Ada Experience Report for BlazeNet, Inc

Mike Kamrad
BlazeNet
1671 Worcester Road
Framingham MA 01701
01.508.370.4343x139

kamrad@blaze-net.com

1. ABSTRACT

BlazeNet is a new data communication company, whose first product line, AppSwitch^{TM}, will be released this spring. AppSwitch^{TM} is a Layer 7-application switch designed to provide application Quality of Service (QOS) and Operational Automation for data communication networks by using *Automatic Application Flow Switching*. Ada was chosen as the programming language for AppSwitch^{TM} software to provide high reliability and portability. This short report summarizes the reasons for Ada's selection and its use on the product.

1.1 Keywords

Layer 7-application switch, embedded system software, language features of Ada

2. INTRODUCTION

BlazeNet is a new data communication company, whose first product line, AppSwitch^{TM}, will be released this spring. AppSwitch^{TM} is a Layer 7-application switch designed to provide application Quality of Service (QOS) and Operational Automation for data communication networks by using *Automatic Application Flow Switching*. With *flow switching*, AppSwitch^{TM} automatically prioritizes network traffic according to the application to which the traffic belongs. It is a coordinated hardware (BlazeFire^{TM}) and software (BlazePath^{TM} and BlazeView^{TM}) solution that has the capacity to provide high performance *flow switching* as well as the assistance to users to easily (re)configure the AppSwitch^{TM} to meet the user's communication needs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. SIGAda '98 Washington, D.C., USA © ACM 1-58113-033-3/98/0011...$5.00

3. WHY Ada?

Ada was chosen as the programming language for AppSwitch^{TM} software to provide high reliability and portability. This software must work right the first time and continue to execute in the presence of faults and software reconfiguration. The software for the AppSwitch^{TM} is expected to have a long lifetime, therefore the programming language must be able to be ported between different target machines and generations of target machines. Ada has the best combination of language features for both high reliability and portability.

GNAT was chosen as the Ada implementation because it is based on GNU GCC technology, which is available on the widest variety of hosts and targets. AppSwitch^{TM} is a multiple processor product, with the target computers, including the MPC860 and the Argonaut RISC Core (ARC) computers. The MPC860 is a PPC-based member of the Motorola QUICC family which consists of the CPU and additional hardware features for handling communications. Argonaut is a British company that develops hit computer games and relevant enabling technologies. The ARC is a powerful and flexible RISC microprocessor solution for embedded applications. Provided as a "soft macro", which is fully supported with industry standard tools, ARC provides customers with a cost-effective design ready-to-use or which you can configure to meet your performance and cost needs. BlazeNet has tailored the ARC to meet the specific needs of the AppSwitch^{TM}.

4. THE USE OF Ada

Among the multiple CPUs, the GNAT support software has been configured to run in both single task and multiple task execution modes. In the single task execution mode, the Ada application software runs "bare machine" with the support of a runtime operating system. In the multiple task execution mode, the Ada application software runs on top of the Real-Time Executive for Military Systems (RTEMS). The nature of the application and the need for efficiency has dictated that the use of multiple task communication and synchronization be kept simple and straightforward. As a result, the communication and synchronization needs are very nearly like those defined in the Ravenscar Profile [1][2]

Besides the restrictions on task communication and synchronization, the use of Ada made little or no use of the following features, for reasons that are fairly obvious:

- Real types: IT makes no use of these types
- Goto, although labels were used extensively
- Use type clause, primarily for lack of expertise
- Package Exceptions: other mechanisms were used
- Child generic units and formal package parameters: primarily for lack of expertise
- I/O, except Stream_IO
- Annexes E-H

It is worth noting that we chose to avoid features of the language that make use of secondary stacks. This means that no functions were used that returned unconstrained types. As an example this restricted our use of Strings; we had to fall back on a constrained type that would handle strings of characters. Once you get accustomed to String type, it becomes hard to give them up.

Otherwise the design of the software made use of most of the language. Several features of the language had a significant influence on the architecture and design of the software:

- OOP: This had a profound effect on the system architecture; it cleaned up the important relationship among "modules", "interface" and "state blocks".
- Root_Storage_Pool type: This feature enables the software to totally manage dynamic state data.
- Controlled and Limited_Controlled types: Similarly, this feature enables the software to form "closure" on the resources the software units used.
- Interfacing to C: There is extensive legacy code that would be hard to ignore.
- Interfaces to hardware: Obviously this is needed for an embedded system like this.

5. REFERENCES

[1] Baker, T. and Vardanega Session Summary: Tasking profiles. in Proceedings of the 8th International Real-Time Ada Workshop, ACM Ada Letters (1997), 5-7.

[2] Burns, A. and Wellings, A.J. Restricted tasking models. in Proceedings of the 8th International Real-Time Ada Workshop, ACM Ada Letters (1997), 27-32