SIGAda '98, Workshop: How do We Expedite the Commercial Use of Ada?

Robert C. Leif, Ada_Med, a Division of Newport Instruments
5648 Toyon Road, San Diego CA 92115
(619)582-0437, E-mail rleif@rleif.com

1. Introduction:

The goal of the Workshop was to promote the use of Ada for commercial off-the-shelf software. Robert C. Leif was the Coordinator. The Workshop consisted of four talks, which were followed by a lively discussion. The four presentations were: 1) "Ada Developers Cooperative Draft License" by Robert C. Leif, Ada_Med; 2) "Ada: A Commercial Flop and Proud of It" by Dave Wood, Aonix; 3) "Ada Database Opportunities" by Gilbert Prine, AdaMATION, Inc. and 4) "Potential New Markets" by Robert C. Leif. Since these presentation descriptions were written after the SIGAda '98 meeting, they include items mentioned during the discussion and other relevant points raised during the meeting. The rest of this article is based on the contents of presentations 1 and 4. The content of presentation 2 is described in a separate article[1] in this issue of Ada letters.

2. Ada Developers Cooperative Draft License by Robert C. Leif, Ada_Med:

Often, the first step in solving a problem is to objectively examine the sources of the problem. Thus, it is necessary to determine why the best technical choice for a programming language, Ada, is a relative commercial failure? Two Critical factors required for commercial success are utility and strong economic benefit. Obviously, Ada's market share is NOT the result of a lack of utility of the language itself. If Utility were the only criterion, there would have not been a need for the workshop. However, economic benefit (profitability) can easily overshadow utility.

An explanation for the lack of acceptance of Ada can be deduced from a study by Capers Jones [2]. This study compared the level of programming languages (Language_Level) and the average number of source statements per function point (Source_Statements_Per_Function_Point). The level of language is supposed to be inversely proportional to the amount of labor required to produce a program. Function_Points := 4*Inputs + 5*Outputs + 4*Inquiries + 10*Data_Files + 7*Interfaces;

Jones has pointed out [3], [4] that the function point formula needs a complexity adjustment and has built upon the concept of function points to create his own metric, feature points. Unfortunately, the very abbreviated set of his data shown in the table below was not reported by Jones in feature points. The best fit with Microsoft® Excel 97 for a subset of 100 languages taken from Jones' data was:

$$\text{Source\_Statements\_Per\_Function\_Point} = 286.91 * \text{Language\_Level}^{-0.9585}$$

From Programming Languages, Table Release 8.2, March 1996 [2]
By Capers Jones, Chairman, Software Productivity Research, Inc.

Language	Level	Average Source Statements Function Point
Assembly (Macro)	1.5	213
ALGOL 68	3	107
Turbo Pascal >5	6.5	49
Modula 2	4	80
Ada 83	4.5	71
Ada 95	**6.5**	**49**
Eiffel	15	21
JAVA	6	53

From Programming Languages, Table Release 8.2, March 1996 [2]
By Capers Jones, Chairman, Software Productivity Research, Inc.

Visual C++	9.5	34
EXCEL 5	57	6
LOTUS 123 DOS	50	6
LOTUS Macros	3	107
MATHCAD	60	5

Please notice that Ada 95 is about in the middle of Jones' table with a language level of 6.5 and requiring 49 lines of source statements per function point. The obvious, albeit superficial, conclusion from this data is that the Department of Defense should have mandated EXCEL. However, a second point, which is true for many "High Level" languages, is shown for LOTUS Macros, which are only twice as efficient as a macro assembler. The benefits of the use of automated code generation can be offset by the costs of extending or tailoring the output of the code generators.

In reality, Jones has actually proven that domain specific libraries are useful! In fact, the creation of source text, even with an Ada 95 compiler, is a very inefficient and expensive process compared to employing a domain specific tool or library. Spreadsheets, such as Excel, and tools like Mathcad are the most efficient and rapid way to obtain a result. The best situation would be to have domain specific tools with an Ada interface and domain specific libraries written in Ada. It should be noted that the quality of these tools and other commercial off-the-shelf products including operating systems would be improved by employing modern software engineering technology including the use of Ada.

Many of the present commercial off-the-shelf software products are large, expensive undertakings. At present, none of the large organizations producing commercial off-the-shelf software use Ada. Instead Microsoft's competitors have assured Microsoft's dominance by employing essentially the same tools as Microsoft. The development of a commercial software product is often facilitated by the use of preexisting libraries, for instance those supplied by Microsoft. Most of these libraries, at present, do not exist in Ada. At best, Ada bindings are available. Therefore, the problem is:

How do we motivate people to create these domain specific and other Ada libraries?

2.1. Motivations for creation of Ada libraries:

There are, at present, 3 main motivations to create Ada software: altruism, risk-free remuneration, and software development as capital investment.

2.1.1 Altruism:

The acclaim and honor received from one's peers for contributing one's work is sufficient reward. However, many creative individuals have responsibilities to support their spouses or significant others, children and/or other relatives. Often when an initially no cost product becomes successful, altruism disappears and the product is transformed into a commercial, paid for product. This is really not very different from giving away the initial beta releases of a new product. The real success of these products is based on their conversion into commercial products.

Compensation for software can be based upon payments for support. This is just another mechanism for making the customer pay. For instance, the yearly cost of a "free" compiler can be greater than that for a conventional commercial compiler. For Ada libraries, the prospects for obtaining revenue for support can be diminished by the ironic problem that well engineered, readable source text permits the user to do their own maintenance and make fixes or create enhancements.

2.1.2 Risk-free Remuneration:

This usually is limited to Government contracts. Because the number of copies is limited, the price per copy is exorbitant compared to that of commercial off-the-shelf products. This is the reason that the US Department of Defense is taking the risk of using commercial products including operating systems from Microsoft, while simultaneously the

Department of Justice is suing Microsoft. William Gates et al. have proven that mass marketing with comparatively low prices is the present best way to maximize profits.

2.1.3 Software Development as Capital Investment (Sweat Equity):

Presently, it takes a very large organization with large amounts of capital to create a commercial application. The combination of Ada and the Web offers a remarkable financial opportunity. It should be possible for a group of independent developers to create applications without a major investment of capital. The feasibility of this has already been demonstrated by Wirth and Gutknecht with Oberon [5]. This operating system can be hosted under Windows and runs on the Macintosh. It includes electronic mail and an editor. English [6] includes an example of a spreadsheet in his text book. AdaSAGE [7] is a very efficient environment for creating commercial applications [8]. However, none of these applications has been adapted to the new environment, the Web.

Most of the users of commercial software do not need nor can they use most of the facilities of powerful products like Microsoft® Office. Products, such as Microsoft® Works are sufficient for their needs. The real utility of the object model of Ada is the capacity to reuse components among applications and to permit these applications to be extended. Ada is the only language that permits large numbers of developers to work efficiently together.

2.1.4 Requirements for Ada Software Development as Capital Investment:

R1. Minimize the up front costs for new Ada products.

R2. Provide developers with a financial interest of in the reuse of their software.

R3. Assure predictable royalty costs.

R4. Minimize risk connected with the use of Ada components.

2.1.5 Creation of an Ada Developers Cooperative License:

The creation and acceptance of a standard license will be of benefit to both the licensors and licensees because: 1) The availability of a significant collection of maintained, software components covered by a single standard license significantly increases the accuracy of the cost estimates for developing a commercial product. 2) Legal costs are minimized. and 3) The license will provide legal protection for the developers' intellectual property, which should significantly increase the probability that developers will receive future royalties based on the utilization of their work in commercial products. This will enable them to license their libraries at an initial low cost or even no cost (R1). Since the developers will only be able to collect significant remuneration, after their libraries have been successfully included in commercial programs, they will have a very compelling reason to provide assistance in the use of their libraries (R2) and (R4). The creation of an Ada Developers Cooperative License will include a means to objectively calculate the royalties for each owner of an Ada licensed software (R3). Providing the libraries as "Open Source" and the creation of a formal mechanism where royalties can be earned by providing improvements to existing libraries assures that even if the libraries are orphaned by their developers, they can be maintained, improved, and extended (R4).

The creation of an object-oriented library without complete knowledge of its usage in a final product can result in only a small part of the library actually being incorporated into an individual final product. The simplest equitable solution is that royalty payments should be based on the actual amount of software used in the final product. In order to avoid the possibility of large legal and accounting expenses, this determination must be both objective and automated. Although the development of an automated, objective means to determine the contribution of each software suppler to the final project is a major problem, it can be solved by the use of Ada technology. The Ada Semantic Interface Specification, ASIS, includes the functionality to create tools to measure the source text that is included in the final linked executable. This contribution could be calculated as: function points, feature points, linked lines of source text, or any other reasonable solution.

2.1.6 Publication of the Source Text (Open Source):

Firstly, Ada sources should be referred to as text rather than code. None of the libraries to be licensed should require cryptographic skills for their use or maintenance. The unambiguous term, "Open Source", will be used. Open Source means the user will have a simple means to obtain the source text. The means employed to acquire the rights to use the source text are not included in the definition of Open Source. The previous term, "Freeware" was ambiguous in that it had at least two meanings: free of charge, or freedom to read the sources (Open Source). A third possible mean-

ing could be that the developers worked free of charge. Since the benefits of Open Source to the users are well known, the discussion below will be limited to the benefits and costs to the vendors of the libraries.

Benefits to the Vendors:

1. The customer already has, by custom, the Ada specifications, which often include the private section. Together, these can provide significant insight into the design and organization of the packages. Therefore, significant information concerning the design has already been released to the customer.
2. Part of the vendor's responsibility for usability is transferred to the customer, who can review the contents of the packages and can create and perform white box (source code based) tests.
3. The customer will review the packages.
   a) which will often result in an improvement in the documentation and/or design in response to a request for an explanation of part of the source,
   b) discovery of errors and suggestions on how to fix the problem,
   c) or better yet, the customer supplies a patch to fix the problem.
4. Changes in compiler library formats will not require the developers to recompile their packages.
   a) The difficulty of managing compiled Ada '83 libraries was one of the main sources of the negative image of Ada.
   b) Source is relatively compact compared with compiled libraries and can be shipped via the Internet.
5. The open source requirement protects against the copying of libraries. Visual or computer comparison of two libraries supplied as open source should permit detection of common elements. Effective copyright protection requires that competitors publish their sources. Competitors who keep their sources secret will have a very significant competitive disadvantage against those who provide open source.

Costs to the Vendors:

1. Reveals to competitors the design and organization of the packages. This is undoubtedly true. However, see Benefits 1.
2. Facilitates software piracy. However in spite of similar piracy problems, source is provided by many profitable industries including: book publishing, video tape, CD-ROM, and music publishing. Piracy often has an advantageous side-effect, a pirated version often serves as a demo product. Once the customer uses a product, some level of interaction with the vendor is often required, such as information on bugs and updates, etc. This interaction requires purchase of a legitimate copy. There is a symbiosis between the customer and the vendor. A customer, who includes a library in his own product has a vested interest in the vendor of the library staying in business.

2.2. Is it too late?

2.2.1 “Java has won the language war”:

One counter argument is that much of the rest of the software development world has selected Java. Before Java, the same argument was made for C++. A few years ago, there was a strong move to SmallTalk, which was led by IBM. From a cold-blooded business perspective, the fact that the competition has gone off on an other language-of-the-year wild goose chase, Java, is good news. Presently, applications compiled into J-code are sufficiently slow to be at a competitive disadvantage with the equivalent compiled applications. Java crawls everywhere and only runs on specialized SUN hardware. It should be noted that a minimal performance on the client side is totally consistent with the aims of server centric companies, such as SUN and IBM. However, the present exponential improvement in price/performance ratio of personal computers and the limited bandwidth of the Internet serve to force maximizing the use of the client. Ada has the further significant advantage of not being targeted to a specific operating system. Ada can be and has been ported to virtually any reasonable operating system or environment including the Java virtual machine.

Returning to the use of specialized processors, the previous experience of Ken Bowles [9] with the UCSD Pascal system is of significant interest. The P-Code version ran considerably slower than the specially microprogrammed,

Western Digital implementation of the Pascal “Microengine” P-Code on a 16-bit, Digital Equipment Corp. LSI-11 microprocessor; however, ultimately programs compiled into executables ran 3 times faster than the hardware implementation. At present, the increase in the instruction size of microprocessors to 32 and 64 bits, as well as the availability of millions of transistors permits the direct hardware implementation of instructions, which should result in very good performance compared to compiled executables and greatly enhanced performance compared to J code interpreters.

The portability of Ada provides the significant advantage of permitting the customer to migrate applications between server and client and between conventional compiled executables, interpreted J code, and ultimately processors which directly execute J codes.

2.2.2 Microsoft has all of the customers?

The prevailing view in the software community and even in the Ada community is the war is over and Microsoft has won on both the operating system and the major applications. Microsoft has significant advantages: 1) Very good management compared to competitors, 2) a very large share of its markets, and 3) very significant financial resources. Microsoft does have significant disadvantages: 1) Very large organizations have large COCOMO coefficients; 2) their products require upward compatibility in software and user interface and, beyond this, old versions must be maintained; and 3) the highly touted virtues of creating object-oriented software applications in C++ has not resulted in Microsoft® Office being assembled from common parts. Word does not inherit its tables from Excel, etc. The number of add-ins or derivative applications is surprisingly small. In Excel and other programs used in business, money should be represented by decimal types rather than the present floating point. Excel has the further weakness that it does not directly permit columns and rows to be given meaningful names. This often results in formulas with meaningless variable names, such as A1*C1; rather than, Interest_Rate*Principle. Admittedly, Excel will permit formulas with meaningfully named variables to be created; however, this requires considerable effort. The situation is analogous to Ada’s built in range checking versus creating this facility in another programming language.

Lastly, although Microsoft, at the time of this writing is being sued by the US Government for antitrust, in all fairness, one great virtue of Microsoft must be mentioned. Microsoft did not invent either C++ or Java; nor has Microsoft spread propaganda concerning the inherent “virtues” of these languages. In fact besides all the obvious technological reasons, the adoption of Ada would provide Microsoft with both an excellent marketing tool to use against the Java jihad launched by its competitors, SUN, IBM, Netscape-American Online et al. and a means to minimize antitrust problems by conforming to open standards: ISO Ada, the Ada POSIX binding, and W3 XML (Section 3.2.).

2.3. The market for reliable, efficient, extensible, software has NOT been tested!

Most consumer software is tolerated rather than loved. The sophisticated customer purchases a product with the understanding that the guarantee is one sided and ludicrous. PCs crash or start sending messages that they are out of room even though one has just purchased an obscene amount of memory. Many of us routinely reboot our PCs because we know that there are memory leaks. Frustration with the present version of a software product is often the strongest reason to purchase an upgrade. The present commercial software culture sells “upgrades” rather than having recalls. A culture based on arcane obscurity, such as that of the present source languages, C and its derivatives, shows through to the user of the application. This is particularly true when the C hackers are permitted to design the user interface. Many of the present products are unreliable, inefficient and awkward to use. The term user-hostile can be applied to many programs. Artificial-stupidity is a rapidly growing field!

Past experience in other industries, such as the US automobile industry, has demonstrated that following the teachings of Total Quality can have devastating effects on competitor companies which are controlled by complacent management. In short, there is money to be made with commercial, quality, well-engineered products written in Ada.

2.4. Ada, software engineering, and the Web provide software entrepreneurs with an excellent opportunity!

Better yet on the Government’s money! Not only has the US Government funded the development of Ada 95, it will provide money for creating applications. Since much of the Ada community is not employed at a University or other institution capable of applying for standard US Government Grants, the appropriate means of obtaining funding is either the Small Business Innovation Research grant or the Small Business Technology Transfer Research grant. Both of these are directed to include a vendor-sponsored Phase 3 where the product is sold in the commercial market for a

profit. For instance, the NIH Phase 1, STTR Form includes a section, “Summary of the potential commercial applications of the research.” Software produced under the Ada Developers Cooperative License specifically includes royalties, which would be the mode of payment for many of the products. Other licenses contain text which could be difficult to explain to a group of reviewers. Specifically, how a project which gives away the product at no cost and does not charge for its usage will be profitable.

It should be noted that since Ada is the preferred language for weapons systems, the National Science Foundation has an obligation according to the mission statement in its charter [10] to enhance and maintain this technology. The NSF's mission: “To promote the progress of science; to advance the national health, prosperity, and welfare; and **to secure the national defense**.” The Foundation's organic legislation authorizes it to engage in the following activities: including “H. Initiate and support specific scientific and engineering activities in connection with matters relating to international cooperation, **national security**, and the effects of scientific and technological applications upon society.”

3. Potential New Markets:

3.1. Very Abbreviated History of PC Software:

Before one looks to the future, it is essential to review and learn from the past. The changing, fluid, environment of software and computers makes the permanent ascendancy of any company extremely problematical. Microsoft has survived and greatly profited during two software epochs. Amusingly, actions by Microsoft’s competitors, principally IBM, are in large part responsible for Microsoft’s success. IBM first farmed out DOS to Microsoft and then ensured the success of Windows by scaring the clone users and purchasers into believing that OS/2 required the proprietary IBM Microchannel Bus. Evidently, IBM also refused to use Microsoft’s Windows as the front end to OS/2.

The third epoch which is the next step beyond Windows was and is the Internet. This is why Microsoft expended very significant resources to create a product, Internet Explorer, that it gave away. The Internet provided motivation for Microsoft® Office 2,000 to convert to an essentially open file format, HTML. Since the capabilities of HTML are limited, the World Wide Web Consortium [11] has created the Extensible Markup Language, XML [12]. The use of XML should circumvent the convoluted architecture of Windows and provide a portable means to both display information on a monitor and print it. This change to XML provides an opportunity for the displacement of old products by new products including those written in Ada.

3.2. XML (Extensible Markup Language):

XML is a major subset of SGML (Standard Generalized Markup Language; ISO 8879:1986). XML includes well-formed HTML (Hypertext Markup Language), which is an SGML application. The major change from HTML is the addition of a DTD (Document Type Definition), which is a stylesheet, that controls the appearance of a document. Since a DTD functions as the equivalent of a collection of styles of a wordprocessor or a spreadsheet, many documents can be formatted based on the content of one DTD. However for each document, this external DTD can be overridden by an Internal DTD.

Virtually all of the Ada community has been interested in the efficient use of memory and file space. The file sizes for a late draft of this document are shown in the table below. The 43% increase in file size of XML versus a pure text file is certainly acceptable. The approximately 19 Kilobyte DTD file which accompanies this XML file can in principle be shared by multiple documents.

Comparison of File Format Sizes

File Type	Kbytes	% of Text
Adobe FrameMaker	191	415%
Adobe Portable Document Format	138	300%
HTML	71	154%
Rich Text Format	77	167%

Comparison of File Format Sizes

File Type	Kbytes	% of Text
Text	46	100%
Microsoft Word 97	135	293%
XML	66	143%

The XML specifiers have tried to do software engineering!

> Quote from R. Light [13], "it is very easy to use clever shortcuts that save a bit of typing here and there but lead to markup whose existence can be inferred by machines, but is not actually present in the document and so is not apparent to the human observer."
>
> "Omitting start tags and end tags is a good example of the dangers of shortcuts."

Preliminary work has already started on building a validation tool for XML [14].

3.2.1 Document Object Model:

The Document Object Model (DOM) [15] "is a set of interfaces and objects designed for managing HTML and XML documents. The DOM may be implemented using language-independent systems like COM or CORBA; it may also be implemented using language-specific bindings like the Java or ECMA Script bindings specified in this document." The DOM interfaces are an abstraction for specifying the methods "to access and manipulate an application's internal representation of a document." These methods should map to Ada package specification subprogram declarations. Appendix C of the DOM document provides IDL Definitions [16], that could be translated into Ada. Appendix D is a Java Language Binding and Appendix E is an ECMA Script Language Binding. The ECMA binding provides useful documentation on both the data types and methods.

3.2.2 XSL (Extensible Stylesheet Language):

XML Style Sheets are created in the Extensible Stylesheet Language, XSL [17]. XSL is based on DSSSL (Document Style and Semantics Specification Language; ISO/IEC 10179:1996)."XSL is a language for expressing stylesheets. Each stylesheet describes rules for presenting a class of XML source documents."

3.2.3 Major support for XML:

The World Wide Web Consortium Issued The Document Object Model [15], DOM, Level 1 as a W3C Recommendation to achieve "Interoperability for Dynamic Web Pages and XML Applications". The Key industry players who have brought their expertise to the W3C DOM Working Group include: ArborText, IBM, iMall, INSO, JavaSoft, Microsoft, Netscape, Novell, Object Management Group, SoftQuad, Inc., SUN, Texcel. Microsoft has shown its support by creating two programs, an XSL Tutorial, which interestingly employs Latin_1, and Microsoft XML Notepad. These can be downloaded at no cost from http://www.microsoft.com/xml/. Adobe FrameMaker 5.5.6 includes a filter which exports files in XML. Both this document and a sample of Ada source text have been saved from FrameMaker in XML format.

3.2.4 XML_Io:

A new package, XML_Io, would permit Ada programs to access the full functionality of XML environments. Since third parties will be supplying software to display on the screen and print XML formatted data including text, Ada should provide a library to interface to this new environment. Besides the present text and bit-mapped graphics available in HTML, XML will permit the specification of vector images. Vectors graphics require significantly less storage space than bit-mapped graphics. This decrease in file size results in a similar decrease in download time.

Encodings are being developed in XSL to display specialized data types such as chemistry and mathematics. If domain-specific tools read and produce these data encodings, it should be possible for Ada to interface with these tools by both writing to and reading from them employing XML format data. This would permit Ada to serve as the glue between these very productive, high level tools.

3.2.4.1 Mathematics Markup Language Example:

Mathematical notation is a problem for conventional programing languages. The XML implementation for mathematics is MathML [18].

> "With XML: The Math WG is naturally affected by eventual changes to XML syntax. MathML's revision will be written in XML 1.0, with the addition of XML namespaces, as far as is possible".
>
> "MathML is cast as an application of XML. As such, with adequate style sheet support, it will ultimately be possible for browsers to natively render mathematical expressions. For the immediate future, several vendors offer applets and plug-ins which can render MathML in place in a browser. Translators and equation editors which can generate HTML pages where the math expressions are represented directly in MathML will be available soon."

Several MathML Implementations have been developed, for instance, Amaya W3C's browser and authoring tool for Web pages. Amaya includes an easy to use editor for MathML; other implementations include: EzMath, IBM techexplorer, Maple, Mathematica, MathType, Publicon, and WebEQ a Java-based collection of tools for authoring and rendering MathML, including a visual editor, a WebTeX to MathML translator, and a rendering applet for interactive math on Web pages.

3.2.4.2 Ada Document Type Definition:

An Ada Document Type Definition would permit the use of commercial XML tools with Ada source text. These tools could include commercial XML based text editors, which could be extended to serve as programing editors. These editors could provide hypertext links to the documentation and withed entities. It should also be possible to create bidirectional hypertext links between subprogram specifications and bodies. Linkage of the public subprogram specifications between the specification and body of a package would ensure consistency and minimize typing. The addition of printing with character styles which included fonts, color, weight and variation would significantly increase the readability of Ada source text. The use of XML files would make possible the direct incorporation into Ada source of XML derivatives, such as MathML, section 3.2.4.1. This would 1) facilitate the checking of equations by domain experts, mathematicians and physicists and 2) permit the incorporation into Ada source of the direct output of domain specific applications.

Ada by virtue of ISO standardization and an effective validation suite is truly portable. ASIS provides a portable way to examine Ada libraries and an Ada document type definition would permit formatted Ada sources to be portable between Ada editors. This threefold portability would be unique to Ada and thus, would provide significant advantageous to Ada developers, which could be justifiably exploited by Ada marketers.

3.2.5 Future relationships with the W3C:

The ARA or even SIGAda should join W3C. The participation of the Ada community will both add significant technical resources to the W3C and will also ensure language independence. This is particularly important for the development of XML and XSL.

3.3. Operating Systems for Embedded Software and Devices:

The two extremes for building embedded systems software are to create a completely stand-alone product or to base the product on an existing operating system. Often, the solution is an intermediate approach where a commercial run-time (something short of an operating system) is employed. The special case where this run-time is written in Ada, although potentially the best solution, provides a philosophical quandary, since the result is also equivalent to a stand-alone product based on an Ada reusable component. For many commercial projects, the cheapest and easiest solution is to employ a commercial operating system or a derivative (PROM version) of a commercial operating system.

3.3.1 Microsoft® Windows 98 and 2,000:

The two major, presently commercially successful operating systems are Microsoft Windows 98 and 2,000. Eventually, these are projected to merge into one product for PC clients. However, neither was developed for nor is suited for real-time software.

3.3.2 DOS:

In the recent past, Microsoft® DOS has been successfully used for real-time Ada software [19],[20]. One of DOS' greatest virtues is that it can be set to do nothing; and thus, it does not get in the way of the real-time software. However, DOS has no future because of three significant problems: 1) It is 16-bit, so it requires an often expensive 32- bit extender. 2) It is against the economic interest of its creator and owner, Microsoft, to continue the life of the product. And 3) DOS does not itself contain a collection of device drivers and there has been little incentive for third parties to write device drivers for new hardware.

3.3.3 DR-DOS:

Caldera [21] provides DR-SBK(™) Systems Builder Kit, which they claim is "a complete DOS/Internet OEM developer kit." This kit includes DR-DOS 7.02, DR-WebSpyder(™) and developer tools. DR-DOS is billed as an operating system that is suitable for embedded applications based on Intel architecture. DR-DOS includes power management, and true multitasking, which are not included in Microsoft DOS. DR-WebSpyder is graphical Web browser, which "fully supports HTML 3.2 including: Netscape compatible frames, GIFs, animated GIFs and JPEG images, Tables and forms, CGI, Client/server side image maps", and 32-bit protected mode applications. The Caldera web page search engine could not find any references to Ada.

3.3.4 Linux:

Both Caldera and Red Hat[22] sell versions of Linux. According to Yager [23], Red Hat eliminated a major impediment to the commercial use of Linux by providing an easy installation and device drivers. "The installation process is a breeze." "Red Hat's installation process is capable of coping with multiple SCSI controllers, as well as mixed IDE and SCSI drives. Advances in this release include new video card and network drivers, and the latest stable Linux kernel (2.0.36)." A major problem with Linux is that it is a true multitasking operating system. The question is, since Linux is provided as open source, can it be scaled down for embedded systems and be retrofitted with an Ada real-time kernel? Admittedly, it is possible to create real-time Ada applications with Linux. However, achieving optimum efficiency in either time or space or some specified combination thereof is much simpler to accomplish when the interacting parts of the system are all written in the same programing language. The use of a single language and real-time semantics provides a significant simplification, particularly for monolingual programmers, for instance the author of this paper.

3.3.5 Windows CE:

Microsoft's Windows CE has several advantages [24]: strong backing, a reasonable 32-bit architecture, scalability combined with tailorability, portability between microprocessors, and it employs a subset of the Windows NT-2,000-98 Applications Programming Interface. Probably, most of the Ada binding to the Windows CE already exists. Significant third party tools [25] including support for Intel microprocessors are already commercially available. Since Windows CE is hosted on Intel, MIPS and Hitachi 32-bit processors, it appears that there must be a common internal form. It is quite possible that hosting on an Ada compiler on Windows CE would provide a means for code generation for multiple microprocessor architectures. From a marketing point of view, if at all possible, vendors of Ada products for Windows CE should try to obtain permission to use the Microsoft logo [26]. Ada products will have the significant advantage of being able to employ 8-bit character representations rather than being forced to employ the 16-bit Unicode representations. The real-time capabilities for Windows CE have recently been described by Gareau and Labrosse [27].

In any event, in order to maximize portability on new platforms, Ada programs should be based on a XML_Io package; rather than the present proprietary windowing environments.

4. Future Consolidation of the Pascal Family of Languages:

Since as was discussed in Section 2, there is positive feedback in the market share of a programming language. The availability of extensive libraries and tools for a language is required for large scale use of a language, which in turn provides the economic motivation for creation of more libraries and tools. If descendants of Pascal are to be credible alternatives to descendants of C, there needs to be a consolidation of the Pascal descendants. The best model for this is a corporate merger of equals. Even if one corporation is significantly larger than the other, it is totally imprudent to incorporate only the features and employees of one organization into the surviving organization. Very often, a Daim-

lerChrysler or Beckman Coulter is created. Similarly, one very particle way to increase the use of Ada and the long term probability of its survival is to merge it with another descendant of Pascal.

4.1. Merger of VHDL and Ada:

VHDL is a hardware description language (HDL) that is similar to Ada. Two articles concerning the merger of Ada and VHDL were published in the SIGAda '98 proceedings. Wong and Levine [28] demonstrated the modeling of both a NAND gate and a dual ranked flip_flop, D-Flopflop, could be done employing a subset of Ada 83. Mills and Peterson [29] mention and give references to, "One effort to transition software engineering standard practices into hardware description languages is the work focused on creating object-oriented VHDL extensions. In essence the proposal extends VHDL with object orientation using similar mechanisms to those of Ada 95." These authors demonstrated the transformation of VHDL to Ada. They also explain the VHDL simulation cycle. In their conclusion, they state 'With the increasing complexity of electronic systems, better integration of the design and analysis for hardware and software is needed." Although Mills and Peterson addressed the electronic part of systems, their conclusion holds for all other parts of a system. The addition to Ada of simulation capability for mechanical hardware and fluidics would facilitate the early detection of design flaws. In essence, simulation tools that would permit animated cartoons that described the functioning of subsystems and the entire system would be very useful. This capability for this approach is not new. For instance, Robinson [30] employed LabVIEW™ (National Instruments) to simulate and provide the user interface for a Proximate Sensing Electronic Unit of an aircraft landing gear.

An Ada compiler which includes the capabilities of present VHDL systems could be sold for a much higher price than today's bargain priced Ada compilers. Since the capital investment to develop a new generation of microprocessors, digital signal processors, and other complex integrated circuits can easily be over one hundred million dollars, the selling price of a software system to reliably design and model these devices can be quite high.

5. Closing Statement and Conclusions:

The problem with Ada is that the technology was too advanced for its time. However, the time for Ada has come. The evolution of the software market into HTML and ultimately XML based applications provides an excellent marketing opportunity for software development in Ada environment which includes Web specific packages (XML_Io). The present growth in size and complexity of operating systems and object based software environments requires software tools suitable for these huge enterprises. The only language with the capability of actually producing efficient, reliable huge systems is Ada.

Ada has great potential advantages to the largest player, Microsoft. The usual product improvement and cost saving virtues of Ada would probably never provide motivation for change. However, Ada provides a solution to two of Microsoft's major problem. 1) Its major competitors are attempting to control the software industry by pushing a technology, Java, as the wave of the future; and worse yet, this technology is controlled by a competitor SUN, who is a sworn enemy. Ada provides a way to gain and more importantly demonstrate technological supremacy. Of even greater significance, this technological supremacy would provide an excellant opportunity to enjoy embarrassing one's competitors as technologically incompetent. Amusingly, the fact that this would be true is almost irrelevant. 2) The same competitors have prodded the United States Justice Department to bring an antitrust action against Microsoft. An IEEE POSIX standard Ada binding to Microsoft's operating systems would be a legitimate way to open them up to satisfy a consent decree. Fortunately for Microsoft, its major competitors have so committed themselves to Java, that they would be unable to make use of an open standard Ada binding.

It is time to forget about past marketing failures, to treat Ada as if she were a new programming language, and to be extremely aggressive in marketing Ada technology. A brand new technology to create reliable efficient, portable software that worked on clients, servers, and the Web would be a very hot new issue on today's (January, 1999) stock exchange. Ada delivers what Java promised. We have both an excellent product and future, if we have the will to make it happen!

6. Acknowledgements:

I wish to thank Dave Wood of Aonix and Gilbert Prine of AdaMATION, Inc. for their excellent presentations, the Workshop attendees for their questions and suggestions. I am indebted to W. Wesley Groleau for his detailed editorial advice. Both Stephanie H. Leif and John T. Apa have made suggestions which have improved this paper. This paper and the Workshop would not have been possible without the efforts of Alok Srivastava, who organized the Workshops, and the organizers of SIGAda '98.

7. References:

1. D. Wood, “Ada Developers Cooperative Draft License” by Robert C. Leif, Ada_Med; 2. “Ada: A Commercial Flop and Proud of It”, Ada Letters (1999)

2. Capers Jones, “Programming Languages Table, Release 8.2, March 1996,” Software Productivity Research, Inc. http://www.spr.com/library/0langtbl.htm. Last visited, 11 December, 1998.

3. C. Jones, “Sizing Up Software”, Scientific American, December 1998, http://www.sciam.com/1998/1298issue/1298jones.html. Last visited, 11 December, 1998.

4. Capers Jones, “What Are Function Points?”, Software Productivity Research, Inc. http://www.spr.com/library/0funcmet.htm. Last visited, 11 December, 1998

5. N. Wirth and J. Gutknecht, *Project Oberon, The Design of an Operating System and Compiler*, Addison-Wesley, ACM Press, 1992, ISBN 0-201-54428-8.

6. J. English, *Ada 95, the craft of object-oriented programming*, Chapter 18, pp. 333-353, Prentice Hall, 1997, ISBN 0-13-230350-7.

7. AdaSAGE, http://sageftp.inel.gov/sage/homepage.htm. Last visited, 7 December, 1998

8. R. C. Leif, R. Rios, M. C. Becker, C. K. Becker, J. T. Self, and S. B. Leif, "The Creation of a Laboratory Instrument Quality Monitoring System with AdaSAGE". *Advanced Techniques in Analytical Cytology, Optical Diagnosis of Living Cells and Biofluids*, Ed. T. Askura, D. L. Farkas, R. C. Leif, A. V. Priezzhev, and B. J. Tromberg. A. Katzir Progress in Biomedical Optics Series Editor SPIE Proceedings Series, Vol. 2678, 232-239 (1996).

9. Ken Bowles, personal communication, (7 December, 1998).

10. “NSF Creation and Mission”, http://www.nsf.gov/home/about/creation.htm. Last visited, 7 December, 1998.

11. “World Wide Web Consortium”, http://www.w3.org. Last visited, 7 December, 1998.

12. “Extensible Markup Language”, XML, http://www.w3.org/XML. Last visited, 7 December, 1998.

13. R. Light, “*Presenting XML”*, Sams net, 1997, ISBN 1-57521-334-6.

14. R. L. Goerwitz III, “XML Validator Built Using Stock Programmer's Tools,” http://www.oasis-open.org/cover/stgParser19981207.html. Last visited, 7 December, 1998.

15. V. Apparao, S. Byrne, and M. Champion (Principal Contributors), “Document Object Model (DOM) Level 1 Specification, Version 1.0, W3C Recommendation 1 October, 1998”, http://www.w3.org/DOM/. Last visited, 7 December, 1998.

16. The IDL files described in “Document Object Model (DOM) Level 1 Specification, Version 1.0, Appendix C: IDL Definitions” are available as, http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/idl.zip.

17. “Extensible Stylesheet Language (XSL) Version 1.0, World Wide Web Consortium Working Draft 18-Aug-98”, http://www.w3.org/TR/1998/WD-xsl-19980818. Last visited, 7 December, 1998.

18. “Mathematics markup Language”, MathML, http://www.w3.org/Math/activity.html. Last visited, 9 December, 1998.

19. R. C. Leif, J. Sara, I. Burgess, M. Kelly, S. B. Leif, and T. Daly, “The Development of Software in the Ada Language for a Mid-Range Hematology Analyzer”, *Tri-Ada* ‘93 340-346 (1993).

20. R. C. Leif and S. B. Leif, “Ada in Embedded Boards for Scientific and Medical Instruments”, *Proceedings, ACM SIGAda Annual International Conference (SIGAda ’98)*, S. Carlson, Proceedings Chair, M. Feldman, Program Chair, pp. 114-120, 1998.

21. “Caldera”, http://www.caldera.com/. Last visited, 8 December, 1998.

22. “Red Hat”, http://www.redhat.com/. Last visited, 8 December, 1998.

23. T. Yager, “Operating system software Easy install puts Red Hat 5.2 on top”, *PRODUCT REVIEWS*, InfoWorld, November 23, 1998 (Vol. 20, Issue 47). http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/47/rredhata.dat.htm. Last visited, 8 December, 1998.

24. “Windows CE, Application Developers”, http://www.microsoft.com/windowsce/developer/. Last visited, 8 December, 1998

25. “Repository for Windows CE Drivers”, http://www.microsoft.com/windowsce/embedded/driverrep.asp. Last vis-

ited, 8 December, 1998.

26. “Windows CE Logo Program”, http://www.microsoft.com/windowsce/logo/default.asp. Last visited, 8 December, 1998.

27. J. Gareau and J. Labrosse, “Developing Applications with Windows CE 2.10”, EmbeddedSystems, Vol. 11, Number 11, October, pp 46-59, 1998.

28. S. Wong and G. Levine, “Kernel Ada to Unify Hardware and Software Design”, *Proceedings, ACM SIGAda Annual International Conference (SIGAda ’98)*, S. Carlson, Proceedings Chair, M. Feldman, Program Chair, pp. 28-31, 1998.

29. M. Mills and G. Peterson, “Hardware/Software Co-design: VHDL and Ada 95 Code Migration and Integrated Analysis”, *Proceedings, ACM SIGAda Annual International Conference (SIGAda ’98)*, S. Carlson, Proceedings Chair, M. Feldman, Program Chair, pp. 14-27, 1998.

30. R. L. Robinson, “Landing Gear Simulation Using LabView™,” Scientific Computing & Automation, Vol. 15, Number 11, pp 27-37, October, 1998.