N86-30369

MEASURING ADA* AS A SOFTWARE DEVELOPMENT TECHNOLOGY IN THE SOFTWARE ENGINEERING LABORATORY (SEL)**

William W. Agresti***
Computer Sciences Corporation
and the SEL Staff

ABSTRACT

An experiment is in progress to measure the effectiveness of Ada in the National Aeronautics and Space Administration/ Goddard Space Flight Center flight dynamics software development environment. The experiment features the parallel development of software in FORTRAN and Ada. The experiment organization, objectives, and status are discussed. Experiences with an Ada training program and data from the development of a 5700-line Ada training exercise are reported.

INTRODUCTION

An experiment is underway to assess the effectiveness of Ada for flight dynamics software development. This paper is an interim report on the experiment, discussing the objectives, organization, preliminary results, and plans for completion.

---

*Ada is a registered trademark of the U.S. Government (Ada Joint Program Office).

**Proceedings, Tenth Annual Software Engineering Workshop, National Aeronautics and Space Administration, Goddard Space Flight Center, December 1985.

***Author's Address: Computer Sciences Corporation, System Sciences Division, 8728 Colesville Road, Silver Spring, Maryland 20910.

The Ada experiment is planned and administered by the Software Engineering Laboratory (SEL) of the National Aeronautics and Space Administration's Goddard Space Flight Center (NASA/GSFC). NASA/GSFC and Computer Sciences Corporation (CSC) are cosponsors of the experiment. Personnel from all three SEL participating organizations (NASA/GSFC, CSC, and the University of Maryland) support the experiment.

TECHNOLOGY ASSESSMENT IN THE SEL

There is a great deal of optimism concerning Ada's potential effect on software development. The SEL seeks to establish an empirical basis for understanding Ada's effectiveness in a particular environment--namely flight dynamics software development at NASA/GSFC. Figure 2* shows some of the characteristics of this development environment. (Reference 1 contains a more detailed description.)

As Figure 2 implies, in seeking to understand the effectiveness of Ada, the SEL is approaching this task as it has addressed the assessment of other software technologies. Some methods that have been demonstrated to be effective in other environments have not been effective in the SEL environment. The SEL is therefore cautious about expecting that reported experiences with Ada will obtain in the SEL environment. Instead, the SEL seeks to conduct an assessment of Ada in its own environment.

The assessment methods used by the SEL have included controlled experiments, case studies, and analytical investigations. The Ada assessment is referred to as an experiment, although it is clearly not a controlled experiment. Identifying this effort as an experiment follows the general use

*All figures are grouped together at the end of the paper.

of the word to denote "any action or process undertaken to discover something" (Reference 2). As the later discussion will make clear, the Ada experiment is a highly instrumental case study of an Ada implementation in parallel with a FORTRAN implementation, with both systems developed in response to the same requirements.

OBJECTIVES

The primary objective of the experiment (Figure 3) is to determine the cost-effectiveness of Ada and its effect on the flight dynamics environment. A related objective is to assess various methodologies that are related to the use of Ada. An initial set of such methodologies includes object-oriented design (Reference 3), the process abstraction method (Reference 4), and the composite specification model (Reference 5). Additional methodologies will be identified as the experiment continues.

Reusability is an important tactic for cost-effective software development, both in a general sense and in the SEL environment. Ada was designed (in part) to facilitate reusability. This experiment seeks to develop approaches for reusability when Ada is the implementation language.

The Space Station is a program of great size, complexity, and significance to NASA. Ada has been recommended as the language to be used for the development of new software for the Space Station. An objective of the Ada experiment is to develop measures that may assist in planning for the large-scale use of Ada in the Space Station program. Examples of such measures are those that relate to size, productivity, or reliability in an Ada implementation.

Because the experiment is not completed, these objectives have not yet been met. However, experiences thus far will contribute to addressing the objective of understanding the effect of Ada.

EXPERIMENT PLANNING

The experiment consists of the parallel development, in FORTRAN and Ada, of the attitude dynamics simulator for the Gamma Ray Observatory (GRO) (Figure 5); which is scheduled to be deployed in May 1988. It is worth noting that the dynamics simulator is part of the standard complement of ground support software planned for the GRO mission. The simulator would routinely be developed in FORTRAN alone; because of the experiment, it is being developed in Ada as well.

When completed, the system is expected to comprise 40,000 source lines of (FORTRAN) code, requiring 18 to 24 months to develop on a VAX-11/780 computer. Each team was staffed initially with seven personnel from NASA/GSFC and CSC. Each development project is expected to require 8 to 10 staff-years of effort.

Three teams have a role in the experiment (Figure 6): the Ada development team; the FORTRAN development team; and an experiment study team consisting of NASA/GSFC, CSC, and University of Maryland personnel. The study team is responsible for planning the experiment, collecting data from the development teams, and evaluating the progress and results of the experiment. The study team will also be able to compare the software products generated by each team.

The profiles of the development teams (Figure 7) reveal that the Ada team on average is familiar with more programming languages and is more experienced than the FORTRAN team.

However, the Ada team is less experienced with dynamics simulators, the application area of interest.

Striking differences exist in the relationships of the teams to their development tasks (Figure 8). The FORTRAN team is able to reuse some design and code from related systems. The Ada team is charged with starting fresh to design a system that can take advantage of Ada-related design approaches. For the Ada team, both the development environment and the language are new.

Figure 9 shows the timeline for the Ada experiment with the activities of the three teams during the expected 2-year duration of the experiment. The timeline shows the FORTRAN team to be slightly more than one development phase ahead of the Ada team. The shift is due to the training in Ada required by the Ada team at the start of the project. The FORTRAN team, by contrast, was able to start immediately with the requirements analysis activity--the first phase in the development process.

The study team is collecting data on both development teams. Figure 10 shows the range of resource, project, and product data collected. Wherever possible, routine SEL forms were used. However, special Ada versions of two forms--the component origination form and the change report form--were developed. The new component form allows the identification of an Ada component as a package, task, generic, or subprogram and further recognizes that a component can be a specification or body. The new change form adds a section to identify separately any Ada-related errors.

TRAINING APPROACHES

A major portion of the experiment thus far has been the Ada training program, which was planned by the study team, in particular by the University of Maryland personnel. The principal training resources (Figure 12) were as follows:

- Ada language reference manual (LRM) (Reference 6)
- Ada textbook (Reference 3)
- Ada videotapes (Reference 7)

The 27 videotapes were viewed by the team over a 1-week period. A University of Maryland graduate student, experienced in Ada, was available to direct the training--that is, to plan the schedule of tape viewing, answer questions about Ada material, stop the tapes to clarify the material, lead the discussion between tapes, and assign reading and small coding assignments. Two sets of diskettes for use on personal computers were available to the team to supplement the videotaped instructions. Lectures on Ada-related design methods--the state-machine abstraction and process abstraction method (Reference 4)--were presented to the team.

A principal component of the Ada training program was the design and implementation in Ada of a practice problem. The purpose of this training exercise was to enable the team to apply what it had been taught about Ada and to begin working together as a team.

Figure 13 shows the coverage of topics by the training elements. The textbook and the training exercise covered all three training topics: the Ada language itself, software engineering with Ada, and Ada-related design methods.

Experience with Ada training led to several recommendations for future sessions (Figure 14). Consistent with several other published recommendations (e.g., Reference 3), the appropriate emphasis should be on <u>software engineering</u> with Ada and not simply the language syntax and semantics. The methods and resources used in training the Ada team--videotapes, class discussion, and a practice problem--were effective. Additional hands-on experience with the Ada compiler (in addition to work on the practice problem) is also beneficial.

Two months of full-time training are recommended for each staff member. After this period, the staff member would be able to join a development team and begin contributing. Ideally, this first assignment as a developer should be carefully chosen and closely monitored by a more senior developer. Reference 8 contains a more thorough assessment of Ada training methods and more detailed recommendations for the design of future Ada training programs.

DATA FROM THE ADA TRAINING EXERCISE

The training exercise (or practice problem) emerged as the single most valuable element of Ada training. It also provided the study team with an opportunity to practice monitoring a small Ada project.

The exercise was to design and develop an electronic message system (EMS) that allows users to send and receive electronic mail and to manage groups of users (Figure 16). EMS has been used as a student programming project at the University of Maryland, where it was implemented in the SIMPL language, requiring typically 1000 to 2000 lines of code.

For the Ada team, EMS was a chance to practice object-oriented design as well as to experiment with Ada. The study team could try out the data collection system and begin measuring a small Ada development.

The completed EMS system in Ada comprised 5730 lines of code (Figure 17), much larger than the student projects in SIMPL. An analysis is currently underway to compare the functionality of the Ada and SIMPL versions. It is already clear that the Ada version has a much more extensive user interface and help facility. Also, the 5730 source lines contained only 1402 executable statements. The drop from source lines to executable statements is more severe than in SEL FORTRAN systems, where reductions of only 2 to 1 are typical.

Developing EMS required 1906 staff-hours (including 570 hours of training). A productivity/cost measure frequently used in the SEL is the number of hours per thousand executable statements. Figure 17 shows the cost of EMS development to be greater than the average cost of developing FORTRAN systems. Of course, the EMS example in Ada represents only a single data point whereas the FORTRAN cost data are taken from hundreds of FORTRAN modules in the SEL data base.

It is wise not to rely too heavily on the EMS data as an indicator of future Ada projects. There are several sound reasons why the costs could be higher or lower than those experienced with EMS.

Costs could be higher in the future because of the following:

- EMS was developed by a highly motivated staff eager to apply Ada. As the use of Ada becomes more routine, the staff may not be as motivated by the novelty of using a new language in an experimental setting.

- EMS had no documentation requirements, unlike typical SEL projects.

- EMS did not involve tasking.

- The application domain of EMS (electronic mail) was easier to understand than the flight dynamics area. As a result, the EMS effort in requirements analysis and acceptance testing was proportionally less than it would be for flight dynamics projects.

Costs of the Ada development may actually be lower than suggested by EMS because of the following:

- The staff will be better trained. Recall that EMS was a training exercise; teams in the future will be more experienced in Ada.

- The Ada team (with seven people) was too large for the EMS assignment. The size of the team was driven by the scope of the GRO dynamics simulator development. The cost of EMS would likely have been less if the team were smaller (approximately three people).

- The Ada development environment for EMS was not only new but also highly unstable. Only unvalidated Ada compilers were available when coding of EMS began. The team progressed through versions 1.3, 1.5, and 2.1 of the Telesoft compiler before the DEC Ada compiler arrived.

Figure 17 shows that the error rate for EMS was lower than that of FORTRAN systems in the SEL data base. Once again, this result should not necessarily be attributed to the use of Ada on EMS. The FORTRAN systems are much more complex, and the testing requirements in the flight dynamics area are much more rigorous than for EMS.

Figure 18 shows the distribution of effort among design, code, and test for EMS and typical FORTRAN systems. Whereas the relative effort for the three activities is roughly equivalent for FORTRAN systems, 60 percent of the EMS Ada effort was spent on design. Of course, the use of Ada raises the question of redefining the cutoff between design and code activities. If Ada is used as a process design language (PDL), the design activity can include the delivery of a design document of compiled specifications, Ada definitions of types, and Ada PDL. In such cases, it may be understandable that more effort is spent on "design" activity, with proportionally less effort on "code." Again, the more substantial testing requirements for FORTRAN flight dynamics systems may explain the difference in relative effort devoted to testing EMS versus typical FORTRAN systems.

The profile of the EMS code in Figure 19 reveals that the EMS Ada modules were smaller on average. The lower percentage of lines of EMS that are blank or comment (39 percent versus 51 percent) may be due to the greater self-description possible with Ada object names and types.

STATUS AND OBSERVATIONS

Figure 21 revisits the experiment timeline to show the actual activity to date. The activity profiles of the two development teams confirm that progress is being made according to plan.

With the Ada experiment not yet complete, no definitive statements can be made on the effectiveness of Ada in the SEL environment. Nevertheless, Ada's influence is being felt on personnel issues, software products, the development environment, and the software development process (Figure 22).

The clearest observations relate to the activity that has dominated the early phases of the experiment--training. The need for effective training is real and should be included explicitly in Ada development plans. Training will occur whether or not it is scheduled; wise managers will plan for it. Two months of full-time training appears to be the right amount. The training exercise emerged as an extremely effective method and is strongly recommended.

The use of Ada led to a larger product than the student versions of EMS in SIMPL. It is premature to state whether Ada products will continue to be larger. EMS did demonstrate that many more design relations are expressible in Ada. The use of Ada will likely lead to changes in recommended intermediate products, for example, at design reviews. Current recommendations are oriented to FORTRAN implementations, so the design products highlight the invocation structure of the code. Ada design products can express other relations in addition to invocation--for example, the "uses" relation, exception handling, and the management of the name space.

The use of Ada has not degraded the performance of the development environment. Stress test are now in progress, but the early indications are that the use of the DEC Ada Compilation System (ACS) is not adversely affecting the performance of the system. Both compilation time and execution time appear to be within acceptable limits, although more complete testing is being performed.

The most important tool is a validated compiler. The DEC ACS has demonstrated that it is a production-quality system. Although other Ada support tools may be used by the team in the future, the DEC ACS has been adequate by itself to support development. The library management facility built into the ACS has been especially helpful.

Although such conculsions may appear less than daring, the Ada experiment has demonstrated that Ada is learnable and that an Ada project is measurable. The results thus far lead the study team to be optimistic that they will be able to meet their experimental objectives and establish an empirical basis for understanding the effect of Ada in the flight dynamics software development environment.

ACKNOWLEDGMENTS

The Ada experiment is managed by F. McGarry and R. Nelson of NASA/GSFC and actively supported by representatives from all SEL participating organizations (NASA/GSFC, CSC, and the University of Maryland)--especially V. Basili, E. Katz, Y. Benoit, G. Page, and V. Church.

REFERENCES

1. Software Engineering Laboratory, SEL-81-104, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

2. Webster's New World Dictionary, World Publishing Co., New York

3. G. Booch, Software Engineering With Ada. Menlo Park, California: Benjamin/Cummings Publishing Co., Inc., 1983

4. G. W. Cherry, "Advanced Software Engineering With Ada--Process Abstraction Method for Embedded Large Applications," Language Automation Associates, Reston, Virginia, 1985

5. W. Agresti, "An Approach to Developing Specification Measures," Proceedings, Ninth Annual Software Engineering Workshop, NASA/GSFC, November 1984

6. American National Standards Institute, Inc., ANSI/MIL-STD-1815A-1983, Reference Manual for the Ada Language, February 17, 1983

7. Alsys, Inc., Waltham, Mass., "Ichbiah, Barnes, and Firth on Ada," videotape series, 1983

8. R. Murphy and M. Stark, Ada Training Evaluation and Recommendations, SEL-85-002, NASA/GSFC, October 1985

THE VIEWGRAPH MATERIALS

for the

W. AGRESTI PRESENTATION FOLLOW

FIGURE 1

MEASURING ADA* AS A SOFTWARE DEVELOPMENT TECHNOLOGY IN THE SEL

BILL AGRESTI
(COMPUTER SCIENCES CORPORATION)

AND THE SEL STAFF

*ADA IS A REGISTERED TRADEMARK OF THE U.S. GOVERNMENT (ADA JOINT PROGRAM OFFICE).

FIGURE 2

TECHNOLOGY ASSESSMENT IN THE SEL

REQUIREMENTS LANGUAGES

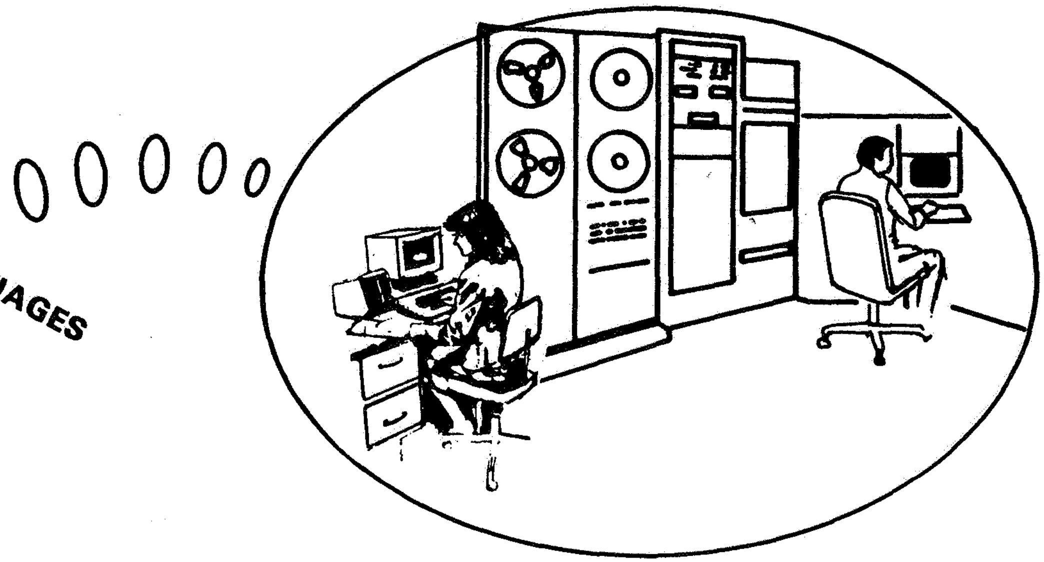
WORKSTATIONS

COST MODELS

TESTING TECHNIQUES

IV&V

CHIEF PROGRAMMER TEAM



NASA/GSFC FLIGHT DYNAMICS SOFTWARE DEVELOPMENT ENVIRONMENT

- SCIENTIFIC GROUND SYSTEMS
- 85% FORTRAN, 15% ASSEMBLER MACROS
- IBM-COMPATIBLE MAINFRAMES, DEC VAX
- SIZE: 3K TO 160K SOURCE LINES OF CODE

831-AGR-(109*a)

FIGURE 3

OBJECTIVES

- DETERMINE COST-EFFECTIVENESS AND IMPACT OF ADA
- ASSESS EFFECTIVENESS OF OBJECT-ORIENTED DESIGN (OOD), COMPOSITE SPECIFICATION MODEL (CSM), ETC.
- DEVELOP APPROACHES FOR REUSABLE SOFTWARE
- DEVELOP MEASURES FOR SPACE STATION

831-AGR-(109b)

FIGURE 4

- **EXPERIMENT PLANNING**
- **TRAINING APPROACHES**
- **DATA FROM ADA TRAINING EXERCISE**
- **STATUS AND OBSERVATIONS**

831-AGR-(109m)*

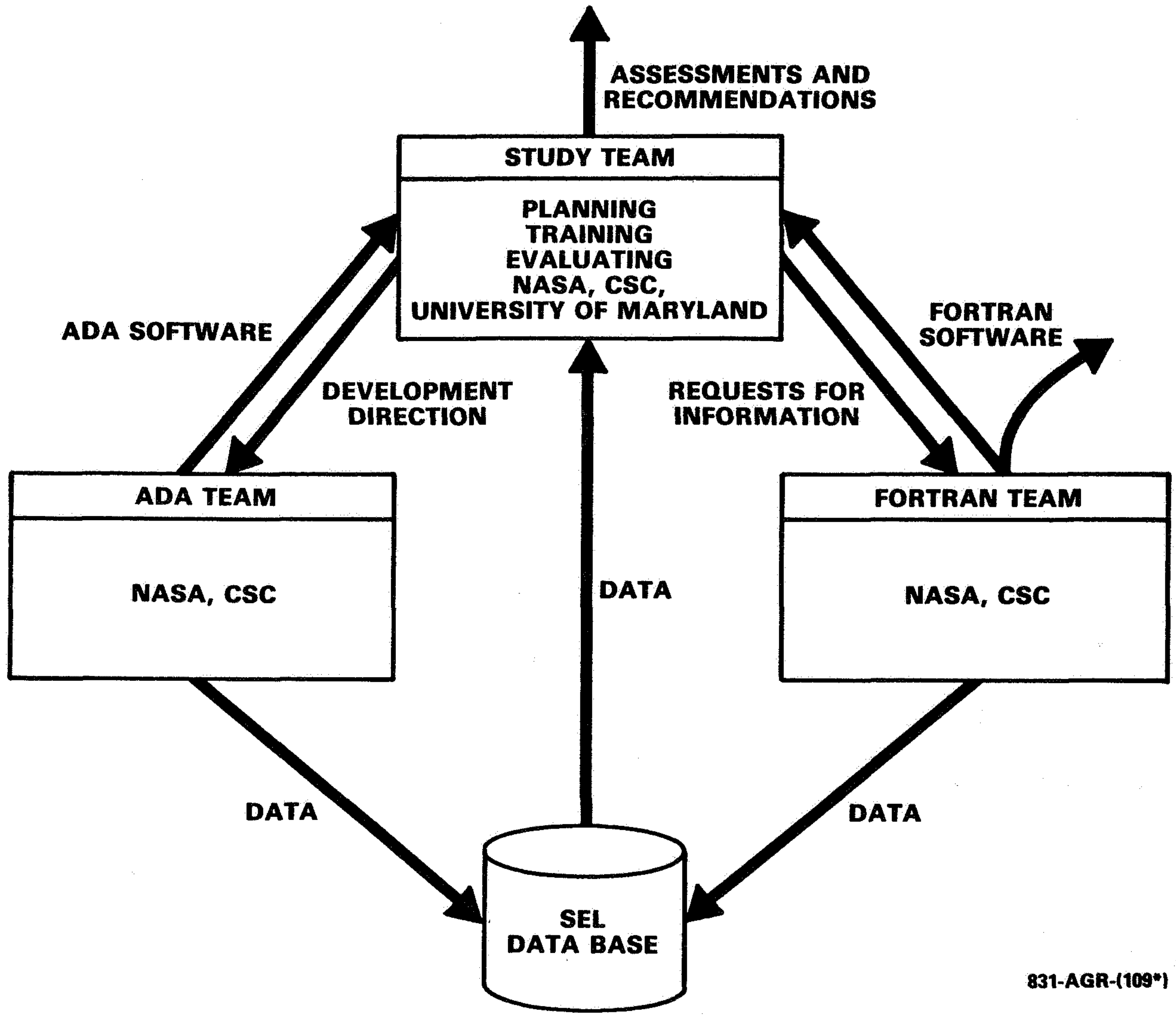
FIGURE 5

SEL ADA EXPERIMENT

- PARALLEL DEVELOPMENT IN FORTRAN AND ADA
- PROJECT: GAMMA RAY OBSERVATORY (GRO) DYNAMICS SIMULATOR
  - SIZE (ESTIMATED): 40,000 (FORTRAN) SOURCE LINES OF CODE
  - DURATION: 18 TO 24 MONTHS
  - ENVIRONMENT: VAX-11/780
  - STAFFING: 7 PEOPLE
  - EFFORT: 8 TO 10 STAFF-YEARS

FIGURE 6

EXPERIMENT ORGANIZATION



831-AGR-(109*)

FIGURE 7

TEAM PROFILES

CHARACTERISTIC	FORTRAN TEAM	ADA TEAM
NUMBER OF LANGUAGES KNOWN (MEDIAN)	3	7
NUMBER OF TYPES OF APPLICATION EXPERIENCE (MEDIAN)	3	4
NUMBER OF YEARS OF SOFTWARE DEVELOPMENT EXPERIENCE (MEAN)	4.8	8.6
TEAM MEMBERS WITH DYNAMICS SIMULATOR EXPERIENCE	66%	43%

831-AGR-(109*n)

CSC COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 8

TEAM DIFFERENCES

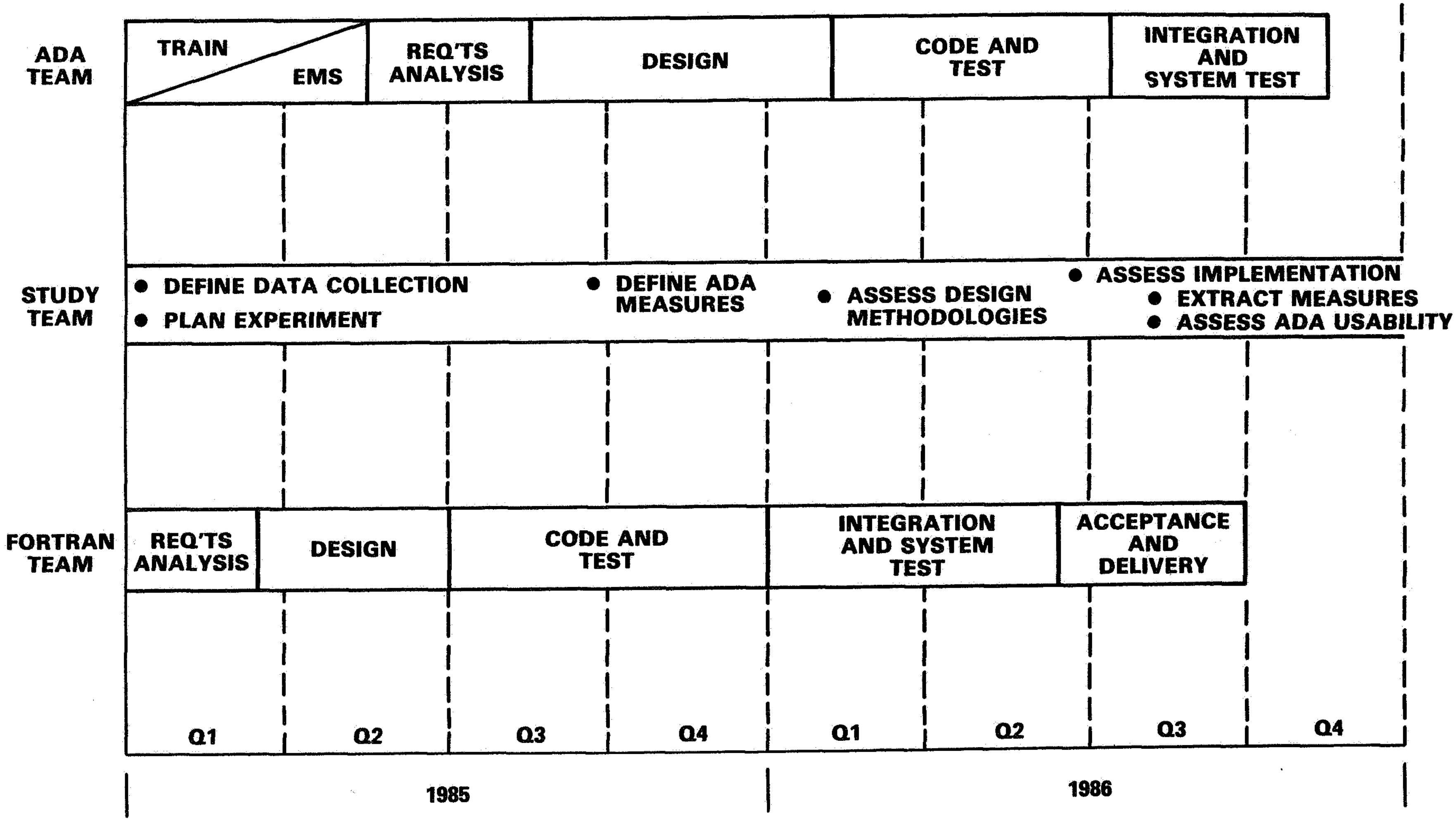
CHARACTERISTIC	FORTRAN TEAM	ADA TEAM
DESIGN HERITAGE	SIMILAR TO PAST SYSTEMS	NEW DESIGN APPROACH
CODE REUSE	15 TO 30%	NONE
DEVELOPMENT ENVIRONMENT	STABLE	NEW COMPILER
LANGUAGE/ METHODOLOGY EXPERIENCE	AVERAGE	NEW LANGUAGE/ METHODOLOGY

831-AGR-(109*)

CSC COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 9

SCHEDULE



831-AGR-(109*)

CSC COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 10

DATA COLLECTION

RESOURCES	PROJECT	PRODUCT
• EFFORT – MANAGEMENT, TECHNICAL, SUPPORT – BY ACTIVITY – BY COMPONENT • COMPUTER USAGE	• STAFF PROFILE • METHODOLOGY/TOOL USAGE • ESTIMATION OF SIZE, EFFORT, SCHEDULE • SUBJECTIVE EVALUATION	• GROWTH HISTORY • DOCUMENTATION • ERRORS • CHANGES

831-AGR-(109*)

CSC COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 11

- EXPERIMENT PLANNING
- **TRAINING APPROACHES**
- DATA FROM ADA TRAINING EXERCISE
- STATUS AND OBSERVATIONS

831-AGR-(109*aa)

FIGURE 12

TRAINING RESOURCES

- ADA LANGUAGE REFERENCE MANUAL (LRM)
- TEXTBOOK – GRADY BOOCH, "SOFTWARE ENGINEERING WITH ADA"
- VIDEOTAPES – ALSYS, INC.
- DISKETTES – HYPERGRAPHICS, INC.
  – ALSYS, INC.
- GEORGE CHERRY SEMINAR

831-AGR-(109*aa)

CSC COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 13

COVERAGE OF TRAINING APPROACHES

TOPIC	ADA LRM	TEXT	VIDEOTAPES, DISKETTES, DISCUSSIONS	LECTURES	TRAINING EXERCISE
ADA LANGUAGE	●	●	●		●
SOFTWARE ENGINEERING WITH ADA		●	●	●	●
VARIOUS DESIGN METHODS		●		●	●

831-AGR-(109*cc)

FIGURE 14

TRAINING RECOMMENDATIONS

- TOPIC — <u>SOFTWARE ENGINEERING</u> WITH ADA
- METHODS — VIDEOTAPES
  CLASS DISCUSSION
  HANDS-ON EXPERIENCE
  TRAINING EXERCISE
- TIME PERIOD — 2 MONTHS FULL-TIME

831-AGR-(109z*)

FIGURE 15

- **EXPERIMENT PLANNING**
- **TRAINING APPROACHES**
- **DATA FROM ADA TRAINING EXERCISE**
- **STATUS AND OBSERVATIONS**

FIGURE 16

ADA TRAINING EXERCISE
ELECTRONIC MESSAGE SYSTEM (EMS)

- **FUNCTION:** SEND/RECEIVE ELECTRONIC MAIL; MANAGE USERS AND GROUPS
- **HERITAGE:**
  - STUDENT GROUP PROJECT AT UNIVERSITY OF MARYLAND
  - 1000-2000 SOURCE LINES OF SIMPL CODE
- **DESIGN APPROACH:** OBJECT-ORIENTED DESIGN

FIGURE 17

ADA EMS PROJECT SUMMARY

SIZE

- 5730 SOURCE LINES OF CODE
- 1402 EXECUTABLE STATEMENTS

EFFORT

- 1336 HOURS
- 570 TRAINING HOURS

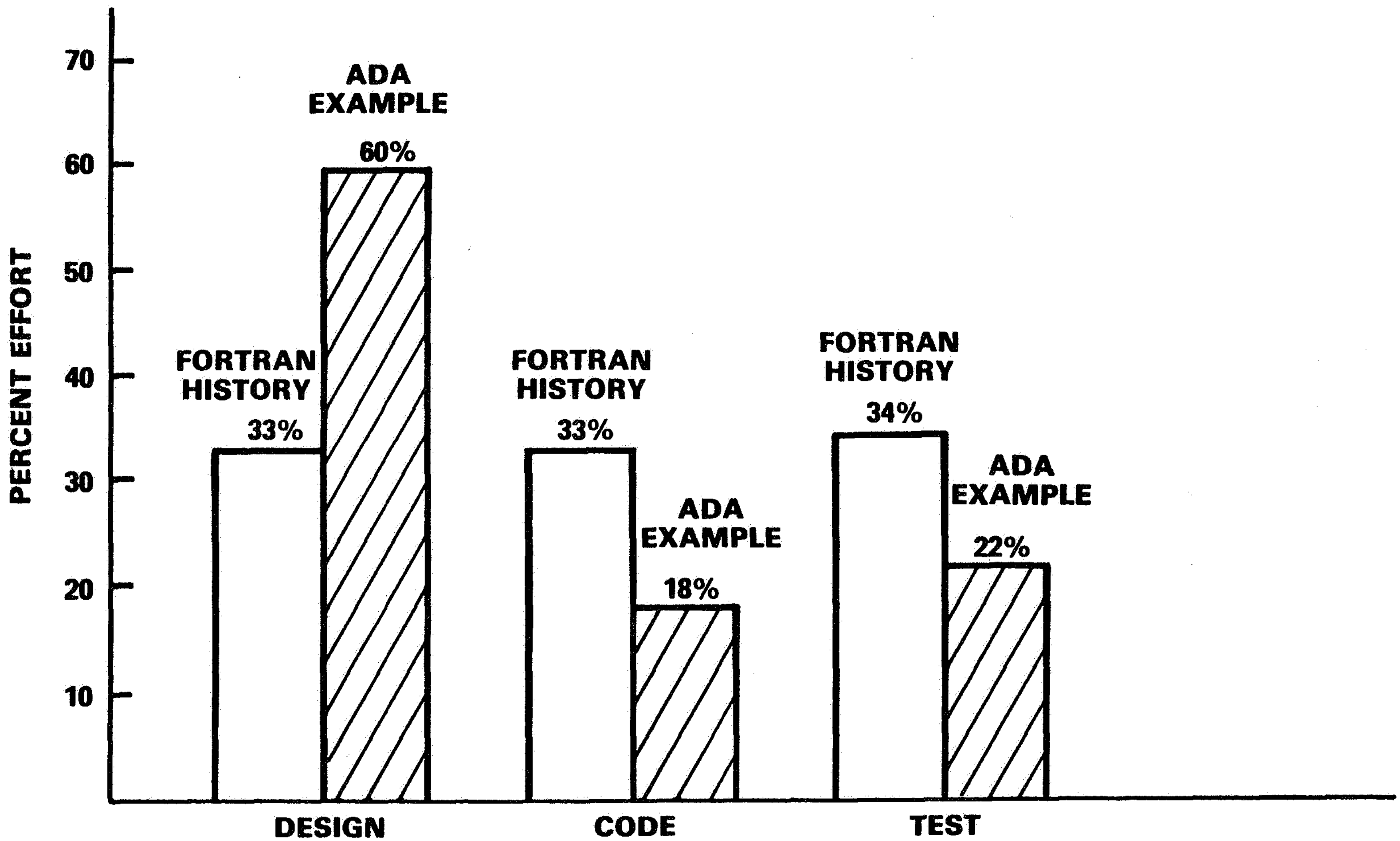
1906 TOTAL HOURS

	ADA EMS EXAMPLE	FORTRAN HISTORY
COST (HOURS/1000 EXEC. STMTS.)	950 (WITHOUT TRAINING HRS.) 1360 (WITH TRAINING HRS.)	720
ERROR RATE (ERRORS/1000 EXEC. STMTS.)	9	12

831-AGR-(109*bb)

FIGURE 18

EFFORT DISTRIBUTION BY ACTIVITY TYPE



831-AGR-(109*aa)

CSC COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 19

PROFILE OF ADA EMS CODE

	FORTRAN HISTORY	ADA EMS EXAMPLE
MODULE SIZE		
• LINES OF CODE	232	143
• EXECUTABLE STATEMENTS	69	35
LINES		
• BLANK OR COMMENT	51%	39%
• PROGRAM TEXT	49%	61%
STATEMENTS		
• DECLARATIONS	42%	32%
• EXECUTABLE	58%	68%

831-AGR-(109*bb)

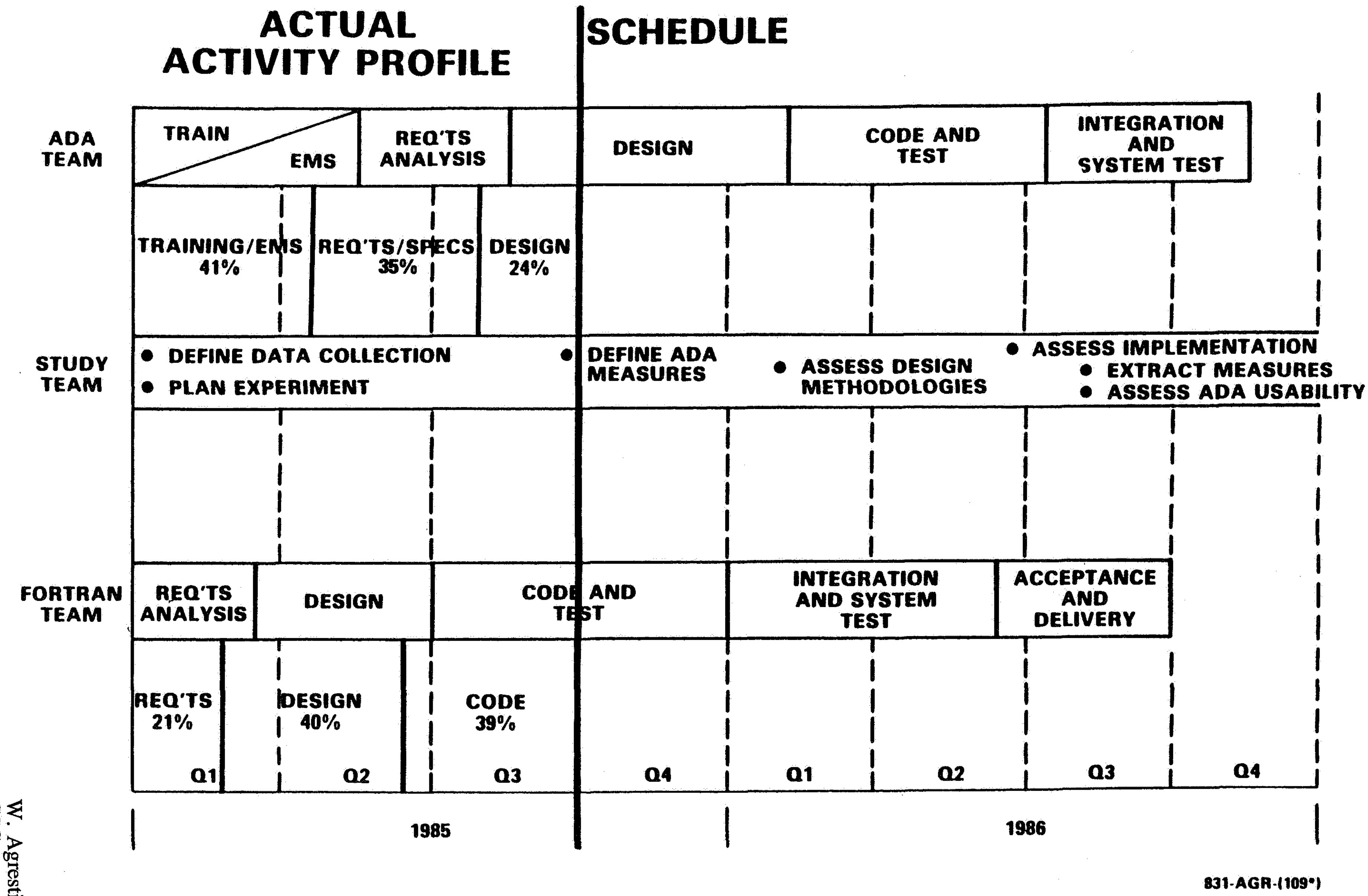
FIGURE 20

- **EXPERIMENT PLANNING**
- **TRAINING APPROACHES**
- **DATA FROM ADA TRAINING EXERCISE**
- **STATUS AND OBSERVATIONS**

831-AGR-(109*bb)

COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 21



CSC COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION

FIGURE 22

OBSERVATIONS ON THE IMPACT OF ADA