SIGAda 99, 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, rleif@rleif.com

The focus of this Workshop, which occurred on October 20, 1999, was on extending the use of Ada into the commercial off-the-shelf (COTS) domain. The goal of this Workshop is to determine what should we do to both make Ada the dominant language for COTS and profit by doing so? The contents of Sections 2. Red Hat, Where the Money Went, a Case Summary and 3. How to Commercialize Ada and Profit have been updated.

1. Speakers:

Robert C. Leif, Vice President Ada_Med (rleif@rleif.com), “Red Hat, Where the Money Went, a Case Summary.” (Please see Section 2.)

Mike Kamrad, Senior Engineer Top Layer Networks, Inc. (kamrad@TopLayer.com) “Developing Data Communication Software for a Layer 7 Switch in Ada95”. An elegant combination of Ada software, specialized hardware, and knowledge of the Internet market was described. The Top Layer Networks’ Layer switch is able to determine the type of application and use that information to control the priority of transmission of a data packet.

Randall L., Brukardt, Director of Technical Operations R.R. Software, Inc. (Randy@rrsoftware.com), “Ada 95 and Commercial Applications: How and Why?”. A major problem with the marketing of commercial off the shelf products, COTS, written in Ada is the creation of a demand for reliable software products. Unfortunately, the customers do not yet realize that it is possible to produce products that are not bug ridden.

Michael P. Card, Lockheed Martin OR&SS (michael.p.card@lmco.com), “FIRM is Ready” - A Real Time Object Database. The FIRM database, which is written in Ada, has incredible performance compared to the standard commercial competitors. It demonstrates how the combination of a well engineered design and the use of Ada can produce a product at incredibly low cost compared with the present commercial C and C++ based products. This talk also demonstrated the extreme difficulty of technology transfer from the Defense Industry to the commercial sector.

Steven Blake, Principal Ada Consultant, Aonix (sblake@AONIX.com), “An ASIS based Tool to Measure what is in an Ada program”. Traditional scientific and engineering disciplines are significantly based on the capacity to measure items in their domains. This new Ada technology makes possible, the objective measurement for software. For instance it is now possible in Ada to measure the contributions of individual packages to the linked executable. This technology also has the unique commercial utility of with a small addition providing an objective measurement of the contributions of individual programmers or organizations to a large project. This in turn can be employed to divide up royalties without the necessity of employing the expensive services of accountants and lawyers.

Robert C. Leif, Vice President Ada_Med (rleif@rleif.com). “How to Commercialize Ada and Profit” (Please see Section. 3).

2. Red Hat, Where the Money Went, a Case Summary

A summary of the financials of a corporation that markets software based on the GNU Public License, GPL, is a necessary preface to a discussion on “How to Commercialize Ada and Profit”. Red Hat was chosen both because Linux is the commercially most significant example of a GPL product and the valuation of Red Hat (RHAT symbol on NASDAQ) is spectacular.

The following information was obtained from the (SEC) Security and Exchange Commission’s Edgar web site.

From: http://www.sec.gov/Archives/edgar/data/1087423/0001047469-99-031070-index.html

- Red Hat Linux represented approximately 56% of new license shipments of Linux-based server operating systems in 1998,
- Approximately $10.8 million in revenue for the fiscal year ended February 28, 1999, primarily from the sale of Official Red Hat Linux.
- Strategic alliances or investment relationships include Compaq, Dell, IBM, Intel, Netscape, Novell, Oracle, SAP and Silicon Graphics.
- Customers who purchase our Official Red Hat Linux Version 6.0 product receive 30 days of telephone installation support or 90 days of e-mail installation support at no additional charge.
- Customers' telephone support agreements range in price from $995 for up to three incidents to $60,000 for 24-hour-a-day unlimited support for one year.
- Red Hat has registered the trademark "Red Hat" in the United States; pending in the Australian, Canadian and European Union trademark offices.
- Red Hat "Shadow Man" logo registered in the U.S., European Union and Australia and have registrations pending for it in the Canadian trademark office.

In terms of Intellectual Property, Red Hat has from a business perspective an absolutely brilliant approach. They obtain much of their software for free by acquiring software covered by the GNU Public License[1],[2] and they protect the corporations assets by employing proprietary trademarks.

2.1. Possible problems:

Red Hat Linux in compressed form consists of approximately 573 megabytes of code (approximately 500 megabytes have been developed by independent third parties). There are 645 distinct software components developed by thousands of individual programmers which Red Hat must assemble and test before a new version can be released. Approximately 10 megabytes of code is contained in the Linux kernel. Without reliable components Red Hat Linux could fail. This would result in serious damage to the company's reputation and have the potential for litigation.

Basing a business on intellectual property where one has no control over many of the developers has its hazards. The release of major product upgrades of Red Hat Linux on a timely basis depends on individuals who do not have a relationship or interest in Red Hat. The Linux kernel, is maintained by third parties. Linus Torvalds, the original developer of the Linux kernel, and a small group of independent engineers are primarily responsible for the development and evolution of the Linux kernel. Actually, Red Hat's financial success is the source of its greatest hazard. The monkey see monkey do rule is directly applicable to many investors including venture capitalists. The Linux creators and maintainers are now very marketable candidates for a start up and other competitors can also market Linux. Caldera has just gone public and Corel is selling its own version of Linux, which will run Corel's software applications.

2.2. Financials:

2.2.1. Initial Public Offering: At the request of Red Hat, the underwriters had reserved up to 800,000 shares of common stock for sale at the initial public offering price through a directed share program, to directors, officers and employees of Red Hat and to open source software developers and other persons that Red Hat believes have contributed to the success of the open source software community and the growth of Red Hat. An optimistic assumption is that 25% of these 800,000 shares or 200,000 were allocated to the open source developers, and that these developers had the money to purchase these shares. It is rumored that some of the developers did not even qualify to open an account at E*Trade, the brokerage that was charged with making these initial public offering shares available.

If the underwriters sold more than 6,000,000 shares of common stock, the underwriters had the option to purchase up to an additional 900,000 shares from Red-Hat at the initial public offering price less the underwriting discount.

The number of shares of common stock after the offering is shares outstanding on July 31, 1999 did not include:

- 5,762,188 shares of common stock issuable upon the exercise of stock options outstanding on July 31, 1999 with a weighted average exercise price of $2.51 per share
- 3,197,450 shares of common stock warrants outstanding on July 31, 1999 with an exercise price of $.0001 per share.
- 8,075,910 shares reserved as of July 31, 1999 for future stock option grants and purchases under Employee Benefit Plans.

Nominally after the initial public offering, the existing stockholders owned 90.9% of the stock. directors, executive officers and their affiliates beneficially owned approximately 67.7% of common stock. However, the underwriters' bonus purchase of 900,000 shares would have easily been subscribed at a large profit; and the already Issued 8,959,638 very low cost options will be eventually converted into common stock. In any event even using the valuation, which corresponds to 20 March, 2000, the date of today's writing and the public offering of its competitor, Caldera, the total valuation of Red-Hat, not allowing for its acquisitions is about $9,514,000,000. Of this at best, $23,700,000 or 0.25% of the stock would have gone to the developers.

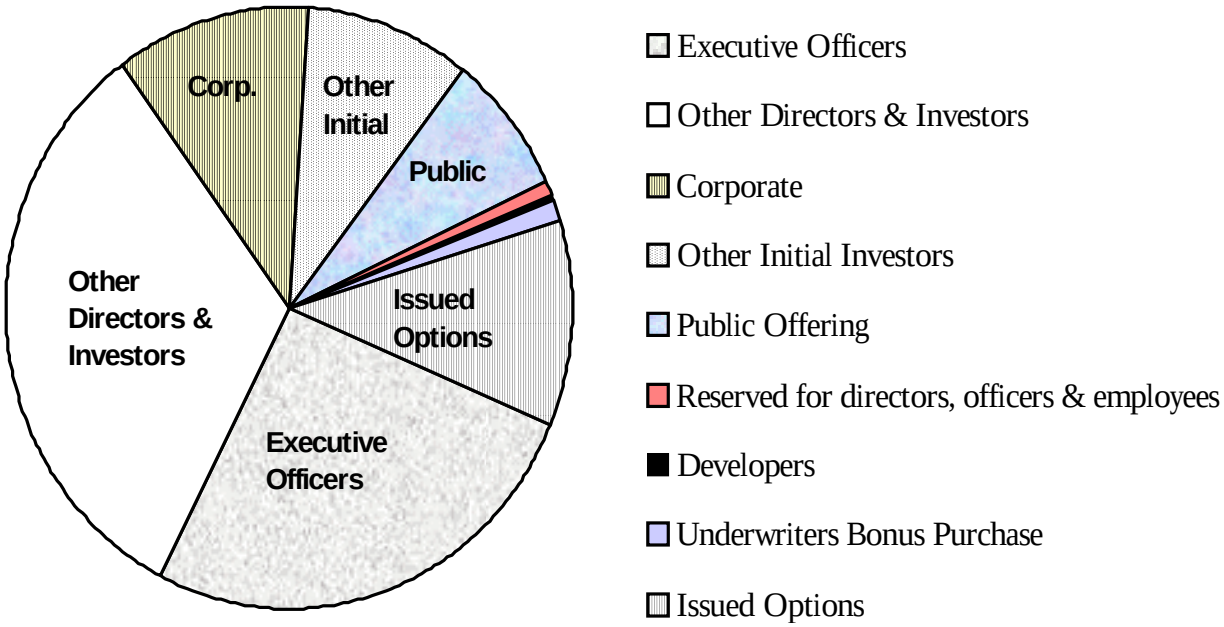

Figure 1. Pie chart showing the distribution of ownership of Red-Hat just subsequent to the completion of its initial public offering. Please notice the thin, solid black sector at 2 o'clock. This is based on the estimate that the original Linux developers both received and exercised the right to purchase 200,000 shares at the open.

A simple calculation will indicate that developers would have been far wiser to rely on owning their own intellectual property rather than relying on corporate charity. There were 645 software components. A reasonable, perhaps somewhat conservative, assumption is to assume 25% of the value of a corporation like Red Hat to be based on its commercial software. It must be emphasized, that the relative value of delivered working software where the corporation has no up front capital investment is significantly higher than that of software internally funded by a corporation. Twenty-five percent of the total value of Red Hat is approximately 2,378,000,000. This divided by 645 equals $3,800,000 per component on 20 March, 2000 as compared with $23,700,000 to be shared by some fraction of the developers.

Red Hat went public and immediately had a value of $925 Million, which has increased to a maximum of about 14 billion and now decreased to about 9.5 billion dollars. This was all based on Free Software, Linux! In short, even Karl Marx would never have prognosticated that the workers would get at best 0.25% (virtually nothing) and the capitalists billions. Under the GNU public license[1] some of us (the distributors) can be much more equal than others (the software engineers). Please see George Orwell's Animal Farm[3].

3. How to Commercialize Ada and Profit

3.1. The Past Need Not Predict the Present:

As a commercial technology, Ada has had three misfortunes that normally should have killed it. These were: 1) Ada arrived too soon. 2) Ada was a purely technology motivated undertaking, and 3) no significant attempt has been made to commercially market Ada. The lack of interest in marketing of Ada is demonstrated by the fact that none of the 20 attendees of this meeting had marketing as their primary responsibility. Ada's survival can only be attributed to its technological excellence compared to its competitors. The present situation is entirely different from the initial period at Ada's introduction, which should have been its window of opportunity. Later on, it was widely stated that Ada could not replace the dominant object-oriented language, C++. Sun demonstrated with Java that a market oriented company could very effectively introduce a new language to compete with and quite possibly displace C++.

3.2. The Time is Now Right for Ada:

3.2.1. Then and Now: When the first validated Ada compilers were released, they were expensive, slow to compile and for PCs required a significant amount of expensive memory. The situation today is very different. No special hardware requirements exist for compiling Ada. A PC that can satisfactorily run Microsoft Office 2000 is sufficient for Ada development. The converse may not be true for the combination of Microsoft Office 2000 and Windows 2000 (http://www.microsoft.com/windows2000/guide/professional/sysreq/default.asp).

3.2.2. Software Products have radically changed: Initially, commercial software products fit on a few 1.2 megabyte floppy diskettes and had sufficiently few features that many of these products could be used by normal people. Today, similar products are now sufficiently complex and large that the they require both CD-ROMs and manuals that can be over one thousand pages long. The major vendors employ very large numbers of programmers and testers to produce these complex products. This type of large product development and its associated maintenance requirements were the initial motivation for the creation of Ada.

3.2.3. Commercial Software Opportunity: The cost of popular software products has increased in significance because it has not decreased while powerful, personal computers can be purchased today for $500 to $1,000. The sum of Windows 2000 and Office is $715 (WWW.CDW.Com).

Windows 2000 and Office 2000 are not suited for home use. Besides being too expensive, they are difficult to learn. For instance, Microsoft Word is a very powerful document creation tool. It contains an enormous complement of features, which are completely beyond the capabilities of a normal user. Unfortunately, most users employ it as a glass-typewriter. They do not create specific paragraph formats for each type of document entity. The full capabilities of paragraph numbering or cross-references are seldom used.

Products like Microsoft Word and Excel are victims of their own history. Object-oriented design and programming languages should be employed to simplify the use of a program. Instead, features continue to be added. In fact, products like Microsoft Works are much better suited for today's nontechnical customer.

3.2.4. Ada Technology is Now Commercially Relevant. One of the best demonstrations of the fallacy of today's C++ based software technology is described in a book, "Project Oberon"[4]. Wirth's latest language, Oberon, like Ada is an object oriented descendant of Pascal. Both the Oberon

language and Operating system are small enough to be described in one 548 page book including source and indices etc.

The combined use of Ada and Software Engineering could extend and capitalize on the discontinuity that Linux has created in the process of commercial software development. Software development in the Windows era has been based on a combination of the large-scale manufacturing and heroic programmer models. Linux has been a distributed venture based on a virtual organization. The CLAW graphics packages and screen builder are a very good example for the organization of future Ada software development. The collaborators live and work in geographically separate states, Wisconsin and California. Admittedly, the initial design and creation of many of the package specifications requires close collaboration. However, the creation of most of the package bodies can be reasonably independent.

Ada technology also will permit a major change in the financing of software projects. A distributed development environment eliminates the need to house an army of workers. The Redmond-Bellview Microsoft facility is an example of a very large, expensive commercial building enterprise. If this type of infrastructure cost is minimized and the employees are compensated in large part by royalties, the cost of entry will be greatly reduced. Open-source when combined with software-engineering and Ada will finally permit the replacement of large monolithic programs by modular constructs. If individuals or small groups of developers can independently create and sell these modules, then the Ada software products will achieve the necessary critical mass to be competitive with today's commercial products. The quality of these Ada products will be greatly enhanced by 1) being released as open source and 2) being subject to competition between developers for bodies and enhancements.

3.3. Ada Commercial Products, Operating System in Ada: One of the major reasons for the popularity of the C language and its descendants is that they have traditionally been used to write operating systems starting with UNIX. The ability to directly and simply link application code directly with operating system code is a significant advantage over other languages including Ada. Ada, of course, has many advantages that in most cases more than make up for these disadvantages. However, the full utility of Ada particularly for real-time applications will be best realized by linking Ada applications to an operating system written in Ada. To that end, plans are underway to write an operating system in Ada (AdaOS-list@adapower.com).

3.3.1. A well designed Operating System should be small. An operating system can be defined as a collection of software components that can work together and be employed to create new software components. Simplicity is an excellent goal for any design process, particularly software. Human frailty often results in costly errors, requires expensive manuals, on-line help, and technical support. A good simple design can reduce development costs and result in products of sufficiently small size to permit distribution via the Internet.

The Oberon operating system developed by Wirth and his colleagues[4] has already proven that a small, simple operating system can be created at minimal cost. If the Oberon operating system had been written in Ada and the graphical user interface had been based on XML (see Section 3.3.4), a very large part of an Ada operating system would have been already completed. However, as in the previous experience with evolution of Pascal to Ada, much of the Oberon design can be reused. The operating system should maximize the present capabilities of Ada and its annexes, in particular those related to real-time operations.

3.3.2. Ada Real-Time Kernel: The first item to be developed should be an operating system independent Ada real-time kernel. This would both fulfill a very significant need for Ada real-time software development and could be marketed as a stand-alone item. The use of the Florida State Ada Linux Kernel Module[5] should eliminate language mismatch with resulting increases in efficiency and reliability. Porting this Kernel to other operating systems such as Windows CE and the Palm Operating system should significantly extend the use of Ada for commercial embedded systems.

Ada software embedded system vendors should very seriously consider combining their technical ability with existing companies that have experience and competence in marketing to embedded system devel-

opers. These companies interest in real-time embedded kernels and their APIs should be increased by Red-Hat's recent purchase of Cygnus Solutions[6] for shares and options valued on 20 March, 2000 at $787,000,000. Cygnus Solutions is developing EL/IX, an Embedded Linux API based on POSIX.

3.3.3. Existing Operating System Components in Ada: The presentations by both Michael P. Card from Lockheed Martin and Randall L. Brukardt from R.R. Software, Inc. were both highly relevant to the development of a modern operating system in Ada. The FIRM database was described in the Software Technology meetings[7],[8]. Ada needs to be extended with the capacity to make persistent objects of its atomic types and records. The concept of using a database to manage the file system is definitely not new. An object-relational database would be able to store reading and writing rights, authorship, etc. It would also be able to include which applications require a given file. Presently, removing applications from Windows 98 is a real-world version of a boolean process. All too often, either too little or too much is removed.

Subsequent to SIGAda99, Tom Moran posted the SmplSrvr demo at www.adapower.com. This program demonstrated that: 1) a JAVA design could be ported to Ada to produce a true executable; 2) how easy it was to use the RRsoftware CLAW components (www.rrsoftware.com), and 3) Ada real-time functionality to make a small, simple server that is able to work with either networked or local HTML pages. Particularly, SmplSrvr is able to obtain the output of the HTML FORM GET and POST operations and translate this output into strings. After the strings were parsed, the FORM data could then be utilized as inputs to a local Ada program. Thus, Ada software that was created to serve as a thick Windows binding is capable of being ultimately used to implement an HTML-XML browser-based GUI for both embedded and distributed applications.

3.3.4 XML-XHTML GUI: The era of the present Microsoft Windows graphical user interface, GUI, software standard is ending. It will be replaced by an Internet based standard. Microsoft is quite correct that the operating system for the PC and the Web should be based on a common set of software components. This is nothing more than employing the principles of object oriented design. The real question is which Internet standard will succeed? The present consensus is Extensible Markup Language, XML. XML is controlled by the World Wide Web consortium, W3C (www.w3.org). This group has the great virtue of not being controlled by either Sun or Microsoft. It also has the advantage of including some very good software engineers and producing public standards. Many companies are supporting XML. These include: IBM (http://www.ibm.com/developer/xml), Sun (http://java.sun.com/xml), and Microsoft (http://msdn.microsoft.com/xml/general/whyxml.asp).

The use of XML-HTML is an effective way to break the Microsoft monopoly. XML- XHTML could serve as the basis of a new portable Graphic User Interface. The browser serves as front end of the operating system. Printers and graphics boards could be programmed to directly execute XML. Direct interpretation of XML by the printer and graphics board eliminates the main underpinning of the Microsoft Windows monopoly, its huge collection of device drivers. The only real problem is to create totally Client based programs with XML. Tom Moran's SmplSrvr has already shown that this is feasible.

3.3.5. Scripting Language: The obvious choice for the macro language is an Ada J code compiler or the equivalent. End-users of commercial software often must create macros, batch files, and formulas. These all can be represented as an Ada parameter-less procedure or classical main program. If necessary, an implicit instead of an explicit "with" and "use" statement can be employed for present conventual usages. Much of this functionality was present in the Alsys AdaWorld programming environment.

A fourth generation programming language environment based on Ada should be created to provide a semi-automated, user-friendly approach for the creation of commercial end-user programs [9]. This should be based on copying and transforming Ada subprograms to display the formal parameters and to employ a text entry box or list box for the specification of the actual parameters. ASIS functionality can be employed to permit type-checking at the time of filling these boxes instead of waiting for a conventional compilation.

3.3.6. License Conflict: Although much of the technology necessary to create an Ada based operating system exists, commingling of standard commercial licensed and The GNU Lesser General Public License[10] could provide a significant problem. The legal question is which would dominate for a product when parts were commingled based entirely on engineering considerations? Please note, that it is assumed that the sources for both the GNU Lesser General Public License and the standard commercial licensed software will be available to the end user. However, the commercial licensed software will require a fee for a user's license. Unfortunately, the amount of useful Ada'95 software is limited. Thus, division of it into two mutually incompatible parts could provide an insurmountable obstacle to the creation of commercial Ada products. One possible solution would be to do the licensing on a package rather than a total system basis.

The GNU Lesser General Public License and the Ada Developers Cooperative License [11] have a major difference in how they treat, "Commercial Executable Products". These are products sold in a format that results in money being transferred. At this point, I have strongly argued [12] that the developers should receive a decent percentage of this money. The Ada community does not need to make a free contribution to the investors in a future Red Hat.

Ada technology permits the evolution of Ada commercialization beyond the current GNU model. The inability to objectively evaluate each developer's contribution to a large product results in the simplest of solutions. In the case of GNU Public License[1] software, everyone works for free. Fortunately, this no longer needs to be the case for Ada. Steve Blake's description of ASIS technology established that it is feasible to enumerate all of the components of the Ada source that are linked. The addition of one variable, vendor number, to each package should permit a reproducible, consistent means of dividing the fruits of the developers' labor. Admittedly, this will require the assignment by humans of weighting factors for each Ada construct. However, once these somewhat subjective factors are agreed upon, the relative contributions of developers can be automatically computed. This is the ultimate in recursion. We use software engineering to divide up the royalties in a reasonable manner; and this technology motivates the use of software engineering.

3.4. The grass really is not greener on the other side: The usual Ada fatal mistake is to believe that the grass really is greener on the other side. A major problem in achieving commercial success is that the Ada developers keep making bindings to the rest of the world instead of using Ada to produce an improved product or extend an existing product by employing Ada technology. The best ways to persuade an engineer to learn new technology are to demonstrate that 1) the use of this technology really reduces the effort needed to complete a project and 2) knowledge of this technology will look good on his/her resume. Ada buys a developer very little, if all it provides are thin bindings to lousy software. The Microsoft design for a windowing system is not the best solution. Each window could be a task and the screen memory could be a protected type. If the system is in sufficient part web based, use of HTML etc. for the interfaces to the operating and other systems makes sense. The Ada community certainly should borrow from existing standards like CORBA or COM and produce products that combine them with the Distribution Annex. In fact, Ada COM interfaces should evolve to work on both Windows and POSIX based operating systems. Ada needs innovative commercial product, such as the Top Layer Networks' Layer switch, described by Mike Kamrad in his talk.

4. Conclusions:

1) The SIGAda '99 Commercializing Ada Workshop has demonstrated the feasibility of creating a very advanced, yet simple Ada based operating system.

2) Groups and individuals that attempt to build or upgrade products in Ada should obtain protection of their intellectual property that will permit them to share in future commercial successes.

3) The spectacular rise of JAVA has proven that it was and is possible to replace the dominant computer language.

4) The complexity, inefficiency, cost, and the lack of reliability of today's commercial software provides a great business opportunity.

5) ASIS provides Ada with a unique commercial advantage. Royalties can be equitably divided between the developers.

6) If Ada were launched as a brand-new programming language which facilitates and expedites the creation of reliable, efficient, portable software that worked on clients, servers, and the Web; it would be a hot new issue on today's (March, 2000) stock exchange.

7) Much talk about virtue (software engineering) has had little if any effect in increasing the use of Ada.

8) Money does talk! Since virtue failed, let's try greed.

5. Acknowledgements:

I wish to thank the speakers for their excellent presentations, the Workshop attendees for their questions and suggestions. Stephanie H. Leif and Tom Moran have made suggestions which have significantly improved this paper. This paper and the Workshop would not have been possible without the efforts of the Workshop organizers and the organizers of SIGAda '99. This project was supported primarily by Newport Instruments internal development funds and in part by Small Business Technology Transfer grant number 1R41CA73089 from the National Cancer Institute. The NCI support was used for the testing of the concept of using web pages as the GUI for controlling an instrument. The opinions stated are solely those of the author.

6. References and further reading:

1. GNU General Public License, Version 2, June 1991, Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA (http://www.gnu.org/copyleft/gpl.html) (1991).

2. Richard Stallman, "The GNU Manifesto", Copyright (C) 1985, 1993 Free Software Foundation, Inc. (http://www.gnu.org/gnu/manifesto.html).

3. George Orwell, "Animal Farm: A Fairy Story", 50th anniversary edition New American Library; ISBN: 0451526341 (1996).

4. N. Wirth & J. Gutknecht, "Project Oberon, The Design of an Operating System and Compiler," Addison-Wesley, ACM Press, ISBN 0-201-54428-8, 1992 (http://www.oberon.ethz.ch/).

5. H. Shen and T. P. Baker, "A Linux Kernel Module Implementation of Restricted Ada Tasking", Ada letters XIX, No 2 pp. 96..103 (1999).

6. Red-Hat Form: 10-Q Filing Date: 1/14/2000, (http://www.corporate-ir.net/ireye/ir_site.zhtml?ticker=RHAT&script=800&layout=10)

7. M. P. Card, "FIRM: An Ada Binding to ODMG-93 1.2", Track: 6 (Object-Oriented Development), Software Technology Conference 96 (1996).

8. M. P. Card and P. G. Springer, "The Best of Both Worlds: Distributing Objects with CORBA and an ODBMS," Slides from the Software Technology Conference 98 (1998).

9. R. C. Leif, "Commercializing Ada". ACM Ada Letters 16 pp. 44-45 (1996).

10. GNU LESSER GENERAL PUBLIC LICENSE, Version 2.1, February 1999, Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA (1999).

11. R. C. Leif, "Ada Developers Cooperative License (Draft) Version 0.3", Ada letters XIX, No 1 pp. 97..107 (1999) (http://www.acm.org/sigada/wg/cauwg/cauwg.html).

12. R. C. Leif, "SIGAda '98, Workshop: How do We Expedite the Commercial Use of Ada?." Ada letters XIX, No 1 pp. 28-39 (1999) (http://www.acm.org/sigada/wg/cauwg/cauwg.html)