ADA® COMPILER PROJECT MANGAEMENT ISSUES

Wendy J. Hudson
Concurrent Computer Corporation
Tinton Falls, New Jersey 07724

ABSTRACT

The use of the Ada language in a development project impacts the traditional approach to project planning. The experience at Concurrent Computer Corporation in the development of an Ada compiler, written in the Ada language, showed that the design phase of the project was longer than anticipated. The increased design time significantly decreased the system integration time. In addition, the time spent learning to use the Ada language effectively was a large portion of the total project time. The coding rate was not unusual and the overall project schedule was only 10% greater than the original plan. Initial results also indicate that the resulting product is more reliable when written in the Ada language.

INTRODUCTION

In January 1985, Concurrent Computer Corporation began the development on an Ada compiler which was written in Ada. At the conclusion of this project, it became evident that the use of Ada significantly altered the phases of the project.

The results of this project were influenced by project planning and by the background of each of the project team members.

Project

As with most companies, we had not written a large amount of software using Ada. Most of the Ada coding had been done on small and experimental projects. Thus, the Ada compiler project was a new experience for both the management and the project team.

The development of the Ada compiler was split into three pieces. The front-end of the compiler was purchased from Systeam, in Karlsruhe, West Germany. This front-end was selected because it was part of a validated compiler and was itself written in Ada. The front-end of the compiler was to be rehosted to our operating system and upgraded to include some implementation dependent features. The front-end of the compiler contains the syntax and semantic checking. The other pieces, and a major portion of the new code, was the development of the back-end and the run-time system. The back-end contains the code generation and optimization. The run-time system contains the support for things such as I/O, tasking, and operating system support.

The compiler consists of approximately 400,000 lines of Ada code of which 50,000 lines of code were newly generated. The back-end contains about 30,000 lines of code of which 24,000 lines were new. The run-time system contains 21,000 lines of code of which 17,000 lines were new. The modified code in the front-end and other sections was about 9,000 lines.

After integration, the entire compiler would require testing and validation. While this project was similar in size to previous compiler projects, about 60,000 - 80,000 lines of code, the testing of the full 400,000 line system was expected to be much greater than any previous compiler integration. In addition, the Ada validation test suite is much larger and more rigorous than any other compiler validation test suite.

The project was also split across multiple locations. Most of the group were based in New Jersey. There were three people in West Germany working on the compiler front-end changes and another person in Virginia working on code generation.

Project team

The make-up of a project team can often

mean success or failure. In this respect, we were very fortunate.

The size of the group varied during the project. The core group contained 11 people and a full-time manager. For short periods of time, an additional three people also worked on tools and testing. There were four consultants who did specific work in the front-end and the run-time system.

In the core group, 5 of the 11 were very familiar with the Ada language when the project began. Those with experience had been designing and coding in the Ada language for 2-3 years. In addition, they had a very strong background in compiler development. Three others in the group had strong compiler backgrounds, but no Ada programming experience. The remaining three people were newly hired with essentially only academic experience in both compiler development and Ada. All but three people in the core group hold graduate degrees. Generally, the group had a strong academic background with significant compiler design and Ada programming experience.

The manager had worked with most of the group for several years. He had previously managed large compiler projects and had a strong compiler background, but very little experience with the Ada language.

Many of the members of the project team had worked together before on other compiler projects. The group had a good spirit of cooperation, a very professional attitude, and a strong commitment to the project. The group was also quite receptive to the plan to design and code in Ada. They believed that using Ada would benefit the project and they worked very hard to make its use successful.

The experience in Ada programming and compilers along with strong management and team spirit combined to create a very effective programming team. The composition of this team was a critical factor in the success of the project.

Project planning

Before the project was fully staffed and had begun, three senior members of the team spent two months on analysis and planning of the project. They studied the existing code in the compiler and designed the high-level strategy for the back-end and the run-time system. Their research led to a breakdown of the work required and the creation of the initial project plan. This evaluation and definition was one of the key factors in the project's success. It provided the baseline for the project analysis.

For the six members of the group who were unfamiliar with the Ada language, learning Ada became a major activity. We found that Ada is more difficult to learn and use effectively than other languages. It is difficult to gauge when someone has gained enough knowledge about a programming language to use it effectively. Generally, this evaluation was done through the design and code walk-throughs. The five experienced Ada programmers on the project made the evaluations easier.

Only one or two people enrolled in a one week introductory course. Most people in the group learned Ada by self-study and read books on the Ada language for the initial instruction. A microcomputer-based course was available which was helpful after the initial introduction to the language.

We had estimated that learning the Ada language would consume about one month of each person's time. After that point, we expected that the individual would be able to code effectively and design small sections of the project. We found that in order to reach this level, it took 2-3 months of training time for each person. As a result, 12 to 18 man-months were added to the schedule.

As a result of our experience, we have developed some Ada "guideline definitions" for people learning Ada. After three weeks of training, a team member was familiar with the Ada language terminology and was considered to be an "Ada coder". After 2-3 months, the programmer became an effective member of the project team. At this point, they were an "Ada programmer". It took nine months to one year before the programmer could effectively design major pieces of Ada code for the project and be considered an "Ada designer". It is interesting to note that the ability to effectively use Ada did not seem related to either the compiler or academic experience of the team member. Experienced compiler designers took as long to learn the Ada language as the new college graduates.

Schedule and manpower results

Table 1 outlines the planned versus actual schedule and manpower results for the design, coding, and system testing phases of the project. The design phase was defined as the design of all the interfaces between all the procedures.

Table 1:

Schedule:

	Plan		Actual			
Phase	Duration	% of Project	Duration	% of Project	% Change	
Design	3 months	20%	5.5 months	34%	83% increase	*
Code/Sub-system test	6 months	40%	7.5 months	45%	25% increase	**
System Integration	6 months	40%	3.5 months	21%	42% decrease	
	15 months		16.5 months		10% increase	

Manpower:

	Plan	Actual		
Phase	Duration	Duration	% Change	
Design	28 man-months	53 man-months	89% increase	*
Code/Sub-system test	88 man-months	112 man-months	27% increase	**
System Integration	63 man-months	45 man-months	28% decrease	
	179 man-months	210 man-months	17% increase	

*** - Increase primarily due to additional time for learning Ada**
**** - Increase primarily due to additional functionality**

The **coding phase** included:

- design of the algorithms for each routine
- coding of the routine
- code walk-through
- unit testing of the routine
- testing of the routine in a sub-system

The **system integration phase** included putting the new code generation sections together with the modified front-end, and testing the complete compiler with the new run-time system. The end of system integration occurred when all the Ada compiler validation and internal tests were successfully completed.

The Ada validation test suite checks for conformance to the language standard. At the time that we completed the validation, the suite contained about 1800 tests. The Ada validation tests are more extensive than the validation suites for other languages. However, since it only checks conformance to the standard, we needed to generate additional tests for quality, performance and system dependent features.

Design

People who have embarked on large Ada projects and have subsequently written about them, have all found the design phase to be longer than anticipated. Certainly, our experience in this project was no different. The schedule was 83% longer and the manpower required was 89% greater. In this phase, we designed, coded, and debugged the package specifications to verify the interfaces between all the procedures. The extra time required in this phase was for the debugging process. We feel that the time spent in this debugging effort was the most significant factor in the ease of integration later in the project.

Coding

During the coding phase, the algorithm design, coding, code walk-throughs and testing were done quickly. Table 2 contains information for one piece of the new coding effort for the code generation section of the compiler. This code was part of the critical path in the project. The code size was 38% larger than planned, but notice, the coding took far less time than planned. We feel that the reduced time was a result of the efficiency and features of the Ada language.

Coding was originally estimated at 11 lines per day per person. The estimate was based on the fact that about half the group had never programmed using the Ada language. This also accounted for the procedure design and testing. The coding rate was actually 47 lines per day. If the design and system integration time were included in the calculation, then the coding rate over the entire project was about 15 lines of code per day per person.

It would seem that coding would be shorter than planned, yet Table 1 shows that the coding phase exceeded its schedule. This was due to additional functionality being added to the product. An improved version of the compiler front-end became available. It was decided that we would use the new front-end; it was believed that this could be done within the same scheduled time frame. The new features in the front-end caused a greater number of changes in other sections than was originally anticipated, therefore the coding phase was lengthened.

System integration

The system integration phase was less intense than planned. The schedule was 42% shorter and this phase used 28% less manpower. Within a few days from the start of integration of the compiler, it was compiling complete programs. This is a very unusual situation in the development of compilers. When problems were detected, they were easy to locate and fix. Steady progress was made in locating and resolving all the problems which were found when running the validation test suite.

During the integration phase of previous compiler projects, the number of problem reports generated were usually about 700. 389 problem reports were generated for the Ada compiler project. This number is especially significant since the Ada compiler has 5 times the number of lines of code than any previous compiler project. Also, the Ada language was new to 6 of the team members. We had anticipated that there would be a larger number of bugs as a result of using this new and complex language.

The success of this phase was a result of the early debugging of the design of the interfaces. The use of Ada naturally led to this work being done early in the project. It lengthened the design phase, shortened the integration, and significantly improved quality and reliability.

Maintenance/Reliability

Since validation, the compiler has been used at six customer sites and at numerous Concurrent Computer field office sites as part of the beta testing period. During the test period, only 52 problem reports were generated against the product. Of these 52 reports, only three were compiler problems. The remaining problem reports concerned the packaging and the compiler operation scripts.

Table 2:

Coding Estimates:

Plan		Actual	
3025	lines	4183	lines (38% increase)
275	man-days	89	man-days (68% decrease)
11	lines/day	47	lines/day
87	procedures	87	procedures
35	lines/procedure	48	lines/procedure

Based on these initial results and the few number of problems detected during the integration phase, it appears that the Ada code will be much more reliable than products written using other languages. The reliability and maintainability cannot be accurately measured without more widespread use. Our experience in the integration and beta phases allows us to anticipate exceptional performance in this area.

Other project observations

The development process on this project was much different from projects which used other languages. The emphasis on design, when using Ada, distributes the machine requirements and compiler load more evenly throughout the project. When the Ada specifications are coded and debugged early in the project cycle, the interfaces stabilize early in the coding cycle. After the specifications became stable, the number of recompilations were significantly reduced. Less time was spent by the programmers sitting at terminals to incrementally program the procedures. The use of the Ada language forced the programmers to think more carefully about the programming process.

The recompilation requirements of Ada also changed the approach to modifications. Ada requires that a routine be recompiled when a change is made to another routine that it depends on. Information found in the Ada compiler project library is used to determine these dependencies. The group became careful in changing Ada specifications which would require major recompilations. Such changes were usually saved until the end of the day. The changes were made and the major recompilations could be done overnight. The group felt comfortable with this approach because the recompilations were generally completed without problems. This attention to changes in the specifications also helped focus group attention on critical modifications. Critical modifications were done only after consultation with the rest of the team.

We had originally planned to add more computers to the project as it progressed. Because of the reduced compilation load, this greater computing power was not needed. We did, however, need larger amounts of disk storage than had been planned. The Ada requirement for project libraries and the source control of 400,000 lines of code contributed to the additional disk requirements.

CONCLUSIONS

The Ada language is excellent for programming. Although it requires a greater amount of design time and takes longer to learn, the integration time is substantially reduced and the product is more reliable and maintainable.

Two separate conclusions can be drawn from our experience. The first conclusion relates to new Ada projects using new Ada people:

- expect a 2-3 month period for learning Ada
- expect effective design of major portions of the project only after 1 year of work in Ada
- expect increased design time due to unfamiliarity with Ada

Generally, the members of a project team should begin learning Ada as soon as possible. Greater project time should be allocated to the design phase and whenever possible, the detailed project planning should be done by those familiar with both the Ada language and the application.

The second set of conclusions that can be drawn would be for the project with seasoned Ada designers and programmers:

- expect the design time to increase about 40%
- expect the integration time to decrease about 30-40%
- Over the **entire project** (design, code, integration, test) expect the coding rate to be about 15-20 lines per day per person
- During times of actual **coding activity**, expect the coding rate to be about 40-50 lines per day per person
- expect reduced problems and more reliable software

In closing, the programming and management team are critical to the success of the project. A team that understands and believes in the concepts of Ada will be more effective. A successful Ada project can be accomplished if both the management and project team understand the differences in designing and programming in an Ada system, and expect the advantages of Ada to accrue.

About the author:

Ms. Hudson is Senior Manager of the Scientific and Support Languages development group at Concurrent Computer Corporation. She holds a BA and an MS degree in Computer Science from Rutgers University. She has 11 years of software development experience involving compilers, operating systems and networking. Since joining Concurrent Computer Corporation in 1979, Ms. Hudson has been involved in the development of FORTRAN and Ada compilers, and symbolic debuggers. She is currently managing the group responsible for the development of the Ada, FORTRAN and Pascal compilers.

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