taylor: We started pretty close to the end of the war. In 1948 a lot of us were still in that frame of mind.

Everett: I think today you will find there is a lot more cynicism in people. It doesn't take the form of people thinking they can't do it; it takes the form that they don't think they'll be allowed to do it.

Wieser: I don't think the workers scared easily. I think some of the confidence, at least on my part, was related to keeping fairly close track of the competitive schemes. I wasn't a computer designer; I was an early customer. We talked earlier about the formidable problems that were faced with the development of the SAGE computer, but the fact that the technology and the ideas were so new meant that there was a lot of elbow room for improvement. In terms of competitive schemes, the University of Michigan's was not the only one. There were others; a big system called 414 was one of those. It used relatively mature analog technology. They had a bad set of problems that they had to cope with and they didn't have all of the new avenues of improvement that we had. They couldn't jump from electrostatic storage to core memory. They were stuck with what they had, and in a sense while



Robert R. Everett
"I have no recollection of ever having been worried that we might not do it. I really believed that we could do anything if we were given enough money and left alone."

414 was a more conservative undertaking, it was also a dead-end street. Today you wouldn't ever go back and try to make an air-defense system using either that technology or those components or the principles on which those competitive systems operated. They had problems in radar-network registration, problems in stability of analog tracking—all those technicians with screwdrivers trying to keep the thing working. You know, analog systems didn't work well even when they worked the way they were designed to work.

Benington: I think we ought to distinguish between the commercial world and doing business with the government. I am now dealing with the government but also a lot with the commercial world, and things in the commercial world in information systems are as lively now or more lively than they were in our day. If you look at the number of companies, large and small, the fantastic variety of projects, the numbers of failures, the number of outstanding successes, it's a very, very thriving world, and I think the country's doing very well in that regard. I'm glad the government isn't meddling too much with it.

Tropp: One thing that often comes up when you're talking about an advancement as important as SAGE is the retrospective look: "If I had to do it all over again, would I do it differently?" As someone who wasn't a part of it and only saw the Q-7 two weeks ago, I look at it and say, "Don't change a thing; in fact, don't move it out!"

Does anyone want to respond with a retrospective view of what you would do differently or what milestones you saw as forcing you to move in a direction that now you wish you hadn't gone in?

Benington: Before somebody does that, can I tell an anecdote? Jack Jacobs and I and some other MITRE officers went down to Fort Lee a couple of years ago where there is a SAGE center. It was very operational and a "gung-ho" activity. The colonel who ran it said it was the most reliable piece of equipment he ever had in the Air Force—one of the most successful

things he had seen; so maybe we shouldn't change too much.

Tropp: I don't think whatever replaces it could be as good, but I'm prejudiced now.

Shiely: I'm sure glad to find out at this stage that nobody was concerned about whether it would work or not. I'm going to be frank and say that I was terrified. It's a terrifying situation to have somebody come in and tell you that your direction center is just half as big as it has got to be or that the machine is just half as big as it has got to be.

Tropp: And the money is only a quarter as big.

Shiely: For example, the system originally started out dealing with the problem of vulnerability by putting the direction centers underground. The first one we started to dig was at Fort Dix, New Jersey. We found that the water table was about three feet under the surface. So we were faced with the problem of whether this had to be a submarine. Then we went up and took test borings at West Point, the second center, and found it was solid granite. It didn't take very much computation after that to figure what all these direction centers were going to cost, so those centers came out of the ground a story at a time. They also got reinforced more every time they came out. I think everybody is aware of the way military construction is programmed in the government system—how it is budgeted, defended, and even appropriated by separate channels. So you can imagine the problems that were involved in explaining to somebody why these buildings were suddenly becoming twice the size, becoming reinforced, and coming out of the ground—all this happening while the program was going on. We were building a building around a machine that hadn't been invented yet in a sense, and it certainly hadn't been built in the form that it had to be housed. I'm glad I found out now that nobody was really worried about it.

Everett: I think you're making a very good point. Those of us who were designing SAGE believed in it, and I don't know how we could have done the job if we didn't. But as the buyer of the thing, you had every right to be terrified. I was amazed at the time and I'm still amazed at the unflagging support of the Air Force. Truly remarkable.

Taylor: You might have been apprehensive, but you didn't let it show too much.

Shiely: The thing that I recall hearing about at that time was the support we got from the very senior people. A top priority for air defense was very, very new. This kind of priority went only to strategic programs at that time. The type of management structure we were allowed to establish also was restricted to very few programs; in fact, it was an experimental management structure that had been applied only to aircraft—and to strategic aircraft at that. To allow us to set up an office between the two acquisition commands—a joint project office involving the two commands—and allowing us the authority to direct activities in those commands was a tremendous departure from the normal operation of the Air Force at that time and took a tremendous amount of support and foresight on the part of people like General Thomas F. Power, commander of the Research and Development Command, and General Rawlings, commander of the Air Materiel Command. Full-time participation by the Operational Command for whom the system was being built was also an innovation that proved invaluable. We were required to brief the two commanders personally once a month on where the program was. It took us ten minutes—that was it. We were never given any direction, except to go ahead. Similar briefings were also given to the Air Defense Command.

Bright: I seem to recall that we were under severe funding restraints somewhere along in the program. As I remember, 23 direction centers were proposed. But because of funding constraints, Charlie Zraket and I and an air force colonel (whose name I don't remember) cut this back to 17 direction centers, without significant loss of air-defense capability. That, of course, had to be reviewed by our bosses. The cutback program was approved.

Forrester: How many were actually finally installed?

Tropp: I think 23 direction centers. Here we are extolling the virtues of the reliability of the Q-7. I went through the literature you sent me, Bob, and I keep finding names like Q-7A, Q-31, and Q-32. From 1959 on it seems that somebody kept wanting to replace it before it was fully installed, fully operational, with transistorized or more state-of-the-art technology. Each of these was dropped for a variety of reasons. The first generation, Q-7, is still operational today and will be until sometime in 1983. But why the rush to get rid of it before it's ever fully completed?

Everett: I don't think they were trying to get rid of it because it didn't work. The problem was its vulnerability to ICBMs. The new program being pursued actually never got built: the so-called super combat center, to be installed in hardened underground shelters. We were talking about a machine of a new generation with more capability, to be made out of transistors. A smaller number of such centers could do the work, which then would greatly reduce the cost.

Tropp: In the records, this isn't very clear. It looks like you're introducing a new airplane before the model has gone through its introduction.

Taylor: The buildings of the direction centers were formidable to look at. Bob Everett and I used to drive to Poughkeepsie and back almost every week for about three years. We had a little Chevrolet wagon that we burned out on that trip. One night we were coming home at about two o'clock in the morning and we were talking about what is important, what will be important, and where do we go from here. Bob said, "You know, Norm, you and I will be buried in some cemetery, and some guy will walk by those buildings and he'll say, 'What the hell do you suppose those guys had in mind?'"

Forrester: Some of our people were tired at the end of the week and occasionally chartered an airplane from Poughkeepsie back to Boston. It was very much contrary to air force contracting officers' principles, so they would come storming in to Nat Sage and say, "You can't do this," and Nat Sage would nod and say, "Isn't that terrible!" and take their document and put it in his desk drawer and never tell us about it.

SAGE "FIRSTS"[3]

Using Lincoln's farsighted initial design as a basis, the Lincoln-Rand team set about developing and implementing the first large-scale information system with capabilities so advanced that a quarter century later they would still be considered the current state of the art. Among the numerous firsts of SAGE:

- A fully real-time system.
- Servicing 100 simultaneous users.
- Acquiring live digital data from many sources.
- Routing data to many destinations.
- Using interactive graphic displays.
- Providing near-instant on-line access to a common data base.
- Having fault tolerance and "graceful" degradation.
- Incorporating a "hot backup" machine.
- Communicating digital data among a dispersed network of computers.
- Handling live operations and simulated exercises simultaneously.

And incorporating in its computer programs:

- Centralized system data structures and control.
- Modular, topdown system organization.
- Discrete structured program modules.
- Overlapped input/output and data processing.
- Simultaneous real-time and computational processing.
- Time-sequenced synchronous scheduling and interrupt for 90 subprograms.
- Centralized data processing with remote input/output and display devices.
- Comprehensive communications pool (compool) defining all global data in the program.
- Table-driven software for ease of modification.
- Built-in test, recording, and data reduction.
- Computer-supported system development and checkout.

[3] Printed with permission from Claude Baum, *The System Builders: The Story of SDC.* Santa Monica, System Development Corporation, 1981, p. 24.

Taylor: I remember coming back on that flight once, when the pilot turned to me and said, "Does anybody see the field down there?"

Everett: I remember a somewhat similar story. It shows you how different things were in those days. We needed more computer time, and somebody came up with the idea of using the ones on the test floor in Poughkeepsie. So we sent one of our staff members to IBM, and he worked out a deal with IBM to make the time available late at night. Then, as we pieced it together later, somebody he was talking with at IBM said, "Oh, by the way, this is going to cost some money." Our guy said, "That's all right." He came home and everything went fine until one day Harris Fahnestock came into my office shaking with fury. He had in his hand a bill for $1 million. I remember saying, "Harris, why are you so upset?" He said, "But you never got permission for this." I said, "Now, Harris, be calm. You know you would have given us permission. It didn't cost a cent more than it should. Now will you go away and pay the bill and not bother me anymore?" And he went away and paid the bill.

Tropp: Claude Baum, in *The System Builders: The Story of SDC,* lists numerous "firsts" for SAGE. The list, reprinted here, seems to be fairly heavily oriented toward software. Herb, do you want to react to it?

Benington: My initial reaction is that it is a pretty good list.

Tropp: Getting into firsts from a historical point of view is always a danger because the minute you say, "So-and-so did something first," you find somebody else who did it. But I think many of these things that are considered firsts in SAGE probably occurred first in Whirlwind, didn't they, Bob?

Everett: It's partly a matter of how you define it. Whirlwind in its Cape Cod manifestation did most of these things. It couldn't have done its job without them. I think that it's not just a matter of who does what first, but whether it gets used in some way. It seems to me one of the things that SAGE did was to spread the knowledge that these things could be done, as well as how to do them—throughout the organization and throughout the country.

Wieser: It may be buried in here some place, but I think marginal checking systems were very, very important. Probably SAGE had built-in automatic tests to a degree that previous systems had never had before.

Harrington: Nobody even tried to do it automatically that I know of. To some extent we still haven't.

Wieser: Marginal checking is not fault tolerance. It permitted the anticipation of a failure caused by grad-

ual deterioration instead of waiting for errors to occur. It was a very important part of both the Whirlwind and the SAGE computers.

Everett: I think this list kind of scants the hardware, but that's not surprising because it was written by software people.

Tropp: But from a software point of view, Herb, do you think it's reasonable?

Benington: I think it adds detail to the concept, particularly the last part of the list—the concept that I mentioned about how you must have a disciplined team to put together a large computer program. Many of these features allow that to happen. For example, one of my favorites was the communications pool, which was one of the great innovations in SAGE. A lot of people who are not using it today are sadder as a result, and they don't know it. With the communications pool, the individual computer programmer could access or change system data (i.e., information such as aircraft altitudes) without knowing many of the details of where the data was stored, when it was moved, or who else used it. In this sense we viewed the system as hardware, software, *and* data. We had the first "system-data managers"—they played a major role in the design of communications. Their updated view of the "communications pool" could be integrated into a newly modified system with minimum disruption to all the other hardware and software participants.

Everett: Yes, but the software business went downhill after SAGE in many ways.

Tropp: Mort Bernstein told me that the communications pool gets reinvented periodically, and I think that's the fault of those of you who did the software and did not document these things.

Everett: Or the fault of those who don't read about it.

Forrester: There were many things in software and hardware that were simply done in Whirlwind and SAGE with the attitude that that's what people would do if they were going to try to design the system. They probably did not think about the new ideas as being innovative, and there was very little emphasis on academic publication. Publication wasn't the goal of most of the people. So we have a situation where many of the firsts on Whirlwind and SAGE are not represented in the literature.

Tropp: I think another facet was that you people were too busy to bother writing papers because the group of people in the industry was small enough that communication was probably by word of mouth and personal contact. That's an environment that's hard for today's professionals to understand.

Herbert D. Benington "The communications pool was one of the great innovations in SAGE. A lot of people who are not using it today are sadder as a result, and they don't know it."

Forrester: However, the Digital Computer Laboratory did publish a quarterly report. It was a glossy-paper, halftone-picture, typeset publication that was distributed to maybe 250 addresses. It reached many people, but I presume one can't find it in libraries because it was not a recognized journal.

Everett: It would probably be very hard to put together a set, except in the archives.

Taylor: There's one item that's not on the SDC list because it's such a controversial item: the light gun. I've edited all kinds of papers in the last five years, and I give credit for inventing the light gun to Robert Everett simply because he told me he invented it.

Everett: If I knew it was that simple, I'd have invented lots of things!

Taylor: You never told me about anything else, so I think you must have done it. Right now you'll find that everybody in the business invented the light gun. Of course, it was invented in the Cape Cod System, which was the prototype for SAGE. We had some crude ones before that, but in the Cape Cod System it was really used. I think Everett ought to get credit for that.

Smith: Why did you invent it? What prodded that?

Everett: There are two answers to that. I "invented" the light gun, in the sense that I didn't know anybody else had done so. And I invented the use of light guns in digital computers. There are lots of cases like that; people had invented the photoelectric cell pickups for other purposes. But my recollection is that one day I came up from the basement where I was trying to make the storage tubes work. We were starting the first experiments in tracking airplanes, and we had the problem of designating a particular track, or particular spot on the tube. Someone had built a joystick for this purpose. As you pushed the stick one way or another, it closed some microswitches and fed a bit stream into a counter that could be read by the computer, which would move the spot accordingly. I looked

at it and said, "Why bother to do that? The computer knows where the spot is. Put a photocell over it and tell the computer that's the one you want." I went back down to the basement. After a while I came up again and was shown a working light gun.

Taylor: That's right. We used it on the storage-tube detection system at first to tell which spot was doing what.

Wieser: It made the conventional joystick obsolete for designating targets on a display tube.

Smith: Why was that superior? Why do it that way? Didn't you want to shoot him? Or did it seem such a good idea on the spot?

Wieser: Instead of steering a displayed circle around the tube face until it fit around the dot (target) you wanted, which was the conventional joystick way to do it, we just found an easier way to designate a particular target. That's all. It worked very well.

Bright: A great many spinoffs from SAGE went into the commercial market later. In the Bell System digital communications was in part a spinoff. It was beneficial in moving up the time frame. Many things from SAGE have resulted in more advanced development—more advanced applications. As I look around Bell Laboratories and Western Electric today, I still see people who were involved in the SAGE program. That won't be true very much longer, because we are getting to that age. But they have made valuable contributions to commercial applications of digital transmission and switching technology.

Forrester: One first in those machines was parallel, synchronous, clock-timed logic. Most of the machines under development in the days of Whirlwind were something else. They were serial, or they were asynchronous, or in some way they were not the parallel, synchronous, clock-timed logic that became the standard for computer logic. Many ideas that were in those machines survived. I believe that a larger percentage of original ideas in those machines survived than from other machines of the time.

Wieser: Another thing that I think we take for granted today is parity checking on all internal information transfers inside the machine. That was a feature of SAGE.

Crago: How many years was it before there was a commercial processor with a full-checked adder? You know, SAGE had a full-checked adder, and it was years before anybody went back and put one in a commercial machine. An invaluable aid in data integrity.

Tropp: I wanted to get into this whole area of the impact of SAGE, but before we do, Tom Smith mentioned a managerial question that he wanted to raise.

Smith: It seems to me from some of the conversations here that while there was this incredible amount of cooperation going on, nevertheless, some decisions had to be made and some options selected. I was wondering what kinds of mechanisms came into being for this within Lincoln, Western Electric, Bell, and, of course, the Air Force Project Office. You couldn't all dance around in sweetness and light. One thought I had in this discussion was: "To hear us talk here, these people almost weren't human. There must have been some knock-down, drag-out fights over genuine issues; that's the way history gets created." So I want to ask: how did all this stuff get implemented, starting from the design level and going on through?

Taylor: I'd like to speak to that, if I may. The first time it came to my attention was when we started to interface with the IBM people. They had a significant number of people on the job. I'm not talking about thousands, but 30 or 40 or 50. We had a pretty good design concept, but it wasn't easy to transfer that on the table, a piece at a time, so we could decide whether we were going to accept it or not.

What Jake Jacobs mentioned before was the vehicle we came up with called the Systems Office. The Systems Office was an unusual concept which I still try to use in my consulting work, because it was so effective in doing just what you said. First, we broke the problem into fairly small parts; in other words, if we had an arithmetic discussion, Jake would put a team together on the Lincoln view of the arithmetic discussion, and IBM would have just two people who worked exclusively on this until they came up with an agreed-upon specification. If that agreed-upon specification seemed to be harmonious, Jake would sign off on it and say that's what we're going to do. He might tell me about it or he might not, depending on whether we had ever discussed it before. If there ever was any degree of controversy between the two parties, we would have a larger meeting—but only when the degree of controversy was nontrivial.

This vehicle grew to be a very powerful management tool because we did not let anything go above a certain level unless it was necessary; that was a judgment that Jake largely had to make. Furthermore, when we started, we did not write a spec and hand it to somebody at IBM and say, "This is what you have to swallow." They had as much right to come to the table with a spec as we had. I remember when Astrahan came with, for instance, a set of primitives. In a computer you have 20 or 30 primitives—the orders you're going to build or design into the machine. I expected to have a real knock-down, drag-out fight on those primitives because what they had in their usual machines wasn't really suitable for SAGE, in my opin-

ion. Astrahan came with, I think, 34 primitives of which we accepted 28. There was a lot of harmony before we got to the point of controversy.

Forrester: Running through the program was an atmosphere of openness, a willingness to listen to differences. Also in the background of the program was the almost absolute control exerted by Lincoln Laboratory. In the contracts between the Air Force and the contractors, Lincoln was given the authority to sign the drawings. Lincoln had to be satisfied. MIT and the Lincoln Laboratory had a substantial role in setting up the air force coordinating office, specifying that it be headed by a general, and insisting on the degree of authority. A strong office was considered necessary so that there would be the authority to get rapid decisions.

Wieser: In today's system-development language you would call that process "configuration management." Jake would have been running a "configuration control board." In the midst of all this togetherness, harmony, and cooperation there was formality in the decision-making process. Accurate records were kept. People worked together when they had problems, and when they got a solution it was documented. It had to be, in order to pass it on to people who were to build the hardware. If you have one group designing a computer and another building it, you need good communications, which means formal communications.

Jacobs: I'd like to say a few words about what made the Systems Office successful. It was necessary to achieve consensus on the design from the air force organizations and the contractors, as well as consensus within Lincoln itself. It was important, therefore, to select a mechanism that was aimed at achieving agreement on what should be done. We wanted to create a technique that treated the organizations as equals, that would give each organization an equal right to propose what should be done. *Direction* was too strong a word, and *coordination* was too weak. The word that was chosen was *concurrence.* Concurrence implied that the organization involved in a design had veto power over the proposals affecting it. Thus all proposals were circulated to the affected participants for their review and comment. The Systems Office would do an analysis that spelled out the alternatives, suggest a choice, and ask the participants for their concurrence. The organization that did not wish to concur had the burden of proof; thus in order to deal with a nonconcurring problem, the organization would have to show how its design was preferable. If they succeeded, the process would be repeated. The thing that is most amazing to me is how small a number of nonconcurrences we were faced with. What had started out as an ad-hoc, informal procedure was eventually institutionalized in the so-called Technical Information Release, which became the order given to the ADES project office as to what we thought they should do. It is true that what we ended up with was a configuration-management technique, as Bob said, but in the beginning it was an experiment in group dynamics aimed at establishing a baseline design.

Taylor: It wasn't so hard when we were talking about computers because we had developed a good relationship with IBM. Furthermore, as Jay just said, there was a piece of paper. That wasn't true with Boeing Aircraft. We had no control over what they did. I remember having one session where we came push-to-shove once in a while. I remember saying to Jay, "If we don't get control of this, it's not going to work." We had a lecture on what is control. It turns out so many times in life that you don't really have control of very many things. So you have to control them either by impeccable logic or by the power of persuasion or just by staying power. We had lots of opportunities with Boeing and some of those other people to do that. The Air Force came in as a very important partner. In other words, we came to a dichotomy. There's no binary way that one is right and the other is wrong in some of these things. So we had to have the Air Force adjudicate some of those. There were a few tough ones, but I was amazed at how many we were able to solve at the working level.

Shiely: There's an important element to this. One thing we did have that was carried on in subsequent years was a mandatory, regular reporting meeting every month, as I recall. Representatives of every organization had to appear at that meeting and report on where they were. We had a master schedule that was broken down into all its parts, which everyone was supposed to follow. (We did have, at least, a hold on cost.) Everybody had to come, whether he liked it

John F. Jacobs
"We wanted to create a technique that treated the organizations as equals, that would give each organization an equal right to propose what should be done."

or not, and stand up and say, "Yes, I am on that schedule, I am doing that," or "No, I'm not and here are the problems I have."

Bright: It was a good mechanism for exposing the problems.

Shiely: The problems that were revealed were kept track of. In several there was a payoff, one way or the other. Either the programmers had to do something, or the equipment had to be changed, or the building had to be changed, or something of this kind. We had some very interesting discussions on a number of these things over the years; in fact, I don't recall that we ever had to go beyond New York for any of those, in spite of the impacts of some of them.

That was a very important management tool. It was almost impossible to keep the problem you were running across hidden for very long, because you had to stand up and speak to it once a month at least and show pictures. Believe me, everyone around that table asked some very penetrating questions, particularly if what the other fellow was doing impacted on what you were doing and you either suspected he was in difficulty or you were having some trouble. We had some very spirited discussions. But as a management tool, that regular reporting of a complex program of that kind, around the table, is something which was carried on and, I think, was one of the most effective tools we had. This was of course, on a much higher level than what we were discussing here. Many, many technical problems were resolved at the lower levels where they didn't impact on any of these overall items.

Forrester: There was a survival all the way into this SAGE system of attitudes that I trace back to Gordon Brown and Nathaniel Sage, Sr. (I do not know with certainty, but some said that the SAGE air-defense system was named after Nathaniel Sage.) Anyway, Sage was the director of the Division of Industrial Cooperation, the contracting office at MIT. He was an unusual person. He had been an army child in his youth and had lived on army bases around the world. He was a civil engineer by training. He had a high and well-justified confidence in his ability to judge people. There were people at MIT he would trust and support. Others he wouldn't trust farther than he could watch them. Gordon Brown, Stark Draper, and I were among the group that enjoyed his confidence and support.

The activities under Brown and Sage had an unusual information flow. There are two things about it worth mentioning. First, bad news flowed uphill. You know a lot of organizations where the reverse is true. There was no need to try to impress people with how well you were doing; instead, people higher up were there to help if there were problems. One had better bring up problems and get help before it was too late. Second, there was complete freedom in jumping levels of hierarchies. In the structure that ran from Sage, to Brown, to me, and to the various levels inside the laboratory, there was no hesitation on the part of anyone to jump over administrative levels to get information. Nobody felt disturbed by being bypassed. Access to information was part of the basis of confidence that Sage had in people. He would drop in and talk to people in the laboratory. He would form his opinion of whether or not those who worked for me seemed to know what they were doing. Also, people who worked for me could go directly to Sage or to Brown. This was just part of the environment. Problems flowing up the hierarchy when help was needed and direct access to information were carried over into the organization of the SAGE system.

Everett: How did all this seem from the other side? How did it look to you?

Bright: It looked to me like a very workable management system because it was the one that we in the Bell System had been employing for many years. That is, to get all the participants on a project together, let them expose what they were pleased about, and then get them to tell what they weren't pleased about. There was full freedom in the so-called phasing meetings to speak up. Speaking up was encouraged. Then there was the more formal part of it where each one reported to the coordinator of the project, which was Western Electric, on how their part of the project was coming along and whether it was in phase with the other elements of the system. This worked very well. The customer was there, all the contractors were there, and the coordinating organization had to make sure that there were full and complete reports on a month-to-month basis with lots of contact in between those formal meetings. My feeling is that it was good.

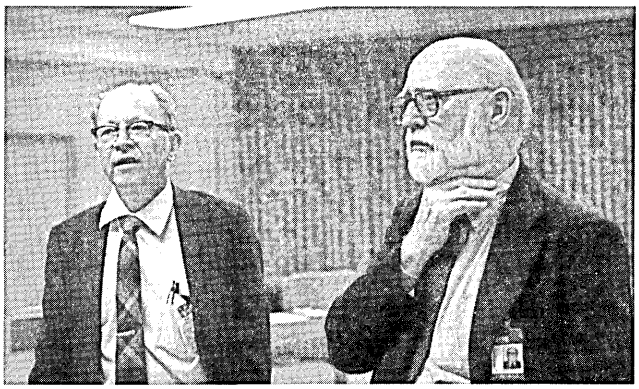
Everett: I thought it was good, too, although, as Jay says, Lincoln had the ultimate authority, and we couldn't be too autocratic. If things weren't normally done on that basis we could get away with it once in a while when we really needed to.

Kent C. Redmond: All of the contracts read that you had the right of concurrence. You just failed to define what happened on nonconcurrence.

Everett: I like the choice of the word *concurrence*, which was Jake's suggestion, and I thought a brilliant suggestion, implying a sort of iron-fist-in-a-velvet-glove approach to life.

Crago: As you say, it was never very clear what would happen if we didn't concur, so the problem never arose.

Taylor: I remember once you disagreed with me on something, and I said: "Would you like to sign that paper to get 45 hours out of this beast?" (I had a



Thomas M. Smith (left) and Kent C. Redmond

verbal agreement with the Air Force that we could obtain a mean free time to failure of 45 hours for each duplex computer. To do this, we needed control of circuits and components.) You said: "What paper?" I never told you there wasn't any, but that's all right.

Smith: I suppose one reason it all worked was the technical competence and the recognition by the parties that "Here are the technical options that everybody sees and here is what seems to be the way to go." Even when there was disagreement, you were still making your judgments on the basis of what I'll call "informed engineering considerations" instead of, say, some inadequately informed, high-level management consideration, like we ran into when they built a new library at our university and decided to reduce costs by eliminating separate air conditioning and humidity control for the rare-books room. They were incompetent to make that decision, but they were at such a high level of management that it never occurred to them to check down and find out. It seems to me that this was one of the things that you people were able to avoid by having control over what you were doing.

Taylor: Some of the time, but not always.

Benington: Life is serendipitous. I remember once a major argument as to how many monitoring lights we'd bring out of the Q-7 to show the operators and maintenance people what was going on inside the machine. The Lincoln Lab people wanted to have tons of lights so you could tell the status of everything, and IBM was dead set against it. Two or three years later the lab people couldn't care less about those lights, and IBM found them essential.

Crago: We did a fine job of defining the interface between the Q-7 and the FST-2. We defined the data format in every regard, except we failed to label which end was which. All the data was coming in backward. Do you remember that one? We had to make a complete change. It was well documented. It was just missing one thing.

Tropp: From the standpoint of IBM management, was there any kind of problem in terms of the number of people you were going to have to train to install these large computers, maintain them, monitor the software, debug, etc., with all these installations as they were coming on from 1958 to 1962—adding that work force in addition to the one that you'd already had to add to develop these computers?

Crago: In staffing the original design work, we were lucky because the company had hired all kinds of college-graduate engineers to be customer engineers in the field, and they wanted change. A lot of the cadre of people who came in to begin working with Lincoln Laboratory were those folks who were happy to get into design work. Yes, the number of people that it took to go out and install and maintain became tremendous—so much so that we finally approached the Air Force and said, "Isn't this a proper skill area in which you ought to be having your own people?" We gradually withdrew from the maintenance, and the Air Force took on the maintenance themselves. That turned out to have real benefits to IBM because just at that time, real-time systems became an important factor in commercial applications such as airline-reservation systems. The people who came out of SAGE had a background of staffing five shifts, seven days a week, 24 hours a day. This was not true of our regular maintenance people. The SAGE field engineers were a tremendous asset in jumping in and making these on-line systems work commercially.

Shiely: In the early days, one of the interesting things was that we began to get in the union business. IBM had all nonunion installers. We went out in the first center, and the electrical union got a look at this machine. We were immediately informed that we could only install this with electricians from the electrical union. We and Western Electric ended up having to chair the solution of this thing, but we had some interesting discussions over whether an electrician was properly qualified to install and check out this digital computer.

Tropp: Bob, can you remember the story about what happened at North Bay with exactly that situation? I guess they installed it at night, so the strike was a moot point?

Bright: We put S.P. (Monk) Schwartz of Western Electric on that problem. He was a great head banger, and he did get it solved, but I've forgotten how.

Everett: I remember those phasing group meetings, particularly because there was a certain amount of discussion about design, but mostly there were long stories about troubles with construction unions. I remember there was a problem with some of the air

conditioning in one of the centers; it was soft soldered instead of hard soldered. Month after month after month, the sad story of the soft solder came up. It taught me a lot about the problems of actually putting things in the field.

Crago: Didn't we also have one where a small boulder was left in the chilled water pipe for XD-1, and the first time the chilled water came on it came ricocheting through the pumps with dramatic effect?

Bright: One of the other management tools that we had was that we were responsible, along with the Air Force, for developing the funding profile and then defending that funding profile along with the Air Force Project Office. That seemed to have a particular influence on getting the job done in some cases. When money was diverted from one account to another, it required the cooperation of all the contractors. Construction funds, research and development, and other categories of money were involved.

Everett: The production funds were different.

Bright: It was all different—and the profile we worked up was done jointly with you.

Shiely: That had to be regularly defended. Anything that cost more money was just like today. It had to be brought up and defended, explained. So the old questions of performance, cost, and schedule were the kinds of things that were the controlling factors after we were actually putting it in. There was a lot of freedom as long as it didn't affect one of those three factors.

Redmond: I have a question that might have an obvious answer. Wasn't SAGE and its antecedents, in a sense, the first demonstrated use of the digital computer in a command-and-control situation? From which came, of course, ticket reservations, periodical subscriptions, inventory control, and so forth.

Bright: I think it was the first digital computer so used, but analog forms had been used earlier, if memory serves me right.

Everett: A lot of control computers were used in World War II. In fact, the fire-control computers had been used for a long time. It was the first electronic digital computer-based system, and it probably had aspects of command in it.

Tropp: One area is very hard to get into because it's so complex—that's the impact of the SAGE project in the years following its first installation. We can discuss it from the standpoint of software technology and hardware technology, people spreading, ideas spreading. For instance, what did IBM get out of this in terms of its role in building computers after it built the Q-7? How do you see that as affecting IBM's growth or development as a mainframe manufacturer?

Crago: A great many people were trained on that program who spread throughout the corporation—not only maintainers, but also designers. I can walk into any IBM plant in the United States and have former SAGE people introduce themselves and express pride in having been a part of the project. So that spreading of knowledge was immensely valuable. The people who went off the program early went on to the SABRE system for American Airlines and applications of that sort. I mentioned already the impact the maintenance folks had. I think the things we were taught and learned together about component reliability, marginal checking, and everything of that sort had impact that you can't measure in the machines that were designed later. There were always trade-offs. There were differences in the economics of what you could put in, but we all knew what the alternative could be, and that was invaluable. Of course, core memories were vital in the early machines, replacing the storage tubes. We also knew more about oil-filled 200-volt capacitors than we ever wanted to know, because we've never used them since except in power supplies. But there is no question that SAGE had a very real impact throughout IBM.

Tropp: Does someone else want to respond from another viewpoint?

Harrington: I'll take another viewpoint. From the viewpoint of the front end of the system, I would say that a lot of the radar signal detection and data handling and processing has shown up to be of enormous benefit to the FAA, in particular, over the years. The machine that the Burroughs Corporation built as a result of Lincoln's work, first known as the FST-2 (it has had other names) is in its second or third generation at various FAA stations. Some of the display equipment that I have seen used was derived from some of the work that was done at the Lincoln Laboratory. A lot of the early modem work was done at Lincoln to make use of existing telephone lines. Modems just didn't exist in 1948 or 1950, and there was a frequency-division system used that was terribly cumbersome. Digital transmission was relatively unknown and was very inefficient. Lincoln developed, I remember, a 1300-bit-per-second system that went on up to 2100 bits per second over a variety of telephone lines. In fact, MIT has a patent on that. That later became a contributor to the Bell System A-1 data system. The techniques have gone well beyond that now, but the early beginning of the modem concept really came out heavily from the need within the SAGE system to net radars to computers.

Bright: Of course, we learned how to build Texas Towers, too, but we haven't had very much use for that in the Bell System.

John V. Harrington
"The early beginning of the modem concept really came out heavily from the need within the SAGE system to net radars to computers."

Forrester: When we first started receiving digital data over phone lines and occasionally found noise disrupting the signals, we got in touch with the Bell System to ask what the specifications were on the lines. Their answer: "The lines are all right if you can talk over them."

Harrington: We had trouble with some of the early models of our modems, and we couldn't understand why we were having troubles. We finally ordered up a loop from the telephone company, which went from someplace in the Cambridge area up to Brunswick, Maine, and back again. It was about a 300-mile loop, and we had the beginning and the end of the line right there so we could look at it. I remember being shocked when we put 1000 cycles in one end of the line and what came out was not 1000 cycles, it was 1000 plus a few cycles. That was typical of the single-sideband K-carrier system.

Bright: We got 43-A1 later that did pretty well and that used the common user group.

Harrington: Anyway, we had to redesign all of our modems.

Bright: But in those days we had to have special treatment on the telephone lines in order for them to do a good job with digital communications. You had to adjust each one. But that was the beginning of the learning curve. We have grown out of that.

Benington: If the human ear had been phase sensitive, we might have had SAGE a year earlier.

Bright: Fletcher and Munson did something about the human ear, you'll recall. They drew a curve of its acuity, and we used that curve in engineering our telephone lines. When we were talking about voice transmission in those days, and/or slow-speed telegraph, it was fine for that but it wasn't applicable to high-speed digital communications without conditioning. That problem's been pretty much overcome now.

Everett: We found out a lot of interesting things about situations we thought were in good shape. For instance, we had to align the radars, and we therefore had to know where they were. We sent surveying teams out to find their locations. They surveyed all the radars, and we cranked up the system and discovered the radars weren't registering. We eventually discovered that sometimes there were mistakes of miles in the surveyed locations. The location given for one radar turned out to be in the middle of Long Island Sound.

Benington: Going back to what followed from SAGE, in my paper I talk about what I think came out of the software. The transfer came through people rather than through publications, and therefore some organizations benefited a great deal. IBM did. SDC had some problems in that regard because they thought they could be nonprofit and do it. They quickly discovered they couldn't do that sort of business as a nonprofit company, and so in the transition they lost a lot of good people. But I hear Jay talk about Nat Sage and Gordon Brown, and clearly they had quite an impact on him. As the junior member of this team, I ought to point out that there are 50 or 100 people whose Nat Sages and Gordon Browns are Jay Forrester. One person is Ken Olsen. I don't think Ken can talk about how he manages DEC—a very successful operation—without using Jay as his role model.

Bright: We haven't talked at all about tests and evaluations on the fully installed SAGE system. We were all very much gratified when McGuire was cut over, tested, and evaluated, with appropriate targets, and it worked. It was time for us to go back to the people who were not so enthusiastic about SAGE and say, "Here, we have something that will track an aircraft and guide an interceptor to the aircraft. And we can do it with multiple targets." What I'm getting at is that the test and evaluation teams consisted of Lincoln Laboratory, Bell Laboratories, Burroughs, IBM (the building people were there to make sure that everything was okay in the building), and Western Electric. The process, with that many organizations involved in test and evaluation, worked very well. I don't recall that we ran into any controversy as these things were cut over and put into operation. It seemed to me pretty smooth. It may be the pink haze of time that makes me think that now, but I can't remember an occasion that we didn't go ahead as a fully integrated team.

Tropp: What constituted the word I saw in the documents I've been looking at: *operational,* as in, "New York sector operational 26 June 1958, Boston sector 11 September," etc? Who defined "operational," and who set up the standards by which a sector either met or failed at that level?

Bright: With a lot of help, the Air Force did it.

Shiely: There were a whole series of tests that, if successful, established that the sector was ready for operational use. As each sector completed its testing,

it was turned over to the operational command for use. In the early sectors, it was a very difficult decision because although originally the Air Force thought that it could operate SAGE and keep the manual air-defense system in existence as a backup, it quickly became apparent that this wasn't possible. Therefore, the decision to accept a sector meant turning the air defense of that section of the country over to the SAGE system.

Tropp: How long did it take the New York sector to run the test process before the crew was comfortable with it? Before you were sure the computer was reliable, the software debugged sufficiently, and so on, to where you declared it operational? Was that a one-year process, a two-year process, a six-month process?

Bright: It wasn't one process. It was an incremental kind of thing. You went in, you checked out one element of the system, then you'd work through the other elements. It was a gradual integration of those elements. How much time that took I can't remember, but a systematic integration of various elements took place.

Shiely: It was measured in months, but it came down as you went down the system. The first one took so many months, then the next one took fewer, and so on. The later sectors were done quite rapidly. The first sector at McGuire must have been under test for several months, I think, before it finally went operational.

Bright: I'm sure it was months, but I have no feel for how long it took.

Shiely: After the first sector, it was a very quick, rapid program. A new sector was scheduled to go operational every two months. The reliability of the machines was superb, but several sectors were operational before we were confident that it was going to continue to operate at those levels.

Everett: The software had to be checked out, too. Although the programs were basically all alike, each one had to be adapted to the peculiarities of the sector, locations of air bases, radars, etc.

Bright: Each sector was different.

Everett: It turned out there were a lot of problems in the geometry of the sectors—the locations of all the radars and airbases, the characteristics of the aircraft, and so on. Then there was the process of keeping the program up to date. There was a new program that went out at regular intervals. I've forgotten what those intervals were.

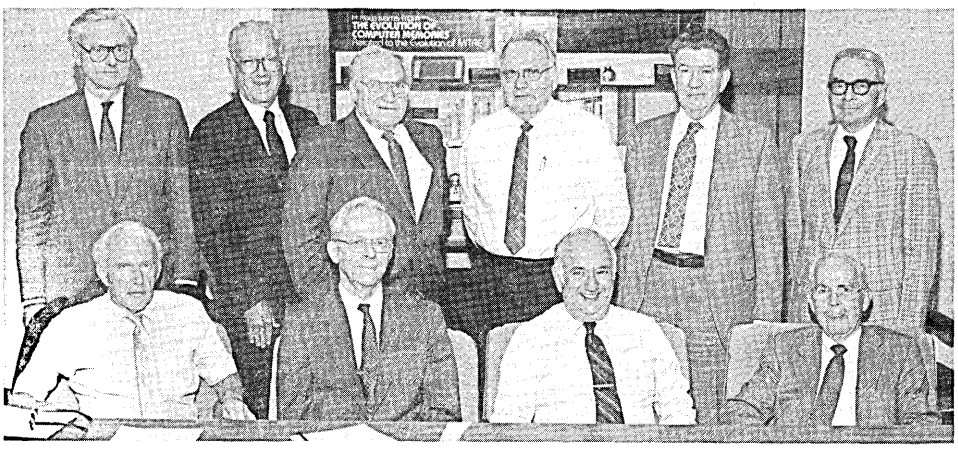
Benington: They called them models, and it was about every six months. We had packages that were sub-models, but SDC, at the end, had a production system that had four groups in it. There were people in the field, there were people at home to service the field, there were people who produced the production program, and there were development people who did the equivalent of the Cape Cod. In order to put BOMARC in, you'd do it first in a development model—you'd know what you were talking about—then you'd production spec it, give it to the field people, and install it—2000 people for the entire process.

While I was with SDC over a period of four years, we missed only one schedule at McGuire by three weeks—and the colonel at McGuire was furious with us. Everything else was done on schedule.

Everett: For those who don't know it, there are, I believe, six SAGE centers still running, and they're to be shut down in 1983. By the end of 1983 they will all be gone.

Tropp: 1958 to 1983—that's not a bad life span.

I want to thank all of you for participating in this discussion today. I particularly want to thank Bob Everett for hosting our gathering and for allowing me the privilege of being able to share in this occasion.



Bottom row, left to right: Jacobs, Forrester, Everett, Bright. Top row, left to right: Benington, Harrington, Taylor, Crago, Wieser, Shiely.

Reliability of Components

Foreword

Because this issue of the *Annals* focuses on the SAGE system, the following excerpt from Christopher Evans's 1975 interview with Jay W. Forrester is timely. (Another excerpt from this tape appeared in the last issue of the *Annals*, Vol. 5, No. 3, pp. 297–301; see Vol. 3, No. 4, p. 417, for details on Evans's "Pioneers of Computing" series of tapes.)

The pioneering series of first-generation electronic computers established the environment, architecture, and fundamentals from which subsequent developments emerged. In fact, some would argue that except for exciting new technology (transistors, chips, printed circuits, etc.), nothing has really changed.

First-generation computer designers and builders were plagued by many new problems in order to get their devices up and running. One overriding concern was that of component reliability, particularly as these pioneers charted new territory. They were concerned with how long the computer they were building would run before some component failed. When you think of the vacuum-tube technology of that era, you can't help but be impressed by the amount of productive work that was done. In fact, if your reaction is like mine, you are awed when you learn that the AN/FSQ-7s in SAGE have not only been operational since 1958, but have an "up" time of 99.83 percent. Can we say this about most of the computers being sold today? How many of them will still be operating in 20 years?

The Q-7 was a direct descendant of Whirlwind, and it is on the Whirlwind project that people like Gordon Brown, Bob Everett, and Jay Forrester set the standards and created the designs that led to the Q-7 and its extraordinary record over the past quarter century. In the account that follows, Forrester speaks directly to the subject of reliability.

—Henry S. Tropp

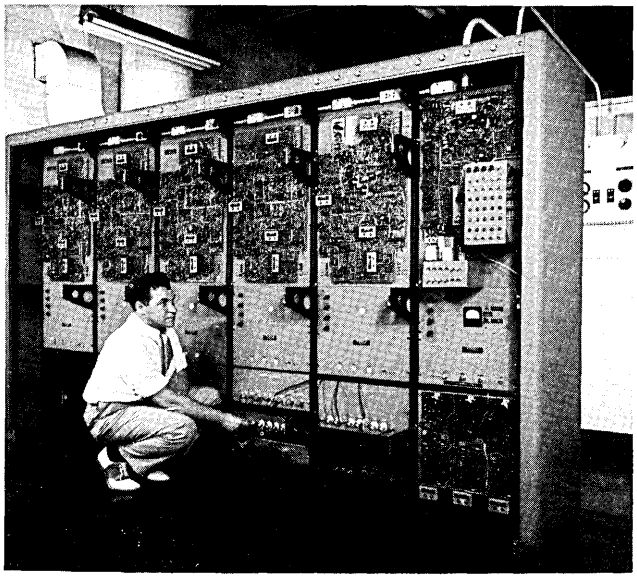
Editor's Note

Jay Forrester is one of the great pioneers of the computer business: director of Digital Computer Laboratory at MIT, builder of Whirlwind, inventor of core memory, and chief engineer for SAGE. His accomplishments are too numerous to mention, but this brief excerpt from an interview shows us something of his attitude toward design.

Jay W. Forrester The Whirlwind I computer was conceived in early 1947. The block diagrams, as eventually executed, were completed by Robert Everett in the spring of 1947. We began to develop the prototype circuits and to make various tests from 1947 on into 1949. At that time there was almost no knowledge of the nature of noise in electronic circuits. There was no certainty that if you wanted to carry out computations at a megacycle rate for days at a time without error, spontaneous random noise wouldn't be in circuits that might interfere with reliability. Almost nothing was known about the life of vacuum tubes beyond 500 or 1000 hours. Everything having to do with reliability and long-term performance had to be explored from the ground up.

In the process of doing this we developed a five-digit multiplier—a first step toward the Whirlwind computer—that we could put on life tests repeatedly solving a specified multiplication, automatically checking the answer, and counting the number of times that it would make a mistake over very long periods of time. We found, for example, that there were missing cycles—switching transients on the power lines that would feed through the equipment. We eventually had to isolate our equipment completely by putting mechanical motors and our own synchronous generators between the power system and the electronic computer to keep streetcars and elevators from introducing an occasional error. After we'd taken all of these precautions we had a device that would run for weeks at a time without a compu-

Adapted from "Pioneers of Computing," Tape 4, Science Museum, London, with permission. Science Museum London copyright.
Categories and Subject Descriptors: K.2 **[History of Computing]** —*hardware, Whirlwind.* General Terms: Design, Experimentation, Reliability. Additional Key Words and Phrases: core memory, MIT.
Photograph courtesy MITRE Archives.



Norman H. Taylor at the five-digit multiplier in 1948.

tational error. We began to show that there was a possibility of building the kind of reliable electronics that our objectives required.

Christopher Evans *Roughly, when was the five-digit multiplier completed and working reliably?*

J.F. It was completed in late 1947 and was used for quite some time after that—certainly through 1948 and probably 1949.

C.E. *What size was this device?*

J.F. It was probably 8 feet high and 10 or 12 feet long. It consisted of five parallel registers that would multiply together two five-digit binary numbers. Now the equivalent circuitry would be put on a chip smaller than a match head.

C.E. *So, now you had this thing completed. Were there any doubts about the way in which you should construct Whirlwind I itself?*

J.F. There are always doubts in any new pioneering effort. The Whirlwind I computer was a 16-binary-digit machine which many people looked upon as too short a register length to be of any practical use because the scientific problems and the problems of interest to mathematicians were thought to require a 30- or 40-binary-digit length. We chose 16 mainly to hold down the size and the complexity of the machine. We looked upon the Whirlwind I computer primarily as an experimental machine—again, to test out the feasibility of a computer. But many people expressed doubts about register length. There were grave doubts about the costs we were encountering in order to achieve the reliability we were after. It was a continuous battle to sustain the necessary funding to carry on the work. People were unconvinced of the idea that a computer could substitute for experience and judgment and decision making. This has always been and still is an issue that produces skepticism, doubt, and a feeling of uneasiness. That was particularly true in the late 1940s when no one had ever seen it happen.

C.E. *Could you say something about the hardware and the software of Whirlwind I?*

J.F. Whirlwind I probably contained more features that survived into today's computers than any other machine of its time. It was a machine that had parallel synchronous logic, meaning that the digits were transmitted in parallel, and they were timed and controlled by a central clock. The machine at the Institute for Advanced Study in Princeton was also parallel, but it used a nonsynchronous logic in which a particular operation ran its course, and at the end of its completion it triggered the next step. It was a different kind of control logic.

We had in the Whirlwind computer the first cathode-ray-tube display that was controlled by the machine itself. We had the first man-machine interaction through a cathode-ray tube in which the cathode-ray tube would display an output, leaving some sort of decision or question for an operator, who could then use a light gun, which he would point at the cathode-ray tube. This would tell the computer what the operator wanted to do and with respect to what part of the display.

Whirlwind I had a marginal checking system in which one could, under the control of the machine itself, alter the voltage on the screen grids of vacuum tubes that would vary the gain and move the particular set of vacuum tubes up or down with respect to their normal range of operation. The computer itself at the same time ran programs to test to see if the tubes still carried on their functions without error. We could thus anticipate deterioration of any component that was occurring gradually. This was perhaps 90 percent of ultimate failures, so it meant that the marginal checking system gave an additional factor of ten to the in-service reliability of the machine.

We had done two things to increase the life of vacuum tubes. Vacuum tubes were thought to have about a 500-hour life. If one is going to put 20,000 vacuum tubes in a computer, it is quite intolerable to have such a short life (a little arithmetic shows that the machine would run only a few minutes at a time). First, we had discovered the primary cause of short life in vacuum tubes. We found that tubes were failing from an apparent loss of emission caused by the

building up of an insulating barrier on the nickel core of the cathode, which produced a self-bias in the tube to cut down current. It was not an inability to emit electrons, but instead a self-bias that simply shut off the flow of electrons. The solution turned out to be the omission of the silicon that had been put in the nickel to make processing easier. This by itself raised the effective life of vacuum tubes from 500 hours to 500,000 hours. The marginal checking gave another factor of ten on that, so the effective life in terms of failures per 1000 tubes per 1000 hours had gone from about 500 hours to 5 million hours. This was quite sufficient to bring vacuum-tube reliability up to a standard that was not met by transistors until 10 or 15 years after they were first invented and were being put into computers.

The best-known innovation that was put into the Whirlwind I computer was the magnetic-core memory—the random-access magnetic-core memory—which was developed in the search for reliability. The Whirlwind computer design was laid out initially in 1947. We felt there was a need for a storage more reliable than the so-called Williams tube, which many people were trying to use at that time. We developed a special storage tube ourselves that had much higher signal levels. Also, the stored binary digits were dynamically self-sustained by a second electron gun. One electron gun was a reasonably ordinary cathode-ray beam that was used to find a spot on the face of the tube and to energize it and to read it. The other electron gun produced a flood of low-energy electrons, with grids at the storage surface and a cathode potential such that electrons flooded the storage surface in a way that sustained either of two stable states. The electrons actively regenerated either the low-potential or the high-potential stored spot. This worked well in principle—it worked well in individual tubes—but it relied on a hot cathode and on the secondary emission characteristics of the storage surface. Both of these are notoriously difficult to keep operating properly. The tube was subject to many practical problems that shortened its life. We were lucky to get a month of life out of such a tube. The tubes stored 1024 bits and cost about $1000 each. That was a cost of $1 per bit per month for high-speed storage. The economics of keeping a machine in operation with that sort of storage cost were entirely at odds with our objectives.

Jay W. Forrester
Systems Dynamics Group
MIT
E-40-294
Cambridge, MA 02139

SAGE at North Bay

Editor's Note

The most obvious characteristic of a SAGE center is its size—courtesy of vacuum-tube technology and emphasis on reliability and maintainability. I hope the readers who get this far have gained some feel for the size of SAGE, but its true character can only be felt by those who wander around inside a SAGE center. Whirlwind had somewhat the same feel, though much smaller, because the cable racks were exposed.

I remember during the design phase of SAGE that when I visited Whirlwind, I had a warm, relaxed feeling walking around inside it, while walking around inside the FSQ-7 prototype at Lincoln filled me with anxiety. I thought for a while that the difference had something to do with the personality of the two machines, but sometime later I found the Q-7 also gave me a warm feeling, so it must have been me after all.

Several years ago, a number of us at MITRE decided we should visit a SAGE site. None of us had been in one for many years. I don't know what we expected—a feeling of decay perhaps, dispirited operators, or that wonder at how small things seem when we revisit places we knew long ago. I could not go at the last moment, but the group returned, their eyes shining with excitement. "It looks brand new," they said. "It's clean as a whistle." "The operators are young, enthusiastic, and proud of SAGE." "It's working just as it was supposed to—downtime a few hours a year." "It looks as big as ever."

Unfortunately, I missed the trip to North Bay also, but I gather that the visitors were impressed. We are fortunate to have a firsthand report.

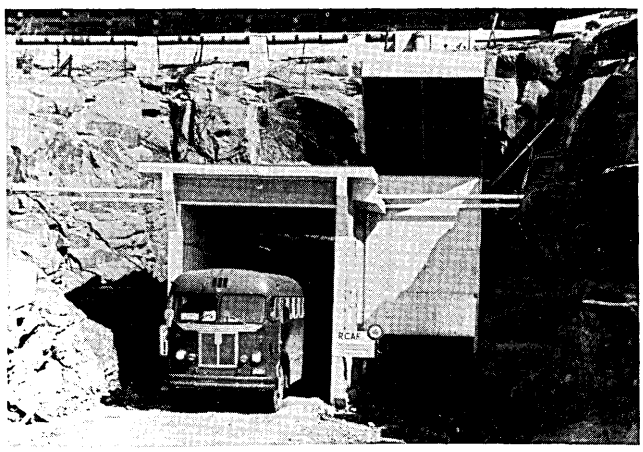
In a recent issue of ACM's *Communications* (Volume 26, Number 2, February 1983, pp. 118–119), Gordon Bell described a visit that I was privileged to be part of to the NORAD site at North Bay, Ontario, Canada. The main purpose of the trip was to view and learn about the SAGE computer, the IBM AN/FSQ-7. When Gwen Bell, director of the Computer Museum, first told me of the plan to organize the trip, my original

Categories and Subject Descriptors: K.2 [**History of Computing**] —*hardware, SAGE, software, systems*. General Terms: Design, Reliability. Additional Key Words and Phrases: North Bay, U.S. Air Force.
Photograph courtesy USAF Electronic Systems Division History Office.

reaction—in addition to a strong desire to be present—was amazement that a first-generation computer was still operating. It turned out that most of the SAGE sites were still using the Q-7, but the U.S. Air Force was in the process of replacing them with Hughes computers. The North Bay site was scheduled for replacement in June 1983, and I was anxious to see the Q-7 in operation before it was removed. (The previous spring I had been taken to NASA Ames twice to see the ILLIAC IV, and both times I found it sitting vacantly with its insides spread all around the floor. Now it's too late; the ILLIAC IV is gone—to the Computer Museum.)

Before the trip I read all I could about SAGE (including the manuscripts for this issue), but mostly I reread material on Project Whirlwind. I knew that Whirlwind had been the basis for much of the computer thinking that went into the design of SAGE. Careful engineering and an emphasis on reliability were characteristics of Project Whirlwind under Jay Forrester, Gordon Brown, and Bob Everett, and I knew that these were continued on into the SAGE project. But this knowledge didn't lessen my astonishment that a 1958 first-generation computer of this size and mission, with 55,000 vacuum tubes that constantly needed replacement, was still operational.

On Friday, October 8, 1982, our group boarded two aircraft at Hanscom Field and flew to North Bay. There we were taken by bus to the NORAD site, were cleared through the gates, and saw in front of us an opening to a cave in a mountain. We had been told that we were going to be 600 feet underground, and I had had a mental image of getting into an elevator and being whisked to the subterranean site. Instead, the bus drove into the cave along a narrow 6000-foot tunnel carved from solid rock. The headlights of the bus were angled sideways, not forward, because the tight fit of the tunnel sides was more crucial than what was ahead. We were awed by the eerie trip and the enormity of the site—inside the mountain is the three-story 150,000-square-foot building that houses the Q-7.



An entrance to North Bay's underground SAGE site in 1963.

It turned out that it really wouldn't have been necessary for me to do a lot of preparatory reading. Members of the site staff gave us an excellent briefing. They explained the hardware architecture, the operation, the software, the mission, and every aspect in great detail and clarity. A scale model of the structure showed all of the rooms on each level, with their purpose and contents clearly identified. We spent the next two hours looking at the Q-7 and all of its components and peripherals—bay after bay of core racks, marginal-testing racks, racks of spare pluggable units, etc. We were allowed to work at one of the consoles where they were running some flight-information simulations. I used a light gun on a radar spot and then read the identification of that particular aircraft on the screen.

Near the end of the tour, Gordon Bell talked one of the officers into taking Gwen Bell, Gordon, and me out of the wing we were in, into a large hole in the mountain between wings of the building that housed the monster-sized diesel generators that could power the installation—marine diesels that could be reversed without tearing themselves apart. When it was time to go back, our guide didn't know the digital combination that opened the door on our side, so there was a brief moment of panic as we realized that we were dependent on someone on the other side hearing the bell we rang and opening the door. Someone did hear it, however, and on rejoining the group we learned that the following Monday was a Canadian holiday; it might have been Tuesday before we were rescued.

Another purpose of the trip was for Gwen Bell to decide what components she would request for the Computer Museum collection before the Q-7 was cannibalized and destroyed. The museum's new quarters in Boston (see "News and Notices" in this issue) will be able to house huge parts of the computer. Some pieces are small; for example, there are six-tube and nine-tube pluggable units—certainly choice items. When we got back on the bus, we noted that a member of the group had walked out with a six-tube unit. We told him that he would never get it through customs, but the next day he told the officers it was scrap, and they casually waved him through.

On the return trip, Gordon Bell and I spent a good deal of time discussing what we had seen and reflecting on it in terms of the contemporary scene. At one point he said, "Do you realize that we walked inside a computer? There is no way you can do that with any computer built today." I absolutely agreed. Three months later to the day, I stood inside a Cray I and felt the cool air blow up around me.

This account has deliberately omitted the technical details of what I saw and learned. Gordon Bell's *CACM* report gives them magnificently, and the papers in this issue give the specifics of the AN/FSQ-7 computers in general. A few numbers continue to awe me, though: 55,000 vacuum tubes, 135,000 diodes, 13,000 transistors, 7000 pluggable units, 12 drums, all of that core memory, and still 99.83 percent availability as a simplex and 99.97 percent as a duplex!

Henry S. Tropp
Department of Mathematics
Humboldt State University
Arcata, CA 95521

A few paragraphs from Gordon Bell's *CACM* note:[1]

Bob Everett's paper on the SAGE computer was published in 1957, and the machine was operational in Canada in 1962. The machine created many patents as by-products, including perhaps the first associative store (using a drum). The machine is duplexed with a warm standby (I mean warm, since the duplexed machine uses about one megawatt of power to heat 55,000 tubes, 175,000 diodes and 13,000 transistors in 7000 plug-ins!). The 6-microsecond, 32-bit-word machine has 4 × 64K × 32-bit core memories and about the same memory in twelve 10.7-inch diameter, 2900-rpm drums, six of which are for secondary memory. There is no use of interrupts, and I/O is done in an elegant fashion by loading/unloading parallel tracks of the drums with the external world completely in parallel with computing. That is, the I/O state becomes part of the computer's memory state. A single I/O channel is then used to move a drum track to and from the primary core memory.

The main I/O is a scan and height radar that tracks targets and finds their altitude. The operator's radar consoles plot the terrain and targets according to operator switch requests. The computer sends information to be plotted on 20-inch round Hughes Charactron (vector and alpha gun) tubes or displayed on small alphanumeric storage tubes for supplementary information. Communication lines connect neighboring air-defense sectors and the overall command. The operating system of 1 million words is stored on 728 tape drives and the drums.

The computer logic is stored in many open bays 15 feet to 30 feet long, each of which has a bay of voltage marginal check switches on the left side, followed by up to a maximum of 15 panels. The vertical panels are about 7 feet high by 2 feet wide and hold about 20 plug-in logic units. The separate right and left half of the arithmetic units are about 30 feet each or about 2 feet per bit. Two sets of the AMD 2901 four-bit microprocessor slice would be an overkill for this 32-bit function today. The machine does vector (of length 2) arithmetic to handle the coordinate operations. The room with one CPU, drum, and memory is about 50 feet × 150 feet, and the room with two CPU consoles, tapes, and card I/O printer is about 25 feet × 50 feet. The several dozen radar consoles are in a very large room.

[1] Reprinted with permission from *Communications of the ACM*, Vol. 26, No. 2, February 1983. Copyright 1983, Association for Computing Machinery, Inc.

Epilogue

As you read this, it is probable that the last SAGE center has shut down and its multitude of parts is on the way to various museums, office shelves, and scrap heaps. It is too bad that most of the things that electronics people build these days have short lives—much shorter than our own. It is our own fault, of course. If we did not improve things so rapidly, they would not become obsolete and end up on scrap heaps quite so soon. Still, it seems to me that engineers had somewhat greater satisfactions in years gone by. A dam, a bridge, even a power plant will probably outlast its designer. As I drive through Cambridge, I pass a filter plant designed by my father 50 years ago. It gives me a good feeling.

Yet SAGE has left behind a legacy of ideas and organizations—a different kind of legacy from a Hoover Dam, but a legacy nonetheless. SAGE was the first computer-based command-and-control system. There are now SAGE-like systems all over the world. The Lincoln Laboratory at MIT, the MITRE Corporation, the System Development Corporation, and the Air Force Electronic Systems Division are a few of the organizations created to build SAGE that are still thriving.

SAGE trained hundreds of digital-system design engineers, thousands of computer programmers, and thousands of digital-computer field engineers who gave great impetus to the new field of digital computers.

Computer-driven displays, on-line terminals, time-sharing, high-reliability computation, digital signal processing, digital transmission over telephone lines, digital track-while-scan, digital simulation, core memories, computer networking, duplex computers—there is an endless list of things done first by SAGE.

All of these things would have been done eventually, and many of them were unrecognized and have since been reinvented, but I think it is clear that SAGE gave the computer field a real boost forward and left its mark on digital computers and on human society.

Robert R. Everett
Editor, Special Issue