Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I
IMECS 2009, March 18 - 20, 2009, Hong Kong

Ada's Vital Role in New US Air Traffic Control Systems

Alok Srivastava *(Member IAENG)*, **Fredrick Woodard,** and **Jeff O'Leary**

Abstract* — Development of high-integrity software requires a programming language that promotes good engineering across domains in highly reliable real-time systems. The new Ada 2005 has introduced more robust capabilities based on user experience. The language offers particular innovations which helps make safety assurance less costly and further improves high integrity features in all major facets. The US Federal Aviation Administration's (FAA) current operational core En Route air traffic control (ATC) systems Display System and User Requested Evaluation Tool use Ada as the primary language. The replacement of the huge En Route Host System, called En Route Automation Modernization (ERAM) is currently completing development and beginning deployment with a majority of critical real-time applications in Ada. These systems were completed ahead of schedule and under budget. The paper discusses the evolution of Ada 2005, its vendor support and several excellent academic initiatives. Discussions include the architecture and software of the ATC systems developed in Ada. Since the FAA has a long-term vested interest in the language for mission-critical systems with a very high availability requirement, it continuously monitors the long-term supportability of Ada and trends in its commercial product evolution.

Index Terms — **ATC, Ada, Reliability, Safety, Security**

I. INTRODUCTION

A. Ada's Evolution

Many languages, such as Sun's Java, Microsoft's C#, and Visual Basic are in a great deal prejudiced by companies. Languages such as Ada are instead specified by ISO with no direct control or enforcement mechanism other than market. The Ada language standard was originally designed in early 1980s (Ada 83) to meet demands of high-integrity systems and a revision in the mid-90s (Ada 95) enhanced its support to full Object Oriented Programming (OOP). Over the decades, the developers of programming languages have been learning from Ada. It has influenced the development of Java, C++, Visual Basic, and even the Microsoft .NET Framework.

In June 2006, an international open forum of International Organization for Standardization (ISO), the Joint Technical Committee (JTC) with 28 participating countries, and another 42 countries as observer and its sub group Working Group (WG) 9, approved the specifications of Ada 2005 with major objective to enhance Ada's position as a safe, high performance, flexible, portable, interoperable, concurrent, real-time, and object-oriented language. Now it offers more safety and portability than Java, and better efficiency and flexibility than of C/C++. In addition, it is an open international standard for real-time and high-integrity system development. Ada implementations, tools, and libraries are available on a wide variety of platforms. Nevertheless, Ada has not enjoyed the commercial success or publicity of other languages [1] most likely due to misperceptions with the relaxing of U.S. and NATO defense industry language mandates.

The new features of Ada 2005 further enhanced the Ada 95 capabilities in areas such as Extension to the Open, Predefined Libraries, Interface Approaches, Enhanced Encapsulation, Access Types and Dependency issues. Ada 2005 adds to the standard packages: environmental variables, time access and manipulation, file/directory manipulation, containers and sorting, wider characters and linear algebra to the standard packages. The version also enhances the earliest deadline first, real-time scheduling, round-robin real-time scheduling, and most importantly The Ravenscar high-integrity, run-time profile. It now supports the notions of interface used in languages such as Java and C#, and architectures such as CORBA. Other enhancements include active and passive synchronized interface types and module/object encapsulation. Addressing a fairly common user need, Ada 2005 adds support for cyclic dependence between types in different packages [2].

B. US Air Traffic Control

The FAA's mission is to provide the safest, most efficient aerospace system in the world by promoting aviation safety and mobility through operating the United States air traffic control system. FAA controls more than half of world's air traffic. Because of its preeminence in high availability and high reliability domains, the Ada software language has become a strategic technology in developing and sustaining those systems [3]. In fact, the FAA's major recent ATC initiative, the vast ERAM program, leverages and significantly expands upon previous Ada based systems so that the FAA has a vital interest in Ada language technology now and for the foreseeable future. The software of the large ERAM System, one of the largest, latest civilian system development projects contracted by the US government has recently been developed ahead of schedule and within the budget.
http://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=7714 The major En Route air traffic control systems are

Alok Srivastava, **Ph.D.**, Northrop Grumman Corporation, 475 School Street, SW, Washington DC 20024 (USA) Tel. 1+202-314-1419 Alok.Srivastava at auatac.com

Fredrick Woodard, Northrop Grumman Corporation, 475 School Street, SW, Washington DC 20024 (USA)

Jeff O'Leary, US Federal Aviation Administration, 800 Independence Avenue, SW, Washington DC 20591 (USA)

Display System (DS), User Requested Evaluation Tool (URET) and ERAM and have been developed mainly in the Ada programming language. Lockheed Martin is the prime contractor of these systems with Computer Sciences Corporation (CSC) as major mission subcontractor for the DS and URET. ERAM development team is led by Lockheed Martin with Raytheon and CSC as major subcontractors. Northrop Grumman's role is to act as Technical Advisor with responsibility to acquire the ATC systems for the FAA.

II. En-route Air Traffic Control Systems

A. Display System (DS)

The Display System (DS) is a major element of the FAA's modernization program which replaced the 30 year old display channel computer complex and dedicated workstations with a local area network-based client-server network of air traffic controller workstations connected to a Host Computer System (HCS). The US $1B plus DS project was completed on time and within budget in May 2000. These displays are used by En Route air traffic controllers at Air Route Traffic Control Centers (ARTCC) nationwide. Its computing infrastructure can support more than 200 workstations and 65 operational sectors of airspace in a single ARTCC. This increase in operational capacity allows the FAA to handle rising traffic loads while maintaining current high levels of service. The three types of DS consoles as shown in figure 1.

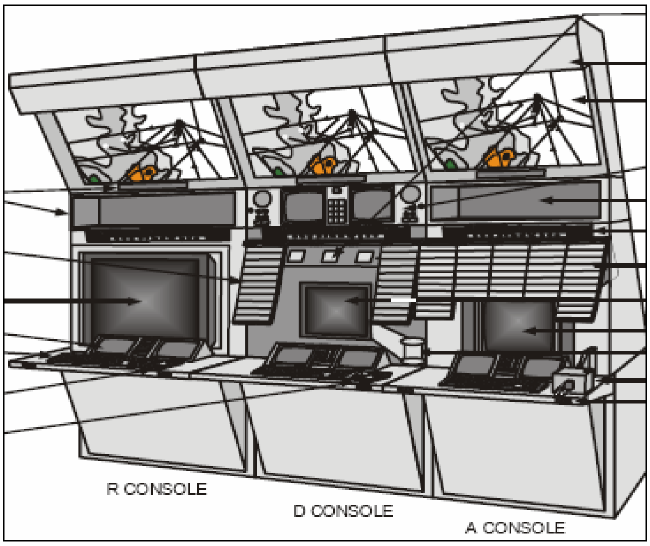

Figure 1 - Three types of DS consoles

The system's fault tolerant software, in combination with redundant hardware and networks, achieves high availability consistent with its critical mission. The new system management functionality includes comprehensive system status and performance monitoring and control, continuous on-line data recording for resource problem determination and trend analysis, and access to both primary and backup networks from any system management console.

The total software size of DS at the time of initial deployment was 794,000 SLOC. The operational environment includes 444,000 SLOC in Ada83 plus a small amount of C code in a distributed UNIX Environment. The support environment includes 350,000 SLOC, the majority in Ada83 plus C, FORTRAN and UNIX Shell languages.

B. User Requested Evaluation Tool (URET)

Current ATC operations in many areas of the United States are highly structured and restrictive. This is in conflict with airspace user's desires for more flexibility, i.e., the freedom to fly more preferred routes and altitudes from origin to destination. The URET is a software program to alert the controllers of potential conflicts or loss of safe separation. The program tool includes: a Conflict Probe (CP) function and a set of automated tools to assist with problem resolution, alternative route planning, and information management. It is installed in the controller's D-Position of the DS console. URET is also better known to controllers as "electronic flight progress strips". It drastically alters the way that air traffic controllers receive and process flight plan information.

URET detects and alerts the air traffic controllers of aircraft-to-aircraft conflicts up to 20 minutes in advance. This "conflict probe" feature uses the system's flight plan and track database, which captures data from the ARTCC's National Airspace System (NAS) host computer and from URET in neighboring control centers. The URET also gives controllers an alert 40 minutes in advance of when an aircraft is predicted to penetrate restricted or prohibited airspace.

The URET incorporates unique Fault Tolerance features. Its console design and system architecture minimize operational impact from system problems. The URET installation involved both the system's commercially available hardware and more than 620,000 SLOC mainly in Ada 95. This system is allowing freedom to pilots to fly more preferred routes and altitudes resulting in significant fuel and time savings and superior conflict probe.

C. En Route Automation Modernization (ERAM)

The current Host Computer System (HCS) operating in the FAA's twenty En Route centers has been the backbone of the US National Airspace System (NAS) for thirty five years. However, the HCS software, primarily written in Jovial and Basic Assembly languages, and its architecture, place increasing limits on the ability of the NAS En Route domain to accommodate growth and new functionality. Finally in 2003 the FAA awarded a contract with initial worth of US $2.2 Billion to replace the existing NAS Host and Direct Access Radar Channel (DARC). ERAM adds improved capabilities of NAS Architecture, Free Flight Initiatives, Advanced Communication, Navigation, Surveillance, Information Management, and Decision Support Technologies that can now be applied to Air Traffic Management (ATM). It will replace the existing mainframe-centric host architecture with a state of the art open and supportable environment.

The 1.3 million SLOC software developments with majority application in Ada have been completed ahead of schedule. The system is currently undergoing through rigorous testing by the air traffic controllers. The operating system software and the interface between application software and operating system conform to the Portable Operating System Interface (POSIX) standard. The system provides enhanced security through a defense-in-depth architecture. The overarching requirements are to provide data confidentiality, data integrity and system availability. The architecture provides multiple layers of security including controls on the external networks, network perimeter, internal networks, operating system, security middleware, and application programs. The software is also designed to verify the syntactic and semantic correctness of commands, inputs, and data from external systems, whenever possible. This verification prevents data triggered software failures and the corruption of valid data. To meet the Reliability, Availability and Maintainability (RMA) requirements of down time of few seconds in a year under emergency mode, the system provides two fully redundant channels. The architecture provides separate platforms for safety critical versus efficiency critical versus routine services to align with the separate availability and recovery time requirements. Accordingly, the architecture provides separate networks for operation control versus maintenance function communication. Product supportability advantages led to the selection of the IBM P series processors, the AIX operating system, and CISCO switches. The system interfaces with a variety of external systems. From airspace users, ERAM receives proposed flight plans. For aircraft that cross facility boundaries, ERAM shares flight data and flight coordination and control data with U.S. and non–U.S. ATC facilities. ERAM transmits active flight data to U.S. government agencies that monitor flights at U.S. borders. Flight data is also transmitted to other systems in the ARTCC. The system receives radar target reports as well as radar precipitation reports from surveillance sources. It receives non–surveillance weather data from the Weather and Radar Processor and the Weather Messaging Switching Center Replacement. ERAM receives Notices To Airmen (NOTAMs) from the Consolidated NOTAM System IP Server.

Each external interface is owned by a single ERAM Subsystem; functioning of that Interface Control Domain is confined to the owning subsystem. ERAM communicates with external interfaces independent of language in which the external system or ERAM interface proxy is developed. The subsystems are decomposed into Computer Software Configuration Items (CSCIs) with clearly defined interfaces and dependencies. The following considerations are used to determine the functional allocation to CSCIs.

From a logical view perspective ERAM software encompass

- Functional cohesion (i.e., do not combine unrelated functions) and encapsulation
- Loose coupling between CSCIs (helps limit the impact of changes to a single CSCI)
- Extensibility and reuse considerations
- Simplicity of interfaces between CSCIs.

From a physical view perspective it can be encapsulated as Failure modes of the CSCI in relation to other elements (i.e., ability to fail independently from other components in order to minimize the number of reconstitutions needed with external systems); failure mode considerations can have the additional benefit of isolating the amount of function that is failed from the user's perspective. In addition to CSCIs, significant elements of the architecture are common shared services (CSSs) which are design–level entities used by multiple CSCIs to provide common services.

Software layers are enforced from a software dependency perspective: each layer may use services only from its own layer or lower. Services have application programming interfaces (APIs) that are invoked either synchronously or asynchronously, via dynamic dispatching (or callbacks). Since the system is distributed, there is data passing between nodes in the network architecture; the messages are encapsulated into functional components, thus shielding the users of the services from implementation details and from changes in data layout. Client-side library code is provided by each functional component and is bound in with the client. It is the client-side library code that presents the APIs. These support Ada and C++ clients for ERAM; other languages could be added if the need were to present itself.

D. Publisher Frame Work Methodology

One of the challenges in building ERAM with distributed fault tolerant systems is keeping application code size and complexity down. This has been done by capturing the nuances of distributed Publisher Frame Work (PFW) [4]. The Publisher Frame Work (PFW) provides a framework for uniform, consistent development of software components. The design pattern as shown in Figure 2, implements support for:

- A server to publish objects to subscribers and to process requests from clients.
- An agent acting as a local subscriber to receive published objects, translate them into messages and multicast them to all remote subscribers.
- A proxy to receive multicast messages, translate them back into objects and republish them to local clients.
- A mirror to augment the proxy by retaining a copy of the data published by the server for use in local queries.

In ERAM there is a plethora of requirements implemented by PFW on behalf of all components that are less interesting to describe in this paper, yet helpful to the component implementers just the same; one example is invoking the recording service and error reporting service (to log commonly recorded events and data) for detected errors.

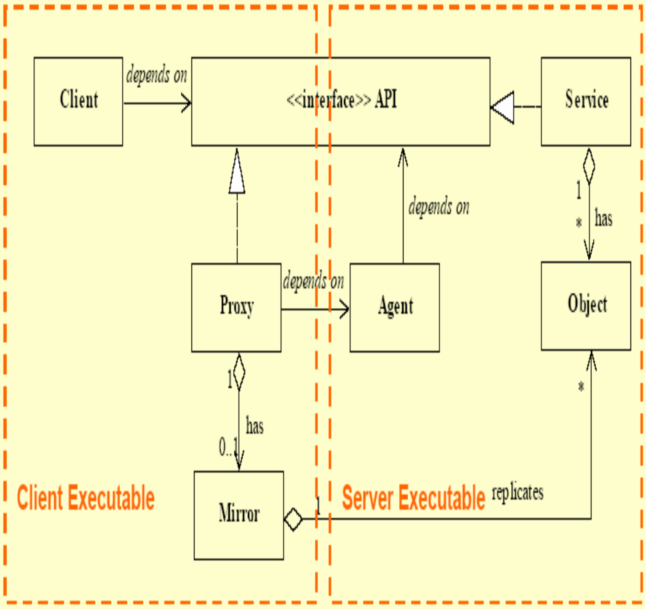

Figure 2 - Publisher Frame Work Concept

E. Some Lessons Learned in ERAM's Software Development

The positive lessons learned in using components are that Factoring out common behavior saved code overall. Especially the Reuse of Proxy/mirror code in many clients saved all those clients from developing similar code many times over, saved those clients from debugging that code many times over. In addition "Clean" one-way dependency rules between components help build the system in an orderly fashion. On the other hand Interfaces between components Application Program Interfaces (APIs) are still volatile, still changing as features are added, learning what should be in the API vs. internal to a component is ongoing. Structuring a component such that the minimum sufficient information is in the API is still ongoing (could have set up stricter controls/rules up-front to facilitate better decision-making). Lessons learned specific to developing systems in Ada includes Ada Semantic Interface Specification (ASIS) interface is very useful and utilized heavily for data exchange specially in

- Operational software (recording data) and Support software (interpreting recorded data)
- Support software (generating adaptation data in compact, binary data) and operational software (reading in the adaptation data)
- Ada software exchanging data with C++ software (crossing language boundary and passing data along)

Ada results in code that is easier to debug due to strong typing/range checking. Much easier to debug an index being out of range when an exception is raised on the first attempt to use it as opposed to stomping on random areas of memory.

On the other hand

- Junior personnel perceived that they are working with "aged" technology, that the skill acquired is not transferable/marketable
- Passing exception across language boundaries (between Ada and C++) is a lot of work; Pragma Interface didn't consider this aspect.
- There is a related issue about calling languages (like C) that do not support exceptions – one has work on all of those, map to return codes.

III. FAA's Monitoring of Ada

Since the FAA has a long-term vested interest in Ada, it monitors the developments in the Ada world. The Ada related annual conferences such as the ACM SIGAda Conference, Ada-Europe are occurring regularly and successfully. New Ada events such as European Ada Working Group conference held in the United Kingdom (UK) in late 2007 was attended by some 120 participants. It is encouraging to note that membership retention rate of ACM's Special Interest Group on Ada (SIGAda) is 80%, the highest among all ACM SIGs and has been financially and technically thriving. Positive feedback in conference evaluation forms indicate that people come for technical sessions, to hear presentations by recognized speakers, and to network with others. It is attracting new vendors to the exhibit hall and has a mix of technical papers, workshops, sponsor presentations, and invited presentations with international participation. The Ada Europe conference has been held for 19 consecutive years. These events appear to be very successful with wide participation from academia, industry, government agencies and IT associations such as ISO.

A. Research in Ada Software Engineering

Research and Development in Ada Software engineering is ongoing. The main journal on Ada published in the USA, ACM Ada Letters, in addition to providing technical articles, publishes Ada Issues (queries and clarifications of the Ada standard), Proceedings of the SIGAda Conference, and Proceedings of the events such as International Real-Time Ada Workshop. Another major Ada Journal is "Ada Users Journal" published by Ada-Europe, which is a federation of several national Ada societies of major European countries including UK, France, Spain, Italy, Belgium, Denmark, Netherlands, Sweden, Switzerland and Germany. This journal has been published regularly four times a year with a wide variety of papers on the Ada programming language, its use, general Ada-related software engineering issues and activities related to Ada.

B. Availability of Ada Programmers

A recent research report in IEEE Software by Chen [5] weighed a multitude of factors and determined that Ada is not likely to decline in popularity among programmers in the near future. The divisions of major FAA contractor Lockheed Martin, CSC, Raytheon and other contractors developing Ada based systems have not reported any difficulty in finding or training in-house software professionals to program in Ada. Teaching Ada to programmers who know other languages has proven to be straightforward.

C. Academic/Industry Joint Ada Programs

Ada academic community and vendors offer several free excellent initiatives because it is whispered that Ada is the right choice for courses in elementary programming, data structures, software engineering and for more advanced courses in compiler construction. Ada embodies the best contemporary ideas in software technology, and students exposed to Ada at an early stage of their career become more skilled and principled programmers. An increasing number of programs offered by Ada vendors encourage university students to gain Ada experience prior to entering the work force. For recent data shows that AdaCore [6] has a major academic initiative with more than one hundred and forty universities in 25 countries participating. Fifty-five U.S. universities including Massachusetts Institute of Technology, New York University, Carnegie Mellon University, Pennsylvania State, George Mason University and the University of Texas participate in this program.

Prof. Michael Feldman, Chair, ACM Ada Education Working Group lists many of the Real-World Projects in wide areas from web applications, medical industry to data communications, developed in Ada at the website http://www.seas.gwu.edu/~mfeldman/ada-project-summary.html

The developer of the subset (simplified) version of Ada “Spark Ada” Praxis High Integrity Systems UK http://www.praxis-his.com/sparkada/index.asp have also reported in significant growth in their academic participation program. They have been appointed by UK’s National Air Traffic Services (NATS) to develop software for a large new system called iFACTS which will trigger the biggest change in ATC since the introduction of radar. Prof. John McCormick has reported a steady growth in selling of his book “Ada plus Data Structures: An Object-oriented Approach book” with significant boost in India. FAA is pleased to learn that Boeing’s recent aircrafts including Boeing 787 and others continue to broadly use Ada in their on-board systems. It is also true for the Airbus’s A380 and other modern commercial and military aircrafts. Boeing and Airbus both have announced major assembly and flight simulation training facilities in China and India.

IV. CONCLUSION

The FAA’s core En Route air traffic control systems DS and URET were developed in an unprecedented short time and within the budget using Ada programming language. These systems are performing very well at every En Route center. Software development of the new, larger ERAM system has also been completed on schedule and within budget. Ada code is easier to debug due to strong typing/range checking and facilitated in FAA’s drive to find the defects earlier in the development. The FAA does not endorse any specific programming language but worldwide Ada appears to meet the expectations of those requiring a high integrity reliable software language for real-time systems. Ada based systems are the mainstay to FAA’s En Route air traffic control systems in the twenty first century. It also appears Ada 2005 is well posed to meet the more advanced and robust systems demand in the future.

ACKNOWLEDGEMENT

Authors wishes to thank FAA ERAM Program Manager Daniel Watts, FAA ERAM Engineering Manager Robert Hanes, Lockheed Martin Software Engineering Leads Judith Klein and Ross Rader for permission to share the information, data inputs and valuable suggestions.

REFERENCES

[1] Ben Brosgol, Ada 2005: A Language for High-Integrity Applications, CrossTalk, August 2006
[2] Richard Conn (Microsoft), Ada 2005, CrossTalk, August 2006
[3] Jeff O’Leary of FAA, Keynote Presentation at 2007 ACM SIGAda Annual International Conference, November 4-8, Washington DC, USA
[4] Judith Klein (Lockheed Martin) and Drasko Sotirovski (Raytheon Corp), Publisher Framework (PFW), ACM Ada Letters, August 2007, pp 12-22,
[5] Chen, Y et al. An Empirical Study of Programming Language Trends. IEEE Software 22, 3 (May/June 2005: pp72-79)
[6] AdaCore, a GNAT Pro Company, New York, NY