View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NASA Technical Reports Server

PSU

243101

P-26

Software Engineering and Ada Training: An Implementation Model for NASA

Sue LeGrand

SofTech, Inc.

Glenn Freedman

University of Houston - Clear Lake

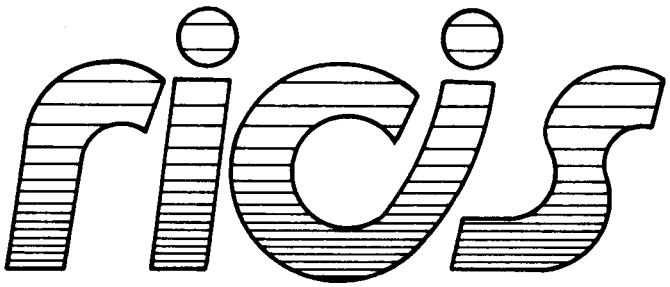
June 27, 1988

Cooperative Agreement NCC 9-16
Research Activity No. *ET.4*

(NASA-CR-186071) SOFTWARE ENGINEERING AND ADA (TRADEMARK) TRAINING: AN IMPLEMENTATION MODEL FOR NASA (Houston Univ.) 26 p CSCL 09B

N90-13966

G3/61 0243101 Unclas



Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information systems in 1986 to encourage NASA Johnson Space Center and local industry to actively support research in the computing and information sciences. As part of this endeavor, UH-Clear Lake proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a three-year cooperative agreement with UH-Clear Lake beginning in May, 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The mission of RICIS is to conduct, coordinate and disseminate research on computing and information systems among researchers, sponsors and users from UH-Clear Lake, NASA/JSC, and other research organizations. Within UH-Clear Lake, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business, Education, Human Sciences and Humanities, and Natural and Applied Sciences.

Other research organizations are involved via the "gateway" concept. UH-Clear Lake establishes relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research.

A major role of RICIS is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. Working jointly with NASA/JSC, RICIS advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research, and integrates technical results into the cooperative goals of UH-Clear Lake and NASA/JSC.

Software Engineering and Ada Training: An Implementation Model for NASA

Preface

This research was conducted under the auspices of the Research Institute for Computing and Information Systems by Sue LeGrand, Program Manager for SofTech, Inc. - Houston Operations. Glenn Freedman, founding Director of the Software Engineering Education Center (SEPEC) of the University of Houston - Clear Lake, provided overall technical direction for this research.

Funding has been provided by the Spacecraft Software Division in the Mission Support Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between NASA Johnson Space Center and the University of Houston - Clear Lake. The NASA Technical Monitor for this activity was Steve Gorman, Deputy Chief of Space Station Office, Mission Support , NASA/JSC.

The views and conclusions contained in this report are those of the authors and should not be interpreted as representative of the official policies, either express or implied, of NASA or the United States Government.

SOFTWARE ENGINEERING AND Ada™ TRAINING: AN IMPLEMENTATION MODEL FOR NASA

Presented by

Sue LeGrand
SofTech, Inc.
Houston, TX

Dr. Glenn Freedman
University of Houston at Clear lake

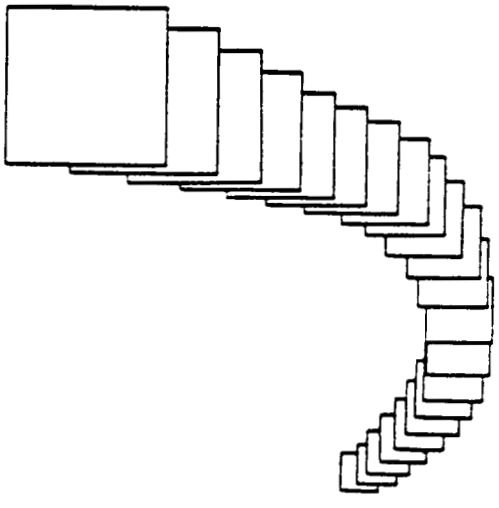
Presented at

Washington Ada™ Symposium
Washington, D.C.
June 27, 1988

™Ada is a trademark of the U. S. Government, Ada Joint Program Office (AJPO).

TP 271

SOFTECH



Software Engineering and Ada Training:

An Implementation Model for NASA

by

Sue LeGrand, SofTech, Inc.

Dr. Glenn Freedman, University of Houston at Clear Lake

Abstract

The choice of software engineering with Ada for projects such as the Space Station has resulted in government and industrial group considering training programs that at once assist workers to become familiar with both a software "culture" and the intricacies of a new compute language. Training costs can be high and management is often reluctant to invest in something as nebulous as "software engineering" training. "After all," a manager might ask, "isn't Ada just another language? Why can't we teach it in three days, like FORTRAN?" Clearly, software engineering with Ada requires more than three days in class and three hours on a computer. But, how much time does it take? How much should an organization invest in training? How should the training be structured?

Software engineering is an emerging, dynamic discipline. Neither industry, government nor university programs are well established in this area, nor is there consensus about who should know what and when should they know it. The Information Systems Services Group of the NASA Space Station Program Office (SSPO) asked for expert advice regarding Software Engineering and Ada training. The authors set out to provide a quantitative assessment reflecting the true requirements for training across NASA, and then to recommend an implementation plan including a suggested curriculum with associated duration per course and suggested means of delivery. Individual copies of the final report of this study may obtained from:

Dr. Glenn Freedman, Director
Software Engineering Professional Education Center
University of Houston - Clear Lake
2700 Bay Area Blvd. Box 270
Houston, TX 77058-1088
(713) 488-9274

Software Engineering Life Cycle

Dr. Charles McKay [13] defines software engineering as the establishment and application of sound engineering environments, tools, methods, models, principles, and concepts combined with appropriate standards, guidelines, and practices to support computing which is correct, modifiable, reliable and safe, efficient, and understandable throughout the life cycle of the application.

The software life cycle has several phases, all of which must be incorporated into an education and training program. A phase is a defined set of input conditions that, when met, trigger an iteration through the phase. There is a defined set of output conditions associated with each triggered iteration. Each phase:

- Has a distinct purpose,
- Has a distinctive set of documentation requirements as the interface to the next phase,
- Is/Should be based upon a model of the requirements associated with conducting the work of the phase,
- Should be complemented by a methodology which features good engineering within the phase, and
- Should be supported by the methodology's own set of technical and management tools to facilitate productivity and quality. [McKay [13]]

These phases, as presented by McKay [13] are consistent with the NASA Life Cycle Model [17]. The seven phases are:

- P1 System's Requirements Analysis
- P2 Software to Hardware to Operational Requirements
- P3 Software-Hardware-Operational Specifications
- P4 Software-Hardware-Operational Design
- P5 Component Development and Integration
- P6 Acceptance Testing
- P7 Sustaining Engineering (Maintenance and Operations)

Education and Training Life Cycle

A review of Ada's history reveals that the language was developed to support the goals and principles of software engineering. Indeed, Ada can be as poorly coded as can any language. It is the sound use of software engineering practices, supported by Ada, or any other appropriate language tools, that results in sound software systems. Further, it is the implementation of education and training programs that result in sound software engineering environments.

Just as there is a software life cycle, so too there is an education and training life cycle for software engineering with Ada. The phases and activities are similar; the consequences for abiding or not abiding by the activities of the phases are also similar.

The record of Ada training in the United States has taught a number of important lessons. One lesson is that many difficulties will be (or may be) overcome by paying more attention to educational requirements definition, analysis and design prior to instruction. Also, just as a good software manager would not expect to reuse code without carefully considering the context and consequences, so too should managers ask if a specific program developed for one audience should be reused by another.

For this report Ada is considered as a programming language, as specified in the Language Reference Manual for the Ada Programming Language. The most effective use of Ada, or any other programming languages, is as a part of the

discipline of software engineering. Recent Ada training reports have indicated that while it may take 5 days for a knowledgeable programmer to learn Ada syntax, it takes 6-9 months to evolve into a software engineer who correctly uses the language to support good design practices and good tool use to help engineer software that is effective throughout its life cycle. Interviews with project managers attest to the phenomena of experienced programmers, with years of FORTRAN or C experience, bucking the transition to Ada. Meanwhile, recent graduates, educated in software engineering, are quicker to adjust to Ada and flourish. Clearly, both groups must be represented in the curriculum.

In summary, like software systems, education and training systems must be well engineered, and they must be engineered for change. A well engineered curriculum will result in a means to develop and sustain personnel skills across many diverse computing environments.

Education vs. Training

Education refers to the processes used in teaching and learning to produce knowledge and highly generalizable skills needed to reason and solve problems. Training, on the other hand, refers to teaching and learning, in the narrower sense, to produce skills to accomplish a specific, practical goal. In brief, education answers the question "Why" and training answers the question "How." A second way of understanding the distinction is to view education as the study of the past to prepare for the future, while training is the study of the present to prepare for today's problems. Both questions are important, obviously, and answering one without the other results in an ill-prepared employee. For this report, the emphasis is on training. Clearly, universities emphasize education and should be included as partners in project implementation.

To educate or train a software engineer, a curriculum must include computing, engineering, project management, and human resource management. This interdisciplinary approach helps explain why software engineering is having a difficult time finding a clear academic home and helps explain why so few universities have well defined software engineering curricula. The Software Engineering Institute is leading the way toward defining a university curriculum, but in the absence of well-integrated, widely accepted academic and training programs, industry and government have developed their own, albeit generally incomplete, training programs.

Often, incomplete training programs have resulted from a misguided perception that knowing Ada syntax means knowing Ada or knowing software engineering. While certainly important, Ada syntax is but a part of a complete software engineering environment that Ada supports. Thus one could possibly be a software engineer without knowing Ada, but one could not use Ada effectively without being a good software engineer. As Ada supports the principles and goals of software engineering successfully, the relationship between Ada and software engineering is quite compatible.

The Clear Lake Model for Software Engineering and Ada Curriculum

The ability to develop a comprehensive training program depends to a great extent on the requirements definitions for the target audience. For an emerging field like software engineering, defining the requirements is akin to hitting

the proverbial moving target. Hence, the course of action was to create a model for defining the field of software engineering in a manner that would allow for flexibility across projects, job descriptions and computing environments while retaining the stability necessary to define a curriculum.

The Clear Lake Model for Software Engineering and Ada Curriculum has six dimensions that must be considered when identifying individual needs and then a curriculum appropriate for an organization or location (See Figure 1). The six dimensions of the Clear Lake Model for Software Engineering and Ada Curriculum are:

1) Job Description
2) Job Activities
3) Software Engineering Knowledge
4) Environments
5) Skill Levels
6) Project Size, Complexity and Extensibility

For a more complete discussion of the Model, see the NASA report [1]. To apply the model to a given organization, one defines precisely the job descriptions of the participants and the number of individuals in each job category. For each job category, the activities of the jobs are then analyzed according to the individuals' involvement in the lie cycle, in software control, in management and in support activities. For those activities, the relevant software engineering knowledge can be mapped.

All of this information has to be considered in terms of the project size and complexity. One assumption we have made is that it is easier to scale-down from more size/complexity to less size/complexity. Thus we recommend that education and training plans be handled similarly to software projects: with full, careful life-cycle consideration.

Importantly, the model must be applied to each context for which a plan is proposed. No two curricula, therefore, will ever be exactly the same, because no two organizations or projects are the same. The Model, however, provides the structure necessary to frame the curriculum.

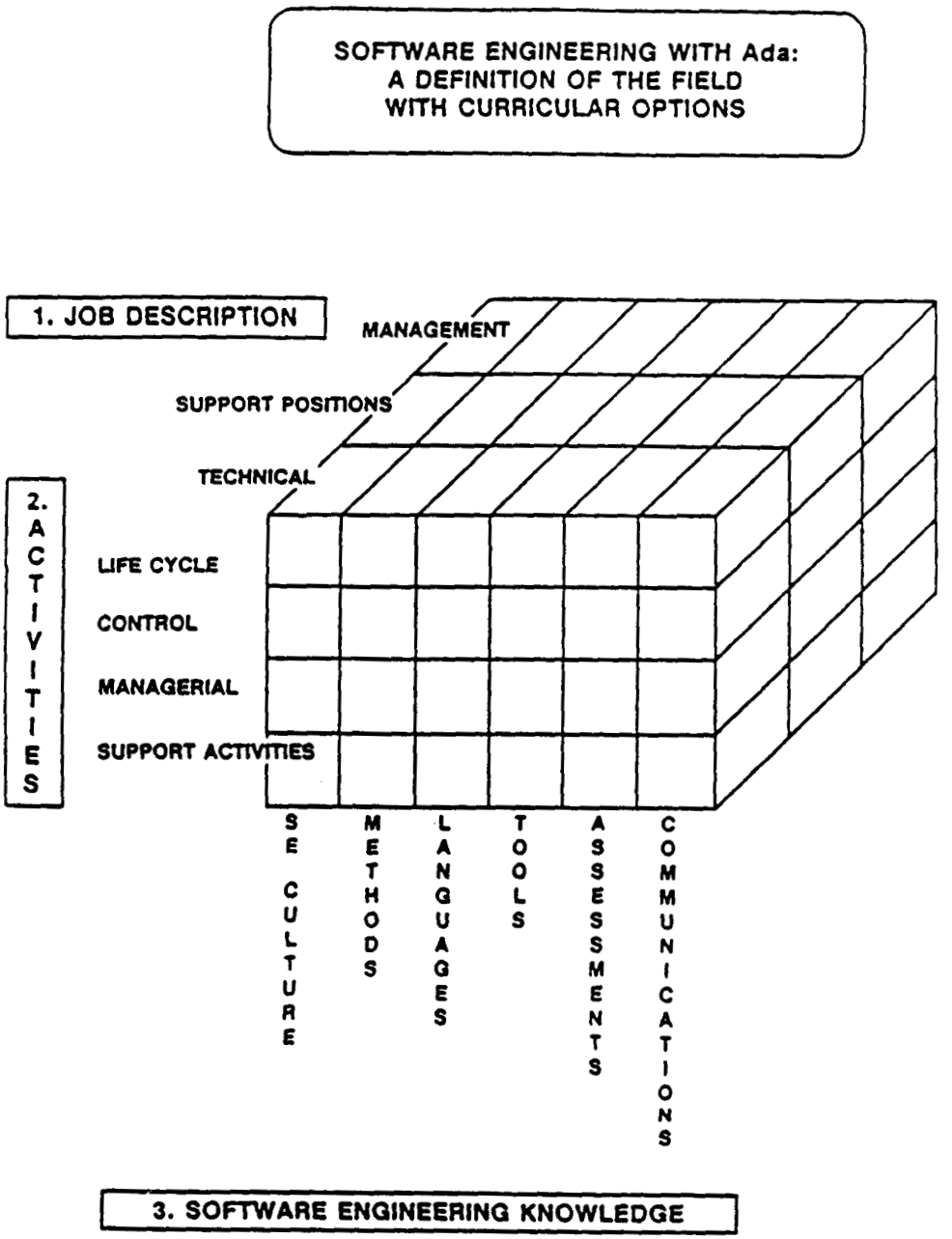


Figure 1 Clear Lake Model for Software Engineering and Ada Curriculum

CURRICULUM

Software Engineering with Ada Curriculum

A comprehensive life cycle curriculum based on the Clear Lake Model assumes that there is a clear sense of the job descriptions involved, a sense of software engineering knowledge and activities, and knowledge of the specific skill levels, computing environments, and projects, domains best defined in the context of a particular organization. Based on a definition of the requirements in the model, a curriculum design can emerge.

To design NASA's comprehensive life cycle curriculum for software, a number of other factors were considered beyond an analysis of the NASA environment. The training curriculum is based on the analysis of NASA environments plus the best of the existing software engineering and Ada education and training programs generally and the needs of NASA specifically. These programs include, for example, those identified in the Ada Joint Program Office's (AJPO's) Catalogue of Resources for Education in Ada and Software Engineering. The most significant Ada and software engineering resources have been the Software Engineering Institute, the now defunct Wang Institute of Graduate Studies, Keesler Air Force Training Command, SofTech, a review of forty-seven commercial vendor's programs and a review of courses from thirty-one universities.

Designing the training curriculum for a complex, distributed organization such as NASA meant that a three-tiered approach was needed to ensure timeliness and the proper targeting of audiences. The first feature of a comprehensive training program was the core curriculum. The second feature, dubbed "technical topics", featured intensive technical, work-related presentations. A third crucial feature of a comprehensive training program was one called "mentoring", referring to on-the-job training, support services, user guides, on-site gurus and references to name a few examples of technical mentoring. Mentoring may also include management features, such as reinforcing good software engineering practice through evaluations, walk-throughs, reviews, and meetings. The goal is to make the software engineering with Ada a part of the organizational culture by infusing it into every layer of software activity. For this brief paper, only a description of the core curriculum will be considered. Both mentoring and technical topics are considered in more detail in the full report to NASA [1].

Based on these features and model's application to the NASA context, a curriculum map was prepared that carefully plotted a core curriculum for NASA. Technical and mentoring activities that augment the core are then based on the needs of a specific software group. See Figure 2.

ORIGINAL PAGE IS
OF POOR QUALITY

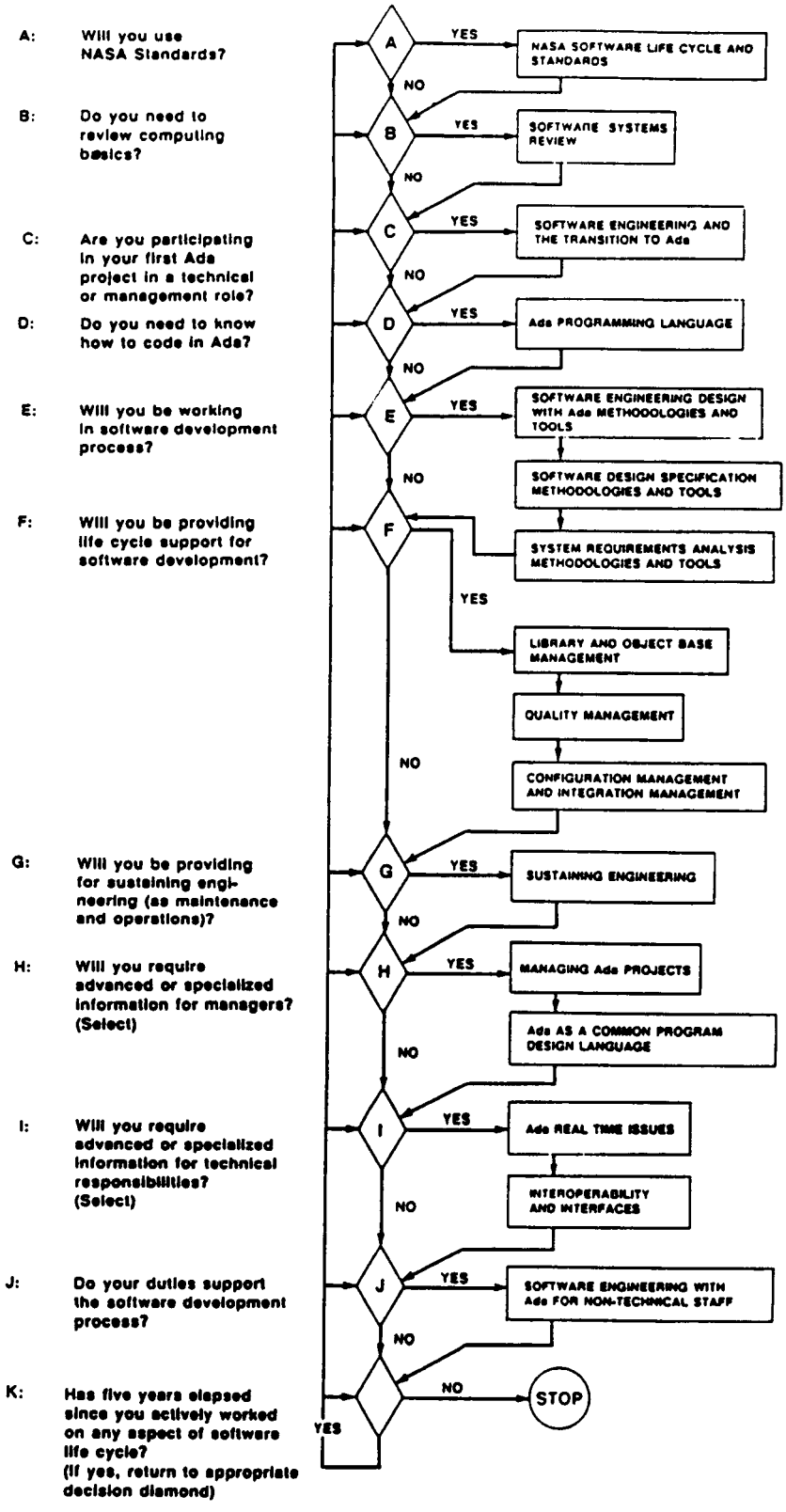


FIGURE 2 CURRICULUM MAP

Core Curriculum

The transition to software engineering with Ada is a transition to a mind set, often called the Ada culture. This culture establishes norms for how good software should be developed and sustained. To achieve successful software engineering and Ada project results, planners must consider technical training, education and on-the-job support as a complete plan. It is easy for one to be seduced by code-centered individualistic approaches to software engineering, but the case histories emerging from large, complex, distributed systems indicate that approaches that perhaps worked well on smaller projects may not scale up to a major software project like the Space Station.

While the Ada programming language is sometimes criticized by detractors as "overly complex" and with "relatively underdeveloped tool sets," each passing month offers new Ada success stories and new, more powerful tools and environments. What does take time and effort is making the significant organizational cultural changes required for a software engineering environment that most effectively leverages Ada. Like any programming language, Ada is a tool. The larger, more significant long-term questions are: How will this tool be used? How rigorous will the engineering environment be? It is safe to assume that rigor is required for hardware development. No less rigor should be tolerated for software. Like all engineering, software engineering requires commitment, effort, and a willingness to adhere to the principles, concepts and models agreed to.

Training hundreds or thousands of practicing programmers to become proficient in correctly applying software engineering principles, in the true sense of the term, will take a major resource commitment. To oversimplify the challenge, for the sake of making a point, one might argue that the problem is akin to taking lifelong house carpenters and expecting them to become architects overnight, with the requisite skills to design, say, a hospital complex. It can possibly be done, but not overnight, not without high cost and risk. In addition, significant training is required beyond the technical skill necessary to do the job.

The curriculum map developed for NASA is based on a set of ten questions that would be posed to an individual or a group of NASA employees. Depending on the job descriptions, one can enter the curriculum at the appropriate level, then select course modules by cycling through the curriculum model. Prerequisites are implied by the ordering of the courses and are not mentioned specifically. Figure 2 illustrates the curriculum map for Ada training for NASA, across center and personnel. For example, all new hires would be exposed to G1: NASA Life Cycle and Standards. A person in legal, however, might not need to answer 'yes' to any other question, except J: Do your duties support the software development process? In contrast, a lead designer might need to participate in the entire curriculum. The curriculum is modularized within courses, so that organizations may select the most appropriate material for the target audiences. The coordination of the curriculum would be handled by the appropriate education staff member in collaboration with management and technical personnel.

The Core Courses

In order to attain good software engineering with Ada, courses in both software engineering and Ada are recommended. A course that teaches Ada syntax is easily labeled an Ada course. Some courses are on the topic of Ada but treat software engineering issues. They include:

- o Managing the Transition to Ada
- o Managing Ada Projects
- o Ada as a Common Program Design Language

Other courses are taught for the purposes of training good software engineers and the Ada language is used in the course. These courses should be categorized as software engineering. The subjects include:

- o Software Systems Review
- o Software Design
- o System Requirements Analysis
- o Library and Object Base Management
- o Quality Management
- o Configuration Management
- o Integration Management
- o Sustaining Engineering
- o Real Time Issues
- o Interoperability and Interfaces

Clearly, courses may be added, modified, or even deleted from the core curriculum once the implementation stage begins, as a normal part of change to meet new requirements.

The Implementation Plan

NASA is currently offering software engineering and Ada courses at its centers. However, the courses are not yet coordinated nor have they been mapped to the overall framework proposed herein. To maintain the NASA's momentum and take full advantage of the results of the study, the following 10 activities have been suggested:

1. Prepare presentations to raise the awareness of the people concerned.

2. Create a feedback loop among the people concerned for tracking:
   New developments and resources
   Recommended improvements

3. Create a database template for centers to use to track the program.

4. Create a baseline set of mentoring resources such as:
   Ada Language Reference Manual
   Vendor List
   Ada Organizations
   Reference Texts
   Experts

5. Assist in identifying programs needed for core, technical and mentoring topics.

6. Map existing resources into a matrix of requirements. Report the differences.

7. Develop a strategy for building existing resources into a prototype curriculum for core, technical and mentoring topics.

8. Report modifications needed in order to use existing training resources.

9. Develop a strategy for providing the remaining resources. Write a report emphasizing integration, synergism and cost effectiveness.

10. Propose a schedule and budget to implement the plans.

Recommendations

The recommendations included in this study do not reflect other training needs; for example, management, hardware and systems engineering and integration issues are vital, but beyond the scope of the study.

Nonetheless, there are final recommendations that need to be enumerated:

1) Promote A Sound Software Engineering Environment. Ada will be most effective if used in an appropriate software engineering environment. Training must be geared to support that culture, including evaluation of courses and instructors according to their contributions to the core curriculum as it becomes fully operational. The work of NASA Software Support Environment (SSE) must be monitored for requirments to update the details of the curriculum.

2) Update the Curriculum on a Timely Basis. The core curriculum will become dated within two to three years if there is no support for including new material, tools, methods and approaches to it.

3) Build on What is Now Available. There are a number of ways to improve existing Ada training programs to match NASA's particular uses. For example, SSE guidelines and procedures will make Ada a working language, one that applies directly to the job.

Ada training templates, reusable components, (both design and code) and library of objectives should be developed and used throughout the agency as a means to demonstrating excellent code examples and for building a library. Ideally, training would take advantage of a large Ada artifact, or a working piece of software that can be enhanced, maintained and studied by the students. Hence, wherever possible real-use examples should be established, especially for documentation and mini-projects included as a part of the course work.

SUMMARY

A software engineering and Ada curriculum for training and education and a proposed implementation plan has been presented that can be adapted for each NASA center according to the needs dictated by each project.

This report is based on a survey taken by meetings, telephone interviews and

written media of the major NASA centers' project and education offices. It is also based on previous research and discussions among education leaders at the Software Engineering Institute (SEI), the Ada Software Engineering Education and Training Team (ASEET), Armed Forces Communications and Electronics Association (AFCEA)[2] and the Research Institute for Computing and Information Systems (RICIS).

ACKNOWLEDGMENTS

The authors wish to thank Lisa Svabek of the Software Engineering Professional Education Center, University of Houston Clear Lake, for her assistance in conducting interviews and summarizing the results [21].

BIBLIOGRAPHY

[1] A Report on NASA Software Engineering and Ada Training Requirements, SofTech, Inc., November 15, 1987.

[2] Armed Forces Communications and Electronics Association, Ada Education and Training Study, Volume 1, 22 July 1987.

[3] Basili, V.R., "The Experimental Aspects of a Professional Degree in Software Engineering" paper presented at the Software Engineering Education: The Educational Needs of the Software Community workshop, 27-28 February 1986.

[4] "Catalog of Resources for Education in Ada* and Software Engineering," Vol. 40, Ada Joint Program Office, May 1987, NTIS Ada 169 892.

[5] Crafts, R.E., ed. "Ada Training -- Selecting a Quality Program," Ada Strategies, Vol. 1, #2, April 1987.

[6] Fairley, R., Software Engineering Concepts, NY McGraw Hill, 1985, p. 2.

[7] Freedman, G.B., A Comprehensive Software Engineering Curriculum Model,, ASEET, Dallas, Texas, June 1987.

[8] Freedman, G.B., Curriculum Options for Transition to Ada, Report for NASA/JSC for ET.1: Software Engineering and Ada, May 1987.

[9] Gibbs, N.E., Fairley, R.E., editors, Software Engineering Education: The Educational Needs of the Software Community. Published by Springer-Verlag.

[10] Godfrey, S., Brophy, C., et al., Assessing the Ada Design Process and its Implications: A Case Study, SEL-87-004, July 1987.

[11] LeGrand, S. and McBride, J., "Ada Language Suited to Space Station Requirements", Government Computer News, September 25, 1987.

[12] Marlowe, G., A Software Engineering Curriculum Proposal, white paper, Summer, 1986.

[13] McKay, C,W., A Proposed Framework for the Tools and Rules to Support the Life Cycle of the Space Station Program, COMPASS Conference, June 1987.

[14] McKay, C.W., Charette, R. and Auty, D. "A Study to Identify Tools Needed to Extend the Minimal Toolset, of the Ada Programming Support Environment (MAPSE) to support the Life Cycle of Large, Complex, Non-Stop, Distributed Systems such as the Space Station Program." NASA/JSC Task Order VHISC B11, March 1986.

[15] Murphy, R. and Stark, M., Ada Training Evaluation and Recommendations From the Gamma Ray Observatory Ada Development Team, SEL-85-002, October 1985.

[16] Murphy, R. and Stark, M., Ada Training Evaluation and Recommendation, Goddard Space Flight Center.

[17] "NASA Software Acquisition Life Cycle Chart", National Aeronautics and Space Administration, Version 3.0, 15 October 86.

[18] National Space Technology Laboratories/Earth Resources Laboratory, NSTL's Ada Lessons Learned, white paper, March 1987.

[19] Pietrasanto, A., Software Engineering Training: Lessons Learned, Ada Expo 86, Charleston, WV, December 1986.

[20] Shaw, M., Beyond Programming in the Large: The Next Challenges for Software Engineering, Pittsburgh: SEI Annual Report, 1985, page 546.

[21] Svabek, L.A., The Development of a Software Engineering Professional Education Center and its Impact on the NASA/JSC Community, white paper to be published December 1987.

[22] Steiner, G.A., Strategic Planning: What Every Manager Must Know, The Free Press, New York, New York.

[23] "Defense System Software Development", DoD-STD-2167, 4 June 1985. Reference Manual for the Ada Programming Language, Department of Defense, January 1983.

[24] Reference Manual for the Ada Programming Language, Department of Defense, January 1983.

SEPEC

Software Engineering Professional Education Center

SOFTWARE ENGINEERING AND Ada TRAINING AN IMPLEMENTATION MODEL FOR NASA

Presented by:

Ms. Sue LeGrand, **SofTech, Inc.**

Dr. Glenn B. Freedman, **University of Houston-Clear Lake**

Presented to:

WAdaS 1988

June 30, 1988

SOFTECH

SEPEC

Software Engineering Professional Education Center

OBJECTIVES OF THE STUDY

- *A qualitative assessment of software engineering and Ada training requirements across NASA*

- *A report recommending a curriculum and implementation plan to the SSPO*

SOFTECH

SEPEC

Software Engineering Professional Education Center

SOFTWARE ENGINEERING

... the establishment and application of
sound engineering environments, tools, methods, models, principles, and concepts

combined with appropriate ***standards, guidelines, and practices***

to support computing which is ***correct, modifiable, reliable*** *and* ***safe, efficient,*** *and* ***understandable*** *throughout the* ***life cycle*** *of the application.*

McKay, 1987

SOFTECH

SEPEC

Software Engineering Professional Education Center

APPROACH

- *Survey NASA Education Offices for Previous and Planned Training Information*
- *Survey NASA Program Offices for information on Ada Plans and Requirements*
- *Form Training Recommendations for Personnel in Management, Technical and Support Areas*
- *Prepare Implementation Plan for*
  - **Core Curriculum**
  - **Technical Topics**
  - **Mentoring**

SOFTECH

SEPEC

Software Engineering Professional Education Center

CONTEXT OF THE STUDY

- *NASA's Transition to Ada and Software Support Environment*
- *Lack of Well Established Education and Training Curricula Suitable for NASA*
- *Need to Establish a Well Engineered, Comprehensive Curriculum in Software Engineering for Distributed, Long-Term Education and Training*

SOFTECH

SEPEC

Software Engineering Professional Education Center

UPDATE

- *Presentation to NASA Headquarters Human Resources OrganizationDevelopment*

- *Report Sent to*

  - ***NASA Project Managers Advisory Board***
  - ***NASA Science and Engineering Advisory Board***
  - ***NASA Centers***
  - ***National Software Engineering Community***

- *Pilot Program at NASA/Johnson Space Center*

SOFTECH

SEPEC

Software Engineering Professional Education Center

TO ORDER COPIES:

Dr. Glenn Freedman
University of Houston-Clear Lake
2700 Bay Area Blvd.
Houston, Texas 77058
(713) 488-9433

SOFTECH