In a stream from the singular space to a distributed world

From the transcript of ICERL'91

Technical Report No. 94-007

Masayuki Ida[*], ed.
Computer Science Research Laboratory
Information Science Research Center

Aoyama Gakuin University
Address: 4-4-25 Shibuya, Shibuya-ku, Tokyo Japan 150

November 5, 1994

[*] ida@csrl.aoyama.ac.jp

From the singular space to a distributed world

This morning, I got an E-mail from Kent Pitman, telling me the current status of the ANSI X3J13 Common Lisp draft.

```
Date: Fri, 4 Nov 1994 11:55-0500
From: kmp@harlequin.com
Subject: X3J13 status update
To: x3j13@ai.sri.com
Cc: kmp@harlequin.com

Just got a call from Lynn Barra at CBEMA.  We have X3 approval on our
document, which they refer to by its project number X3.226!

Now just ONE MORE STEP: approval by ANSI BSR.  Until this occurs, we are
STILL NOT an ANSI standard.  But we are moving forward, so PLEASE hold off
on the ticker tape parades until then.
...
Anyway, although she was unable to guarantee ANSI's action, Lynn seemed
quite optimistic that we could look forward to an ANSI standard before
Christmas with a "94" date on it.

Cross your fingers and watch under your Christmas tree!
 -kmp
```

Finally, really finally, we seem to be approaching the goal. It took long long days. We had experienced various hills and bottoms. This goal, the ANSI standard for Common Lisp, is also a starting point for our next step, which we have been seeking. This technical report is compiled to give the thoughts on it.

Masayuki Ida
November, 5th, 1994

Preface

This is from the ICERL-91 transcript. ICERL-91 (International Workshop on the Cooperative Extension and Refinement of LISP 1991) was held for March 26 and 27, 1991 at Hyatt Rickey's, Palo Alto, California. This report contains the whole record of the day two recorded on the audio tape, including the speeches and discussions, and the position papers. We could not compile the day one record in this report.

This report basically contains the record of the conference held in 1991, *not* in 1994. Three years passed. Let me explain why I decided to write and compile this report.

In April 1994, returning from the one year stay at AI Lab, MIT, I needed to clean up my desk to resume research life in Tokyo. In the mountain of many garbages, I found one file which contains the draft documents I had had no time to finish before the departure to Boston in April 1993. By re-reading them again, I got convinced that they should be published quickly, since they are still valuable to know the process of making Common Lisp. I am sure it would also serve as a case study story to learn pro and cons for the International standardizations, how language specificiation design process was troubled even though many excellent people joined to the group. Yes, we can learn a lot from this vivid story.

Let me describe what was the year 1991. In March we had this workshop, ICERL'91. In November we had JPAL-91 in Tokyo. At JPAL-91, we had 15 exhibitors and several technical sessions. But, I must say the numbers of the attendees had decreased from such events of 1990.

In USA, Lucid and Franz were the tops. David Moon left Symbolics and joined Apple Cambridge Lab. Xerox Common Lisp went to a smaller firm. Apple was to announce MCL2.0. X3J13 expected to finalize the draft but found tens of modification requests.

In Europe, Eulisp seemed to win. For the ISO standard, Europe seemed to be stronger.

Viewing for computer software and systems in general, Windows was becoming to be ready. Internet was almost ready to get huge enthusiasm.

In this stream, I felt the Lisp community needs to cooperate.

This is the exact reason why I organized ICERL'91.

The title of this report is "In a stream from the singular space to a distributed world." This title was *not* appeared at the ICERL'91 workshop. I gave this title to this report, since the old and classic Lisp community was very singular space, I must say, and we are heading to a distributed world of co-existence. We are on the way to some distributed world, though I or anyone can not predict what is the destination in detail.

Please read the report sentence by sentence, and think of the future.

Masayuki Ida

Acknowledgement

The author would like to express his gratitude to many persons, especially to Mr Mark Son-Bell who worked hard as a co-chair of the program committee and who has patiently made the draft trascript from the audio recorded tape, and to the attendees. Special thanks are due to Gregor Kiczales and Dick Garbiel who had gave us excellent talks at ICERL91.

— No Migration but Co-existence —

In the paper, most of the personal names are quoted using the following table.

```
********
Alcabes = Harvey Alcabes, Apple
ASmith = Andrew Smith, Harlequin
Gregor = Gregor Kiczales, Xerox PARC
Hanoch = Hanoch Eiron, Franz
HES = Howie Shrobe, Symbolics
Ida = Masayuki Ida, Aoyama Gakuin University
Joaquin = James Joachin, Apple
Kobryn = Cris Kobryn, SAIC
Kosaka = Takashi Kosaka, Aoyama Gakuin University/CSK
Kromer = Gene Kromer, Franz
MAS-B = Mark Son-Bell, ILA
PGA = Phill Apley, DEC
RPG = Dick Gabriel, Lucid
RWK = Bob Kerns, DEC/MCC/ILA
Sato = Mamoru Sato, TIRL/Canon
Shiota = Eiji Shota, Nihon Symbolics
SWM = Scott McKay, Symbolics
Tanaka = Keisuke Tanaka, Aoyama Gakuin University
Veitch = Jim Veitch, Franz

********
```

Notes: In the text, several portions of the tape recorded were very difficult to transcript. such part are indicated by three dots, ...

Contents

1 How We evaluate the Past, Present and Future of Lisp **3**
1.1 Introduction 4
1.2 PC & WS Growth 5
1.2.1 Personal Computer Production in 1989 5
1.2.2 Workstations / Mini-computer / Mini-supercomputer Production in 1989 6
1.3 Determining Our Research Direction in the Network Era 7
1.3.1 Overall View 7
1.3.2 Considerations for Aoyama Gakuin 8
1.3.3 Key Issue to Consider is: 9
1.4 *YyonX* 9
1.4.1 Outline 9
1.4.2 *Yy* Goal and the *YyonX* Implementation Characteristics 10
1.5 Jeida Common Lisp Committee (JCLC) 12
1.5.1 It's Overview and the History 12
1.5.2 Questionnaire Survey for 1985, 1987, 1990 14
1.5.3 1987 Questionnaire Summary 15
1.5.4 1990 Questionnaire Summary 16
1.5.5 JCLC Working Groups Activities 18
1.5.6 Common Lisp Implementations and Related Software History 19
1.5.7 Lisp in 1989 19
1.5.8 JPAL'91 20
1.6 Where do we go ? 20
1.7 Discussions 22

2 Talk by Gregor Kiczales **23**

3 Talk by Richard Gabriel **33**

4 Discussions **39**
4.1 Discussion 1 40
4.2 First Afternoon Session 50
4.3 Second Afternoon Session 64

5 Position Papers **73**

ICERL'91 Summary

ICERL'91 was sponsored by Aoyama Gakuin University, Computer Science Research Lab., and International Lisp Associates, Inc. and held under the auspices of International Foundation for AI in Japan.

The program committee was formed by Masayuki Ida (Aoyama Gakuin University) for Japan, Mark Son-Bell (International Lisp Associates) for USA, and Jo Marks (Harlequin Ltd.) for Europe. The local arrangement is done by Bill York (ILA). The workshop secretary was Keisuke Tanaka (Aoyama Gakuin University).

Here is the summary of the workshop.

Dates: March 26 and 27, 1991

Place: Hyatt Rickey's, Palo Alto, California

Workshop Objectives: To bring together a select group of international representatives from academia, industry, and government to present their recent research and development efforts related to the Lisp programming language. The workshop is by invitation only and will provide a forum for the dissemination of technical information and for the discussion of potential avenues for future cooperation. The workshop will have a particular emphasis on user-interface issues, but may also touch on the topics of distributed and parallel computation, application delivery, and object persistence.

Workshop Format: Two days of presentations and moderated discussion. The first day will be reserved for detailed technical presentations, especially of a number of Lisp UI projects. A basis will be sought for future standardization efforts in the Lisp UI area, and especially for the unification of the CLIM and *YY* specifications. The second day will consist of brief presentations of the attendees' current activities and future plans, followed by a discussion of potential areas for cooperation. (Managers and executives may choose to attend only the second day.)

Invitations: One of the goals of the workshop is to be balanced geographically, within a limit of 45 chairs (where a "chair" means an organization, not a person). Each of the three regions (Europe, Japan, USA) is allocated 15 chairs. The Program Committee is therefore calling for presentations from interested parties in each of the regions who may not have been invited directly because their work was not familiar to the members of the Program Committee. Submissions must be in the form of a summary statement of work completed or in progress (not to exceed 300 words) and must be sent by Feb. 25, 1991 to one of the Program Committee members at the addresses listed at the top of this call. Those accepted for invitation will be notified by March 1, 1991.

Workshop Schedule: • **Day 1:** User Interface and Lisp (featuring the Unification of CLIM and *Yy*)

- Morning: Technical presentations on platform-independent APIs and platform-specific toolkits development tools (the purpose of the morning session is to provide all the attendees of this session with a solid technical understanding of all of the efforts)
- Afternoon: Roundtable Discussion I: Integration of the external specification (API) of Lisp UI tools, from three perspectives: A common API (CLIM+YY), platform-specific tools + a common API, and user desiderata.
- Evening: Banquet

• **Day 2**

- Morning: Keynote speeches and Regional Overviews: by Masayuki Ida, Gregor Kiczales, Richard P. Gabriel.
  Brief statement by each attendee of current status, activities, and interests for discussion.
- Afternoon: (directed discussion on) "Basic Standards" question: ANSI CL, EuLisp, "Lisp Lite", "Big Lisp (i.e. CL+CLOS+CLIM/YY)" or "integrated Lisp", possible areas for future cooperation, distributed computation, more UI projects, persistent objects, better development and delivery tools, possible formalizations of cooperation, Hawaii research center for "Open Symbolic Software Foundation".
- Evening: Dinner (informal)

Position Papers: Anyone wishing to attend the workshop - whether by direct invitation from the Program Committee or after acceptance of a submitted response - must complete the registration and position form included below and send it to one of the members of the program committee (e-mail preferred) before Feb. 25, 1991. The position papers will be collected for distribution in the workshop proceedings.

Chapter 1

How We evaluate the Past, Present and Future of Lisp

1.1 Introduction

(In the morning, March 27th, 1991.)

The recent Lisp community seems to believe Lisp with its 1) industrial strength, 2) formal aesthetics, and 3) tool for Laboratory experiments. As a result, a new question arises, that is, "What Industrial Lisp should be ?"

According to the 1990 questionnaire summary of Jeida, 1) Prototyping (76), 2) Research (56), 3) Products (51), 4) Internal Use (38) are the top four usage. Meanwhile, people seem to complain about Common Lisp implementations for the following points. 1) Too Large. In Specification and/or At Run-time. 2) Too Slow; Irritating GC, Slow Compilation, Slow execution. 3) Too Rich Functionality (Culture, Education problem).

Most of the current Lisp usage is for prototyping and research. And the source codes developed in Lisp were converted to a different language like C.

Why people rewrite Lisp sources they have into other languages ?

Or, why people migrate their systems from Lisp machines to other platforms ?

Is this the right thing to do ?

Where is our Future ?

These are the original questions.

Here is the analysis to answer the above. There are two types of environments:

1. Development environment:choice of the developers
2. Runtime environment:choice of the customers, end-users

We must realize these two environments do not match usually. And there exist lots of useful software/database (written in other languages or as a package software) Lisp programs want to collaborate. Then, a choice among the following is typical.

1. Rewriting into other Language
2. Migration through the Common Lisp Portability
3. Common Lisp at Run-time. As is. Or as an delivery version.
4. Bridge between user's environment and application server using computer network.

The last one is almost equivalent to the authors view. The approach by the author is symbolized by "No migration but Co-existence" which is the slogan of *Yy* project.

In this talk, I like to introduce the following data too.

- PC and WS market trend: Growth of PC and Workstation world. Because most of the platforms to implement the user interaction part at least will be on such machines.
- The group activity at JCLC (Jeida Common Lisp Committee) as what is the history of Common Lisp Implementations in Japan.

1.2 PC & WS Growth

Any computer languages and application software require hardware to run. The analysis on the hardware trend is important to know how we are. First of all, we cannot ignore the growth of personal computer world and work station world, because people/civil life is with them and domestic culture is usually tied with them.

One of the exceptional situation of Japanese market is, NEC PC is 50% share of the total Japanese PC market. IBM is not the number one from the beginning.

1.2.1 Personal Computer Production in 1989

According to the Report of the Jeida Personal Computer Board (90-PA-1, 1990, March), we have the following data.

- 2.4 million personal computers were sold during 1989. (1.66 million units for japan domestic, 0.75 million units for export. 22% are 32 bits machines)
- 11 Billion Dollars (1,771,000,000,000 yen) was the total amount.
- 122% increase from 1988 for total production.
- The numbers of units for export is down to 92% of 1988.
- To North America, down to 80%,(0.19 million units).
- To Europe, down to 97% (0.36 million units).
- (US exports 0.88 million units to the world 1988).

The characteristic point is "70 % is consumed in the domestic market." In other words, only 30 % were exported. It's smaller number than I expected. And the total amounts of export (0.75 million) is decreasing from the previous year, and is less than US exports to the world (0.88 million units).

In the same report of Jeida, PC Manufacturers Choice to 1990 is described. It is summarized into the following list.

- High Functionality, and Variety of OS
- Book size or Notebook size Computer
- Color LapTops

The first one means, MSDOS will not be considered to be the only OS. These three principles are well executed after all, and bring the current PC market situation. That is, the recent popular personal computer is color desktop or color notebook computer with 486 DX4 or pentium as its CPU, running MSDOS, Windows 3.1, Windows NT, and various versions of UNIX OS.

Here is the table of the survey among 246 answers from industrial PC users on the choice of operating system. Since it was done in 1989, there is no explicit appearance of Windows, the expection is with OS/2 and UNIX.

— OS for PC —

OS name	current	Near Future
MSDOS	206	↘ 62
OS/2	15	↗ 110
UNIX	26	↗ 127
TRON	0	↗ 29
Original OS	38	↘ 17

Here is the table of the survey for the same people on the application trends. It shows Data Base, Graphics and Electronic Communication are highly expected. Though there was no word like 'multi media', we might feel the emergence of the multi media era.

— PC Applications —

Application	Current	Near Future
Word Processing	212	↘ 85
Spread Sheet	149	↘ 67
Electronic Communication	68	↗ 98
Data Base	83	↗ 114
Graphics	72	↗ 99
Package Software	52	→ 52

1.2.2 Workstations / Mini-computer / Mini-supercomputer Production in 1989

According to the Report of the Jeida Mini Computer Board (89-Kei-20, 1990, March), we have the following data.

- 31,975 units, 319,600,000,000yen (2.3 billion dollars) in 1989.

- Statistics includes CTC (SUN), OKI, Concurrent, Strutus, Ustation, CEC, MODCOMP, Tosbac, Toshiba SUN, Olivetti, Data General, Digital Japan, NEC, JAC, Hitachi, PFU, Fujitsu Sun, Melcom, Yewcom, HP, AT&T 3B
- 158% of 1988 in units
- Expected Growth Rate in the next 5 years:
  25 % each year in units, (4 times in 5 years)
  14 % each year in yen. (2 times in 5 years, 700 billion yen in 1994)

Since this is one of the early data Jeida gathered for the workstation market, they have no further data. We can expect more in the future.

According to the Report of the Jeida Mini Computer Board (89-Kei-20, 1990, March), there was a questionnaire for advanced 60 organizations, and the summary was reported.

There were Alliant, Ardent (Titan), Convex and other 6. Most of the applications are on Simulation. 78.6 % is Unix, and 12.5 % is Unix-like. Languages used on mini-super are, 91.1 % Fortran, 87.5 % C, 7.1 % Pascal, and 3.6 % Lisp. 87.5 % are connected to computer network (Standalone use is 12.5%).

So, the total figure of mini-super computer is, Unix based Fortran machine connected to a network as a server for computation like simulation.

This story on mini-super gives us an interesting view for the future. That is, "We can co-exist with UNIX, through Network environment."

1.3 Determining Our Research Direction in the Network Era

1.3.1 Overall View

The internationalization and the computer system relation to it is my keyword to design a new system. International Internetwork is now widely popularized. Technology Transfer to NIES is one of the important Japanese role in asia. Also we need to keep good communication with Europe and USA. Globalization is a popular keyword and the reality of "Border-less Society" is coming. While personalization, one of the basic trends, is in progress, we must standardize the industrial matters. It should be with a principle for co-existence of different life style. There are world for open software and joint development.

Distributed computer system should support this environment.

The following is the list of the elements for further design and development.

1. We are in the age of Network / Personalization of computing environment LAN / WAN and Internet connection. → Lisp should be one part of network.
2. Lisp is suitable for personalization. (Lisp might be a successor of Basic with much more refined features.)
3. Lisp is suitable for software development. (Lisp is an integrated tool over design, coding, debugging, and production run.)
4. Activation of Lisp Community = Free/Open Competition.

1.3.2 Considerations for Aoyama Gakuin

Aoyama Gakuin University was established in 1874. There are 19,000 students, 3 campuses, and 6 colleges. Information Science Research Center (ISRC) is responsible for overall computing equipments. Computer Science Research Lab (CSRL) of ISRC which is my laboratory is to explore the frontiers. CSRL implemented the experimental campus network over three campuses and the connection with the Internet. CSRL is the gateway to the Internet.

The Important Principles at CSRL is to esteem the user choices and variety of the life styles.

The Unified System ? → NO !

It means language and system choice should be done by users side:
C, Fortran , Lisp, ...MS/DOS, UNIX, Host OS, ...

And, *No Migration but Co-existence* is the principle.

In other words, there should be no extra works to port software among computing equipments since "Migration" means everything leave from one place to another, and nothing worth with our co-exstence principle. Just the waste of man power.

To make this clear, let me show several examples; Sun WS cannot replace the super computer. Workstations cannot run all the Japanese personal computer applications. Symbolics machines cannot have all the UN*X tools.

And here is the personal comment on Symbolics.

- Public acceptance of the superiority of Dynamic Windows.
- Sharing the PDS between US and Japan. More E-mail communication. User community in Japan is not much grown yet. To do so, we need much more trigger.

- Fully equipped CL applications force the UN*X workstation as if it were a personal computer.
  Allegro CL/ Composer needs 12Mbytes and GnuEmacs needs 8Mbytes or more on my SUN-4 260C, making SUN-4 260C a personal computer. Sometimes I got a message 'no more memory' with 32Mbyte swap space. (Practically impossible to run several person's job.)

- As a workstation, needs much more functionality on Japanese character handling in every text handling. → if it is practically impossible, one choice is to be dedicated board manufacturer for standard platform workstations/PCs.

The Choice for Symbolics to Survive is:
"Sell it as a powerfull highest-end Work Station with some user choice application installed "

⇒ To Co-exist in a Network

1.3.3 Key Issue to Consider is:

Our conclusion (our current research challenge) is,

Keep the software on the machine, Change the user interface to *Yy* and then, applications are accessed from other WSs with *Yy* .

Remote (X-) Screen for Symbolics means the usage of a software developed on Symbolics as is from UN*X workstation. With several reasons, there exist several software which cannot be ported to UN*X. Which is better, one big computer or several standard sized workstations ?

The answer is several standard sized workstations which are connected on the same network and share the roles.

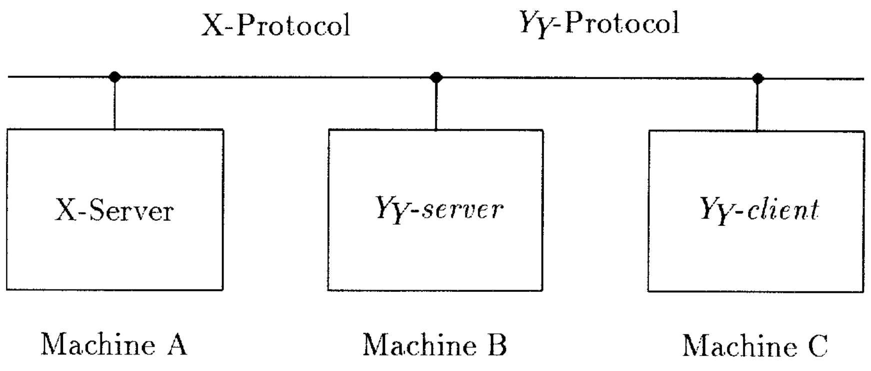
1.4 *YyonX*

1.4.1 Outline

The basic explanation of *Yy* and *YyonX* are in my position paper. In this section, I will focus on the issues I am thinking. *Yy*-client can be on the remote IP-reachable machine. In other words, *Yy* is suitable to access remote application server using a local GUI/user-environment.

A (remote) Lisp oriented application server (*Yy*-client) on a network can make interaction via a (local) *Yy*-server (a X-client) with a user (X-server).

Since application server is a client for the GUI environment, we named it as a client. The next figure is a symbolized relation among three processes.



The *Yy* experimental development project started in 1989, as a research project on a portable window for Common Lisp at Aoyama Gakuin University. Assessment was done before March 1989. In April 1989, investigation of the existing Common Lisp window systems were done and the basic design was started. From October 1989 through March 1990, a prototype was made. The supporting members and the development group are from CEC, CSK, Fujitsu, Hitachi, NEC, Nihon Unisys, Nichimen Graphics (Nihon Symbolics). In April 1990, the first version was there. And now, one project in IPA is using *Yy*.

All the sources and documents are available for Internet. Anonymous FTP address is on `ftp.csrl.aoyama.ac.jp`. Mailing list is named `yyonx@csrl.aoyama.ac.jp`.

1.4.2 *Yy* Goal and the *YyonX* Implementation Characteristics

We had summarized the projected features and goal of *Yy* as follows.

- Because *Yy* supports the No Migration but Co-existence Principle,
- No need to port your applications from a machine to other workstations.
- No need to rewrite user interface on porting.
- No need to combine several functional modules into one executable image.
- Your Propriety is guarded while can make co-operation with other software.

An implementation of *Yy* is called *YyonX*. Here is the summary of its characteristics.

- *YyonX* is a Pilot Implementation of *Yy* on X-window.
- It uses a server/client model.
  - *Yy-server* is a X-client for NWSI and low level stuffs of YYWS.
  - *Yy-client* is a X-client for high level stuffs of YYWS and YYAPI.
- CLOS oriented portable window tool kit
- The Server-Client Model to lighten the CPU / Memory loads

  AI Application requires large memory and CPU. Separate Application process from Interface process. Experimental Research on Load balancing / distribution between them Experimental Research on The Protocol between them.
- Never Make a Xlib (CLX) call by user

YyonX is views as a system using a server-client model. At the same time, *YyonX* is a layered window toolkit system. As a server-client implementation, all the window operations are done by the cooperation of *Yy-server* and *Yy-client* processes. As a layered window toolkit, there are three componets; NWSI, YYWS and YYAPI. There is a conceptual correspondence between the layered model and sever-client implementation model designed.

Yy-server has two parts; a NWSI for X-window and a kernel of YYWS. *Yy-client* has two parts; the application part of YYWS and the YYAPI. *Yy-server* and *Yy-client* are implemented as X-clients. Then, they can be loaded on separate machines on the network. There is *Yy* protocol.

Server-Client Model Implementation of *Yy* Three Layers

$$\begin{tikzcd}
\text{YYAPI} \arrow[d, leftrightarrow] & {} \arrow[d, leftrightarrow, "Yy\text{-client}"] \\
\text{YYWS} \arrow[d, leftrightarrow] & {} \arrow[d, leftrightarrow, "Yy\text{-server}"] \\
\text{NWSI} & {}
\end{tikzcd}$$

The *Yy* Protocol is the next key issue on the design. Performance and functionality of *YyonX* is determined and measured by observing and evaluating the protocl, which is used for the communication between *Yy-server* and *Yy-client*.

Here is the quick view of the protocol. All the actual images are stored and managed in the *Yy-server* (*Yy-client* has no bitmaps). The objects of *Yy-server* are called *TERRITORY*. *Yy-client* has a sub process to dispatch events from *Yy-server* .

Yy protocol has currently 50 Commands. The packet format is 4 bytes × (2 + Command Arguments), and is implemented on top of UNIX socket interface, which is facility for Inter Process Communication through TCP.

There are three command types.

- **Notification** (server → client. 2)
- **Command** (client → server without requiring ack. 36)
- **Instruction** (client → server requiring ack/answer 10)

The synchronization of the processes in server side and client side is done using the intermittent synchronization mechanism.

1.5 Jeida Common Lisp Committee (JCLC)

1.5.1 It's Overview and the History

There was Jeida Common Lisp Committee (JCLC) during 1985 through 1991. It played the role as a driving force on the mutual understanding body for the technical and the pre-standard matters on Common Lisp. This section is to report the activities of JCLC. The purposes of JCLC do NOT include the issue of making a domestic standard lisp in Japan. The work was governed by the following:

- Understanding of the Common Lisp specification
- Discussions on implementation techniques
- Investigations of existing Common Lisp implementations
- Communications with international Common Lisp community
- Investigations of Common Lisp applications

We had monthly meetings since May 1985. For once a year JCLC had workshop/seminar events. And there were several working groups time to time. The questionnaire were made in 1985, in 1987, and in 1990.

The members of JCLC were came from the following institutions:

Aoyama Gakuin Univ., CEC, CSK, DEC Japan, Electro-Technical Lab., Fuji Electronic, Fuji Facom, Fuji Xerox, Fujitsu, Hitachi, JRC, Matsushita(panasonic), Meidensha, Mitsubishi, NEC, NTT, Nippon Data General, Nippon Unisys, Omron, Oki Electric, PanaFacom, Ricoh, Sinko, Sanyo, Sharp, Sony, Sord, Sumitomo, Symbolics, Toshiba, Yamatake-Honeywell, Yokokawa, YHP. (alphabetic order)

The numbers of the members for each year were not constant. Starting with 31 persons in 1985, the highest days had 48 persons joined. The detail is shown in the following table. The activities of all the members were so impressive and we had enormous fruits.

Year	Number of members
1985	31 (28 organizations)
1986	37 (30 organizations)
1987	24 (47 for WGs, 27 organizations)
1988	27 (48 for WGs, 26 organizations)
1989	20
1990	20

(Some comments on the table: 1985 - 1988 : observer members are included, 1989 - : Jeida administration took the policy to permit the only principal members of the Jeida body to participate.)

Here is the list of activities. The highest event was the Common Lisp Forum in 1989. We had gathered 250 persons.

- **1985.9.11-12**: Workshop. 28 persons; VAX Lisp talk, IJCAI-85 report, Implementation issues, Subsetting, Objects, questionnaire
- **1986.07.08**: Common Lisp / Core seminar. 60 persons; discussions on CL/Core
- **1988.06.06**: Common Lisp Symposium. 110 persons; Common Lisp Overview, Exception handling, Kanji proposal, Loop macro, Questionnaire, CLOS
- **1989.06.06**: Common Lisp Forum. 250 persons; Overview, CLOS, Window Tools (CLX and CLUE), Lisp vs C, GNU Emacs Lisp, Japanization of CL implementation
- **1990.02.08**: Lisp Forum. 120 persons; Overview, CLOS productivity case study report, Real time GC, High Lights of X3J13 clean up issues, CLIM by Bill York and *YY* by Masayuki Ida
- **1991.02.19-20**: Common Lisp Forum. 100 persons; Common Lisp Window Toolkits, *YY*, Jeida CL Guideline, Programming Style Guide, CLOS Tutorial by Jon L White (Lucid)
- **1991.11.26-27**: JPAL-91 Technical Forum and Exhibition

1.5.2 Questionnaire Survey for 1985, 1987, 1990

This section is to summarize the surveys made in 1985, 1987 and 1990. The 1985 survey was for general survey. The 1987 survey was for utility of Lisp and for the follow ups. The 1990 survey was for the deployment of Lisp applications.

The following table is the summary.

– 1985 –	– 1987 –	– 1990 –
350 sent 135 replies 66 CL Users	200 sent 128 replies 100 CL Users	- 222 replies 168 Lisp Users
General Survey of AI Tools. Common Lisp Pro and Con	Utility of Lisp among AI tools. Experiences with Common Lisp	Follow Up.

Seveal interesting data were obtained from the surveys.

Among them, the following is the comments on the 1985 survey; Needs for Subsetting was significant, and Top 11 languages used to develop AI software in 1985 were C(89), FranzLisp (60), Fortran(51), Pascal(49), C-Prolog(46), Prolog-KABA (44), Vax Lisp (40), Smalltalk (38), Interlisp (33), UtiLisp (33), Basic (30), and so on.

The following is the table of the machines used.

– 1985 –	– 1987 –	– 1990 –
Vax (72)	Vax (56)	Sun (90)
Sun (28)	Sun (32)	NEC PC98 (37)
Xerox-1100 (27)	NEC PC98 (16)	Symbolics (30)
		(MacIvory(6)included)
Symbolics (22)	Symbolics (12)	Vax (23)
Fujitsu (17)	Xerox 1121 (12)	Sony News (17)
ACOS (17)	Hitachi (10)	Fujitsu PC (17)
Hitachi (16)	Fujitsu (9)	Xerox 1121 (17)
U-station (15)	Explorer (9)	Mac (16)
DEC-20 (15)	Fujitsu PC (8)	Fujitsu (15)
		ELIS (15)
-	ELIS (7)	NEC EWS (14)
		Hitachi (14)

The following is the implementation used.

– 1985 –	– 1987 –	– 1990 –
Franz Lisp (60)	KCl (42)	KCl (48)
Vax Lisp (40)	Franz Lisp (26)	UtiLisp (35)
InterLisp/-D (33)	Common Lisp (24)	Sun CL (30)
UtiLisp (33)	Vax Lisp (23)	Allegro (25)
		Emacs (25)
		GCLisp (25)
		Symbolics (25)
Zetalisp (18)	UtiLisp (11)	Xerox Lisp (17)
KCl (17)	Interlisp-D (11)	Lucid CL (16)
-	GCLisp (11)	Elis CL (14)
		Franz Lisp (14)
-	Zetalisp (8)	Interlisp-D (12)
		Vax Lisp (12)
-	TAO (7)	ICLisp (9)
		VOS3 Lisp (9)
-	VOS3 Lisp (7)	XLisp (7)
		Scheme (7)

(UtiLisp is not a CL dialect.)

1.5.3 1987 Questionnaire Summary

- Among 128 replies, private companies' development sections (39.1%), research labs (28.1%), universities (10.2%).
- Application area: Expert system building (59), Intelligent man/machine interface (26), Natural language processing (15).
- The ratio of Lisp among development languages:
  Lisp is not/seldomly used (9), Lisp is the major language (53), C is used with Lisp (47).
- Development type: Research (65), prototyping (66), Internal use (27), Commercial product (36), others (8).
- Advantages of Common Lisp: its flexibility (43), productivity (39), rich environment (19).
- Problem with Common Lisp: slow speed (32), large memory requirement (22), irritating GC (13).
- Manys complain about the functionality of debugging facilities, 60% complain about the graphic interface, Satisfaction level with japanese handling is half and half,

- The top four reasons to choose Common Lisp; 1. defacto standardness, 2. compatibility of application software, 3. rich functionality, 4. suitable for large scale development.
- The top four claims on Common Lisp disadvantages are; 1. huge implementation, 2. complex specification, 3. slow interpreter, 4. function naming with long spelling.

The 1987 survey asked the directions for extensions.

Among the proposal to improve CL, the extension of CL specification for the following items are significant;

- object oriented facility,
- loop facility,
- multi process,
- multi national character handling,
- window tool kit,
- graphic functions.

(With this result, the author encouraged X3J13 to adopt CLOS, Loop, character extension proposals. Also started a window tool kit project.)

1.5.4 1990 Questionnaire Summary

- Why you use Lisp? Because Lisp has -
  - High Flexibility / Adaptability 110
  - High Productivity 97
  - Good Environment 59
- Why you use CL among Lisp Dialects ?
  - defacto standard 88
  - rich functions 45
  - portability 33
- What is Bad with the current CL Implementations ?
  - Huge 103

  - Complicated Specification 42
  - Too slow interpreter 36
  - Long name to type 17

- Issues to discuss for CL future
  - Standardization of windowing 79
  - Foreign Function Interface Standardization 74
  - Multi Process handling Standard 60
  - International Character Set handling 40

And the following is the GUI/Window toolkits survey.

- Window Toolkits Used/Experienced
  1. Others (32)
  2. CLX (29)
  3. MS-Windows (20)
  4. Presentation Manager (12)
  5. Common Windows (10)
  6. CLUE (4)
  7. YyonX (3)
  8. CLIM (2)
  9. LispView (1)
  10. Winterp (1)
  11. GINA (0)
  12. GARNET (0)
  13. Express Window (0)

$\Longleftrightarrow$ No Standard Yet

- How you develop UI ?
  1. With Languages other than Lisp (54)
  2. Lisp (53)

3. Window Toolkit (46)
4. Object Oriented System (27)
5. UI tool (14)

- Expectations to the near future of CL Window toolkit

1. Object Oriented Integration (75)
2. Uniform/Standard Operation Interface (70)
3. User Customization Feature (43)
4. Multi Media Extension (36)
5. Reducing the Memory size, Performance up

1.5.5 JCLC Working Groups Activities

- **Object oriented** working group (OOWG 1985, 1986, reformed as OOSG 1987, 1988): CLOS

- **Subset** working group (subsetWG 1985, 1986): Common Lisp/Core includes **356 functions and 20 variables and constants**.

- **Kanji** working group (kanji WG 1986): Jeida Proposal of 1987 to ANSI X3J13. Reformed as Jeida Nihongo Guideline WG 1991.

- **Bboard** working group (bboardWG 1986)

- **Application** working group (apWG 1987, 1988, 1989): Questionnaires, Lisp vs C, CLOS productivity case study. In 1990 and 1991, Windowtoolkits are investigated.

- **Technology** working group (techWG 1987, 1988, 1989): Exception Handling, Scheme, Real Time GC. In 1990 and 1991, Programming Style guideline is being formed.

Temporary groups were occasionally gathered to translate Loop facility documents, exception handling documents, and so on.

1.5.6 Common Lisp Implementations and Related Software History

1984	VAX LISP (DEC), Kyoto Common Lisp
1985	ESHELL/FM (Fujitsu)
	KBMS (NTT), EXCORE (NEC)
	GCLisp (CEC), KEE (CSK)
1986	KS-300 (Nihon Unisys Ltd.)
	KEE/KS-300
1987	VOS3LISP (Hitachi), TAO (NTT)
	Lisp System (Fujitsu)
	Symbolics Common Lisp
	ABCL/1 (Tokyo Institute of Tech.)
	Nihongo San Marco Lisp Explorer (CEC)
1988	Hi-UX LISP (Hitachi)
	Lucid Common Lisp 3.0 (CSK, ...)
	Allegro CL (Nippon Steel, ...)
1989	VOS3LISP E2 (Hitachi)
	ELIS Common Lisp (NTT IT)
	Nihongo Common Windows (CSK)
	YY on X (Aoyama Gakuin University)
1990	NX-LISP (NEC)
	Hi-UX Lisp E2 (Hitachi)
1991	Nichimen Graphics (Formally Nihon Symbolics)
	OKI Common Lisp

1.5.7 Lisp in 1989

The year 1989 is a hot year for Common Lisp community as a result of several years efforts from 1984.

Lisp = 7 % of total presentations at 1989 IPSJ annual conf.
82 out of 1168 papers

- 15% : Applications on top of Lisp-based Shell
- 27% : Lisp Applications
- 29% : Lisp based Experimental / Basic Research
- 29% : Lisp Implementations / Environments

This conference covers all the Computer related topics, including hardware, communication, OS, software, application, ... Then, the fact that 82 out of 1168 is quite good evidence

to indicate
"Lisp is used in various fields."
The invited talk was "Common Lisp Object System" by Gregor Kiczales (Xerox Parc). (IPSJ = Information Processing Society Japan)

1.5.8 JPAL'91

JPAL'91 : Japan Practical Applications of Lisp Forum and Exhibition 91

- Date: 1991.11.26-27
- Place: The Forum at New Ohtani, Tokyo, Japan
- Sponsor: Jeida
- Technical Forum:
  - Invited Talk from Europe (by Al Roth), USA (by Mark Son-Bell)
  - Introduction of University Researchs on Lisp
  - Interleaf related talk
  - Graphics application talk
  - Vendor Presentations
  - Jeida CL Committee interim report
  - Jeida CL Guideline interim report
- Exhibition:
  Oki, ILA, NTT-IT, Nippon Steel, IPA, CSK, Dia-Semicon, Nichimen Graphics, NEC, Hitachi, Herlequin, Fujitsu, Line Systems, Intellitech, (Nihon Sun Micro Systems, Apple Japan provide some machines.)

1.6 Where do we go ?

- Nowadays, Lisp, or Common Lisp is used not only by researchers but also by people in real application fields directly and indirectly.
- Much more works on applications, educations, development environments are needed. As a kernel of such works, to tell people 'why' aspect is important.
- Much more on user side by people in every field will be needed and expected.

- We may start to make a serious discussion on "how wee can make Common Lisp slim ? "

The Current Plan

- We started to discuss about how to inherit the output of our activities and seem to come to the end with a conclusion that we should make a publicly accessible database on the internet as one of the works to do.
- We think it should contain the document which describes the language, public source codes, and many useful information.
- We are facing the needs to have a common user interface among various Common Lisp implementations. There is no standard for window toolkits or GUI/UIMS. Furthermore, lots of window systems are there. To cope with this situation, as one of the trials, $\mathcal{Y}\mathcal{Y}$ project has started to provide a common window tool kit on several window systems.

Acknowledgments

This article contains the results of contributions of many persons. The author would like to express sincere thanks to all the anonymous contributors especially from JCLC. Communications with X3J13 members assisted a lot on several stages for decisions. This paper is not the official report from Japanese national organizations, but a private one.

1.7 Discussions

[Questions about Japan Lisp conference trends.]

SWM: Why the drop from 1989 to other years?

Ida: 100 persons in Japan are the true Lisp lovers.

SWM: Why did the other 150 decide not to attend?

Ida: Fashion - or to try out enthusiasms that were there [in Lisp at the time].

[closing remarks]

Ida: I want to add another statement: We have started to make a JEIDA [Japanese Electronics Industry ... Association] guideline for Japanese character handling. Draft is on an ftp-able machine. In it we had started to (do) a common understanding for implementation-defined part of CLtL2 or ANSI Common Lisp. We do not want to extend the specification, but we want to clarify the understanding of the implementation-defined things. Common Lisp itself has basic provisions for character handling, so we do not need any extensions to the specification. Instead we want to clarify the understanding of the specification.

From April ('91) on, we will have a new distributed environment project. With this project we will make a new style of interprocess communications for Lisp and for distributed intelligent data-access and distributed window toolkit. Basically, the client side will be suitable for Lisp and Common Lisp. If the protocol is conformed (to), other languages may be possible. But, we like Lisp, so the first step is to do everything in Lisp.

Amazing facts (around me): While I have been VP of SLUG-JP, last September I was elected as one of the board members of Sun User Group Japan. For some people this is quite funny or unbelievable. But in Japan status, the network is quite important for every persons. It is also true for Lisp-related applications in AI fields.

Chapter 2

Talk by Gregor Kiczales

(In the morning, March 27th, 1991, after Masayuki Ida's talk.)

I was told that I could speak almost extemporaneously, so I'm going to speak almost extemporaneously about the good news about the status of Lisp in the United States. I actually think I can quite honestly say some good news things. The perspective that I have to speak from is sort of my own. I do research and I work for a very large company [Xerox PARC]. So, some of what I have to say may be a little bit premature, perhaps, for some of the different markets, but I think it is applicable anyways.

I'm going to talk about five things quickly: I'm going to talk about market trends that I see and the kinds of systems that people are building. I'm going to talk about what it looks like Common Lisp vendors and Lisp vendors in general have started to learn. I'm going to talk briefly about the larger Lisp scope and ISO standardization. I'm going to talk about Object Oriented Programming and how that effects the Lisp market, and, finally, it's sort of like the academy awards –I'm no fool– I'm going to make a plug for my own research and to say how in the future that might affect Lisp standardization.

So, the first thing is the trends toward different kinds of systems. It seems to me –and I'm going to take some examples from work being done at PARC, and at Xerox, in general– that what we're seeing now is people building different kinds of programs than have been built before. They're much larger, they're glued together from different, pre-existing components; sometimes they're even distributed. Interoperability is a key word. So, for example, we have a document processing system that we're working on at PARC, which take different document formats. Takes any old random document format, and converts it to any old other random document format. So you can sort of take PostScript and fax it and take fax and PostScript it, and do all sorts of things like that. Now, a lot of the software in that wasn't written at PARC. In fact, it's just public domain or commercially available software. But, in the middle of it, there's this little expert system that sits there going OK, you want to make this conversion, let me try to figure out how to do the conversion. That seems to me to sort of be typical of these new kinds of systems, where there's a lot of software that's been glued together. The key point is, it's sort of like when BMW builds cars, right? BMW builds very nice cars (depending on your religious beliefs), but they surely don't build them with the aid of only screwdrivers, because it just takes more different kinds of tools to do the job than that. It seems like, with these larger, more complicated systems, it's going to more and more take different kinds of tools to do it. I think there's actually a growing awareness of that.

I'll give you one other example: Xerox has a new copier product which is out now, it was announced, I guess, actually some time ago. This big, huge thing has a touch sensitive display and the software has a million and a half lines of code and 32 processors in it, including a Lisp interpreter because that's the right thing to write the display, the interactive portion of it in. I think there's growing awareness of this, and I'm going to use as evidence of this the difference between OOPSLA'89 and OOPSLA'90: At the beginning of OOPSLA'89, there was this completely absurd panel in which a prominent member of the computer science community was making the absurd point that you wanted to have single paradigm languages and that everything should be built in a single paradigm language. And at the end of OOPSLA '89 there was another panel in which three people who I know and one person who I know very well, Peter Deutsch (sp?), Danny Bobrow, and Bjorn Stroustrip (sp?) and I were talking about how there were these several Object Oriented languages which were familiar to us and how each was good for different kinds of situations. No matter what you were building, those different kinds of situations were likely to come.

The interesting thing was that by the time OOPSLA'90 came around, at least I heard –maybe it's just what people were saying around me– but I heard a lot fewer paper saying, well, you know, you should build everything in C++, right? There seems to be a growing awareness that you should build in C++ what needs to be built in C++, and build in Smalltalk what needs to be built in Smalltalk, and build in Lisp what needs to be built in Lisp. So, that's sort of a growing, a shift in the kinds of systems that I think people are going to be building.

On a different note, there seems to be a lot of discussion of, about the company of the 90s, and how the company of the 90s needs to be a learning organization and how marketing in the 90s...Regis McKenna and all these other marketing firms are talking about this. How companies need to be more adaptable to their customers, how they need to much more quickly respond to their customer needs. I think that's true. I think –here's a part that's bad news for somebody like me, because I come from a very big company, and very big companies have been very bad at responding to their customers– that, for example, for a company like Xerox, we're going to see in the 90s, people saying to us "we like your document processing system and if you can make it talk to this database before anyone else can, we will buy $10 million worth of it. (But), if you come in second place, we'll buy zero dollars worth of it."

So I think there's going to be an increased premium on adaptability of software, and I think that the business community, the Harvard Business Review style business community, is starting to recognize that. So, it seems to me that as the cost of MIPS goes why down –HP just sort of made it go way down– and the premium on adaptability and tailorability of these systems goes up, once again, that opens up an opportunity for the kinds of belief in software that underlie our affections for Lisp. I don't think our affections for Lisp should be called that, as much as affections for high level programming languages and dislike of programming in assembler.

OK, so that's a little bit about trends, the second thing I want to say is it seems to me that another piece of good news is that the vendors are learning what their customers really want. When I go around, and talk to people, I say, well, what do you really want? They say: It's too big. It's too slow. They say: I want foreign function calling. They say: I want a portable window interface. It seems to me that the vendors are finally learning that that's what the customers really want, and, more importantly, learning how to do it. I think no vendor here would tell you that their release of 3.0 or 2.0 products were really anywhere near as good as their 4.0 products are, or their 5.0 products should be. I think some of these size problems are finally starting to get addressed. At the same time, of course, what small means has gone up, so the problems are getting a little bit easier. But we have people, for example, who would just like to be able to deliver into eight megabytes, and some of the Lisp vendors are actually learning how to do that. And that seems like good news. At the same time X has come along and gotten real big, so what you have to be better than had gotten sort of easier to beat.

But, along with the theme I was talking about before, which is the rest of the commercial world is realizing that you need different kinds of tools for different kinds of jobs, I think that Lisp marketing organizations need to stop trying to have one-[programming]-language customers. The only chance for Lisp to be commercially successful, is, in fact, if commercial organizations are willing to be, are willing to be not monotheistic. I looked in the thesaurus last night and couldn't figure out what the antonym of monotheistic was, so...

(someone): "Polytheistic"

Is it? See, I had a crappy dictionary. We need to appeal to people who want to write in multiple languages, and that means that we, ourselves, in our own rhetoric can't discriminate against people who choose to write two thirds of their system in C++ and one third in Common Lisp. The Lisp community and the Lisp technical organizations have to make life easy for those people, because those are our only customers. No one is going to write anything except for research organizations and, as I got a message from Dick Gabriel the other day that reminded me that the slogan was ... what is it? "Researchers for show and dopes for dough." Researchers are not where you're going to make all of your money. You got to make your money off product organizations and they're going to write in more than one language. So I think we need to –in the same way that the C++ community is willing to accept us– we sort of need to accept them.

We'll deal with that later. OK, now the third part of what I want to say is that Lisp is not equal to Common Lisp. Right? This is my only prepared slide, is this book ["Structure and Interpretation of Computer Programs," Abelson and Sussman]. I've been actually saying this for quite some time, and as more time comes along, more evidence supports my theory.

Every year more and more throughout the world of the best of the people who receive the best educations in software engineering, receive it by way of this book. The book isn't about Lisp, and it isn't even about Scheme. It's about how to do software engineering. But, there's no doubt that by the time you get to the end of the book, and you've learned some neat ways to program, C++ don't look so good anymore.

My claim is that what happens is that people who take this course, they get out there in the world and, when there's only one of them by themselves, you can force them to use C++. But, then when they tend to cluster together in bunches, they have little revolutions, and say screw it, we're going to program in Scheme. Or, you can fire us. And they're good, so they don't get fired. There's at least one large computer company that I know of that I'm not going to name, which is doing a project, which, as near as I can tell is a fairly large project of producing for production quality use something which is vaguely like Scheme plus CLOS(?), but with the concept of compiling for delivery taken very, very seriously. So that comes back to this stuff that the vendors are learning about. The users really want to compile for delivery, and that's sort of a Lisp-like language. It's not Common Lisp, and it isn't even exactly Scheme, either, but it's Lisp in the mainstream computer community. So I think we need to also not be afraid of Scheme, because I think that Scheme is a potential large growth market.

Now, let me talk a little bit about the long term. I've recently been, and Dick was before, the ISO representative for the United States. ISO recently made, I think, a major shift. It was previously trying to make a short term standard, and couldn't come to agreement about a short term standard, and now what's it's trying to do, they made a significant shift to be a long term organization. To look for a long term standard. And, so far the only identified basis for that long term standard is EuLisp, which is a great little language. It's bigger than Scheme and smaller than Common Lisp, two thing which are easy to do. But it's really quite a nice little language. It's learned a lot of lessons from both Scheme and Common Lisp, and we should be open to that in the future as well because, you know, come the year 2000, I sure as hell don't want to be programming in CLOS ten years from now, even though I think it's a fine little language today. And I don't think anyone in this room would honestly want to be doing the same either, even though there'll have to be backwards compatibility and stuff.

So that's the long term of ..., that, to me, seems to be sign of health. EuLisp is a big new language that has just been developed, and there's a lot of backing behind it, so that's strength in Lisp.

Two more things very quickly: Object Oriented Programming is hot. Everybody knows that, right? All you have to do is call it Object Oriented and maybe you can sell a little bit of it. But that's good for Lisp. It's good for Common Lisp because Common Lisp actually has a pretty good Object Oriented language in it now, which is CLOS. And some of us have worked very hard, and some companies' marketing organizations have worked hard -some harder than others- to make sure that the Object Oriented community is willing to call CLOS an Object Oriented language. And that's no mean feat, because there's some people out there who write papers saying No, No, No, No, No: It's not Object Oriented! But the editors of sort of the rag publications have all been convinced to call CLOS an Object Oriented language. So that's good for marketing organizations, I think, and they should take advantage of it.

But a more important reason why Object Oriented Programming being hot is good for Lisp is it means that for the first time, out there in the commercial software community the terms "productivity," "elegance," "reusability," "adaptability," all those terms have come into play. They weren't really as much in play before, but now they are in play. People are tossing those words around, using them. The business community is trying to come to grips with what those words mean. And those are the kinds of words on which, if you're trying go make a debate for writing part of the system in Lisp, you can actually win. We can't win on absolute speed or absolute size, but we can win on productivity, elegance and terms like that. And so I think that, in some sense, the Object Oriented community has done us a favor, which is that it's put into play exactly the vocabulary that the Lisp community needs in the business world. Now, that's a serious marketing and technical challenge, which, the kinds of people who are in this room need to meet.

Finally, I think Lisp, what we're really, what we really liked about Lisp, is not Lisp per se, but rather that its a high level language as opposed to, you know, the assembler that UNIX is written in. And I think that there's actually a lot of interesting and fascinating work being done in high level languages, and there's some breakthroughs coming in high level languages and we won't have to go back and program in C. Common Lisp may not make it through the next ten years, but nice things will make it, and of that I'm quite confident. That's all I have to say.

Discussion

SWM: Can I actually say one thing?

Gregor: No. Forget it.

SWM: I'd like to. I'll be brief. The one thing, the one thing that always strikes me as ironic is the only label that people like to apply better to technologies than "hot" is "once hot," right? So, my fear is that Object Oriented technology is going to one of these once hot technologies unless people are very careful about what they apply that label to.

Gregor: I think that's exactly right. I think the good news is that some of the senior people in the OO community are actually looking very carefully at what caused AI hype and AI winter, and are trying to avoid that mess.

SWM: That's great. My little thing is let, my little motto for myself is to not make Object Oriented programming the AI of the 90s.

Gregor: But, but it comes back to the last thing I was saying about Object Oriented programming. I think if you just try to go out and sell Lisp as a good Object Oriented language, you could fall into that trap. If you take advantage of the fact that the business community is now writing about reusability and productivity, you can't really, because you're not using the OO word, you're using the underlying concepts. But I think that's a real trap that needs to be avoided.

MAS-B: Could you say a little bit more about EuLisp.

Gregor: What would I say about it?

MAS-B: I mean, you said...well, I don't know very much about it; I don't know who else does in here. How does it differ from Common Lisp? How is it interesting?

Gregor: EuLisp is a–I forget what side the number goes on–it's a "Lisp 1", right?

RPG: "Lisp 1".

Gregor: I can never remember whether it was "1-Lisp" or a "Lisp 1." It's a "1-Lisp" if you're Brian Smith. EuLisp is a Lisp 1, which means that function namespace and variable binding namespace are the same namespace. It's a, you'll correct me if I'm wrong, it's a stratified language which has three layers. (It) has a very small kernel on which the upper two layers can be built. The topmost layer is not as big as Common Lisp, but is much bigger than Scheme. The bottom layer is a little bit smaller than Scheme probably.

RPG: Except that it has generic functions ...?

Gregor: Single argument, specializing.

RPG: Right, and the bottom layer has only required arguments, I think, right? And then optional arguments come in at the second layer, multimethods come in at the second layers, and multiple inheritance comes in at the second layer.

(???): I mean, we don't want you to give us a complete description of the language, but if we want to read about it, where do we look?

Gregor: Dick (Gabriel) has a copy of the recent spec.

RPG: It's being written by Julian Paget, that is to say the specification is, at the University of Bath, which is "jap@AC.Maths.AC.Bath.UK" (or something like that).

Gregor: What's the address of the EuLisp list?

RPG: "eulisp@inria.fr".

...

RPG: There's a significant thing that Gregor left out about EuLisp which I think does make a significant difference to a lot of folks, which is it has a very good (module system).

Gregor: That's right.

Hanoch: What is the position of American representative on this committee as to introducing another standard?

Gregor: The position of the U.S. delegate [Gregor] to ISO is dictated by the position X3J13, which is the ... we all know what X3J13 is. And the current position of the U.S. delegate is basically to keep the ISO committee alive, and to shift it to a long-term focus by doing so. For a long time we were trying to get Common Lisp to be an ISO standard and we now realize that that's just not practical. And what we want have not happen is for there to be in the short term an ISO standard, which isn't Common Lisp. So the basic strategy is to shift ISO to long-term focus, so that it will come out in the late 90s with really nifty language.

Hanoch: I see.

Veitch: I'd be curious. what would you think of that, the possibility you, I mean, there's at least a couple of people here, in this room could be strongly involved in the language design of Common Lisp of revamping Common Lisp to do the same sort of thing that EuLisp has done. Is that completely impossible?

RPG: We would be better advised just to start with EuLisp than to do that.

HES: And then look at a compatibility package (or other upgrade path).

RPG: Because you've got the Lisp 1, Lisp 2 thing worry about.

Veitch: So we're going to be talking about at least three competiting Lisp dialects in the medium term future: There'll be EuLisp; there'll be Scheme; there'll be Common Lisp.

RPG: Yeah, within the European arena I would expect two of those to do have preeminence fairly soon, once there's a real implementation of EuLisp. Right now the only implementations I know of are interpreter only. Whatever, because of this great that MIPS are cheap and plentiful, these interpreter only implementations are being used for real work. I sat and observed it happening three months ago.

Gregor: It [EuLisp] has macros in layer 2, I think.

RPG: Right.

Gregor: I don't want to let you have characterized what I said implicitly as that MIPS are cheap and plentiful. Because I didn't say something as (mindless) as the old song about, oh, well, don't worry about it, computers are cheap enough.

(inaudible discussion)

Gregor: I have another question of this area. I mean, how, is it likely at all to actually position Common Lisp as the third layer of, you know, basically accepting maybe two layers [of Eulisp] then Common Lisp for the third layer of this, or..? Why is that?

Gregor: They've gone relatively far in that direction, and, in fact I think, I would say, correct me if I'm wrong, they've accepted the Lisp 1, Lisp 2 issue of they're relatively close to Steele [CLtL] for one, in their third layer. But I don't think they're going to move much more to that. They sort of –quite frankly, did the same thing, which I think the U.S. had wrested the international– which I think the world Lisp community did: They got tired of waiting for X3J13. And, they sort of decided what they're going to do, and they're not waiting around for X3J13 to decide what it's going to do any more.

RPG: At one point, that community asked us, X3J13, whether we would like to cooperate in making the Common Lisp be the third layer or something very close to it. But we basically didn't do that.

Gregor: It's hard.

SWM: How big a mistake do you think that was?

RPG: Speaking as an individual, I don't think it was a big mistake, and I think EuLisp is better for not having done that.

SWM: Oh, I asked a different question than you answered then.

RPG: Well, I answered the question I wanted to answer.

SWM: Let me try one more question that's so, you know, like it or not, in fact X3J13 has, in fact, done some useful work. Now, there's some useful things that have been learned in that process. And, the mistake I'm referring to is one of knowledge transfer, right. Just because someone who is actively participating in X3J13 decides to participate in the EuLisp effort, doesn't mean that that person is saying can we please turn EuLisp into Common Lisp. What that really means is, I've learned some things, can I contribute to this process? What is the possibility of doing that? You can answer too, Gregor, since I see you squirming here.

Gregor: I think Ronnie Lott did the right thing by going to the Raiders. :-)

RPG: Well, they did learn, I mean, people like Chailloux, Jerome Chailloux, who's the, sort of the leader of the French, and to some extent, a high influence person in Europe, attended may X3J13 meetings. Folks who designed the EuLisp Object Oriented system knew fairly well what CLOS was and why it was the way it is. So, they learned that way. Speaking as a representative of the United States –if I've ever been called that– I think we did make a mistake in not joining up with them earlier.

Gregor: I think it was a mistake that we couldn't avoid making. Because we didn't have the wherewithal to join up with them.

RPG: I don't think we were smart enough either..

Gregor: That's the wherewithal.

HES: As an obnoxious person sitting on the sidelines ...

–end of tape–

... the ANSI standard. So, now, you know, you're in a position which sounds very reasonable to me arguing for this other thing to become the successor standard, perhaps.

If that's what you're saying. The only problem with that is it's the successor to something which is not yet standard. I don't have any sense personally, of how fast you guys think the ANSI standard's going to converge. You know, the answer should have been about a year and a half ago.

Gregor: It's not going to converge that fast.

RPG: Also you're right at the U.S. position was exactly what you said, it's for the first two or three years of the ISO process, the U.S. found the sneakiest, most bastardly person they could find to argue the case. You have, it's myself. (laughter) And I did a good job of it.

MAS-B: That's a great transition comment to (the talk you're going to give). The schedule says that Jo Marks was going to give an overview of things happening in Europe. As I indicated yesterday, he wasn't able to make it. So, I think Andrew Smith from Harlequin has agreed, during the (session) in the second half of the morning where everyone kind of explains what their company's doing, that he will address some of these issues, at least to the extent he's able.

Chapter 3

Talk by Richard Gabriel

(In the morning, March 27th, 1991, after Gregor Kiczales's talk.)

Professor Ida asked me to make a couple of remarks, and they are even more extemporaneous that Gregor's, but I took notes during his talk to try to say something. I think what I'm going to try to do is address, to some extent, what I think is the purpose of this meeting, which is cooperation. And for those of you who went to the X3J13 meeting last week, you heard me say some of these things there, and I'm going to say them again in a different way.

First thing I want to say, is that over the last year I've had a number of interesting experiences that bear on the question of how we as a Lisp community is going to survive. It all started, coincidentally, one year ago, almost one year ago today at the EUROPAL conference where I made a speech called "Lisp: Good News; Bad News; How to Win Big." And some of you may have seen excerpts from that over the network recently, which I didn't send out, called, "Worse is Better."

And, also recently some folks in the U.S. community have taken me to task and had seen this paper and said, well, why did you make this speech saying Lisp is dead. I didn't say Lisp is dead and what I tried to do is something that I think Larry Rowe tried to do yesterday, and I wish he was here today, which is to slap us on the face and say why don't we try waking up and looking at what's going on, and be honest about where we are and what we're doing, which I don't think we have ever done as a community before.

Now, there have been two EUROPAL conferences in the last year. The one which was one year ago in Cambridge, England, and then another one which was on parallel Lisp, which occurred in November of this last year, just outside of London. I have two experiences I think were worth commenting on. One was, I ran a little bit of an interaction session at the first EUROPAL, where there was group of users, mostly from Europe, who were discussion their experiences with Lisp in sort of a commercial setting or at least a setting where they have to deliver products within their organization. The uniform comment that I heard was gee, we need to be able to have integration. We need to be able to have means of delivery. In our organizations we could hold out for about another year, and then we're going to be shut down. So, please, Lisp companies can you do something about this for us. That's what I heard one year ago today.

In November, which was about four months ago, there was a similar discussion, although in a much smaller forum, where people were talking about the future of Lisp, and the comments I heard was, well, Mr. Lisp Companies, what I need now is a super cheap Common Lisp because my boss has told me and my group has told me that I can't use Common Lisp at all, I have to use C or C++ or Ada. And I made a deal with the guy where I told him that I would do the project in C++ all right, but at the same time I would do the project at home on my own time on Common Lisp. Prove to him I could do it faster and better in Lisp, and that's how I plan to keep Lisp alive at my company. So what we need now is a $200 Lisp so that I can afford to buy it myself. And this all in the space of about nine months is changed.

Now, the question is: is Lisp in a winter? I'm not sure whether we're in a winter, but we're wearing a lot of warm coats over where I work these days, and I think it's time that we need to circle the wagons a lot more than we have. I think that circling the wagons means a lot of different things, and I'm going to tell you some of them. Then I'm going to end with a couple of interesting stories which I think goes towards what Gregor was saying what the companies have to do.

I think the first thing that we have to do is this: There are some people who believe, I'm not one of them, but there are some people who believe that it's imperative that there become an officially recognized standard for Common Lisp. Since I've lived long enough, and been wrong often enough, I will bow to this attitude that we must do this, and argue for it myself. And I belive that it is important that we have a standard for Common Lisp very soon. The way to do that is to stop the people in X3J13 from doing what they're doing. It was two years ago, on the pleasant islands of Hawaii, the meeting that was supposed to be the last meeting to make any technical changes to the language. We agreed beforehand. We agreed during the meeting. I stood up, made lots of speeches, bowed, ... Gregor made a lot of speeches. We went to the beach a lot of times. We ate a lot of good food and were happy and got suntanned and Hawaiian shirts. We all agreed that was the end. And now, two years later, we just finished a meeting where there were approximately 65 technical issues that were decided. Some of them small, some of them big, some of them incompatibly changing the language from CLTL1 and CLTL2 ... only (Bob) Kerns, I think was there for the, my second performance as a real bastard where I pretty much made it clear that I, personally, wouldn't tolerate any more changes to language, but I'm not sure that we're going to see a moratorium on more changes.

So, the first thing that we have to do, which is mostly directed at the representatives to X3J13 is to say No. No more changes; and get it done. I think it's important that we also keep WG16, ISO committee going because of the possibility of bad press from not doing that. The bad press would be: Oh, it's those Lisp guys, aren't they funny, they just can't agree on anything. They've got the Scheme people who won't talk to the Common Lisp people who won't talk to the EuLisp people. They started an ISO committee at great expense to nations around the world, and they yapped and yapped and they travelled all over the place and they didn't do anything. And that just shows that they can't get their act together so why should we even take them seriously.

I wanted to make a couple of orthogonal remarks to what I've been saying so as to set up the final stories which I have, which are directed towards this issue of cooperation.

Some of you may have noticed that the company that I happen to work for has branched out into other areas, including the dreaded enemies, C and C++. And one of the reasons that we did that is an interesting one and has to do with, has to do with our love of Lisp, which is: we don't know for a fact that Lisp isn't going to go kablam in the next year or two. However, we're hoping that when it goes kablam there's a trampoline there that will bounce it back up somehow. I would still like to have my company around when that happens, so I wanted to develop some other businesses so that some other source of money can keep us alive during this period of time. I sort of feel that if we don't do that, then the only companies that will be doing Lisp will be these big companies. Gregor won't name one, but I'll name one: Apple – which is doing Lisp, probably the amount of money they spend is so small compared to what they do that they don't really care. I don't know about you guys, but, since we're the companies who love Lisp enough to start a company, I feel I would rather have a company that's successful in Lisp than a big company like Apple that does lots of screwy things I don't like anyway.

Now, the fact that I'm in these two camps, Lisp and C/C++, means that I see a lot of different experiences from the rest of you guys. I get to see everything that happens in Lisp. Now I get to see a lot of what happens in C and C++. There are some interesting things. Professor Ida put up a slide that said that, among the most hated parts of Lisp was it size. "Huge" and "gigantic" were the words that he wrote. Now, one interesting thing is, C++ is taking off and there are actually very good debuggers for C++, and when you write C++ code, and you try to debug it, the size of the image is as big or bigger than Common Lisp. We have a product which is about one and a half megabytes of C++ code. When we compile it so we can debug it, it's about eight or nine megabytes. So the size issue, I've said this before, a lot of times people will use excuses for why they won't use Lisp or something like it, and I think one of the excuses is the size.

I think there are some other reasons that people don't use Lisp. I don't know that those reasons are, but I don't think it's the size. I don't think it's the speed. I don't think it's a lot of things. If it's anything, it's probably that there happened to be, as Gregor hinted, more C programmers available to hire cheaper than Lisp programmers. Because even though, well, there aren't as many of us, and we're more expensive.

So now, I want to get the point I want to make which is we need not only cooperation among different organizations like standards organizations and universities and companies ... we need more cooperation among the companies than we have ever had in the past.

(Everything I've said up until now you can take as a real statement from my company. Now, I'm going to make statements as myself so I really wish you that you would attribute all of the bad things that I say to me and not to my company.)

I think that part of the reason for AI Winter and certainly the reasons for Lisp winter, if that's coming, has to do with the business policies of the companies that we all work for. I know that when I started my company the success of that company was predicated on being able to help the AI community – the AI business community – be successful. The interesting thing about that is you would think that the AI companies would feel the same way and there'd be a large degree of cooperations where they would share their plans with us, we would share their plans with them, we would make common plans through them, adapt the implementations to the needs of the users and adapt the user's code to the realities of the implementations ... it never happened. It never even began to happen. We never had any conversations with IntelliCorp, or Inference along those lines. It never happened. So that's one part of the attitude that's a big problem.

A worse attitude exists between the Lisp companies. So now, I'm going to tell my interesting stories. On the one hand, I'm going to tell you about what it is like to be in a non-cooperative Lisp business environment. On the other hand, I'm going to tell you what it's like to be in a cooperative C/C++ environment.

There are two companies. I'll call them J and P to protect their identities. Company J will not sell ... will not sell its Lisp to company P. Both are Lisp companies. Moreover, Company J, if it receives an order from a person, an ordinary person, and Company J thinks that person is a friend of Company P, they will push in that person to find out if that person is going to make a slip, a copy of the Lisp of Company P. So that's the story on one side. Whereas some Lisp companies will take an order wherever they can get it, Company J won't even cooperate to that degree with Company P.

Next story. As you know or maybe don't know, my company bought a compiler company a few months ago. A C/C++ and Fortan compiler company. Before that we started work on a project, which is the main line of products that we're doing, based on programming environments for non-Lisp languages. As part of the technical requirements of that effort, we need to modify slightly compilers – C compilers, C++ compilers – and the strategy is to be able to work with a variety of compilers. So we went and we talked to a company (whose) name is Oregon software – it's a C/C++ compiler company up near Portland – and we said, "we're doing this nifty programming environment for C++, we would like to help you win big, please modify your compiler the following way." They said, "well, we're an awfully small company. We don't have resources. Yes, we want to participate in this very much, but could you do it for us?" And we said, "sure." So they gave us a free source code license for their compiler and we moved it down to Menlo Park and it sits there and we stare at it, we modify it all the time.

In December, we bought their competitor. We bought their competitor, and before their competitor moved into our building and hooked themselves up to our ethernet, we decided to call up Oregon Software and explain the situation so that they could flip or whatever they're goint to do. So we called them up and we said, "we bought your competitor, we're going to bring out a C/C++ compiler, we're going to try to beat the shit out of you anyway we can, but we still want to hook up our environment to your compiler, do you want us to delete all the sources?" So they sat and thought and they said, "well, gee it's too bad you bought that company, because we were hoping you'd buy us ... besides, we still want to participate in this, so how about this: Why don't you just use common sense and don't steal any of our ideas. Please continue to modify our compiler to hook up ..."

By the way, we exchanged source code regularly with Oregon Software so that they are updated and we're updated so that we don't have a year-old copy of their software. And they said, "just be careful and try not to use our stuff and let's see, our CEO is doing a business trip in Europe and then he's going to go on vacation for a while, so we'll get back to you in April or May. We'll work out the details." Now that's a pretty different situation that we see with the C/C++ community on one side and the Lisp community on the other side.

There are three or four other compiler companies that we work with on almost as open a basis. There's MetaWare, there's AT&T, there's a few others. OK? Very different attitudes.

Bill Scherlis [DARPA, Program Manager, Information Science & Technology Office] once came to X3J13 and said, "gee, it's interesting that the Lisp communities are one of the few communities that doesn't have industrial organizations or cooperations. Why don't you try doing that?" And we smiled at him and said, "Yeah, that's right Bill ... goodbye." I think it's imperative that we circle the wagons and work together.

There are things that we need to do, as Gregor said, to learn to adapt to become more aware of what our customers need and there's no need for us to search for enemies within when we have enough enemies without. So I think it's important that we get together as companies and open up the doors, define what the things that we need to do to survive, and do them; or, we won't survive. We will not survive. OK, it might be interesting to define a language, a common syntax for blue rectangles, but, as Larry Rowe said, that doesn't make any difference. I don't think it makes any difference. if we don't learn to cooperate, we're not going to get together again. Or, if we do, we'll be at the Scheme meetings or the EuLisp meetings. We will have only the researchers for show and no dopes for dough. So, I guess I'm pleading with you. Let's cooperate more than we have in the past because we have more things in common than we have not in common.

(applause)

Chapter 4

Discussions

4.1 Discussion 1

(In the morning, March 27th, 1991, after the three speeches included in this report.)

MAS-B: In this session what we want to do is go around and have brief statements from all of the participants here. Rather than just going around the room or going alphabetically or something, I'll try to select the order based on my understanding of who's doing what and try to get some rhythm or combination going of geographic mix, and vendor and user sides.

So I think the first very brief statement I'd like to ask for is Mr. Kosaka's, from CSK in Japan. Very briefly, just, I think in this section there's no need for any formal presentations. If you have a statement you'd like to make, that's great. Some of the vendors have been asked to bring product literature, talk specifically about products. This is really the opportunity to get what you may have brought to the workshop out for the larger group. So I think Mr. Kosaka first and then perhaps the Apple folks. Because I think you all (Apple) have some answers, actually, to some of the issues that were just raised before the break. So, Mr. Kosaka: a very brief statement.

Kosaka: I would like to talk about two things: One thing is mouse processing. The other is interrupts.

[Before that, though, on the subject of Lisp efficiency:] If users use just Lisp, Lisp is very fast. Many users do not care about size and speed: it's already enough. But if we consider (the) total system: if users using Lisp and other processes are running, or maybe the Lisp displays something, then this causes problem. Total system speed is not only one process's speed. Total system performance is quite different.

Mouse processing: Common Lisp has only one terminal I/O. Also different processes use (a single) terminal I/O. This is very confusing. I have experience building YY using multi-tasking. (If I entered debugger, it was impossible to determine where I was in the program execution. Because different processes got the input events, I couldn't tell what had been running when the error occured.)

Ida: I want to follow up. Mr. Kosaka is very technology oriented, as you understand. [To Kosaka] Please tell us your background and affiliations ...

Kosaka: Yes. First, at CSK I was the author of Nihongo Common Windows. This is a nihongo [japanese language input front-end] facility with Common Windows. This was also a client server model. I designed that. Also, at Aoyama Gakuin University, I designed the YY client, and a little bit (of the YY) protocol. I also designed the intermediate synchronization facility, to speed up the total system speed.

MAS-B: Could you explain a little bit what CSK does in Japan with Lisp? CSK distributes LISP?

Kosaka: Yes. We make Lucid Common Lisp on News and other japanese workstations ...

MAS-B: Sony News [Unix workstations]?

Kosaka: Yes, Sony news. Also other confidential workstations ...

(laughter)

MAS-B: OK. I see. Is there any direct LISP development other than the Nihongo work done by CSK, or is it all distribution?

Kosaka: Yes. CSK make a Nihongo Common Lisp –Japanese Common Lisp– in IBM 6000...

MAS-B: The RS/6000?

Kosaka: Yes.

MAS-B: On the workstation?

Kosaka: Yes. With Japanese features. Also Nihongo Common Windows. IBM's one. Sony's one. The other confidential ones.

(laughter)

MAS-B: That's fine. OK. Thanks. Harvey from Apple.

Alcabes: I'm Harvey Alcabes from Apple Computer; with me is James Joaquin. To give just a little bit of history of Apple's involvement in Lisp, since the early days of the MacIntosh, there have been several third party companies that had Lisp environments available for the MacIntosh, and Apple encourages those products because our attitude on development tools is to encourage any company that has products that help people program for our hardware. One of those products, one of those companies was Coral Software, which is a small company in Cambridge, Massachusetts. And with the help of Franz they had developed a MacIntosh Allegro Common Lisp product. In late 1989, Apple acquired Coral and started selling MacIntosh Allegro Common Lisp– at the time it was version 1.2 and then version 1.3.2.

We also used, starting from Coral, we grew a research lab in Cambridge, which includes the product development for Common Lisp product and includes some future research. We've been hiring additional Lisp researchers into that lab more recently. One of the most notable, a few months back, was Dave Moon, previously at Symbolics.

Just recently, we've announced MacIntosh Common Lisp 2.0. Because we've diverged somewhat from what we've been doing with Franz, and now we're working on this on our own, we've dropped the Allegro from the name. It's just "MacIntosh Common Lisp." The version 2.0 supports CLtL2, and supports CLOS. It requires two and half megabytes of memory to run, although we do recommend four megabytes. And, on the MacIntosh that's considered a lot of memory. Although I understand that in the Lisp community, that actually can be considered fairly small. But the full Common Lisp and CLOS can run small programs at two and half megabytes and moderate sized programs in four meg. There are certainly some people who have very large programs and one of the limitations that we've had in the past was that you couldn't use more than eight megabytes. With Version 2.0 running on top of System 7 [ed: with its virtual memory OS feature] – which Apple is releasing in a couple of months – you'll be able to use virtually unlimited memory size if you write very large programs.

What we generally do with our development tools is when we have a very solid, usable beta version of our tools, we will sell the beta and then when we have a final, obviously, we will sell that. So, starting in a week or two, we will be selling MacIntosh Common Lisp 2.0 Beta Release and customers can buy that for $495. They'll will get free upgrades of the final on that as it comes out later this year. In fact, I brought along some (copies of) the two-page press release that we put out last month and some order/information sheets, if you'll pass these around? Just so that you have an idea what sort of details are included in the product, or if you want to pass this information on to other customers or other people who'd be interested in finding out about the product.

As I mentioned, the reason that we do programming tools and languages is to encourage people to develop more software for our own hardware and also to use it ourselves. Certainly, we're using MacIntosh Common Lisp within Apple for research and some of our software developers are using it. Where many of our development tools are used mostly by our commercial software developers – people like Aldus, Microsoft, Ashton-Tate, Claris, and many smaller developers who are selling commercial applications – Lisp tends to be used much more by in-house developers. That's partly because of the size and performance issues discussed here earlier. While the size and performance of MacIntosh Common Lisp are pretty good, it's still not, it doesn't run quite as fast as a C or C++ or Pascal, and it doesn't create programs quite as small.

There is a little bit of interest in Lisp from our commercial development market community, but mostly the interest is in the in-house development market and also, of course, in the university market. Our in-house developers, some of them are using MacIntosh Common Lisp, there's quite a few other tools that people are using. One tool that's used quite heavily by in-house developers in the MacIntosh market is HyperCard, because it is very accessible, very easy to put together user interface, put together a simple application. But we find that HyperCard is good for a certain scale of projects. But as people are getting into very big projects, they start with HyperCard and they found that really that doesn't necessarily meet their needs for a large project. I'd really like to be moving those type of customers to Lisp. I think that a product like MacIntosh Common Lisp can meet the needs of a lot of those kind of customers. And I'd really like to see essentially the audience for Lisp broadening. Instead of trying to make the best possible product to meet the needs of the people who are currently using Lisp – I certainly want to make a good product that meets those needs – but I'd also like to be developing our product so that it broadens the market and reaches out to more people who might not have considered using Lisp in the past. Because I do think that a lot of the power of Lisp could really benefit those kind of customers.

Gregor had mentioned that earlier, that vendors are starting to learn what customers want and, in the past, we've, we're learning that in-house developers are particularly interested in cross platform development and portability. So that they can write a program that will run on a MacIntosh, on a DOS-based system running Windows, on an X-Windows or other UNIX platforms and so on. While, in the past, Apple is pretty, was somewhat resistant to having development tools that help people write programs that can be ported through MacIntosh and other platforms, we are recognizing that our customers demand it and we're open to that.

We don't have a whole plan in place for how we're going to deliver all of that to our customers and, in fact, I don't think it should be only Apple delivering that to our customers. We want to work with other people so that, for instance, we can deliver a standard Common Lisp on MacIntosh and other companies can be delivering standard Common Lisp systems on other platforms, and especially using tools such as CLIM from ILA or Action, which is a product from Expertelligence ... that customers can do their user interface design work, write their algorithms and run their code on multiple platforms. Since our customers want to do that we're certainly open to, you know, exploring those directions.

Gregor also mentioned the idea of polytheism: not having a single religion and not saying that one language is the solution to everything. We certainly believe that. My job – part of my job – is product and marketing manager for Lisp, but my title, my entire job is actually product and marketing manager for Object Oriented languages. So this includes Lisp, C++, ObjectPascal. Our customers are not interested in– they're not coming back and saying: "we're interested in using a particular language." They are coming back and saying: "We're interested in accomplishing something, how can we do this? What languages will help us do this?" So, I'm not trying to push any one religion on them. I'm trying to help them meet their needs. I think Lisp can offer a lot of things that will help meet their needs, but I think that as Gregor said, there are other languages that can also do things that customers need. I think it's important that we evolve Lisp and Lisp products so that they can work with other languages, so the customers are not only locked into a particular way of doing things.

We do see that Object Oriented programming languages are very important. We see that Lisp and Lisp-like languages are important and interesting to the future, so we're going to be continuing to be working on all of those fronts. Right now, we're trying to determine what our future directions will be beyond the release of Lisp 2.0. We're certainly committed to continuing to work on Lisp and we're interested in following standards of the community and also in helping advance things in the Lisp community and growth the community itself. So we're open to input from other people who are working with Lisp as to what our future direction should be. But, our goal in all of this is to support our customers who we hope will be creating more software for our hardware and thus, we'll be selling more hardware. And we don't see that that means that we have to compete with other Lisp vendors. We'll certainly cooperate with vendors who have products that will help our customers to meet their needs. I hope that gives you a little bit of a sense of Apple's attitude about Lisp. I don't know if Mark wants me to be dealing with any questions now ... I could do that or I could talk to you afterward when we have a break.

MAS-B: I have a couple of things I wanted to follow on from what you had said. How does Apple position its Lisp product vis-a-vis these others? So, you know, you said you're responsible for managing C++ and ObjectPascal and so on. And you say that Apple's statement back to the users is that there is some difference between Lisp or Lisp-like languages (and the others). What is that statement?

Alcabes: Maybe I didn't quite understand what you were quoting me as saying in terms of ...

MAS-B: OK, what I think I heard you say was: "customers come to Apple saying 'what's the right tool for the problem?' and you have this tool kit that you can reach into and pull out the right tool, and sometimes you pull out C++ and sometimes you pull out Common Lisp." How do you differentiate?

Joaquin: Sometimes you pull out both.

Alcabes: Yeah.

Joaquin: What Harvey was saying, well there is a multi-language solution that we provide.

MAS-B: But I'm (asking) what are the chief differences that you see and what classes of problems do you see that they solve?

Alcabes: At the moment, quite honestly, I don't have a good answer of how we position it because we're still trying to figure that out and I'm very interested in feedback from that. But, at the moment, when a customer talks to me, I say that because the programs you deliver with Lisp ... and, just as I mentioned that our Lisp system requires two and a half megabytes to run and generally four for any reasonable sized product, we let people deliver standalone applications. They pay us one small fee and they can deliver unlimited quantity. But those standalone applications are also going to be, I mean we just strip out a little bit, basically the compiler and the editor, so they are also going to be about those same sizes.

And so I tell people that they make the right program in Lisp very quickly, that they get it up and running, you know, prototype, makes changes to user interface very quickly, but that they are going to, if they deliver that application, it's going to be relatively large by MacIntosh standards.

What that tends to do is that people sometimes use it for a user interface prototyping; sometimes use it for projects that are internal to a company where they are going to be ... whether either for the company itself or for a consultant who is working for a company that's going to be developing software that going to be used within the company ... but, generally not for a product that's going to be packaged up and sold to lots of people.

I think we sort of point out the features of the various products, and we don't try to say this one is better or this one is better for that. In the press release that I passed out, I wrote in a few common uses of Lisp systems: "Especially well suited for rapid prototyping, custom development for business and education, scientific and engineering applications and artificial intelligence research." Those are some of the things that I'm saying that people use Lisp for, but I'm not saying that it's exclusively for that or that Lisp is the only language for doing those sort of things.

So I guess the answer is, I'm still– we're still trying to figure out how we should be positioning relative to other products. And, we're open to working on it.

Joaquin: I think the stage we're in right now is educating our developers that MacIntosh Common Lisp is here and is an option. So we're still getting the message out that we have a Lisp platform and it has a place in the development community.

Alcabes: I think there's also some misconceptions throughout the world that we're trying to counteract. I mean people hear Lisp and they assume that it's a slow, interpretative language because some of the first Lisps were that way. But, so one of the things that's important is that were getting out the message that it is a fast compiled language – not as fast as something like C++, but certainly performance is quite good and adequate for a lot of applications.

Other questions or issues right now? I'm not sure how long we'll be staying around here today, but I can certainly stay around for a few minutes after, until the next break to–

MAS-B: Given your time constraint, I want to ask Dick (Gabriel): Would their product have solved the problem you encountered at Europal back in November? When people said they needed a cheap, home Lisp?

RPG: Probably.

MAS-B: 'Cause, in my view, that's about as small and cheap as we're going to get.

RPG: Well, you can argue with that, but that's probably the smallest and cheapest you can expect to get. Also, well the particular folks who were saying this, though, tended to come from shops that use Suns.

MAS-B: I see.

RPG: So they would have to deliver it on the Suns.

MAS-B: I see.

RPG: So develop on MacIntosh and deliver on Sun: There's a concept!

(laughter)

Alcabes: Realistically, it's doable. I mean, my understanding, if you're writing, using the Common Lisp, you know, just using the Common Lisp and CLOS and not using any features, you know, MacIntosh specific features, should be able to take that code and port it to any other systems with Common Lisp and CLOS. So, in theory it should work.

RPG: One of the folks who said this had to deliver on Suns using Motif, so they would have ... it was a user-interface intensive application.

Alcabes: MacIntosh Common Lisp and an eventually CLIM for MacIntosh Common Lisp.

MAS-B: Yeah, that's the theory.

MAS-B: OK. Let's continue around the room. I want to have the Harlequin folks say something now.

ASmith: (First) some background. The company is just over five years old. It was five years old last week. It has 60 people at the moment and has doubled in size within the last year and still growing.

— end of tape - about a 15 minute gap in the record —

-discussion is about delivery tools for the stock hardware Lisps-

Veitch: ... every layer uses some little chunk of the layer underneath it. So what will happen is that the actual application way out here will only actually touch very tiny percentages of the layers in between. So in terms of the code component, you'll probably get increasing benefit with the size of the application, but you'll still have to deal with the data.

RWK: Are you talking about working set now?

Veitch: It affects both swap space and working set size.

RWK: But not image file size.

Veitch: Presto is pretty simple. Code is put in a big library file that lives on some disk somewhere. In network land, it lives on the network maybe. When Presto runs, it pulls in code vectors as it needs them, and then runs them.

SWM: It's doing an execution simulation instead of executing them ...

Veitch: That's right: It's doing an execution simulation, and that's why you get this sort of free paging performance enhancement. Because if A calls B then probably A calls B a lot. There's a heuristic there. So you train the application. When you run the application, then the odds are that you won't need any more.

What you do is freeze what's in the application [image file] and that becomes the disk file, for which you need swap space. But the library file is still out there if you need.

SWM: What does it do about all of the dirty data?

Veitch: What do you mean? You mean in the build procedure?

SWM: Yeah.

Veitch: Right now it doesn't do a very good job of that.

???: So your savings in disk space assume that you already do have a copy of Franz at the site?

Veitch: It does indeed. That's correct. In Unix land today, you need to configure swap space, which is permanent disk that you give away for ever. On every machine. Every CPU needs swap. Well... that's not quite true. There's a thing called TEX for read-only memory where they'll page directly off the file.

But that's the basic thing, you can configure smaller swap spaces by doing something like this.

RWK: The reason for my questions has to do not with what I think is important, but rather with the frequently heard "To deliver an application [in Lisp] is too large" presupposes a particular mindset which [Allegro Presto] doesn't really address, but which I don't think is the right one to address anyway ...

Veitch: I think the Lisp is too large. I agree that the Lisp is too big. Larry Rowe said yesterday that the equivalent of Picasso in a C-based toolkit .. I forget which one? Sapphire ... is 5-6 MB, in SmallTalk is 8-9 MB, and in Allegro is 13 MB. Well, that's still a factor of two. Memory may be getting cheaper, but it still ain't free, that's for sure. So, a factor of two is a lot, I think. I think it's too big.

RPG: This is a funny story. What Kerns is getting at is that's there's reality and perception. I have an anecdote that shows how tough a row we have to hoe. Now Jim was explaining, and we all understand that working set is the important thing, and they [Franz] have a reorganization technique, as does Lucid, and to us, that's the important thing to make sure that programs run faster.

A year ago, when we were out hyping this thing, we sent one of our real computer scientists out on the road to generate some enthusiasm, and he went to reasearch labs and various places like MCC to pitch the thing, Andersen Consulting and places likes that. He found that about half the time, he was talking to the developers using Lisp, they did not understand the concept of working set, and therefore this entire pitch which explained this benefit was completely not understood. And once they understood it, they couldn't believe that it worked, or that working set had any effect on performance. So ...

SWM: It's the only thing that has any effect on performance.

Veitch: I have an interesting story that's related: We have a whole delivery technology

– it does things like putting things into read-only memory and blah blah blah. It turns out that none of the VARs use it. (This is pre-Presto stuff.) So you ask them why, well, they're too busy to ever actually understand it. I never thought it was that complicated.

The SmallTalk guys... Digitalk were showing a demo that I went to quite recently and they put a little panel up that says "Build an Application", and you go click; poof, then it goes away and does something, and out of this comes a really small executable file. That's the type of delivery technology that a Lisp vendor has to supply ...

MAS-B: If I can say, this is exactly the kind of discussion that we had hoped to get into for this workshop. But we are pressed for time. So I'd like to have one final statement, Howie Shrobe's from MIT and Symbolics, before we break for lunch.

HES: I should say that Scott McKay is here, who is full-time at Symbolics. I guess I was asked to come because I'm eloquent or something ..

MAS-B: An eminent person.

HES: An eminent person, right... Until quite recently, I was full-time VP of Technology at Symbolics. Now I'm half-time VP of Technology at Symbolics, and I spend the other half of my time reincarnating a previous Howie which was an AI researcher at MIT, and I'm going back to doing the half-time.

Most of you know Symbolics, we were one of the first Lisp companies. We are the remaining Lisp hardware vendor - Lisp machine hardware vendor. That is a business that we continue to maintain, and will continue to maintain for some time. The company has essentially four kinds of products at this point: One is a very high end graphics system for animation and entertainment kinds of graphics development, which is in its market niche regarded as a leader.

Many of you saw an announcement from the company recently which was misinterpreted as saying we have changed the company into that as the only business. I know because all the folks I'm seeing over at the AAAI meeting are saying "Gee I heard you flushed Lisp and are now a graphics company." The short story is, no, that's not what we did. It was an announcement that we're trying to put some emphasis on making the graphics business successful, as a Lisp application, which indeed it is. It currently is a product that runs on our hardware. There's a long-term view about porting that and getting a stock-hardware graphics application, perhaps with some special graphics hardware that we develop for it.

The second business is the Lisp machine and Genera software business, which is still the largest component of company revenue (I believe). And that's a modestly healthy business (surprisingly enough, since it's been declared dead about twenty times in my tenure there). There are some new hardware products that are coming out in the short-term future that are pretty spiffy. They're not 75 mips for $20K, but they're nice Lisp products.

The third thing we have is a Lisp system for 386 class PCs running DOS or Windows, called CLOE.

Finally, we have a set of layered software products that are sold on top of Genera, which include the Joshua AI programming system; Statice, which is an object oriented database; Concordia, which is a hypertext documentation system.

Then there are finally some contract-driven customer solution work, which are becoming an increasingly large part of what we do. For example, I personally right now, my work at Symbolics is on a contract with a utility company in the area of scheduling and planning. There's another very large contract which was prototyped in Lisp and the delivery work will be done in C++, which is for an airline company for operations design making – flight control and that kind of stuff.

There's some work in doing some customized versions of both the hardware and software to fit into specialized application niches. I'm not sure how much about one of those I can say – it's pretty exciting, but I don't know how public it is, so I won't say much...

SWM: "A large company with three letters in its name..."

HES: Yeah, a large company with three letters in its name...

SWM: "... has repudiated using C."

HES: You probably won't guess the right company with three letters in its name. There are lots of them anyway.

The company [Symbolics] is a Lisp company. It's committed to that. I think we are pretty much in harmony with the kinds of things that people have been saying here. We are both a developer of Lisp technology and a consumer of it, as application developers. We face absolutely all the same issues that everyone else has been identifying: There are significant barriers to deploying Lisp applications. Some of them are the technology's fault. It is too big, by and large. There aren't good enough tools for shrinking it down to what you actually need. Some of those tools aren't easy enough to use.

Things have gotten better about cross-platform compatibility over the last several years, and that's a big plus. But, there are still significant areas where that needs to be fixed, particularly user interface. (The stuff being done with that with CLIM is great.) We're committed to helping those to the extent that we're capable of doing. We're a company right now with a yearly revenue of somewhere in the $30-40M range at current rates ... I think about $40M at current rates. And so we don't have ... we're not the size and leverage that we once were in the marketplace. But, we still regard ourselves as a still significant player in the Lisp business, and we want to do as much as we possibly can to help make that business successful.

We've contributed significantly to the CLIM effort. (Just in fairness to Scott... people keep talking about CLIM as ILA's thing ... this guy did an awful lot of the implementation work, and I think deserves an lot of the credit for it. Not that he feels taken by that, but I am jealous of that credit for him. I think he deserves it.)

We are anxious to cooperate with the other vendors and users in this field to try and make Lisp be a viable technology. We think it's crucial to build in the kinds of application systems that we see as being the growth of our business areas. Those are complex, decision- making applications that require lots of heterogeneous kinds of components ... big integration ... tend to be the larger such systems. We don't see technology in other languages that's comparable to Lisp at that level. It's the best glue we have.

I think we agree that we don't want to be arguing over the entire world with the C folks. That's silly - they're going to win. But it's important to establish where Lisp is important, where it has viability, and to demonstate that success. I think we'd like very much to cooperate with the rest of the people in this room on doing that.

For example, we hope these guys you [Harlequin] mentioned before –the 20 guys outdoing the 400 guys– we hope dearly that they win, and that you talk about it as much as you can, and that you help the rest of us talk about it as much as we can. It's much more important to establish that that's a successful big Lisp application than whether it's your customers or our customers to us.

SWM: Can I just say the one other thing that Symbolics is committed to, is this notion of co-existence. Co-existence and cooperation is much more important than winning out over C. There are just too many of them. I mean, you can shoot them dead, and all that will do is provide a wall of C programmers behind which there will be more C programmers that you can't do anything about...

MAS-B: A little hedge in front of the Cobol programmers...

(laughter)

SWM: The only thing we can be doing is solving this by carefully investigating things like cross-language calling and cooperative data representation, so that, for instance, we can leverage off the half-dozen companies that are producing object oriented database for C++ right now.

MAS-B: That's a very good point. We had hoped as organizers of this [workshop] in the afternoon today to get these themes that everyone's been talking about for a day and a half now out there, and say, okay "How?" To steal Dick's term, how do we circle the wagons? What steps do we as companies and individuals take to see that that happens.

–lunch break—

4.2 First Afternoon Session

(In the afternoon, March 27th, 1991, just after the lunch break.)

MAS-B: Let's see if we can wrap up the statements of individuals' positions. There are a few more to go. I'd like to have the remaining representatives of vendors say a little about their position. Then we have two representatives from the user side, SAIC and the MCC/Cyc/MCC folks who have interesting statements of their needs, I think, and particularly that address the high end of the complexity space, so I'd like to give them an opportunity to explain in a little more detail their kinds of requirements and the things they need.

So, to kick off, if Eiji Shiota from Nihon Symbolics could tell us a little bit about them... this is to connect Howie Shrobe's final statement [before lunch] about Symbolics Inc, Mr. Shiota can speak for Nihon Symbolics.

Shiota: My name is Eiji Shiota from Nihon Symbolics. Our company is a distributor of Symbolics in Japan. Our business is almost all localizing Symbolics products, Japanizing and some Japanese device driver development.

Our company was founded five and a half years ago. Almost all these five years, the same number of machines installed every year. Every year, more than 70 machine installed. Not this year, and not two years ago. Three years ago, in Japan, the AI boom is coming. At that time, everybody wants Symbolics and Lisp machine. This year and last year, most people don't want. AI boom is over.

But at this time, Graphics boom is coming in Japan.

Dr. Howard Shrobe said several things. First is graphics. In Japan, almost 70-80machines. That is the situation in Japan.

We already installed more than 500 machines in Japan: 3600 series is more than 300; MacIvory is less than 150; UX is *inaudible* is less than 10; XL400 is more than 50.

The next thing: Now the important things in our company, one is Graphics. That means application. So, computer graphics applications, is like AI programming: Trial-and-error style is needed. That style is basically Lisp style. Lisp is flexibility, extensibility... Why does computer graphics use Symbolics machine? Because of these things.

My position is that I am one of the *Yy* members, but not so much working on *Yy* , more just interest. Sometimes commenting or assisting the *Yy* project.

About CLIM: CLIM is announced about two years ago, 1989 in June, at SLUG. But has not come yet, so I am waiting about two years. So please come up *inaudible*

The important thing is that we need applications. This means we need end user. How to get end user? We need to assist the programmer. We need some documents. First is we need a programmer's document, like an API or something. That needs many examples, or samples. The second document we need is for implementors. These two documents are different. Sometimes I confuse these documents. One is a law book, the other is a bible. We need both law books and bible.

Future directions: I like Symbolics, and I like multi-system using the network. Each system has [it own benefits]. If you use Symbolics, C is no good. If you use Unix, C is the best one. So, multi-system using network is a very good way for us. So the *Yy* project is one example of this, so I agreed, and I assisted the *Yy* project with this need.

MAS-B: Thanks. Intellicorp.

Dholakia: I'm from Intellicorp. Most of you know what we've been doing for the last four or five years. The mainstay of our product line was a Lisp-based product called KEE, which was like a software engineering environment, which provided object oriented programming with some truth maintenace system, and Common Windows, a user interface system...

We've seen a decline in that business, and we've tried to use some of the stuff that Lucid provided to help us in *inaudible* We've had some limited success with all of that, but all in all, we've still seen a fairly significant decline in the Lisp business. So, Intellicorp being a tools company has moved its focus and, for better or worse, its future and fate is tied to the success of a new product called ProKappa. It's a C-based product that rides on top of Saber, right now. It's intended to be Saber-less at some point, but it begins to provide you with an object system on top of C.

It's as heard a sell in many senses as KEE was. Just being object oriented doesn't ... at least in gets you into the places that used to say "Lisp? No". At least now we're getting into some of those places. If anyone expresses an interest in seeing that environment, we can set up a demo.

As far as the Lisp business is concerned, we saw some disarray over the last eight months or so where there was some confusion over whether ProKappa was a KEE re-implementation in C. It is not, it's a product designed from the ground up. We saw this sort of up and down in the KEE business, and I'm happy to say that the KEE business has stabilized and actually grown over the last couple of quarters. Part of the reason for that is that we've regrouped and now have a little business unit inside of Intellicorp that focuses on this product line, and my neck, and Mark's, and a couple of other people are tied to the numbers we make. So far it's secure, and we are trying to push to see what else we can do: We have an object system that has not been significantly updated in the last 4 or 5 years; and a user interface is pretty old.

We're trying to do things with it, and we're looking for ideas, which is why we are here to hear about the things going on in the Lisp world. We still face the same constraints that we had before. Delivery is an enormous issue, but not as significant as it used to be. We have a customer in Europe who is getting reasonably acceptable performance on a Sun under 8 MB, with just the core and Common Windows. They're happy with the application that they have, and they're willing to let their sales agents go around to various places and *inaudible*.

So we've seen some successes and our strength right now is in holding on to the customers that we have, in making them successful, making them happy. The hard thing for us now is to go out and sell more licenses. That's a significant barrier there. Especially given that there is some schizophrenia in the market about which product to sell and how to emphasize it. Our position right now is to hold on to the customers that we have. *inaudible*

MAS-B: Mr. Tanaka.

Tanaka: I am a member of the *Yy* project, but I usually work on the Unix machine in C. I mainly do research about networking, especially TCP/IP networking. I implemented the *Yy-server* part of *YyonX*. The *Yy-server* is written in C. This project is led by Prof. Ida.

YyonX is an example of cooperation between a C process and a Lisp process. Lisp process does the more high level operations. C process does the lower level, primitive operations. In our case, some problems occurred. One problem has been described by Mr. Kosaka this morning: Lisp processes in Unix environment push out other processes. The Lisp process needs too large a memory in Unix environment, so it pushes out other processes. Then Lisp itself runs very fast, but other processes stop because of the Lisp process. That is a problem in our project, but I think that this problem will occur in many other cases in the Unix environment. I think Lisp implementors should consider this problem.

If you feel that *Yy* is not so fast, then look at the demo of *Yy*. This [poor performance] comes from that problem. Mr. Kosaka?

Kosaka: I have tried to use the MacIntosh, a MacIvory, a Sun, a Symbolics 3620, I feel that it [the 3620] is faster than Sun...

Dholakia: I forgot to add two things. One is that we got through with an implementation of Common Windows on CLX. We used to send out sources of that stuff, if anyone's interested we could probably do it again. If it's of any use to anyone else, we could release the source code. We'd be happy to do that.

The other thing is that I'm personally interested in seeing if any of the vendors or anyone else is interested in the Object Request Broker stuff that the OMG (the Object Management Group) is working on. This is this object oriented backbone that sits on your network, and you write your applications to it... supposedly everything is updated by this object *inaudible* of some kind. I think client/server models like that help us in the long run. The fact that we have something like X right now... we have done something very similar to X with a product where software ran on a VAX and was displayed on PCs. X allows us to separate those, so that we've gotten actually not too reluctant about running a large Lisp application on a mainframe and distribute the interface out to network servers. So I think that distributed processing technology holds some degree of promise for *inaudible*.

MAS-B: Sun is involved in that [OMG], isn't it?

Dholakia: Yes, the proposals are all in, and I've looked at some of that.

MAS-B: Related to that, Texas Instruments just got a big contract from the government to work on an object oriented database and C++ and CLOS access to it.. I've only heard a little about it, but I talked to one of the guys at TI who's running it... do you know anything about this?

RPG: If it's the group I think it is, it just shows the government incompetence at awarding contracts.

MAS-B: I see, did they give the contract before asking for proposals?

RPG: I don't know. If it's the group I think it is at TI, it's a *inaudible*.

MAS-B: On your [Dholakia] offer to make the sources of Common Windows available, mechanically how do we do that?

Dholakia: My e-mail address is: dholakia@intellicorp.com

MAS-B: Good. Next, I wanted the Cyc folks to talk about what they're doing, because I think it's very interesting, and because it represents the class of application for which the claim is still being made that it can't be done in anything but Lisp. In talking over lunch, I threw that challenge in their face to articulate why it is that Lisp is necessary for Cyc.

Pratt: I'll give an overview. The Cyc Project is a project being run out of MCC, which, as you probably know is a consortium of American companies to do long range computer-related research. The Cyc project is one of the most long range projects at MCC, it originally had a 10-year horizon. That horizon may be being stretched by the current administration. We're in it for the long run. Our project in particular is being supported by a number of shareholders, but the ones that would be most interesting are DEC, Apple, NCR, ..., *inaudible*, and Kodak.

The point of our project is to create an extremely large knowledge base of common sense facts about the world and to provide a reasoning layer with that knowledge such that it can be used as a knowledge and inference utility by arbitrary programs. That of course means that you have to do an application in order to show anything off, so we're both in the prototype application business along with our shareholders, along with formalizing .. for example, I'm presenting a paper tomorrow morning at the AAAI conference on a naive theory of money.

The program itself is a large Lisp application. It makes your SMBX image increase by about 40-50MB, and it winds up with an image of around 100MB under Lucid or Franz. We typically like to run with at least 150-200 MB of swap space. It's supposed to get a lot bigger. The better we get at our data entry techniques the faster it's going to grow. We had one jump this fall in terms of a great amount of data and we hope to make it jump another couple of notches, so we'll be looking at 300 MB images within another year or two. At least we hope so.

SWM: So you're actively working on data compression, right?

Pratt: Well, we'll work on that as we need be. It's true that one of our problems is paging. Some of this we'd like to control ourselves. We'd like to reason about what parts of the kb are germane what parts aren't and control their location in memory. So one of the things we'd like from all you (Lisp) vendors, we're interested in as many hooks to control locality as we can get.

Veitch: I'd like to say to the hardware vendors: Are you going to give us any hooks?

Pratt: We support a functional interface for people to writ apps which attempts to bring the system up to predicate calc level. There are several interfaces that run under Genera and on the Macintosh. Our recent work with Bob Kerns is creating a frame editor that runs under CLIM. On top of that our basic desire is to have our efforts with CLIM work out well enough so that we can go to a single unified interface across all platforms. We do not have the luxury of settling on any particular platforms or Lisp vendor because of the politics of dealing with our multiple shareholders. So we are in the process of porting to or have committed to port to the Sun4, the DEC 3100, 5000 series, IBM RS/6000, Intel x486/386 under Unix (because of NCR (?) being a shareholder), and the Macintosh.

I've managed to sell our shareholders who are skittish about Lisp on our project producing black boxes ... it's going to sit on the network, it'll get predicate calculus expressions in, and give predicate calculus expressions back out. Or something along those lines. But they do not have to have Lisp up and running on all of their platforms. It could very well be that you could have a 100K application running on a dinky system that nevertheless could go and ask a tough question of this resource.

In order to pull that off, one thing on our wish list is efficient round-trip communications in order to ask such questions. Our naive implementation right now with TCP/IP streams, just doing null operations, we've kind of gotten stymied at about 20/sec, which just brings our editing to its knees. I'm sure that it's because we're losing somehow and we ought to know better how to do this. But we want to stay busy doing knowledge engineering and so we'd be very very happy with anyone who comes over and says: Here's a solution to your problem and it's portable across platforms and vendors.

Efficient multi-processing is something that underlies our desire to have multiple server processes simultaneously accessing the knowledge base.

One thing I'd close with is that, I'd like to suggest to the vendors here - as a former vendor: I used to work at the late lamented LMI - is that you find ways to have products that can run across all platforms, all Lisps, and allow *inaudible* value added ...

– end of tape –

... my network problem solution that would run across all the diff plat that I have to support here. I don't want as much as a full Common Lisp to get that solution. But the solution I need is like 8 pages of the right code. But getting that right 8 pages is quite a few man-weeks maybe man-months of effort and another certain commitment of maintenance. It's not as big as Common Lisp, but it's more than I want to get into on our project right now.

Bob?

RWK: Dexter talked about what the goals of the Cyc project are - in terms of portability and so on. So I want to address a little bit about what the concerns are from the standpoint of what this means for the Lisp application and a little more about Why Lisp?

One of the things that we hear about a lot is the size of the Lisp. For this class of application - not to denigrate the importance of the issue, but recognize that for the traditional Lisp market, the size of the Lisp was not such an issue, and that it's still true for this kind of application, where the size of the Lisp is nothing compared to the data. I'd like to observe that this fits with the trend of a lot of other kinds of applications not written in Lisp: the size of the data being manipulated, be it 24-bit true color images, or large CAD databases, or most any other thing ... Programs are dealing with larger and larger volumes of data. It is a trend. The size of the application is not yet irrelevant, but I think it is becoming less of an issue, and it's important for the near term survival of Lisp, but it's not a long-term issue.

To speak a little more about what the concerns are from the user interface standpoint: The ability to port a look-and-feel that is appropriate to the environment is an important thing. Part of the Cyc Project's mandate, as it were, is to produce something that can be brought into the shareholder organizations. *inaudible* The individual shareholder companies have barriers against Lisp because they want things in ways they already know how to absorb and integrate into their way of doing things. This also applies to user interface.

While at the same time you can't afford to implement yourself different user interfaces across different platforms (I think the Cyc project has gone farther than most any other project I've seen in terms of developing a variety of user interfaces on different platforms) I don't think that ultimately we can afford to do that, yet at the same time for acceptance, we have to deal with the reality of delivering on multiple platforms in ways which integrate nicely with the environment there because otherwise you won't get the acceptance. I think that CLIM isn't there today, but the Cyc project certainly would endorse that direction that CLIM is moving in.

MAS-B: Can I ask a couple of follow-ups there? The situation of Lisp at MCC I think is of general interest. There's a comment in the book blurb that Peter Norvig gave out from Dave Loeffler [of MCC] that says he could have saved Lisp [at MCC]. There's a rumor that Lisp is dead [at MCC], and the new guy who came in - Craig Fields from DARPA - that he hates Lisp ...

Pratt: That's not true ...

MAS-B: Well, this is what I'm saying - there are these rumours out there - can you speak to that?

Pratt: If anything, things are better under Fields because Fields is technically apt. There are those in the room who have had more dealings with him than I have, but as far as I can tell from what changes have happened at MCC since his arrival, he's somebody who would be willing to make a reasonable technical argument to, he's not trying to get you to give him a "Yes or No: Lisp is good/Lisp is bad" kind of message.

However, he's very pragmatic, and he's not about to go and reverse what has happened, notably in the CAD program, long before he got there. There are a number of different tellings of the CAD program story - I wasn't there for any of them, but the one that rings true to me is that they started as a bunch of little independent research projects, and somebody decided they needed to play together as a big framework, they jammed it all together and it didn't run very fast when it was jammed all together ... not surprisingly. Lisp immediately was scapegoated..

SWM: More accurately, Symbolics was scapegoated and Lisp was tarred with the same brush. To put it bluntly we got reamed by that fiasco - it was pretty bad.

Pratt: The carry-over was that Lisp did indeed get tarred with the same brush such that all other programs had to constantly defend themselves over "Why are you still using Lisp? It's never going to run fast - it can't possibly run fast." And they really missed the point that even if C may have an execution speed advantage, when it finally comes down to paging, as a lot of these big applications do, if anything, the Unix boxes don't page large things very well and it doesn't matter how fast your inner loop is running if you're waiting for the disk most of the time.

So the way you make your system run faster is by intelligent construction of the architecture so that you stay local, and that's more important than wasting your time fighting with an implementation system that's trying to bite your hand all the time.

Cyc is safe [for Lisp], basically because we're "doing good" right now. We have happy shareholders and happy shareholders means that we're pretty much invulnerable to any pressure to go and do that. And also, we can say to them, "If you make us convert to C++, the project would not move for three years because of the volume of code and the need to completely retrain our staff."

MAS-B: Are there any other enclaves or bastions of Lisp work within MCC?

Pratt: The Natural Languages group, who are currently porting to both Franz and Lucid [from Symbolics].

RWK: I wanted to say a little bit more about "Why Lisp?" and why why Cyc couldn't really move to C++ reasonably. The traditional arguments, which I'd have to say are valid, have to do with programmer productivity, but there's a number of other issues that also relate to it.

One is the mutability of the software. The data being manipulated here has a much longer life than the software involved. The software is continually evolving - it's a very large and long term project. To do an edit-compile-link-and-reload the data ... how long does that take?

Pratt: 2.5 hours, 3 hours.

RWK: That's only going to get worse as the size of the knowledge base grows. To be able to do incremental development is crucial for large-scale, dynamic projects, where you don't have this nice PERT chart .. step by step, you know here's what you have to do and here are all the pieces.

Another area has to do with robustness, reliability, and error-checking. Because this kb does not typically come from source files, but is data in the kb, it's in your address space ... C programs, perhaps worst of all the commonly used languages now, essentially has no way to detect errors in programming logic, except as the programmer explicitly foresaw. To have pointers be dereferenced the wrong number of times, or falling off the ends of arrays and so on ... things which Lisp programmers even forget you have to worry about as a problem - are sort of inherent in the architecture and design of the C language. Even other languages don't address run-time checking nearly to the extent that Lisp does.

To have your ongoing data which survives now for 10 years be subject to an occasional reference off the end of an array or do something that wasn't a pointer that you thought was a pointer *inaudible*, that's simply intolerable. Even though you may write very careful code, and very careful test suites, just the statistical argument indicates that as your program grows and as the lifetime or your data grows, that becomes more and more of a problem. More and more danger exists.

Veitch: A lot of the issues you're talking about are really database issues. There was a big project at MCC called Orion, are you guys interested in that at all? At the moment your database is everything in VM.

Pratt: Orion didn't have the requisite expressiveness at the time. We actually used very simple objects because this project started long before we had a portable object mechanism. So we just had DEFSTRUCTS with large attached lists and lists of lists. As I understand it, the Orion technology may now actually lead to using non-object-oriented databases to shuffle off large chunks of data.

An active project right now is one where we are able to ask a question of the system, you don't know it off the top of your head, you formulate a query to an appropriate database, where you have representations of the database and its schema and what kind of knowledge you expect to find in it. That extends in two directions: One for database integration of arbitrary ... databases, but also integration of a captive database where you can count on its precise organization.

Veitch: Does that mean that your long-term path for Cyc would be a database component and an interface into the database?

Pratt: We will always have a very large working set. The point of the database will be when the knowledge that we're working with is ... and we only want to have 200 Mbit in storage. My off-the-cuff guess - which is not the Cyc group's in general - is that in order to hold a conversation with the user using natural language interface, and doing any sort of job of understanding where you are in the conversation, you're probably talking about creating MBytes of structure and storing MBytes of structure every few seconds, and ... consider the range of interpretations that the conversation might have ... I don't think we're going to be doing that in the next couple of years...

SWM: Doesn't this argue for a language that has more explicit control over memory management than Lisp does?

Pratt: I'd love to have something that gave me a sort of intermediate quantity - more than I have now, but sort of soft, sort of an advice system, where I say "This is probably important, but don't make me actually have to MALLOC that area." Under Genera, we already do a certain amount of consing in areas, and that seems to help.

RWK: I think that the database issue, while it does somewhat help the issue of robustness over time, .. as long as you're actually manipulating data, and that data has a long lifetime, and that data is in virtual memory, it is vulnerable to ... It's a product of your address space times time that in some sense is the measure of the risk. As long as that is large, it's something you have to consider.

If what happens is that your editor blows out and you lose one file, that's not at all the kind of scale of problem that we're considering here. You might not find that for two years you have a pointer off into some .. you have no way of telling where that came from. ... I think this is a challenge for all of the Lisp vendors, including those who have assist from hardware checking ..

The kinds of regressive error checking that can help to track down problems, I think it helps everybody .. The sooner that an error is detected - whether it be a malformed declaration or an access to a structure ... - whatever kind of checking, the more you do of it, the more robust the software. One of the applications of this rightly should be the developing of robust software. The [Common Lisp] condition system is better than any other condition system that I'm aware of out there. The run-time type-checking has potential for being ... it probably is better.

The only place that people can say we fall down is compile-time analysis of types, and there's nothing inherent in the language that prevents that. ... Compile-time checking is the first line of defense. Adding to that can only help. In the final analysis it the run-time checking that's the final line of defense against data corruption or ...

PGA: I have a question ... A lot of what Dexter said about consing MBytes of data in the context of natural language in a user conversation involves a lot of knowledge that is temporally very short-lived. It's hard to determine which knowledge, though, will be short-lived. It has a lot to do with a very very complicated and very very context-dependent features of that particular piece of knowledge, so while I think there is a lot of ... a simple GC scheme, or simple allocation/deallocation scheme would not normally be a solution to [Cyc's problem].

In the meantime, the way that it's dealt with is to do something like have a reference count on the knowledge or .. so that you can GC it conventionally.

RWK: I think the best known - as opposed to hypothesized - solution for that type of thing is the ephemeral/temporal based GC where the system automatically makes notes persistence of pieces of data and determines how best to manage the lifetimes of these objects.

PGA: I think there might be room for some enhancements to that. For example, you might want to have reference counts associated with a piece of knowledge and have a GC that might throw something away that was still referenced, because it was [determined to be] unimportant enough that you could throw it away. That's a completely radical view of GC.

MAS-B: One of the things I wanted to do is put you on the sport a little bit, given that you [PGA] do work for DEC ... and I understand that you are not in a Lisp product group at DEC ... but what can you say about Digital's position with respect to Lisp, or activities that you are aware of. I heard a statistic several years ago, that counting the value of hardware sold to go along with AI applications and tools that DEC was the biggest overall AI and Lisp company. They didn't sell a lot of copies of Lisp in terms of numbers, but they sold a lot of VAXes that it got run on. Can you tell us anything about that?

PGA: There's a lot I don't know, but one of the things that I've observed in the time that I've been at DEC, which is just under two years, there's been a lot of reshuffling and a decreasing number of people within DEC that are actually using Lisp. And obviously, the entire Lisp product line got dumped ... the direct Lisp being developed and maintained by DEC ... so we're now dependent on Lucid for our primary [Lisp] support.

There are several groups doing applications in Lisp, but even there there's a lot ... the bottom line is that it's not going very well...

MAS-B: So DEC is not a hotbed of internal Lisp activity...

PGA: It definitely is not, although there still are a lot of people who are concerned with it and looking for an opportunity to bring it back. I think I've already spoken at length about my personal strategy for trying to do that.

I'm trying to think if there's anything ... We even had a large knowledge representation project going on - which I believe I can name it: it's called MKA ("Management of Knowledge Assets" or something like that) - and it's C++. So, we'd like to see - the people in my research lab - would like to see Cyc gradually getting a foothold within Digital, which is one reason we're putting as much into it as we are.

MAS-B: Good. Mr. Sato, even though you presented yesterday, could you tell us briefly about TIRL.

Sato: Thank you for giving time for me. I am Mamoru Sato from TIRL, and also I am an employee of Canon. TIRL is "Telematique International Research Library." Research about electronic library.

This company is a five year project, and this September, we will be finished, so I will stay with TIRL for three years. I am studying about windowing directory locating system for electronic library. This library is for general persons, not only professionals. So the directory system is very difficult, so we are using loading picture for retrieval. The user can feel he is in the real library, so, if the user wants to read a book, the user can pick it up by [using the] mouse and open it, and read it.

The data is very large, about ... I have an experimental system named San Wu Kin: this system has only 5 books with whole contents, and 20 with only covers and some hypertext links. This system needs 100 MBytes of memory. So we cannot load whole system at the same time.

But we also use distributed environment, so we can use 3 computers - each has a very large memory, 32 Mbytes, or 64 Mbytes, so this is the profile of San Wu Kin.

As I told yesterday, the data retrieval must be familiar [to the user]. Also, programming of the data retrieval must be familiar. So I use Lisp for the programming of the data retrieval. The programming of the data retrieval means the design of moving pictures for the retrieval. This program needs extensibility, not only selectability.

Lisp has interactive development tools; and extensibility, not only selectability; and trial-and-error environment; and also a good system.

By the way, when I joined Canon, I made first major processing program by Fortran. Two years later, I made image processing programming in C. Finally, I made small own Lisp interpreter for programming image processing user interface. I love Lisp for user interface, but I don't like to use Lisp for image processing itself, because image processing programming is array processing, so it is fit for C, for Fortran. So I don't like using Lisp for image processing itself. But I want to use Lisp for user interface programming.

This is why I use Lisp for the programming for San Wu Kin. I use C programming for image processing for San Wu Kin. (San Wu Kin is the name for the experimental electronic library system.)

Finally, I will talk about my opinion. In large system, especially multi-user and multi-processor system, we must consider every processor never has same attributes. Each has different attributes.

MAS-B: Okay, thanks. To finish up Cris (Kobryn) has a fairly long presentation. I think you represent a class of user that has been the bread and butter of Common Lisp.

Kobryn: We've spent a lot of money on Common Lisp. I'm Cris Kobryn from SAIC, the applied AI branch.

To give you a quick overview of what/who SAI is, we're a diversified enginerring company, with about 11,000 employees, and last year we did over $1B in business. We have several centers for AI software development. The other notable thing about my company is that even though we are over $1B/year, we are still privately held.

Regarding the applied AI branch, which I represent, we specialize in KBSes and intelligent user interfaces, and I'll explain what I mean by intelligent UI in a moment. We have a fair amount of experience with various Common Lisps, back to LispM days, with Symbolics, Explorers, and D-machines. We're also thoroughly familiar with all the Unix implementations, as well as on the Mac - Coral which became MCL. We've done a lot of alpha and beta testing of these products.

From a top-level perspective, we don't look at ourselves as Lisp fanatics certainly; we're fond of Lisp, but we're also relatively even-handed in that we use Prolog when it's appropriate. What we actually prefer is a Lisp-based syntax Prolog within the Lisp. In order to be effective, that Prolog needs to perform at about the 20 KLip range.

Currently we are using Harlequin's LispWorks for most of our product development.

Some of the major product development activities that I've been associated with in the Applied AI Branch would be: distributed processing in the large, what we refer to as Distributed Application Operating Systems, using Cornell University's ISIS system for the last iteration. The distributed system that I'm most familiar with is a $20-25 Million project funded by DARPA which does signal processing in Norway, blasts the data to [Washington] DC to do knowledge-based system processing, and sometimes to San Diego as well. So it's WAN-based as well as LAN-based.

Another area that we've been working with Inference Corp on is performance analysis for knowledge-based systems. That is, doing validation of knowledge-bases cached by ART.

The third area we see the best growth for future development in is what we refer to as Intelligent Information Systems. There we'll be doing a lot of work in hybrid graphic/Natural-Language/UI supporting relational database. We're familiar with most of the commercial NL products. We have a close working relationship with [Natural Language Inc's] natural language group out of Berkeley. And also dynamic hypermedia. The idea that it uses dynamic links instead of static links.

The third area of grwoth would be spacial databses - by that I mean geographic information systems and CAD spacial databases. To show that concept graphically - what we refer to as graphic media ... Our baseline is that we do knowledge-based systems well, and in the process of building this large distributed processing system for DARPA, we've come to know relational databases relatively intimately also. Also in that process, we have perceived the need for handling on-line documentation in a civilized manner, and have seriously explored the use of hypermedia in the large.

The recent growth area that I was pointing to is this GIS CAD activity. We recognized about a year ago - because a lot of our work has to do with environmental sciences: nuclear waste disposal and that sort of thing - that geographic information systems were growing quite a bit, and they're yet another type of database that is difficult to integrate with other entities, specifically knowledge-based systems, and we have a very large project at Ogden AFB to plan explosive ordnance sites and store explosives efficiently.

Some of the top-level goals for "DynaMedia" are to develop a unified perception model for accessing, organizing, presenting this distributed information, and, of course, to implement it in a practical manner. Some of the technical issues that we've divided into front-end and back-end issues: On the front-end side, UI standards of course. This tricky complexity/power balance. Make it easy to use, yet powerful enough to do interesting things. That's tied in with our concepts for intelligent user interfaces. And the idea by data caching, I mean, in the practical sense, selecting subsets of a relational database to make large-scale queries effective or practical.

For instance, in terms of seismology, our analysts often want to analyze data over 10 years. We might first extract a subset over time using another feature, then using that subset, do a more detailed query on that subset. That gets into data capture, data exchange, and performance.

To give you some idea of what we've developed, I've got a couple of screen dumps of the system. What we're using is Harlequin's HyperBook hypermedia system, What's interesting about this system is that it's based on a little filing card metaphor - based on a library/book metaphor. As you can see, the book has the appearance of a book, with tabs, a page corner ... It's a hypermedia system in the sense that it also includes graphics as well as text. [The active link here (in the example) would be Arrival.]

What's important about this we think for large hypermedia systems is that it gives us the ability to implement dynamic links. So that at select time we can do basic inferencing to go to the appropriate link ...

–end of tape– ...

4.3 Second Afternoon Session

(In the afternoon, March 27th, 1991)

mid-discussion

Veitch: ... CLIM might have had a much different fate, because there's a lot of resource that can be brought to bear.

MAS-B: Exactly my point: If you can start with the consensus on what to do, you can get there efficiently.

Kromer: For instance, if a trade association can identify specifications ... let's take, for example, X3J13: If a trade association was to take on the identification of standards that specs need to be written for, it can sponsor such things. There's a small group of us that are contributing to finishing X3J13, and this trade association -or whatever you want to call it- could do things like that, and I would be very much in favor of it. That's one area that would be of a great deal of value.

There are other things such an association might do that I'm not taking a position on, bu, for instance, there was a difficult development task that had to be done, and, for most of the [Lisp] companies, their resources are thin, and so one might be able to write a general proposal for such a thing and go to government agencies to get funding to help build things. And that might be part of it, but you have to take things one at a time.

MAS-B: Two follow-ons to that. First, should that, which you just identified, be the starting point? I don't know what the formalization is for the funding that's been given to Kent Pitman. Is he a one-man corporation, a non-profit corporation that you all set up? Is that the right to start this up?

Kromer: That could happen in the future. The way that the Pitman thing was set up is that Symbolics and Lucid and Franz -and hopefully Sun and Apple- are just contributing funding that's given via consulting contracts to Symbolics to pay the guy's salary and expenses as he finishes writing the spec. It's just a jury-rigged thing to get it done.

MAS-B: Off the top of my head, that sounds like as good a seed as any to try to transmogrify into a trade association.

Kromer: Well, it would certainly be something that could be part of one, yeah. If you want to do a trade association, somebody ought to take the task to write down some of the possibilities, and if you can get consensus out of whether this is a user and vendor trade association, or just a vendor association, or what is it? I mean, you have to get those things sorted out.

MAS-B: Well, the user side seems to be coming together...

Kromer: Yeah, I can see that.

MAS-B: There's a kind of a willingness, it sounds like, on the vendor side, following the Japanese model –what we heard– is that maybe we can start from the bottom up, and if the vendors can agree, then the users will be interested in hooking up with that somehow.

Ida: I have some comments. The JEIDA Common Lisp committee is no organization for protectionism. At the very beginning, we had principal members from Digital and other US companies. Since I have been –and I am the chair of this committee– any efforts for nationalization or something is not subject for the committee. Concentrate on the technical issues and concentrate on the communication of the users and vendors only. Usually, such industry consortiums or committees are misunderstood as monopoly-making by several companies ... sometimes misunderstood. We have carefully to reject such ideas and keep the health of the committee for five years.

Though I myself said it should be a national one, maybe there are some reasons for openness to international activities are these things.

MAS-B: That was my second follow-on: Whether the feeling should be that we should try to start with a US association which would then, in very short order, be connected to the Japanese association, which seems, in this context to be farther along, and what the right model would be for such a thing in Europe. How would Europe respond to a "Lisp Trade Industry Association" of the companies that Gene listed? Would that provoke a protective response, or would that be something that Europe would want to connect to?

ASmith: I can't speak for Europe. I can speak for Harlequin. We would be very interested in being involved in a trade association. A large part of our business is in the US anyway, and we would want to do more by way of cooperation. The precise nature would of course (inaudible)

MAS-B: I want to finish my sort of conversation with Gene. I still feel like the question of whether what you all have already done informally should be the right starting position for a formal structure ... I kind of want a yes or no answer to that. What do you think?

Kromer: I'm not sure, and I'll tell you why. Because I haven't thought about it, is one reason. The second reason is I don't know what ... if an organization starts collecting funds and using them for some reason –I mean we're not talking about a large amount of funds, but here there are some funds going around, and they're large to us, and they're large to Lucid, and ... I know this because we argue about the size of the funds– but nonetheless, when organizations start collecting funds to pay someone to write a spec or whatever else might go on, there happen to be legal problems, and I don't happen to know what the typical legal organization is for a trade association. I don't know if you incorporate such a thing or what you do, but there are unfortunately, at least in the United State –we're so encumbered with legal things– but there's got to be some structure. So, since I don't know anything about it, it's hard for me to say.

But, I do believe there's a need for something. And there are benefits. I don't know what they all are, but there's one example: getting the [X3J13] specification done. Another example would be: do we need a specification for a minimal Lisp or for certain interfaces or what have you. That could be set up to be done through a trade association. (I've said this before) You could even find something that no one wanted to do on their own, but that was good for the Lisp world and one might prepare a proposal toward various governments to get money. All of those are possibilities, but soon you'd run into (the need) for a legal structure, and I don't know what that is, but it needs investigation.

MAS-B: Right.

RPG: I would hope ... well, having a trade association that sets specifications, etc. is not going to help all that much, to be honest. It's going to help a little bit, but I know one of the things that Lucid does is we sit around with our crystal ball -which happens to be me- and we sort of guess "wonder what the users really want?" and we make big lists and say this is the number one thing, the number two thing...

Almost everything that sits in that list now happens to be of a significant cost, to us, to do it, so for us to do it is a significant risk, because we're not swimming in developers, swimming in money -I don't know anyone who is, except maybe Apple- but we're not swimming in those things. I would prefer to see something where, maybe even in conjunction with user groups, for example, we've been talking about DEFSYSTEM: We have a little DEFSYSTEM; Franz has a little DEFSYSTEM, so does everyone else. If we could be assured -this is a really bad example- that this would help the user community, then one developer from each company could develop *THE DEFSYSTEM* that we all used, and that would be common to the users. They'd be happy. I'd only have to spend one person instead of three, which would make me happy. It would make you [Kromer] happy the same way, and I wouldn't be taking such a risk guessing that this is good, because I'd be sharing the risk with everyone else. I'm not taking the risk that I guessed wrong and Veitch guessed right ... which would really piss me off...

Veitch: No problem.

(laughter)

RPG: ... so I'd rather see an industry organization modelled maybe more on the MCC model than on the straight trade association model. There's a third model to bring up, which I don't know how to make work, which is the Jo Marks model. You know the Jo Marks model?

ASmith: Why one do you mean?

RPG: The tri-continental model ... Maybe you want to explain what that is... But the idea is to set up a tri-continental trade association which is really like a consortium or like an institute or like a company which acquires funding from three different areas –Pacific Rim, US/America, and Europe– and thereby spreads the risk. So that when the US is up and Europe is down, or the US is down and everyone else is up, which is sort of the case now, funding can be hard there. It's just a bigger interprise. Jo Marks wants this to be something more than that, and he's got more personal gain involved... By I know that my company is diversifying beyond Lisp. Harlequin has always been diversified beyond Lisp. In fact, its Lisp is sort of like a side line, as far as I can tell. And I know Franz wants to, or is in the process of, or already has diversified beyond Lisp. I don't have a lot of hopes that I'm going to get rich off of Lisp. But, I sure as heck don't want to see it die, and I'd like to get some money in from it. Maybe it's time to pool our resources even that way and have just one Lisp company.

MAS-B: "Open Symbolic Software Foundation." There are a couple of other Lisp vendors down there that haven't piped up:

SWM: This person has no authority in these matters, and consequently has nothing to say. I personally think it's a good idea, but what I think is of no merit.

Dholakia: I think I am in a similar position. But, having a business model (along these lines) makes it more possible for me to take something to management (inaudible) ... So if somebody would hand me a proposal, I'd be glad to take it to management.

Mori: Our company, yeah, I think they'd support it. Yeah, cooperation is important.

MAS-B: Elaborate.

Mori: Any form of cooperation is fine. If it's an international Lisp company, that's great. If it's the seperate Lisp vendors collaborating that's great. To what extent the benefits are, I'm not sure. But the thing should be written up. There's no reason that we wouldn't support it. (As for) funding, I'm not sure... I can't speak for that ...

MAS-B: Understood...

Kobryn: What can we do to get DARPA funding?

MAS-B: If we can agree –loudly– maybe they can come along.

Kobryn: Well, why not get them in the loop? They're a major customer of ours. They know that we've used every Lisp that moves. They're well aware of the issues. Can we do some things concurrently is what I'm asking? How many folks here have DARPA contacts besides SAI?

RWK: (inaudible) ... made strong noises that DARPA would be interested. They made strong noises that ...

RPG: That could make sense. Yeah, he did say something like that. It could make sense because now they are facing the prospect of developing means of reusable software on different platforms. There have been some shakeups at ARPA –which you may or may not know about– which makes the prospects slightly less tantalizing right now than they would have been a year ago.

I want to point out an interesting point of synchronicity: It was almost exactly 10 years ago today –the first or second week in April of 1981– that ARPA sponsored a Lisp meeting at SRI, and from that meeting came the seeds of the Common Lisp group. So, starting anew now would make a nice coincidence of it.

MAS-B: I want to have another little conversation –picking up on that– with Gene, because Gene is a pragmatic guy. What are some steps that can come out of this two day get-together that would lead to something constructive. One of my concerns is that we all come here and pledge eternal devotion, and then we all have to walk back out into the real world. Nobody's coming forward volunteering to spend three months setting up what we all agree to set up. Thought? How should it happen?

Kromer: I don't know how. Who's going to take the load to put something together? That's a problem. Because, tomorrow morning I have new problems, so do these other guys who have been gone for a couple of days. One of the problems is that some of the people who have been here doing the talking are in extremely fragile companies and have to do a payroll every two weeks, and worry a hell of a lot more about how you're going to meet payroll that you do about industry trade associations. That's a problem.

But I was hoping that ... One of the ways to start is that somebody leaves here and writes down the notes that they took as far as forming a trade association: Here would be the things that this trade association would do. And there would be certain things that are objectives. Some people will say "I'll go along with the first three, but I'm not going to do the last two." And somebody else would say "I like 1, 5, and 6." And maybe out of that you get a consensus where there'd be enough meat left in the thing that people could get together to start making some action happen out of the ones that everybody agrees on.

For instance: Dick has proposed doing some developments together. There's an idea of something that we can't fund, so each company, or several companies put some people in and they put it in the public domain and anybody can pick it up and make a product out of it that wants to. Probably us selfish guys over in Berkeley will say "Some of those things will do, and some of them we won't." So we don't want to get into an association we support and get hammered on these things.

So that's the kind of thing you'd want to think about. So maybe there's some others that we'd all be willing to do. Maybe we're willing to fund somebody to go write a spec. Or maybe we're willing to go join with Lucid and Harlequin to go do something right away ... I don't know.

But the first thing is that somebody when he leaves here ought to go and write "Here are my ideas for an industry trade association and here would be its duties." And see if you can get a consensus out of all the vendors or companies that want to be part of this trade association and see what you come up with. If you come up with enough pieces that people are willing to come to a meeting to make it happen, then you've made progress.

MAS-B: I should just state that the two of us [Ida and MAS-B] consider ourselves the instigators of this exercise...

Kromer: Why don't you try it? There's a suggestion: Write it down and see what kind of a consensus you get. There's a lot of people in the room who'll take a message back to their respective organizations. A lot of people will get turned off it. Some of them will get it supported.

MAS-B: The material product of this two-day session will be a report. We've had this tape recorder in front of us for a couple days and some of us have been taking notes. There is going to be a proceedings...

Kromer: Right. But if you want an industry trade association, have a thing that says "Industry trade association; here are the ideas" and see if you can get consensus on what that might be. That's my suggestion. I don't think you're going to get it in this discussion...

MAS-B: We're not looking for it in this discussion...

Kromer: You and I and Dick and a couple of others might be able to go off and have a discussion to see if there's something, but it won't have any punch unless we can hear from a few others like Sun and Apple, and hopefully Digital...

MAS-B: The two things I think we can beat around productively in this session are the question of that larger, but still inner circle. There's the old [Lisp] gang that group you just enumerated represents, and then there's the new gang, and we've had some of them for parts of this, and then there's non-US representation, and some of them, like Harlequin in the larger community are now from the other geographic areas. I think what we can try to talk about is that we do go off on our own and float these ideas and look for some reactions where they say "Gee, that's great – If you all would do that we sure put our shoulder into it."

Kromer: I'm not trying to restrict anything to just the US or North America. I would begin this thing on an international basis, until you get down to where it had to be divided up somehow for some unknown reason.

MAS-B: I would ask the question to some of the Japanese delegation: What is the process to connect what you have that exists –the JEIDA Lisp Committee– to this idea that's been floated here? How would we go about doing that?

Ida: My idea is symbolized with several key words. Most ideal thing is that we would have a kind of base on the Moon ... it's a joke ...

MAS-B: You just mean, somewhere not on the earth?

Ida: Yes ... Instead, we may have tight connections, even if they each live in their own countries. That is, by the network. If we use the network, then it is kind of driving force for organization, or cooperation.

The other thing is the content of the activities. That should be symbolized by something like "Common Lisp Technology Center" or evolved into the "Open Symbolic Software Foundation."

We already have close relations ... and we know we have much more potential market in the three areas ... and, of course, inside each of the three areas there are lots of potential markets from the other two areas. Say, in Japan, there are lots of potential market which will welcome US and Europe products. Also, we must never forget Asian or African or South American countries. With the assistance of the Foreign Ministry of Japan, I have been teaching the Introduction of AI in Lisp for exchange students in Japan. Though they do not have a big budget or are not driving forces back there [in their home countries], they are human beings.

My idea is that distributed connections of the kinds of centers [in each of the three areas] is better than one location.

Also we know that in Japan, Lisp is still a minor language. It is not the major language. It is like the Christian population in Japan. This means that unity of such a minority can be publicly acceptable also in the USA too. We must discuss about projects, and financial things, and sending people to the technology centers.

Since we have slightly more formal organizations, but not much budget yet, for 1991, we cannot invite a USA liaison, but in 1992, we hope to invite one or two technical persons to Japan. If the USA is possible, we can send one person to the USA to assist development, or as worker or something.

A lot of random such items (are my ideas).

Why don't we have a mailing list or something, to keep the discussions going?

MAS-B: We have most everyone's e-mail address from the registration forms –if we don't, please give it to us, and we'll keep the list at ILA.

One of the things the two of us have discussed is whether this [ICERL'91] is a one-shot affair. My opinion is yes. This thing, as I've said, was largely filling a void filled last year by the Europeal session, which is an excuse to get people who care a lot about Lisp together to talk about what they're doing and hear about what others are doing and discuss common interests. It sounds to me like there are better vehicles for capturing this spirit of cooperation... it sounds to me like whatever we can all do to help this unified Lisp User conference is going to be energy well spent.

I also think it sounds like energy expended to investigate this industry trade association idea is also going to be well spent. There's a lot of work to be done, but Gene's challenge to set down a document, then go looking for international consensus sounds like the right way to proceed.

PGA: I'd like to say that one of the things I can offer –not terribly definitive–

MAS-B: Well, Professor Ida, final remarks please.

Ida: To close the discussion and the ICERL'91, on behalf of the programming committee, I like to sing a song, instead of making a speech. That is the song for us, at least in a part. Barmar made it on the tune "We are the World."

– Sing the song –

```
 We :Save (the) Worlds
    Copyright 1990 by Barry Margolin

       (to the tune of "We Are the World" by Jackson and Richie)

There comes a time, when we make a function call,
  when the industry must come together as one.
There are programs dying, and it's time to lend a hand
  to lisp, the greatest gift of all.

We can't gone on, malloc()ing byte by byte,
  hoping that someone will call free().
We are all a part of McCarthy's family,
  and the truth, you know, Lisp is all we need.

CHORUS-1:
We build the world, we optimize it.
  We are the ones who patch the bugs, so let's start hacking.
There's a file we're writing, we're blocked in "Chaos Out",
  it's true, wedged forever, you and me.

Symbolic code, with CDR, CAR, and CONS,
  our programs will be more bug-free.
As Moon has shown us, with presentation types,
  so we all must by an Iv'ry board.

CHORUS-2:
We save the world, it's incremental.
  We are the ones who make the better code, so let's start consing.
There's a tape we're making, we're saving our own jobs.
  It's true we're more productive, just you and me.

When you're swapping out, there seems no hope at all.
  But if you just GC, there's no way we can fall.
Let us optimize, and change will only come
  when we--- page together as one.

;;; Repeat choruses until out of swap space
```

Chapter 5

Position Papers

Acuff, Richard: Systems Programmer, Stanford University Knowledge Systems Lab.

We in the Knowledge Systems Lab believe that one of the key requirements for Lisp to flourish is that it must have a good user interface development environment. So-called "conventional" development environments such as NeXT Step or Saber C have realized great improvements in recent time, eroding Lisp's standing as the preeminent development environment. For Lisp to continue, it must be possible for fairly inexpert users to produce complicated programs with sophisticated interfaces quickly relative to conventional environments. Further, these programs must be portable across platforms. Thus, we strongly support the adoption of a standardized interface building environment for use with Common Lisp though we believe it is critical that this environment be easy to learn and use for the usual cases.

We have had experience building complex interfaces to a variety of expert systems, simulators, system tools, medical applications, etc. In particular we have been interested in the role of the interface in acceptance of computerized aids in medical environments.

Alcabes, Harvey: Product Marketing Manager, Object-Oriented Languages, Apple Computer, Inc. Product Marketing Manager for Macintosh Common Lisp (and other object-oriented languages)

Apple Computer, Inc. sells Macintosh Common Lisp 2.0, a Common Lisp programming environment fully integrated with the Macintosh personal computing platform. It supports Common Lisp and the Common Lisp Object System (CLOS). It includes an Interface Toolkit which allows developers to graphically author interfaces for applications with a Macintosh look-and-feel.

Macintosh Common Lisp also includes an incremental compiler, a window-based debugger, a source code stepper, a dynamic object inspector, and a Macintosh-style Lisp program editor. These tools work together to present all programming activities in a high-level, object-oriented fashion. Interface files are provided for access to Macintosh Toolbox routines, including routines that will take advantage of powerful new features in the forthcoming Macintosh System 7.0 system software release.

A beta version of Macintosh Common Lisp 2.0 is now available.

Apley, Phillip G.: Member of Research Staff, Digital Equipment Corporation

Member of Research Staff, developing applications based on the CYC common-sense knowledge base, and involved in technology transfer issues for CYC. I'm involved in the project to port CYC to a portable implementation including CLIM, and have been involved with Bob Kerns (who will also be attending) and the CYC group in designing the portable interface for CYC.

I'm interested in seeing CLIM evolve into a robust product, and I have interest in seeing extensions in several areas. I'm working on a number of experimental user-interface projects, and I'd like to see CLIM evolve to meet my needs.

I'm also interested in all the other issues which were mentioned in the workshop announcement.

Atarashi, Atsushi: NEC Corporation, Researcher, C&C systems research laboratory

"How Should Window Toolkit for Lisp be Implemented?"

At the workshop, I would like to discuss, how window toolkits for Lisp should be implemented, and what kind of extensions are then necessary to Lisp. It is now an urgent task to provide powerful and portable Lisp window toolkits, so that Lisp can be widely used. I've been involved in one of such projects, *Yy* at Aoyama-Gakuin University. Its major concept is 'No Migration, but Co-Existence'. Therefore, *Yy* is designed to be native window system (abbreviated as NWS hereafter) independent. I strongly believe this is a promising approach, but I also find some problems in this approach.

One of the problems is mouse event handling; Window servers are responsible of handling all the mouse events, while the mouse is within the Lisp window system region. This results in heavy CPU consumption, because each time mouse motion is detected, the server must process it.

I would like to discuss whether NWS can provide new type of mouse event handling interface, which will decrease such overhead in NWS dependent server.

Another problem is the look and feel. Because X is so widely used, many Lisp programmers are manipulating X-widgets directly within Lisp. They are against introducing non-X type look and feel. So, I would like to discuss, whether it would be possible to incorporate NWS look and feel into Lisp window toolkits in a elegant fashion.

Apart from these problems, I would like to discuss what kind of extensions are necessary to Lisp, in NWS independent implementation. User interface programs generally require event-driven style programming, but no current Lisp system provides such primitives. Symbolics, Lucid and Franz provide multi-processing facility, which is effective to implement such event-driven programming. But I'm afraid that the multi-processing facility might be too heavy for the purpose.

I would like to discuss whether new kind of event interface (such as Unix signal) can be incorporated into the specification of Lisp language.

Eiron, Hanoch: Director of Product Marketing, Franz Inc.

Franz, Inc. will send two technical and one marketing representatives to the International Workshop. They are:

- Jim Veitch, Director of Engineering (3/26, 3/27)
- Chris Richardson, Engineering Manager (3/26, 3/27)
- Hanoch Eiron, Director of Product Marketing (3/26)

Our statement of purpose for the workshop is two-fold:

o In the 1970's and 80's Lisp was closely associated with many of the leading edge technologies. Lisp also saw some commercial setbacks. Today's increase in price/performance of stock hardware is reshaping the realm of possibilities for commercial deployment of Lisp applications. We need to set a new agenda and re-position Lisp as the language of choice for advance software development.

o CLIM is an important component of the product mix offered by Franz. It is Franz' objective to bring a high-quality CLIM to the Lisp community and explore a future path for CLIM with other members of the Lisp community.

Gabriel, Richard: Chief Technical Officer, Lucid Inc.

Ida, Masayuki: Associate Professor, director of the *Yy* project, Aoyama Gakuin Univ.

1) Window toolkit is one of the important keys to success. User interfaces including specific Look and Feel oriented operation, simple GUI oriented interaction, graphic outputs, natural language interfaces, or scientific visualiation are developed on top of the window systems using several window toolkits. From personal computer to super computer, the importance of the window toolkits becomes clear for every application fields. Also window toolkit level portability is needed for some application.

2) Rich functionality is needed. Industrial strength of the vivid computer software is supported by the layers from machine-OS level, language specification, library and off-the-shelfs, to the endeavor of users (or software developers) to write their own codes. In many cases, middle two parts are replaced by hiring shells or application packages. To reduce user's task and to make a higher level culture, presence of rich functionality of the language and its implementations is needed.

3) Real application with rich user interfacing facilities needs huge memory and CPU power. Rich functionality means large executable modules to be loaded at runtime, if simple scheme is employed. It is better for user to add nice user interface module to a

large application. But, sometimes it forces the load factor to the machine worse at the same time. To improve the situation, to use special hardware like Lisp machine is one choice. Or to employ a server-client approach to distribute the loads is another choice.

4) We cannot stop talking about LAN and network, which is a base for heterogeneous environment.

5) X-windows is a server-client model window system and is a defact standard.

6) *Yy* is a server-client model window toolkit.

7) YyonX is a implementation of *Yy* for X-windows and is available through anonymous FTP.

8) YyonX is a co-operation example of the functionality of C and Lisp. *details and highlights of the YyonX implementations are suppressed.*

9) *Yy*-Protocol enables the co-operation of several processes on the net. *Yy*-Protocol gives us a new market. Vendor A *Yy-server* can connect to vendor B *Yy-client.*

10) No migration but Co-existence.

11) *Yy-server* can talk to other UN*X processes.

12) Collorary: why Lisp is suitable to develop UI tools. UI tool development requires the following;

A: trial and error is needed.
B: provision for defining environment and runtime (production) environment is needed. And these two should be continuous.
C: Successful UI requires not only to arrange visual components and predefined menu manipulation but also to provisions for extension or alteration with procedural 'patch'.
D: Interactive development environment is suitable.
E: (not only the developers but also) users want to customize the UI, along with the daily operation.

For all of the above issues, Lisp is one of the most suitable language to use.

Kerns, Robert: DEC & ILA

Kiczales, Gregor: Xerox PARC

Kobryn, Cris: Sr. Software Engineer, Science Applications International Corporation (SAIC)

Mr. Kobryn is a senior software engineer in SAIC's Geophysics Division, where he is the technical leader of the Applied AI Branch. The branch specializes in knowledge-based systems and intelligent user interfaces, and has done extensive work with natural language interfaces and distributed processing. Mr. Kobryn's responsibilities range from project management and requirements analysis to software design and implementation.

SAIC is a diversified research and engineering company with ~11,000 employees and >$1 billion in annual revenues. The corporation maintains several centers for AI software development. The Applied AI Branch of the Geophysics Division specializes in knowledge-based systems and intelligent user interfaces.

Most of our developers are ex-LispM programmers, and we are experienced with all of the major UNIX Common Lisp environments (Allegro/Composer, Lucid/Sun/SPE, and Harlequin LispWorks) as well as with Macintosh Common Lisp (MACL/MCL). We regularly alpha/beta test for new products. Currently we use Harlequin LispWorks for most of our new application development.

Our technical work has involved the use of Common Lisp and Prolog for the following software development activities:

- Distributed Application Operating Systems
- Performance Analysis of KBS's
- Intelligent Information Systems
  Hybrid graphic/NL UI's for querying relational databases, Dynamic hypermedia, Spatial database integration

The following statements summarize our major technical concerns related to the workshop agenda:

COMMERCIAL APPLICATIONS FOCUS: We believe that the production of high-quality, functionally-superior commercial CL software packages is vital to CL's success. We are concerned about the current dearth of commercial CL applications (which sometimes begs an A*a comparison). We are dismayed that successful commercial products which use Lisp (e.g., AutoCAD) don't use CL.

ENVIRONMENTS: We recognize an urgent need to converge on a GUI standard which will provide transparent support for Motif and Windows 3.0. Other CL standards which would significantly improve applications development include an API, a relational database interface, persistent object support, and a defsystem. We think that 386/486 boxes (with their momentum from Windows 3.0 and laptop models) may prove to be an excellent delivery platform for AI applications.

INTERNATIONALIZATION: GUI standards should provide support for multi-lingual text objects. We recognize the enormous potential of the Japanese software market for American software developers. We consider excellent support for kanji critical for penetrating the Japanese market. We suggest that NL interfaces using normalized grammars may be useful for developing multi-lingual applications.

Kromer, Gene: CEO, Franz Inc.

Kosaka, Takashi: Research Assistant, Aoyama Gakuin Univ. and CSK corp. Aoyama Gakuin Univ. / CSK

I am a principal author of *Yy* client and Nihongo Common Windows.

I want to discuss 2 points. One point is a mouse event interface. The mouse event method is a user defined method and called by pushing mouse button, moving mouse or other operation. In window system, there are two major ways to implement the mouse event method. One way is to use loop facility and looking a event always in the loop, like a X Windows system. In this case, sometime the loop facility is used multitasking facility on Lisp implementations. When a event is caught, the mouse event method is called immediately. The another way is to use an interrupting facility. When a mouse event has occurred, an interrupting function is called and then a user defined mouse event method is called.

If Lisp implementations have not a multitasking facility and to use loop facility for a mouse event interface, the interactive provision of Lisp will be possible to lose especially Lisp listener. Some current Lisp implementations has not a multitasking facility and if it has, I think we should discuss the multitasking functionality more than now. Also the mostly OS have an interrupting facility. Why dose the lisp have not an interrupting facility ? If a mouse event interface is used by an interrupting facility, the interactive provision of Lisp will not lose. And I think implementing an interrupt mechanism is not so difficult.

Second point is a what is going on *Yy* and CLIM. I am principal author of *Yy-client*, so that I have an interest in how to integrate *Yy* and CLIM.

Linden, David: International Lisp Associates, Inc.

Mr. Linden is responsible for coordinating the CLIM Version 2 specification.

Males, Kevin: Harlequin, Ltd

Marks, Jo: Chairman, Harlequin, Ltd

Harlequin has been in existence for over 5 years and has had its LispWorks product on the market for around 18 months. From the beginning there was a recognition of the importance of predicting and following formal and industry standards. LispWorks was therefore developed for Common Lisp (a more controversial decision in Europe perhaps than in the US) running on Unix workstations with X Windows. Problems can arise when no single industry standard is established: the current variety of windowing systems is an example. Supporting several standards can be painful, particularly if one of the them is technically superior, but we have to live in the real world.

LispWorks is aimed primarily at the industrial developer of large, complex systems, so the development of a comprehensive and flexible environment went hand in hand with, or even preceded, the development of the basic system. Meeting the needs of applications developers with a range of new and enhanced tools is still a major component of Harlequin's activity. The implementation of the environment required an efficient object system so LispWorks has had Harlequin's own implementation of CLOS since the product's initial launch. CLOS is used throughout the LispWorks implementation, including speed sensitive areas. The power of CLOS is the most persuasive reason for the use of Lisp but work still remains to be done in areas such as persistence.

However, even if a Lisp system has the most comprehensive and powerful development environment, applications developers are not going to use it unless they know that they will be able to deliver their applications in a compact and efficient form. Lisp vendors have made some progress in this direction, but much still remains to be done.

As an applications developers, Harlequin has always been aware of the importance of integrating with other systems such as other (programming) languages and perhaps more importantly with systems such as existing industry standard databases. Supporting the free interworking of tools developed separately in different software environments is a major challenge but one which will need to be faced if the use of Lisp is to break out into the mainstream.

McKay, Scott: Symbolics Inc., Consulting Member of the Technical Staff

Of major interest to Symbolics are any potential cooperative efforts which may enhance the commercial usefulness of LISP. Of particular interests are projects which enable the effective delivery of applications for which Lisp is particularly suited as a development or delivery language. An example of this kind of effort is CLIM. Symbolics has been instrumental in bringing CLIM 1.0 to the LISP community, both via technical and marketing contributions.

We are interested in seeing the cooperative CLIM process continue. For example, we are very interested in efforts to have CLIM use standard high-level toolkits such as Motif and SunView.

CLIM provides a unique approach to user-interface building, integrating the UI of an application closely with its semantics. It requires an API which is custom tailored to its needs, especially during its current attempt to find broader acceptance in the LISP community. Thus, while we generally applaud standardization and unification efforts, we do not see a great deal of room for CLIM to compromise on its API.

Another area in which we would like to see further work done is the area of reducing the runtime overhead of LISP. In particular, time and space efficient compilation of object oriented applications is needed.

An "Open Symbolic Software Foundation" is an interesting idea, in particular as a vehicle for formalizing efforts to cooperate on improving the commercial acceptance of LISP.

Mori, Robert: Sun Microsystems, Inc.

- What's happening to the CL market?
- What are realistic parameters for CL applications that will be delivered to Unix workstation customers? How big, how robust, how expensive?
- Is it time to establish a CL foreign interface standard?
- Is there a (small) kernel of essential presentation system definitions that could be standardized on?
- Over the last 5 years how many major CL GUI packages have been developed? What does this say about CL programmers?

Muller, Hans: Sun Microsystems Inc. Chief Architect and Developer of LispView(TM)

Norvig, Peter: Research Computer Scientist, UC Berkeley

Research into AI solutions for language processing problems

I am finishing a textbook on advanced Lisp programming for AI applications. I would like to make this book known to the other attendees, and welcome any interaction towards developing on-line interactive educational materials based on the book. Here is a synopsis of the book compiled by my publisher, Morgan Kaufman:

Paradigms of AI Programming: A Common Lisp Approach Peter Norvig

This book teaches AI programming through case studies of nine high-quality AI application programs, plus five AI tools. These are complex reconstructions of significant research efforts, not "toy" examples. The case studies unfold in an incremental manner: first a simplified version is developed, then its limitations are revealed and improved versions are shown.

The book also presents Lips programming techniques at a more advanced level than other texts. The chapter on efficiency is unique in considering both general and specific techniques for increasing performance. Where most texts emphasize writing interpreters for embedded languages, this text presents compilers. The coding style presented is concise and elegant, and does not shy away from advanced features of Common Lisp when they are appropriate. The Common Lisp Object System is given full coverage.

The programs in this book are designed to be clear examples of good programming style–paradigms of programming. They are also paradigms of AI research–programs that use widely-applicable techniques to solve important problems. Readers of this book can expect to come away with an appreciation of some of the major problem areas and techniques of AI, with understanding of some important AI programs, and with an ability to read, modify and create programs using COMMON LISP.

Ohta, Yukio: CEC / Aoyama Gakuin Univ.

I have researched and developed a YyonX that is a portable window tool kit with a *Yy* research group under Prof. Ida since '90.7. And, I have programmed some applications run on GCLisp, as my company CEC is an agency of GCLisp in Japan.

I have programmed on a personal computer before, then I implemented a simple window system that is dependent each machines or each applications. In the personal computer it is used alone, I think that window system function suffice to have a low level interface or a device interface. But in use on network, I have some points to discuss of window tool kit. What is need for an application program interface to be independent of UI. Application programs must assimilate user environment or user interface without an application own look&feel. Well, I have designed an image processing and a simple animation system on YyonX. I am interested in an indication or a manipulation of any commands. The image mechanism works on YyonX now, I will develop an imaging and graphical application program.

Pratt, Dexter: MCC

Rao, Ramana: Member Research Staff, Xerox PARC

Research Scientist in the User Interface area of the System Sciences Lab (SSL)

SSL has a broad range of efforts including artificial intelligence, natural language, innovative applications (hypertext, collaborative systems), user interfaces, participatory design, and language design. We do a fair amount of lisp development, but in the last few years much system-building effort has been moving to other things (C).

Our user interface group currently has two major thrusts: exploring interfaces for visualizing information in the same way that scientist currently visualize scientific data and designing new interfaces that are more appropriate to casual users (for example simplified copier and multifunction machine interfaces).

I am one of the primary designers of CLIM, and was the lead designer of Silica (CLIM's window system interface) and Windshield (the prototype for CLIM's gadget-oriented toolkit).

I am personally (with one hat on) interested in helping with the effort of integrating the CLIM and *Yy* efforts.

I with another hat (based on the needs of my group and SSL) am interested in developing applications in Lisp which can be used by a broader group of people. In particular, we need to be able to deliver robust and efficient applications to our local population of SparcStation 1's and 2's. Two areas that are important to us are the ability to build complex custom user interfaces and lisp application delivery mechanisms (like Allegro Presto and Lucid Distill).

Richardson, Chris: Engineering Manager, Franz Inc.

Rowe, Lawrence A.: Professor, University of California at Berkeley

Building a full-function GUI API is very complex and time consuming.

Common Lisp is big and sometimes slow => impractical for most real applications.

Picasso is wonderful and obviously the answer to all problems :-)!

Sato, Mamoru: Telematique International Research Laboratories and Vision and Image Processing Div. Information Systems Research Center CANON INC. Shin-Kawasaki Office

I am a member of High Quality Color Picture Processing and Picture Coding team in CANON, and I am in charge of retrieval technologies for multimedia (literary) database system in TIRL.

We have experiences of building experimental book retrieval system which is characterized by multi-media user interface using lisp based CG animation. The system presents animated video displays to make the user feel that he/she is in a real library.

We have developed three prototypes on the different computer communication environments. First prototype is stand alone, its main purpose is presentation for such CG animation based services. And it is build on a LISP machine because it has excellent CG production system. Second prototype can actually be used remotely on the LAN environment, and it is consist of UNIX machines because they have color window systems, high speed communication facilities, large memories and so on. In this prototype, CG production system is used only database creation phase. So, the user may feel it is packaged media like CD-ROM or LD. Third prototype, currently developed, can be used both retrieve and modify book databases interactively on the user workstation. This prototype is consist of a LISP machine and a number of UNIX machines. Using this system, book databases could be lively and up to date.

I wish to mention that LISP based object-oriented system suits the interactive multimedia management because they have dynamic behavior. And the cooperation with UNIX or main-frame powers are needed for each (multi)media processing which is simple but vast, and for media synthesis.

I may show a demonstration video at the workshop.

Currently, the *Yy* project and we are cooperated to build our experimental GUI on *Yy*.

Shiota, Eiji: Deputy General Manager, Nihon Symbolics Corpration

I'm also a member of *Yy* project.

I would like to discuss about GUI with Lisp and network multi process environment.

Our company is supporting symbolics computer. And some device drivers (ex: printer) are developing. Also our stuff support and extended Japanese system. I can use CLIM (prototype) and *Yy*. I will use both. And I will make some extend of both.

Shrobe, Howie: MIT & Symbolics

Son-Bell, Mark A.: President, International Lisp Associates, Inc.

ILA is the company that organized the CLIM effort. We continue to coordinate CLIM specification and development activities. At ICERL'91, our staff will present a detailed status report that will cover both the technical and business aspects of the project.

On the technical side, we will discuss our recent experiences with porting the Adapting Toolkit Module of CLIM to Motif and the Macintosh. We will report on the progress we are making in the attempts to unify the two dialects of CLIM, and to help bring CLIM products to market.

We will also be presenting ideas for the unification of CLIM and *Yy*, and for cooperation between the members of both groups.

We are interested in the issue of connecting CLIM to other UIMSes, GUIs, toolkits, and Interface Builders.

Beyond the issue of User Interface, we hope to discuss the larger question of how to establish or improve cooperative activities among the attendees of the workshop and the Lisp community at large. We will present a short history and (self-)critique of the experiences with the process of organizing the CLIM effort.

In conjunction with the other organizers of the ICERL'91 workshop, we will present suggestions for areas of technological cooperation, and ideas for establishing a foundation to pursue such a mission.

We also want to discuss the idea of an international Lisp user group.

Spiers, Tim: Harlequin, Ltd

Tanaka, Keisuke: Research Associate of Computer Science, Aoyama Gakuin Univ.

- one of authors of *YY* Window Tool Kit (YYonX) (principal author for a server part of YYonX)
- administrator for UNIX machines in my organization, and Campus Network in Aoyama Gakuin University.

I have two issues:

One is "Inter Process Communication". We already have many computers which have LAN facilities or Internetworking facilities. This situation causes us to consider networking on the designing of a system. Or we must design/implement a service with consideration of the use on a network. To implement a service on a network, language must support an interface of inter process communication. Many Lisp system have its own interface. But, almost of them are designed to be interface with C or UNIX environment. I think this gives poor IPC performance to Lisp system. We should discuss the better and uniformed interface for IPC on Lisp.

Another is "Distributed Environment". I don't think we can support any service with only LISP or only C/UNIX. Some jobs are suitable to be implemented with Lisp and some are for C or on UNIX, and others may be implemented with other language. The important thing is these services can cooperate with each other to provide an integrated service to users. Because of this, any language should support an IPC mechanism and must have concepts to help the implementation of a distributed system. For example, we must have a integrated concept about a user on a network, or device/computer resources on a network. I want to discuss about this issue.

I am one of authors of *Yy* system. This system is designed on server-client model and it works on the cooperation of a Lisp process and a C process. I think this is one of examples for these two issues.

Veitch, Jim: Director of Engineering, Franz Inc.

Weinecke, Paul: Lucid Inc.

York, William: Managing Director, Software Products, International Lisp Associates, Inc.

Mr. York is responsible for management of all Lisp Software Product development at ILA. He is the principal architect of CLIM and one of its chief designers and implementors. He has served as the CLIM specification coordinator since the beginning of the effort.