MANAGING ADA DEVELOPMENT

by

James R. Green
Manager, Standard Products

Systems and Software Engineering Operations
Dalmo Victor Incorporated
The Singer Company
6365 East Tanque Verde Road
Tucson, Arizona 85715
Phone: (602)721-0500

C-1

Managing Ada Development — James R. Green

Introduction

The Ada programming language was developed under the sponsorship of the Department of Defense to address the soaring costs associated with software development and maintenance. Ada is powerful, and yet to take full advantage of its power, it is sufficiently complex and different from current programming approaches that there is considerable risk associated with committing a program to be done in Ada. There are also few programs of any substantial size that have been implemented using Ada that may be studied to determine those management methods that resulted in a successful Ada project.

As the Manager of Standard Products, I have the responsibility for developing software products that will be offered for sale on the open market. One of the products which has been developed is implemented entirely in Ada and its success demonstrates that a project can be successfully done using Ada. The program itself comprises over 130,000 source lines of Ada code. This project, although not large by today's standards of software development, did cause me to face the frustrations and the difficult management tasks associated with implementing an entire program in Ada at a time when the Ada development environment was less than desirable. The items presented in this paper are my opinions which have been formed as a result of going through this experience. The difficulties faced, risks assumed, management methods applied, and lessons learned, and most importantly, the techniques that were successful are all valuable sources of management information for those managers ready to assume major Ada developments projects.

People - A Key Ingredient

Projects are implemented by people. The right people are definitely a key ingredient to the success of any project. Ada is no different. Management must realize this at the beginning of a project and ensure that "the right people" are selected. Ada has new concepts which are different from other languages. Concepts such as packages, specifications, body, information hiding, generics, instantiation, and multi-tasking are all examples of concepts and features of the Ada language. Since many of these concepts do not exist in other languages, management must be prudent in selecting personnel for assignment to the Ada project itself.

The real power of the Ada language lies in the concepts not necessarily available in other languages. The people in key positions of the project must relate to these concepts and management must ensure that the people that are initially selected do relate to these concepts. It is possible to write code in Ada that utilizes only the most elementary concepts. An analogy to this would be writing Ada code using only the constructs allowed by FORTRAN. Clearly, the code may work, but you will not realize the benefits of the Ada language.

Those people selected to work on an Ada project most probably will have prior working experiences in other languages. Their effectiveness on the Ada project will be related to how easily they accept the new language features and strive to use them effectively. As a manager, you do not want the powers afforded by Ada, to be eclipsed by an engineering staff of parochial vision.

I have found that people who have recent degrees in computer science relate well to these concepts. In addition, personnel who are well versed in Pascal programming, seem to transition quite easily into the Ada world. In my experience, I was fortunate to find talent that related these concepts. At the beginning of the project, no one on my team had any Ada experience, and further, few of them had any knowledge of what Ada was

all about. The success of this project is a tribute to their talents.

Management must seriously scrutinize the qualifications of those people they select to implement Ada projects. Selecting the right people will definitely increase your probability of implementing a successful Ada project.

Training Programs - A Key Ingredient

Ideally, you would want to hire people who have performed successfully on other Ada projects. However, there is a limited number of people who are proficient in Ada and I highlight the word *proficient.* Proficient means that the people understand the complex concepts of this new language and understand how to apply them. This is different than just knowing the syntax and semantics of the new language. There is a large body of software people that are well versed in FORTRAN, JOVIAL, COBOL, and other well established, high-level software languages. Many of these people will be transitioning to work in the Ada environment. Management must provide a means for these people to transition successfully into the Ada world. This leads me to the second key ingredient to success--training programs.

The program which I managed began with people that were unfamiliar with Ada. There was a wide variety of background experience among the people selected for the project. It was clear at the outset that a key ingredient to the success of the program would be the implementation of an effective training program. The training program would provide two benefits. First, it would establish a common baseline of knowledge for all people on the project at that time. The varied experiences of the people, and their knowledge of software engineering was an unknown. By covering these topics in a training course, I could be sure that every one on the project was in synch with respect to vocabulary, concepts, approaches, methodologies, and techniques. The Ada programming language and the concepts and methodologies to be used when designing Ada programs could be covered in detail. In addition, there would be a benefit of discussing

application techniques--that is how to implement certain features using the Ada language in practical applications. The training program that I implemented actually consisted of five courses and comprised 132 classroom hours of instruction.

The courses developed and given were:

Introduction to Software Engineering.....32 hours
Software Design Methodology.....40 hours
Coding Methodology.....16 hours
Ada Programming Support Environment..... 4 hours
Ada Programming.....40 hours

The Software Engineering, Design Methodology and Coding Methodology courses were developed in-house. These courses are specifically designated to provide a sound understanding of the software development process, software life cycle and design methodologies. The Ada courses used books and lecture material that was available at the time. It dwelled primarily on the syntax and semantics of the Ada language.

The length of time that was taken for training may seem excessive, and indeed, at the time I thought it was excessive. However, during later stages of the project, it was clear that the time spent at the front-end of the project for training, was time well spent.

I cannot stress strongly enough the need for the development team to understand good software engineering principles and design methodologies. To fully realize the power of Ada in your program, these principles must be understood and used. Although, in my case, all individuals went through the same level of training, I would recommend different levels of training for different project people. I would recommend two to four weeks of intense training for key technical people on the project, and possibly one to two weeks for junior people. The Ada training for the junior people will be augmented through on-the-job-training and the assignment of tasks under the guidance of the more senior and more experienced project

personnel.

The training issued, and the time and money which should be allocated during the project for training, is quite controversial. There are a number of training programs currently available. However, many training courses are short and cover the syntax and semantics of the language primarily. For the training program to be truly successful, it must include the software engineering and design principles needed by those people designing the Ada program. These people must understand these principles, and they must understand how they relate to Ada and this means that significant time must be spent on the software engineering aspects of software design.

I recognize that the effectiveness of a training program is largely realizable only after you are well into your project.

The 132 classroon hours that I allocated for training at the front-end of my project, was excruciatingly difficult to justify at the time. It appeared for several weeks that the project was making no progress in accomplishing its real objective of designing and implementing a software program. However, I now firmly believe that the time spent on the basic fundamentals of software design reaped enormous benefits later in the program. The issue of training must be taken seriously by management as well as the training programs themselves. How well the skills and methodologies are learned by your personnel will greatly affect the success of the project. The training must be effective and the personnel assigned to the Ada project must be aware that management considers the training crucial to success and that they must take it seriously. I believe that training related exercises may indeed be integrated with initial project tasks in a sort of a real laboratory exercise.

At this point, you as a manager would theoretically have qualified people who are trained and capable of implementing an Ada project. The next issue you may worry about are the schedule issues. How can you best be assured that the project is progressing on schedule and whether the

schedule is realistic. There are numerous models and rules of thumb which apply for FORTRAN and COBOL and other languages as to the relative amount of time that is spent during the requirements and design phase of the project, how much time is spent during the actual coding, and how much time is spent during the test and integration portion of the project. I have traditionally used the 40-20-40 rule-of-thumb, where 40 percent of the time is allocated to requirements and design, 20 percent to code, and 40 percent to test and integration. It is my experience, however, that when implementing a program in Ada, significantly more time and effort should be expended during the requirements and design phase. Possibly as much as 55 percent to 60 percent of the time should be allocated and expected to be spent during the requirements and design phase. Only 15 percent of the time need be allocated to code, and 25 percent to 30 percent of the time should be spent in testing and integration.

It has been the experience of people on my project, that if the Ada code compiles, chances are good that it will run. I had teams of workers implementing different elements of the program. All of the parts of the program had to work successfully together in order for the entire project to work. We have found that the test and integration phase is extremely shortened using Ada. If a module compiles, chances are very high that it will run except if there are design errors which go back to the extra time spent for requirements and design. You must ensure that your design is correct and sound.

There are many features in Ada which will result in the program being accomplished very quickly. One of these features is a concept called "generic." The concept of generics is that you design a template to do a certain function, and each time the template is invoked at various places in the program, it is instantiated or initialized to the values needed for that particular function. Generics are extremely powerful. However, in order to get the most benefit from this feature of the Ada language, a lot of effort must be put in the design of these generic packages. This is an example of how the training and the software engineering elements work together to ultimately benefit you project schedule. Approximately, 50

percent of the program I was responsible for, is implemented in generics, and the success of a particular generic was largely dependent upon the amount of time that was spent in determining the requirements of each instance that generic would be used and ensuring that the design of that generic package was sound for all instances.

Another key consideration for successfully managing an Ada project, has to do with creating an atmosphere which is conducive to accomplishment. When management is planning the schedule for an Ada program, there must be enough time at the front-end for the technical people to be accustomed to and familiar with the new language. Progress on the project may be excruciatingly slow during this time period, but as the technical people become more accustomed to the features and capabilities of the Ada language, they will be able to better apply this knowledge during the actual application required in the project. An atmosphere for accomplishment, I believe, will encourage experimentation and pushing the language to its limits. In my particular instance, the Ada compiler which I had available at the time that training was occurring, was of poor quality and several of my people found that the Ada compiler really did not operate in accordance with the Ada Language Reference Manual. They took it upon themselves, as part of their training exercises, to determine all those features of the Ada Language Reference Manual which did work. I encouraged this sort of activity as it broadened their horizons and it held their interest in the project during the period of time that training was occurring. This atmosphere of accomplishment meant that the technical people were not afraid to try things and risk new methods of implementation. They became less fearful of failure and concentrated more on success. This attitude is extremely important to maintain for a successful Ada project. The technical people will engage in frustrations and difficult things, but they must be able to experiment and they must be able to feel the freedom to try new things. You must develop a "can do" attitude in your technical people.

Risks and Cost Considerations

Managing Ada Development **James R. Green**

An area of major concern to management to committing a project in Ada is the unquantified cost associated with it. There will be cost associated with training, there will be cost associated with new compilers and software tools, computers, and there is not guarantee that the people that are hired on the project will be able to accomplish the project. Indeed, the risks to doing a project in Ada are formidable. In order to control these risks, and maintain the project on cost and schedule, management must aggressively be involved with all aspects of the project. By this, I mean you don't have to know how to program in Ada. Indeed, I do not. However, you must be able to relate and understand those concepts and those methodologies which are successful.

Management should consider that doing a project in Ada will involve many risks that are not present in projects using other traditional languages. The risks to be faced are not unmanageable, and as a result, the aggressive manager--that is one who can quickly spot trends leading to success as well as trends leading to failure and can direct actions appropriate to either trend, will be able to successfully complete an Ada project.

Since risk equates to cost, the Ada project manager will want to reduce risk as much as possible. I believe this may be done through prudent selection of personnel, good training programs and agressive involved management.

This short discussion on Managing a Program in Ada has touched only a few of the elements management must be concerned with. These, however, are keys to success.