Knocking Down the Corporate Barriers to Lisp

Randal V. Zoeller
Itasca Systems, Inc.
7850 Metro Parkway
Minneapolis, MN 55425
(612) 851-3154

Jeff Galarneau
Itasca Systems, Inc.
7850 Metro Parkway
Minneapolis, MN 55425
(612) 851-3155

Abstract

It is a well-known fact that the software development industry has not embraced Lisp as a mainstream language. Despite the flexibility, power, and heritage of the language, the fact still remains—Lisp is a niche development language. While there remain staunch proponents of Lisp, their recommendations, proposals, and productivity gains are often overlooked. In many cases, the programming language used to develop an application is mandated by management, often for reasons that appear to make sense (e.g., availability of tools, portability of the application, and acceptability by the customer). How then do Lisp advocates gain acceptance against the groundswell generated by mainstream languages such as C and C++? Knocking down the barriers to Lisp involves not only co-existing with the mainstream languages, but also complementing them. This paper describes business issues that must be addressed to enable Lisp to become accepted in the computer industry and in Corporate America as a whole. Additionally, it explains what it means to "complement" mainstream languages and describes some mechanisms used to accomplish this task.

1 Introduction

Lisp was developed almost a half century ago as a simple, flexible programming language to allow easy manipulation of list constructs. During the last 40 years, Lisp evolved into a complete programming environment suitable for a wide range of applications. Unfortunately, Lisp was also saddled with the burden of being known as a research, or prototype language, unsuitable for application deployment. What went wrong?

In the early 1980's, during the heyday of the "Artificial Intelligence boom," Lisp was a prominent language for software development. Real products appeared that were implemented in Lisp, and successful Lisp-based hardware and software companies were launched. Unfortunately, this powerful symbolic language became overshadowed by static and less flexible languages like C and Pascal. The advent of object-oriented languages like C++ has only furthered the demise of this language. What can be done to change this situation?

The purpose of this paper is to discuss some of the issues involved with delivering commercial applications developed in Lisp. More importantly, it sets out to promote discussion about what must be done to put Lisp into the mainstream of Corporate America.

1.1 The Motivation

Why is it important for Itasca Systems personnel to get this message across? Because Itasca Systems develops and markets an object database management system ("ODBMS") that is implemented almost entirely in Lisp: The ITASCA Distributed ODBMS. Who better to comment on commercial Lisp applications than a company whose main product is implemented in Lisp. We present to you the observations of a company that successfully markets a Lisp-based application in a predominantly C and C++ world.

ITASCA is the most dynamic, feature-rich ODBMS on the market today. For Lisp people, these dynamic features may seem commonplace, but in the object database industry, these features set us apart from our competition. For example, like CLOS, ITASCA allows the user to modify the schema definition dynamically without having to shutdown the database and recompile the applications. Virtually any schema change can be made anywhere in the class hierarchy. Try doing that with a C++-based ODBMS.

ITASCA's Lisp implementation gives us a big advantage over our competitors by enabling us to provide capabilities that our competitors cannot approach [1]. Yet, as we now enter our 4th year of business, we still hear criticism of our Lisp ancestry. The degree of misinformation and the amount of hearsay is amazing. In some cases, people dismiss the technology because 20 years ago they worked with Lisp, and it was interpreted and slow. Obviously, education about Lisp and its capabilities is in order.

2 Coexisting with Other Development Languages

In order for Lisp to thrive as a viable development alternative, the concerns from developers working in other programming languages must be addressed. Many of these concerns revolve around the lack of common ground. Ideally, development systems should work for developers, rather than against them. Too often a tool or system gets in the way of a developer, particularly one who is unfamiliar with the environment. For example, if a Lisp developer is forced by management to use C as a programming language, that developer is also forced to use C debugging tools. Not only is the language possibly foreign, but the tools are foreign as well. In this example, the tools are working against the developer. A common ground must be established so that the developer can work in the environment where he/she is most productive.

2.1 What is Needed?

A key to successful Lisp applications is to establish common ground with people who like to work in C, C++, Smalltalk, ADA, etc. This common ground goes beyond foreign function interfaces. It is necessary to establish a basis where development tools, data, resources, and even ideas can be shared among people working in different programming languages. As an analogy, consider two people who speak different languages. A foreign function interface would merely serve as an interpreter, while a paper, pencil, and the ability to draw pictures provides a common ground for understanding. With the common ground established, the two people can appreciate the differences in the languages, yet still share information and cooperatively coexist.

With common ground established, the Lisp developer can not only work side by side with developers using other languages, they can complement them. With this in mind, what competent manager could possibly refuse developers to work in the environment where they're most productive?

As for deployed Lisp applications, they must be able to handle data and data access from a wide variety of sources. By providing more "open" interfaces, the barriers and objections put up by many misinformed or under-informed developers can be put to rest. Foreign functions are just one piece in the puzzle. Transparent access and interoperability for C++ programmers (and others) is a necessity.

Unfortunately, if no changes occur in the goals of Lisp developers and vendors, a different kind of change will occur. Even Lisp users themselves are under pressure to make changes. To quote one Lisp user [4], "We'd like to continue to use Lisp for the life of the application, but the pressures of the market may force us to change over completely to C++ in the next few years."

2.2 What is the Common Ground?

One of the most important factors for cooperative systems in a development environment is the ability to share data. As long as applications written in diverse languages can easily share data among one another, they have a basis for communicating *and* understanding.

The technology which provides the basis for sharing data among applications is a Database

Management System ("DBMS"). The problem today is that very few DBMSs support the sharing of data from a wide variety of programming languages. More alarming is that with the strong emphasis on object technology today, there is little or no support for a mixture of object-oriented languages among most ODBMSs, except for one: ITASCA.

Why is ITASCA important to the readers and to the Lisp industry in general? Because ITASCA is written in Lisp and has application programming interfaces ("APIs") for CLOS, Lisp, C++, C, and Ada. More importantly, it provides that common ground necessary for Lisp applications to cooperatively coexist with applications written in other languages. Applications written in CLOS and C++ can actually share objects!

The dynamic capabilities of Lisp are truly unequalled by any other commercial development environment. Due to its Lisp heritage, ITASCA is able to benefit from this flexibility. ITASCA provides direct access to literally any programming language that can open a TCP/IP connection. Because the system stores the data in a language neutral format, these languages (and potentially any other) can store and manipulate data using their own native mechanisms.

What this flexibility means is that a C++ programmer manipulates C++ constructs, just as a Lisp programmer manipulates Lisp constructs. The key is that the stored data is maintained in a dynamic, neutral format that is automatically changed as necessary to conform to the accessing language.

To further the portability across languages, every database function is provided in each API. This enables programmers and users to work entirely in their preferred programming environment without ever accessing Lisp directly. Therefore, ITASCA serves as the common ground to tie languages together persistently.

The net result of these features is a robust and complete database that is suitable for a wide range of application domains. These areas are as varied as decision support, concurrent engineering, multimedia, GIS, CAD, and CASE [3]. This support means that a Lisp developer can recommend ITASCA to their management, confident that developers in other languages will be appeased, and the project can succeed for the long haul.

3 Making Lisp Deployment a Reality

A key to successful Lisp application deployment into mainstream Corporate America involves not only cooperative coexistence, but also education and value-added complimentary technology. Without each piece, a Lisp project runs the risk of being moved to a more traditional development language, or worse yet—cancelled.

3.1 Educate

One aid in making Lisp deployment a reality is to increase the level of understanding and education about Lisp and Lisp development environments. Just like the person who quoted their 20 year old experiences, many people are making decisions based on outdated and incorrect information.

The goal is not necessarily to convince a majority of people that Lisp is good. It is a well known fact that Lisp and CLOS are powerful, flexible, high-level programming languages that enable programmers to be extremely productive [5]. However, there is still an incorrect belief that Lisp is inefficient and monolithic.

What we see as important is to eliminate some of these stigmas that are associated with Lisp without offending those proponents of more mainstream programming languages. This, in combination with cooperative coexistence, can be an effective persuasion tool.

3.2 Don't *Just* Educate. *Impress.*

Rather than merely attempting to educate the non-Lisp community about the virtues of Lisp, ease their worries. Many concerns can be addressed by providing functionality that is not normally available in non-Lisp environments without taking those environments away. This should be done as transparently as possible.

In some cases, showing the virtues of Lisp may require hiding Lisp completely. Other times it may require using Lisp just to enhance func-

tionality that is beyond the capability of other languages or tools. Either way, the functionality provided by Lisp can be used to develop applications that are certainly quicker to deliver, and provide consistently more functionality. The key to success is adding value—not attempting to replace, which will surely fail.

3.3 Do You Hide the Lisp?

One concern often expressed is the need for in-house Lisp expertise in order to purchase a Lisp application. This should not be true, especially in the case of turn-key applications. Applications developed in typical static languages like C, C++, or ADA do not reveal their implementation details to end-users. It is virtually impossible for a user to determine what programming language was used to develop these applications. Most end-users do not care what language was used to develop the application. They just want an application that works reliably and consistently.

Perhaps a possible goal is to deliver a Lisp-based application that is indistinguishable from a C-based application—with the obvious differences in ease of programming, flexibility, and dynamics. Then, extend this basic model for sophisticated users.

4 Summary

In this paper, we attempted to provide a sense of the requirements necessary to develop a successful Lisp application in today's market. What we have observed and recorded is certainly not the only experience. They are merely the observations of a successful company who markets a Lisp-based product.

Rather than competing for static or even dwindling market share, the Lisp community must band together to promote the virtues of Lisp as a viable production and development language. Perhaps even an organization like the "Milk Council," whose goal is to promote and educate. Maybe something like, "Lisp, it does a programmer good."

The main thing we hope we have accomplished is to provide the reader with some insight on how Itasca Systems has approached the task of developing and selling a complex Lisp application in a predominantly C and C++ world.

5 References

[1] Ahmed, S., et al., "A Comparison of Object-Oriented Database Management Systems for Engineering Application," MIT Technical Report IESL-90-03, October, 1990.

[2] Itasca Systems, Inc., "ITASCA Distributed Object-Oriented Database Management System Technical Summary Release 2.2," 1993.

[3] Liu, L. and Horowitz, E., "Object Database Support for CASE", *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD*, Prentice-Hall, Inc., 1991.

[4] Newquist, Harvey P. III, "In Practice - All It Takes Is Finding the Right Recipe," *AI Expert*, May, 1992.

[5] Richardson, C., "Software Development with CLOS", *Object Magazine*, March-April 1993.

[6] Stein, J., "OODBMS Object-Oriented Technologies for Complex Data-Management Systems", *SunWorld*, Vol. 4, No. 5, May 1991.

[7] Zoeller, R. V., Galarneau, J., et al, "The Development of the ITASCA ODBMS", *Journal of Object-Oriented Programming*, July 1991.