Technical Report CS81020-E

THE DEVELOPMENT OF SOFTWARE ENGINEERS: A VIEW FROM A USER

Walter P. Warner
Naval Surface Weapons Center
Dahlgren, Virginia

Richard E. Nance
Virginia Polytechnic Institute and State University
Blacksburg, Virginia

15 December 1981

*Invited paper for the 1982 National Computer Conference, Houston, Texas, June 7-10, 1982.

ABSTRACT

Like many organizations, the Naval Surface Weapons Center (NSWC) has recognized a tremendous growth in the use of computing resources. How NSWC focused attention on the crucial role of software development technology and devised a plan for dealing with the scarcity of competent software development personnel is the subject of this paper.

The necessary knowledge areas for software engineering are identified in discussing the academic requirements and some conclusions arrived at during the deliberations are mentioned.

KEYWORDS: software engineering, education, knowledge areas, training programs.

Computing Reviews Categories: C. 2. 0; J. 3. 2

INTRODUCTION

Like many organizations, the Naval Surface Weapons Center (NSWC) has recognized a tremendous growth in the use of computing resources. This growth is reflected in several statistics; a simple indicator is the increase in users of the central main frame from 1350 in 1978 to approximately 1800 in 1981. The development of computer software delivered to the Navy and Marine Corps estimated to consume 610 staff-years in 1979, was projected at 640 staff-years in 1980. Such rapid growth has surfaced several difficulties, but the overriding problem is the scarcity of software development talent. How NSWC focused attention on the crucial role of software development technology and devised a plan for dealing with the scarcity of competent personnel is the subject of this paper.

Aware of the Bureau of Labor Statistics (Reference 1) estimate that the need for software development personnel is to double between 1978 and 1981, NSWC conducted an informal survey of internal requirements. The results were startling: an addition of 117 to the workforce of 2084 professionals in FY 1981 and an addition over five years of 373. The National Science Foundation prediction of a 3.5:1 ratio of positions to master's and bachelor's graduates in computer science clearly acknowledges the global nature of the problem observed by NSWC (Reference 1).

The NSWC approach to the problem of scarcity of software development talent should not be viewed as a preconceived insightful long-range strategy. Admittedly, the plan has evolved, and this description of the evolution takes the following order:

(1) a background sketch of the NSWC showing its early and long-standing role in Naval computing efforts and software development;

(2) a picture of the current weapons system context, in which the software development task is conducted;

(3) the NSWC initiatives that collectively define the plan; and

(4) the conclusions arising from the various initiatives that have exerted significant influences.

NSWC BACKGROUND IN SOFTWARE DEVELOPMENT

The history of software development at the Naval Surface Weapons Center began with the advent of large digital computers for laboratory usage in the late 1940s. Because of its longstanding mission responsibility for the numerical data required to aim, target, or control Navy weapons, the Naval Proving Ground (combined with the Naval Ordnance Laboratory to form NSWC in 1975) was the first Naval activity to have a large-scale computer. Correspondingly, the Center was the first Navy activity to develop and support software for Navy deployed operational digital systems, beginning about 1960. With the continually evolving Navy requirements for digital computer applications, the Center has sustained its leading role in digital computer applications and software development expertise.

The Center is responsible for the complete weapons control software package development, testing, and operational support for the Fleet Ballistic Missile Systems POLARIS, POSEIDON and TRIDENT. It has provided the wrap-around simulations and facilities for testing the digital fire control programs for the TARTAR, TERRIER, and TALOS surface missile systems. Similar support has been provided for the Mk 86 and Mk 92 gunfire control systems. In connection with the Navy's Gunnery Improvement Program, the Center has developed the software for the Mk 68 digital fire control system for 5-inch guns.

The Center has pioneered the application of minicomputers for Fleet electronic warfare (EW) systems and ELINT processing systems. These digital systems have enabled orders of magnitude advancements in processing quality and quantity, in data response time, and in overall Fleet EW effectiveness. Two current major examples are the development of the Airborne ESM Data Analysis Systems and the support of the AN/SLQ-32 EW Countermeasures Suite. The Intelligence Analysis Center for the Marine Air/Ground Intelligence System (MAGIS) and the

shipboard Intelligence Center for LHA and CVV installation are additional examples of systems developed in-house by the Center. These are the first major deployed intelligence database oriented systems incorporating Navy standard computers. Among the large software systems developed on general purpose computers are the TRIDENT Advanced Weapon System Simulation and CELEST (a satellite orbit determination program).

COMPUTING SUPPORT FOR NAVAL WEAPONS SYSTEMS

Operational software might also be described as software for "embedded computer systems". The principles and techniques underlying the software development task for large, complex systems apply regardless of the applications context. However the enclosing system and the application often impose constraints and limitations of time (time-critical requirements) and storage that are shared only by the most challenging general purpose programs (e.g., operating systems, run-time control systems, etc.).

The typical Naval weapon system is becoming extremely complex. The AEGIS weapon system, for which the Center has life cycle support of the software, is an illustrative example, see Figure 1. Specific configurations for different ship classes may have different subsystems, but most of the subsystems, shown pictorially around the periphery, contain embedded computers. The total software development task includes:

(1) the operational software for the embedded computing support of the weapons subsystem,

(2) the integration software enabling the necessary communications among subsystems,

(3) the control software by which the weapons subsystems function as a total system, and

AEGIS SHIP COMBAT SYSTEM -- BASELINE

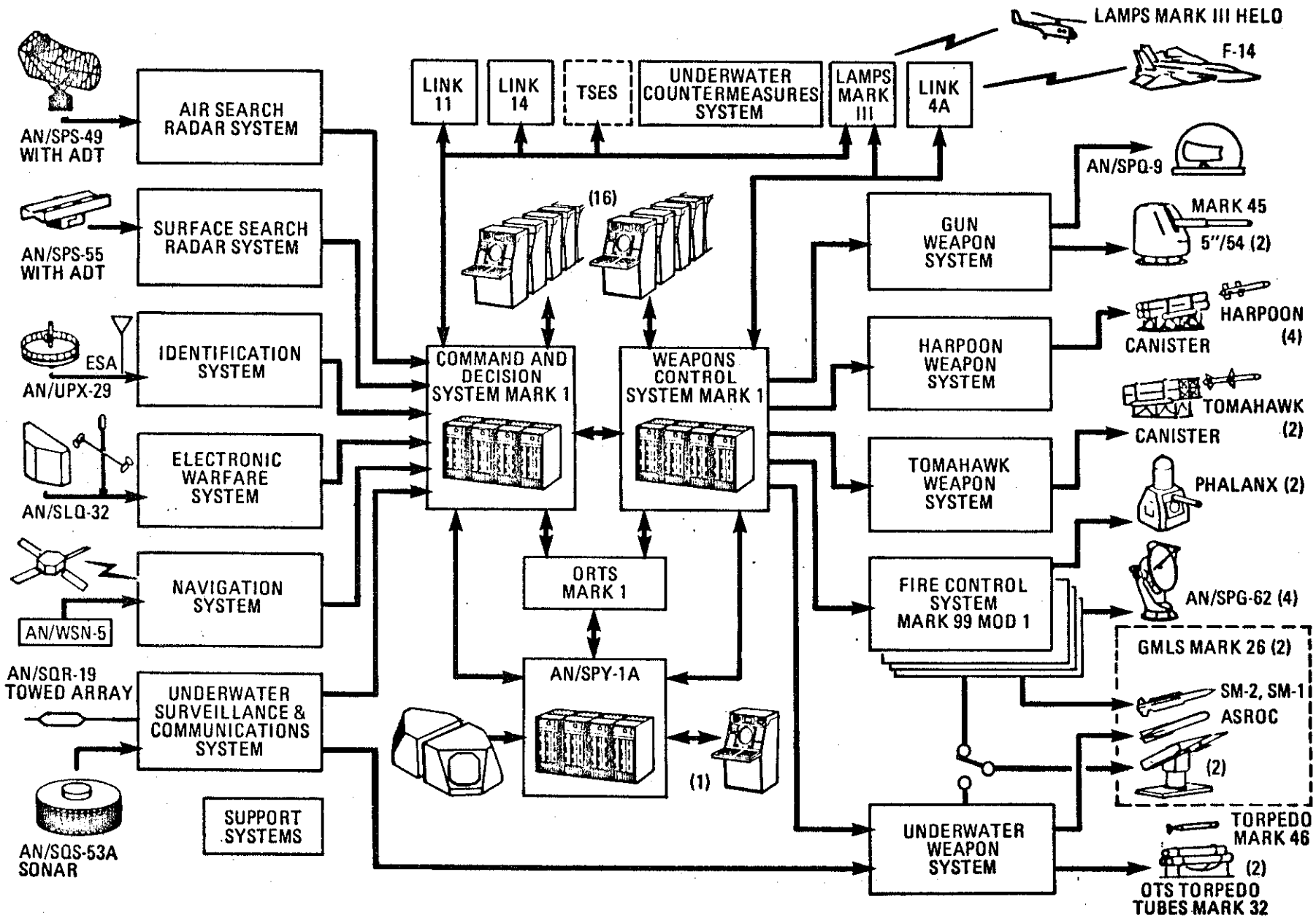

Figure 1

(4) the life cycle support software necessary for testing, verification and validation, and maintenance and enhancements.

The magnitude of this task is underscored by the simple statistic that 45 computers are required in some AEGIS configurations.

NSWC INITIATIVES

The evolutionary plan has proceeded primarily through the work of three committees.

(1) A committee that developed a computing technology seminar for top-level management. This committee consisted of an NSWC middlelevel manager with software development experience, an industry representative and a university faculty member.

(2) A second committee that examined the required background knowledge for software development at NSWC. This committee identified the necessary knowledge areas, evaluated their relative importance and assessed the utility of on-the-job training, in-house or academic courses in each area. This committee included experienced project managers and a computer science faculty member.

(3) A third committee, comprised of representatives from technical departments responsible for software development, that explored the alternatives for developing the needed expertise. This committee addressed formal academic programs, internal training and employee development possibilities.

COMPUTING TECHNOLOGY: A SEMINAR FOR SENIOR EXECUTIVES

A seminar, consisting of talks by top experts in the country, was conducted by NSWC (Reference 2). The purpose of this seminar was that computing technology in general and software development in particular would be better understood at all levels in the organization. The historical development and the perceived future technology, integrating the hardware and software evolution, was presented. Other topics were the software development process, distributed computing and computer networks, the regulatory environment, and information support systems for management. This seminar, presented in a 3-4 day concentrated format, has been given to all managers from the Commanding Officer and Technical Director to first line supervisors and also to two groups of Naval Flag rank officers (admirals), who are some of the NSWC sponsors.

IDENTIFYING THE ACADEMIC PREPARATION

An NSWC study was made to determine the best academic preparation for the kind of software engineering performed at NSWC (Reference 3).

Needs specification

The participants in this study could not come to an agreement on a definition of a software engineer; therefore, we began by identifying the knowledge areas important to the development of the Navy systems for which NSWC has, or could have, responsibility. These knowledge areas, considered to be necessary for those individuals developing successful systems, are defined in Table 1. A definition not withstanding, in the remainder of this paper we use the term "software engineer" to identify the required software personnel.

Areas of Academic Preparation

The study committee encountered a problem experienced by many groups in examining the note of the software engineer. That is the inability to differentiate between the function of the systems engineer who has total systems responsibilities and the software engineer who is responsible for the software subsystem and its effect on that total system.

After reviewing several sources (References 4, 5, 6, 7, 8, 9), we recognized that a consensus exists as to the definition of software engineering (and, consequently some direction was provided in defining a "software engineer")

- The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines (Reference 10, pg. 530).

We were pleased to discover that this definition did not differ in principle from that given in the Department of Defense Directive 5000.29 of 26 April 1976:

- (The) science of design, development, implementation, test, evaluation, and maintenance of computer software over its life cycle.

In order to circumvent the problem of differentiating between the responsibilities of the systems engineer and the software engineer, we sought to compare or contrast the two (software and systems) using the 13 knowledge areas. Applying the scale 4–in-depth knowledge, 3–good working knowledge, 2–some knowledge, and 1–little or no knowledge, we produced a censensus estimate of the importance of each area in fulfilling the duties of the software and the systems engineer (see Table I).

KNOWLEDGE AREAS:	KNOWLEDGE BEST PROVIDED BY: ACADEMIC	ON-THE-JOB	KNOWLEDGE AREA IMPORTANCE: SOFTWARE ENGINEER	SYSTEMS ENGINEER
1. Controls - controls, information feedback systems, basic systems distinctions (open loop, closed loop, hierarchical, etc.).	x		2.5	4
2. Process exposure - dynamic interrelationships-time-dependent behavior, system interactions, the "process" concept, cross effects, binding time, process communication, cooperation, and competition.		x(a)	4	4
3. Design principles - the principles of engineering for the scientific method), elements of the design activity (specification, analysis, decomposition, syntheses, testing), maintenance and reliability.		x(b)	4	4
4. Communication skills - written and verbal communication, team participation, and team leadership.	*	*	3.5	4
5. Digital hardware - logical structure and composition. (This area was recognized to be potentially divisible into computer hardware and digital non-computer hardware.)	x		3	2
6. Software design - system life cycle, specification techniques (e.g., PSL/PSA, workbook, Jackson, etc.), development techniques (e.g., chief programmer, structured walk-through, design review, builds, code reading), documentation, modification, maintenance and configuration management.	**	x(c)	4	2
7. Evaluation - systems analysis techniques, models and modeling, identification or creation of alternatives, characterization of trade-offs.	x	**	3	4
8. Systems integration - component and subsystem testing, systems reliability, progressive testing, diagnostic capability, degraded mode options, recovery.		x(d)	4	4
9. Programming - programming languages, systems programs, structured programming, modularity, stubs, program documentation, program testing.	x		4	4
10. Human factors - human/machine interface, dialogue design, prompting, "trainability" and "learnability," adaptability and design for change. (This area was recognized as potentially divisible into software design for human use and hardware design for human use.)	x		3	4
11. Information structures - logical and physical organization of data, data definition, abstract data types, database technology.	x		4	2
12. Communications technology - digital communications, devices, data transmission, coding techniques, protocols, security.	x		2	2
13. Systems simulation - experimentation and system testing using simulation, discrete event and continuous simulation models, wrap-around simulators, emulation.	*	*	3	3

NOTES: * neither or both; ** better job needed; (a) development and support of real-time systems; (b) design of systems emphasizing hardware, software, and firmware interfaces from user needs specifications; (c) development of large software systems; (d) design and testing of systems with software and hardware components.

TABLE I

Academic or on-the-job knowledge

In some knowledge areas academic preparation is believed to be less effective than job experience. Table I reflects the consensus regarding the better source of knowledge, and in some cases a closer description of the contributing experience is given.

Integration and summary

Obviously, no academic program is likely to offer preparation in all the areas marked in the academic column of Table I (Areas 1, 5, 7, 9, 10, 11, 12, 13). The identification of the most essential areas of academic preparation should follow from the inspection of Table I. A reasonable first cut is to begin with the academic areas (those where academic preparation was judged better) designated as requiring in-depth knowledge (level 4). This discrimination identifies only the areas of programming systems techniques (Area 9) and information structures (Area 11). Relaxing the criterion slightly admits interpersonal communication skills (Area 4), which is not clearly designated as better with academic preparation.

At this point we examine the remaining areas, judging the importance of the area considering the better source of knowledge. The consensus is that Areas 5 and 6 should be added as primary knowledge, and the remaining designated as either "secondary" or "useful." This decision produces the resulting classification:

Primary

- Interpersonal communications skills
- Functional capabilities of digital hardware
- Software design technology
- Programming systems techniques
- Information structures

Secondary

Process exposure/dynamic interrelationships

Design principles

Systems integration

Human factors engineering

Systems simulation

Useful

Controls knowledge

Evaluation

Communications technology

We believe that an academic preparation exposing the student to the primary areas is essential. As many of the secondary areas as possible should be included.

University contacts

Based on our study of the needs for and the necessary training of software engineers we have communicated with the presidents of several universities in the local area and plan to follow this up with visits and presentations to their administrations. Our purpose is to try to encourage them to include the academic training needed for software engineering since it is from these universities that we obtain many of our new hires. Their initial response has been favorable but it is well known how long it takes a university to shift its gears. It is also well understood that they have a real problem in obtaining adequate faculty for their present programs.

SOFTWARE ENGINEERING DEVELOPMENT PROGRAM

In order to try to meet NSWC needs for software engineers another committee developed a program to retrain some of our own in-house people and to better educate those who are already doing software engineering tasks (Reference 11).

We reviewed and accepted the 13 knowledge areas identified as being critical to software engineering in the previous study regarding the academic preparation of software engineers.

Three core courses were selected as providing the necessary foundation for learning software engineering; (Structured) Fortran for Scientists and Engineers, Computers Systems Organization, and Software Development.

Fortran for Scientists and Engineers provides an introduction to computer programming utilizing Fortran IV and the CDC 6700 computer. Practice problems, dealing with topics from mathematics, will not be written in an arbitrary fashion, heavy emphasis will be placed on some of the countemporary "Structured Programming" techniques designed to produce readable, coherent, and structured programs. To meet this goal, students will be expected to follow coding conventions provided by the instructor. The instructor will stress (1) efficiency of algorithm and code, (2) program structure and style, (3) documentation (in-line commentary), and (4) correctness of answers and form.

The Computer System Organization course is an introduction to the generic organization, or structure, of digital computers. It also includes instruction in machine, and assembly language programming which requires a more extensive knowledge of computer hardware than does the higher level language programming covered in Course I. The material is taught from a hardware user's standpoint. The course will not prepare a person for work in designing computers. It is oriented toward preparing the software engineer to specify or select computers for various needs and to better understand the implications of hardware design on the software.

Software Development is a course on the development of software for large single and multi-computer applications using both assembly and higher level languages. Development extends from definition of requirements through introduction into use. Life support functions are also included from the standpoint that for many applications development continues throughout much of the life of a system. This course, of necessity, is taught within the framework of systems

engineering methodology, but avoids addressing the full scope of the technology involved in the broader process. Three examples are carried through the course to illustrate the material and provide the basis for work assignments.

The three core courses at the Dahlgren Laboratory are being sponsored by Mary Washington College. Since they are just getting started in their own Computer Science curriculum we are teaching them with our own personnel. The courses are being taught at the White Oak Laboratory under the auspices of the University of Maryland.

Having completed the three core courses the trainees should have an understanding of the basics of software. It is planned to follow up with the following series of short (3-4 days) courses covering the entire spectrum of software engineering:

1. Comparative Software Engineering: A summary seminar which acts as an introduction to the technical subject matter covered by the curriculum. The seminar provides a broad overview of the alternatives, experiences and issues encompassed within the field.

2. Comparative Design and Analysis Techniques: A seminar which discusses in detail how various modern requirements and design techniques can be used to reduce rework costs and improve quality through better specifications.

3. Modern Programming Techniques: A seminar which describes how structured programming concepts and principles (i.e. structured design, top-down development, modular coding, etc.) can be practically implemented in weapons systems within the Navy.

4. Software Testing: A seminar which describes different test approaches, techniques and tools that can be used to realize more quantative goals set for software and system testing.

5. Embedded Computer System Architectural Engineering: A summary course which provides the attendee with the knowledge needed to make informed hardware/firmware/software engineering trade-off decisions.

6. Software Project Management: A summary seminar which addresses the subjects of project planning, organizing, staffing, directing and controlling real-time weapons projects.

7. Software Economics: This seminar surveys the dynamic field of software cost estimation, compares methods and discusses experiences both pro and con of various software cost estimation models. Productivity measures and evaluation methods are described.

8. Software Configuration Management: This seminar discusses the subject of configuration identification, change control, status accounting and verification in the context of Navy systems and documentation requirements (e. g., MIL-STD-1679, SECNAVINST 3560.1, etc.).

9. Software Quality Assurance: This seminar discusses how a software quality assurance program responsive to MIL-S-52779 can be implemented to provide for an objective set of checks and balances. It also treats the subject of software testing.

10. Software Acquisition Management: A seminar which discusses the issues and experiences associated with planning and conducting software development in an aquisition environment.

CONCLUSIONS AND SUMMARY

1. The shortage of software engineers and the needs of software development activities has caused NSWC to take a hard look at how to solve the problem. College and Universities are not and cannot supply them in sufficient quantities in time to meet the needs.

2. There is a general lack of appreciation for the role of software at all levels of management. There is also a lack of agreement on what software engineering is and therefore what the duties of a software engineer are.

3. The NSWC seminar for senior executives has given management a better appreciation for the problems of developing software and has done much to foster the acceptance of new software development techniques. It has also focused the attention of top level management on the need to do something to solve the problem of the supply of software engineers.

4. The knowledge areas designated as primary in this paper are essential in any software engineering program. As many of the secondary areas as possible should be included. This list should be helpful in developing software engineering programs and for evaluating existing or proposed programs in an academic, commercial, or an internal setting.

5. No amount of formal training, either academic or in-house, can produce expert software engineers and the best that can be done with an in-house program is to give people enough knowledge to get them started in the field. It is believed that those who successfully complete the in-house training program, by additional on-the-job experience and formal training, can progressively assume the responsibilities of a software engineer.

6. The decision that the three core courses should be taught under the auspices of a college reflected the conviction that the participants would perceive the necessity of their commitment to the training since it begins in academic environment with attendant homework and grades. It is also hoped that receiving college credit for the courses will encourage some of the trainees to pursue a second degree in a computer related field.

7. Trainees who wish to change career fields to software engineering will stay in their present jobs while taking the core courses. Thus those who do not succeed in the program will still be in appropriate positions. It is also felt that any amount of knowledge they gain from the courses will be beneficial in any job in an R&D organization such as NSWC.

8. The training plan is initially being restricted to holders of scientific degrees in order to have a somewhat homogeneous background and scientific maturity in the classes. It is recognized that there will be others without degrees who may have the capability of succeeding in the program and they will be considered on an individual basic.

9. The response of the program is encouraging, There have been 115 employees sign up for the Software Engineering Program. These are all non-programmers with degrees in engineering, physics, chemistry, materials sciences, etc. Of this group 58 have taken the Fortran class, the others having had some experience with Fortran will be taught structured programming.

10. Activities such as NSWC should encourage their personnel to attend universities, granting advanced degrees in software engineering, under government sponsored programs.

11. An effort should be made between academic, industry, and government to arrive at a consensus of what a software engineer is and the duties thereof. This definition should clearly define the differences between a systems engineer and a software engineer similar to the distinction that now exists between systems engineers and electronic engineers.

12. Colleges and universities should move more rapidly in the direction of educating software engineers. Industry and Government should cooperate in the delineation of the needs and providing an interchange of personnel and experience with universities.

REFERENCES

1. COMPUTERWORLD, Vol. XIV, No. 45; November 3, 1980.

2. Warner, W. P., R. E. Nance, J. H. Manley. "Computing Technology: A Seminar for Senior Executives." NSWC TR 79-174. June 1979.

3. Nance, R. E., W. P. Warner, "Anticipating the Software Engineer: The Academic Preparation." DTIC AD-A086827 (NSWC TR 80-108. May 1980).

4. Hoffman, A., A. J. Personal Correspondence, February 18, 1980.

5. Stucki, L. G., L. J. Peters. "A Software Engineering Graduate Curriculum." Proceedings of the 1978 ACM Annual Meeting, Washington D. C., (December 4-6, 1978), pp. 63-67.

6. Fairley, R. E. "Educational Issues in Software Engineering." Proceedings of the 1978 ACM Annual Meeting, Washington D. C. (December 4-6, 1978), pp. 58-62.

7. Fairley, R. E. "Software Engineering Education." Proceedings of the Thirteenth Annual Hawaii Interational Conference on System Sciences Hawaii (1980), pp. 70-75.

8. Jensen, R. W., C. C. Tonies, W. I. Fletcher. "A Proposed 4-year Software Engineering Curriculum." SIGCSE Bulletin, Vol. 10, August 1978, pp. 84-92.

9. Hoffman, A. A. J. "A Proposed Masters Degree in Software Engineering." Proceedings of the 1978 ACM Annual Meeting, Washington D. C. (December 4-6, 1978), pp. 54-57.

10. Bauer, F. L. "Software Engineering." Information Processing 71 (1971). North Holland Publishing Co.

11. "Report of the Software Engineering Committee." NSWC AP 81-314, August 1981.