Equitable Software (Draft)

Robert C. Leif, Ada_Med, a Division of Newport Instruments

Two critical factors required for the commercial success of a technology are 1) the technology must have some inherent utility and 2) there must be a strong economic benefit connected with its adaptation. The relative commercial failure of Ada obviously is not the result of a lack of utility. If utility were the only criterion, there would be no need for this essay. However, economic benefit can easily overshadow utility.

There is no question that Sun has done an excellent job marketing Java with the result that there is a very substantial library of Java components. (Fortunately, these and those developed in other languages can be interfaced to Ada.) Unfortunately, the quality of marketing does not have any direct relation to the quality of the resulting libraries or large benefit to the users who actually create software. Sun's costly advertisements in the Wall Street Journal may provide management with the belief that their software problems have been solved; unfortunately, it does not provide the technology for the software engineers to produce an efficient reliable, portable product.

The problem to be solved is creating economic motivation for the use of Ada and concomitant good software engineering practices. The quality of the design of a programming language and of the available compilers although of significance are often of less significance than the quality of the programming environments and libraries. If a library does not exist then it can not win on quality. Therefore, the question is how to provide a strong economic incentive for the creation and use of Ada libraries.

There are several alternatives:
1) Altruism. The creator of the library sees his or her work included in many programs and receives much acclaim and honor from his or her peers. There are very few examples of this occurring with real products and the measure of success based on their conversion into commercial products. It should be noted that many creative individuals are responsible for others. They have spouses or significant others and children or other relatives who are either dependants or who share support responsibilities. These other individuals can not be expected to look favorably on one spending large amounts of time on what is essentially a hobby. However, they can be expected to be supportive of a venture with real potential to increase their standard of living.
The argument that the compensation for software can be based upon payments for support ironically has a significant flaw for Ada products. One of the great virtues of Ada source text is that usually is readable. Both the Ada language and culture enforce this. Thus often, a user will be able do their own maintenance and make fixes or create enhancements.

2) Quick Buck: This approach is based on selling a limited number of copies at an exorbitant price. Defense contractors even have the advantage of being paid prior to

delivery and limiting their production to items that are funded by the purchaser. William Gates has stated that mass marketing is the correct approach. His obvious commercial success proves that he is correct.

3) The Developer as a Capitalist: Treating software development as capital creation as opposed to a service does not have the inherent problems of the above two alternatives. This requires keeping the initial cost of the libraries low in return for future royalties to be derived from their actual use in commercial products. A developer by licensing Ada packages would be contributing capital to the organization that creates a product. This would result in minimizing the development costs of new Ada products. It also has the very interesting property that the financial interest of the original developer in the reuse of that software provides a strong motivation to help the user. This is a major benefit compared to the present situation for unsupported (nonpaying) users of “Free” Software.
The major problem with this approach is determining the contribution of each software suppler to the final project. This measurement and the determination of the pricing of components should be objective and automated in order to avoid legal and accounting expenses.

Fortunately Ada already provides a unique solution, which is presently unavailable in other languages. A tool based on the Ada Semantic Interface Specification, ASIS, can be created to measure the number of Function Points present in the final linked program that were provided by each package. Function points are a reasonable solution. The use of the other alternative, counting semicolons from lines which are not totally comments, is completely inappropriate for Ada. The inclusion of reusable structures based on object oriented class-wide programming and Ada’s an excellent facility for generics renders obsolete simple linear measurements, such as the aforementioned counting of semicolons. Although the Ada package structure is excellent for creating and organizing libraries for re-use, it can result in the user being forced to acquire a libraries which contain a very significant amount of extraneous material. Basing payment on the actual amount of software used, is the equitable solution to the pricing of the fractional use of large libraries.

The term source text will be used instead of the more common source code. This discussion is totally irrelevant to software artifacts that require cryptography skills to interpret. The subject of providing the source text of the packages was not directly discussed in the above discussion, since, it is not materially relevant. Of greater significance, the availability of source has significant advantages to both the seller (producer) and customer (user). Since the benefits to the user are well known, this discussion will be limited to the benefits to the seller.
1) With Ada, customarily, the user already has the specifications, which often include the private section. This information provides significant insight into the design and organization of the packages.
2) Part of the vendor's responsibility for usability is transferred to the customer.

3) The customer will often review the packages, which will often result in a request for an explanation of part of the source, discovery of errors and suggestion of how to fix the problem, or better yet, supplies a patch to fixe the problem.
4) When the compiler vendors change library formats, the package vendor will not have to recompile their packages. Most Ada packages in source can be used by any Ada compiler that includes annexes which were used in the source.
5) The difficulties of managing compiled Ada '83 libraries was one of the main sources of the negative image of Ada.
6) Source is relatively compact compared with libraries and can be shipped via the internet.

One argument against providing source was that it facilitated software piracy. One very simple counter argument is that there are many successful industries that do provide source. These include: book publishing, video tape, CD-ROM, and music publishing. One other argument is the well known fact that some piracy may actually benefit the vendor. Firstly, the pirated version serves as a demo product. Secondly, once the customer uses a product, some level of interaction with the vendor is often required. One obvious benefit to the customer of purchasing software is to inform the vendor that the customer is a user and thus, should be informed of bugs and updates, etc. In fact, there is a symbiosis between the customer and the vendor. The customer has a vested interest in the vendor staying in business.

No Future: 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 applications. Actually, Microsoft has several significant disadvantages, in spite of the obvious capabilities of its management. Firstly, it is a very large organization. My past experience has convinced me that the COCOMO coefficients for projects developed at large organizations are very large. In spite of the highly touted virtues of creating object oriented software applications in C++. Microsoft Office does not appear to have been assembled from common parts. Word does not inherit its tables from Excel, etc. The use of floating point numbers to represent money, at present, has no justification. Fortunately, for Microsoft and one of the major reasons for its dominant position is the sheer technical and commercial incompetence of its competitors.

No one has tested the market with reliable, efficient extensible, software. Dear reader you can determine the accuracy of this statement by visiting your nearby computer software store. The only extensively reused item is the operating system. The other standard applications are all stand-alone. They are not based on Microsoft Office applications. Certainly, many DLLs (ActivX components) are shared. However, considering the number of copies sold, the number of add-ins or derivative applications is surprisingly small.

In short there is money to be made with commercial, quality products written in Ada . The long term viability of the present commercial software culture which sells “upgrades” rather than having recalls is infuriating. The inherent confusion of the source language C or its derivatives shows through. Most of the present products are unreliable, inefficient and awkward to use. The two key definitions are user hostile and artificial stupidity. During the nineteen fifties, the Japanese automobile and electronics manufacturers followed the teachings of Total Quality with devastating effects. Ada provides software entrepreneurs with a similar opportunity.