ADA Accelerates Time-to-Market

Under the heat of competitive time-to-market pressures, Vision Systems, Ltd. of Australia had to find ways to reduce their overall product development time for a new multi-tasking video security system. The company undertook two steps: They established a baseline for their key development parameters and they switched from C to Ada95 as their programming language. As a result, the product development team realized dramatic cost savings in debug, test time, and integration due to inherent advantages in the Ada95 language.

The Video System and Design Targets

The new product, the ADPRO Amux-20CD, records images from up to 20 cameras on a single videotape. The advanced video multiplexing technology allows operators to control as many as 20 cameras from one monitor and display as many as 16 cameras on one screen (see figure 1). The system tapes events in the observed area and in the event of any criminal activity, the system captures taped images of the suspects. The system updates at either 50 (PAL format) or 60 (NTSC) frames per second and delivers resolution of 720 x 576 (PAL inputs) or 720 x 480 (NTSC inputs). A typical installed system consists of the controller box which supports up to 20 cameras and up to four monitors, one or two VCRs, a keyboard, and some alarm detectors. (See block diagram, fig. 1.)

To accelerate and improve their design process, the company began by following the lead of large software development companies like Motorola, Hewlett-Packard, and Boeing. Two years ago, the product development manager began keeping track of defect rates and lines of code per day to establish baseline parameters. From the beginning of the program, all of the developers entered their errors in PVCS (r) Tracker. The program tracks how long it takes to fix each error, the number of errors, and the severity. This information was used to compare current results with historical results.

The baseline parameters identified two potential areas of improvement: First, a design approach that would reduce the defects generated and also reduces the time to find and fix errors. If the design team could achieve these objectives, they would accelerate their time-to-market, giving them a competitive advantage. And second, with productivity improvements to their development process including expanded reuse of code, the company could also introduce subsequent generations in their product line more rapidly.

The development team assessed a switch from the C language to Ada95 for the new product. One of the advantages of Ada is that the it is formally certified by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The standard is called International Standard ISO/IEC 8652; 1995. (1)

No centralized standards committee assures that the C and C++ are compliant with a standard. Additionally, C is notorious for allowing invalid parameters to be used in function calls. When a team is developing code, they work individually for an extended period. In C, it is easy for a developer to make assumptions about how a segment will interface that varies substantially from how it is actually implemented. When integrated, this C code can and does pass invalid parameters across the interface, a significant source of error generation. The Ada language, however, forces each developer to maintain tightly specified interfaces which makes it nearly impossible to pass invalid parameters across the interface. Ada then performs rigorous compile-time checking to test code developed by more than one person to assure that the interfaces are correct. This feature of the language prevents propagation of invalid parameters. The C and C++ languages have no compile-time checking to prevent such errors from propagating.

Because of the increased system complexity of the new security system, the design team wanted to switch from a single developer per project to a team of six software developers. Had they made this switch in C, the project manager realized that they would be multiplying the potential for propagating invalid parameters as compared to Ada. This proved to be a substantial factor in the company's decision to switch to the new language.

For their Ada development environment, the design team had to choose between two vendors, both of whom offered systems for the new processor (PowerPC 860). They selected Green Hills Software's AdaMULTI® development environment hosted on a Solaris Unix workstation. The new system was designed and targeted for a higher-end PowerPC platform, a multi-generational upgrade from the prior Intel 80186-based processor.

Design Team Results

Surprisingly, the design team found that writing a line of Ada did not require any more time than writing a line of C. However, ridding their software production of the undefined pointers and array index out of bounds errors found in C translated to significant savings. Traditionally, these types of errors in C are the most time consuming errors to find and fix. In Ada, the engineers found one error every 270 lines compared to C, which produced one error every 80 lines. Additionally, the time spent fixing errors plummeted from the traditional four hours each in C to only 20 minutes each in Ada as documented by PVCS ® Tracker. (see fig. 2)

Fig. 2. Time to Fix Errors
Comparison based on 10,000 lines of Code.

Language	Lines/Error	Errors/10,000 lines	Time/Error	Total/10,000 (hrs.)
Ada	270	37.04	20 min.	12.35 hrs.
C	80	125	4 hours	500 hrs.

When the design team was ready to integrate completed modules, they experienced a substantial improvement in integration times over prior projects written in C. In fact, integration times collapsed to one day or less with next to no rework in Ada. Although this example is only a single, non-rigorous comparison between an Ada and C project, Ada showed a 3:1 improvement in integration time over C.

When errors occur in C, they are often hard to find, may corrupt other code, and may cause the system to crash. It can take the test engineer a long time to back track to the error from a crash. However, when an engineer using integrated error-checking in the project's Ada development environment identified a run-time error in the application, the system brought up that piece of code in the debugger. This allowed the engineer to view the state of the application when it occurred. The ability to error check without causing corruption or system crash is another inherent advantage of Ada.

The design team was able to complete testing on the new product in less than four months. By comparison, the prior product written in C took nine months to test. The Product Development Manager believes that the company can shorten test times on their next project by implementing reuse more broadly. The company used a good Object Oriented design, which Ada supports, with the idea that they would implement substantial reuse. The project manager had hoped to achieve up to 50% reuse in the second project, but results have been closer to 20%. From the original project approximately 10% of the Ada packages were reused without change, and a further 10% were reused with minor changes. However, the improvements made in the second project, have flowed back into the first, so efficiencies are still being gained. As the code base matures the degree of reuse should continue to improve. The team did experience faster program development times because the language, tool set, and development tools were all reused.

The Ada design environment also included the ability to seamlessly debug in both Ada and C, a big advantage on many projects. However, this project had only minimal C in the shared code environment. The design team wrote the PowerPC almost entirely in Ada. The only C was for the RTOS and interrupt support. A second processor, which handled the lower level routines for video management was written in C because it was too small a processor to be a supported target for Ada.

Summary

The company began implementing productivity improvements over two years ago by establishing a baseline for defect rates and lines of code per day. The original statistics applied to new vision systems developed in C, but this project written in Ada95 allowed management to make a direct comparison between the two languages. Because of its tightly controlled interfaces, Ada eliminated errors arising from invalid parameters, a notorious problem in C. This also resulted in dramatically reduced defect rates, from one defect every 80 lines in C to one defect every 270 lines in Ada. Time spent fixing each error plummeted from four hours in C to 20 minutes with Ada. Integration times for the company's expanded project design team collapsed to one day or less with virtually no rework. And overall testing was cut in half.

Following the completion of the multiplexing controller product, the design team has started two more projects using the Ada language and the same Ada design environment. The first project demonstrated the advantages of Ada and the design environment. As the design team has developed improved expertise, they have built their own set of reusable components in Ada. According to the project manager, the team's next productivity improvement will increase their utilization of the large library of public domain Ada modules for reuse in their designs.

[Sidebar]

What is Ada and how does it compare to C (and C++)?

The Department of Defense named a software engine "Ada" in 1979 in honor of Lord Byron's daughter, Ada Byron, Lady Lovelace. Raised as a mathematician by her mother, Lady Lovelace began working with Charles Babbage, inventor of the Analytical Engine. She suggested a plan for him to generate Bernoulli numbers on his machine. This plan is generally considered the first software program ever written. (2) Today's current version of Ada, which was released in 1995, is generally referred to as Ada 95 and is an object oriented language (OOL).

The following three points, error prone C and C++ notation, run-time errors, and real-time applications just begin to scratch the surface of the advantages of Ada over C or C++.

1. According to the Ada 95 Reference Manual, "The rules of Ada require that program variables be explicitly declared and that their type be specified. Since the type of variable is invariant, compilers can ensure that operations on variables are compatible with the properties intended for objects of the type. Furthermore, error-prone notations have been avoided, and the syntax of the language avoids the use of encoded forms in favor of more English-like constructs. Finally, the language offers support for separate compilation of program units in a way that facilitates program development and maintenance, and which provides the same degree of checking between units as within a unit." (3) This reference is to the error prone notation in C and C++.

2. In regard to pointers and arrays, C++ and Ada differ sharply. "C++ and C follow the same approach: Arrays are closely related to pointers, and the indexing operation is described directly in terms of pointer arithmetic. This approach has several consequences, one of which is that the indexing operation is not automatically checked (as it is in Ada). There are no indices out-of-bounds checks (the most common run-time error). This makes use of arrays in C or C++ inherently less safe than Ada…. Programming without index checking is much like driving without seat belts based on the optimistic conviction that most drivers are sober." (4)

3. Ada was the first mainstream language to incorporate constructs for multi-tasking and for real-time programming. Is still virtually the only language to do so in a coherent fashion. Ada 95 has much more developed real-time and synchronization facilities, in particular, constructs for data synchronization (protected records). There are, as yet, no winning proposals for tasking facilities in C++ and no hint of notions of real-time and scheduling tools. In this area, C++ is a non-starter, even compared with Ada 93. Ada 95 provides tasking facilities for asynchronous transfer of control, protected records, better user access to scheduling primitives, additional forms of delay statements, and

parameterized tasks. In the area of real-time programming, Ada 95 is still without any serious competition. (5)

[End Sidebar]

References:

1. Reference found at: www.adahome.com
2. "Who is Ada?" at www.adahome.com
3. "The Ada 95 Reference Manual, Introduction", www.adahome.com
4. "Contrasts: Ada 95 and C++," The Ada Information Clearinghouse, www.adaic.com. This paper has a much more detailed comparison than space allows here.
5. Ibid.

* * * END * * *

(Word Count: 2,067)