Workshop: *Third CLOS Users and Implementors Workshop*

Organizer:
Andreas Paepcke
Hewlett-Packard Laboratories

This year's CLOS workshop served two purposes. The first was to bring users and implementors together for the purpose of exchanging ideas and feedback. The second was to serve the coordination for an upcoming publication that will collect papers about CLOS, covering a broad range of issues connected with the language.

All participants contributed a position paper describing concerns or work done on or with CLOS. These included issues of programming style, applications of the CLOS Metaobject Protocol and other facilities special to CLOS, and suggestions on how the language could be enhanced.

John Collins' contribution is concerned with documenting CLOS code. He identifies three consumers of documentation: maintenance programmers who need an 'internal protocol,' functionality users who need an 'external protocol' and programmers who will extend the system and need a 'specialization protocol.'

Roman Cunis points out in which parts he feels CLOS is not reflective and what could be done to change that.

Scott Cyphers and David Moon explain several of the efficiency mechanisms in the Symbolics CLOS implementation. This includes issues such as object representation, slot access, dispatch, instance creation and the modification of classes and existing instances.

Rick Dinitz, Philip McBride, Hans Muller and John Rose describe how they used CLOS features in their work with Lisp View, a Lisp interface to Open Look and X, which is itself written in CLOS. In particular, they explain their use of multiple inheritance, multimethod dispatch of various kinds, protocols with polymorphism, introspection, class evolution and method combination.

Jiri Dvorak and Horst Bunke link object-oriented programming with rule-based work and explain how CLOS helps with the implementation of the database, general system design and enhanced flexibility.

Bruce Esrig and James Hook introduce static typing into CLOS by figuring out the types of expressions through pattern matching. They show the relationship of this work to ML and point to some optimizations that would be necessary in a CLOS implementation for efficient execution of such a mechanism.

Steve Ford explains that the integration of Common Lisp and CLOS is not complete enough, citing several examples, including the need for Common Lisp functions to be generic. He points to several problems arising when persistence, distribution, and sharing are added to CLOS: the partitioning of the name space by packages is insufficient with persistence increasing the life time and volume of names. Transactions are needed, a sense of location for a computation must be introduced and consistency maintenance beyond CLOS' class/instance consistency become necessary. Other issues pointed to are the need for the concept of set/collection and a portable representation of objects (among CLOS implementations and, more generally, among other object-oriented languages).

Alan Gunderson, Mark Adler and Steve Schwartz describe a CLOS implementation of knowledge representation that is suitable for computer network modelling for reasoning and diagnostics. They use the Metaobject Protocol to introduce relations into the language.

Benoit Habert has used modifications to the generic dispatch to reflect ever narrowing constraints during the parsing of natural language expressions.

Simon M. Kaplan and Alan M. Carroll use CLOS for a cooperative computing environment. They show how they use accessor method specialization, inheritance and dispatch control. They also point out the drawbacks they found in their use of the language and available environments, stressing the lack of an ability to cause only a subset of the :before and :after methods to be run, as well as the need for persistence.

Jonathan A. Pierce and Joshua Lubell suggest a way of specifying constraints on slot values, including constraints between the values of slots in the same or different instances. They also suggest various extensions to the language: a new slot allocation scheme that makes the slot shared for the class, but which causes each subclass to have its private copy to be shared only among the instances of that subclass. They propose a change to allow keyword arguments to 'setf' methods and would like a protocol for managing named instances to accommodate knowledgebases.

Ramana Rao argues that the concept of metalevel architectures is important outside of language development and illustrates this with Silica, a portable window system layer that is part of an emerging standard on Common Lisp user interfacing.

The workshop was kicked off with a talk by Danny Bobrow on CLOS' intellectual history and context. He did this by giving definitions, language comparisons and speculations about possible extensions for various aspects of language design and implementation.

Danny began with a tradeoff analysis of polymorphic operators, pointing out the design dimensions, such as who may influence how polymorphism chooses among alternative behaviors, when selection occurs and whether this selection is based on single or multiple arguments.

He followed the history of factoring code (reusable code segments) from subroutines through the idea of modules to the principle of factoring by object-oriented techniques. He pointed out aspects of reflection in languages, distinguishing between reflection on structure, program and process. Reflection on structure refers to class descriptions and hierarchies. Reflection on program refers to the building blocks of the language, such as methods, and reflection on process addresses issues of how the program works, its flow of control, stack manipulation, etc.

Support for development was another language aspect he addressed. This includes program analysis, the dynamic addition of elements, such as methods, the tracking of changes and 'reconstructability,' such as the saving and restoring of a run-time object collection.

The next major block of the workshop program was chaired by Gregor Kiczales and was dedicated to the Metaobject Protocol (MOP). It consisted of an update on the status of the MOP definition and its documentation, a technical summary of its details and a 'design review and muscle display' which involved all participants of the workshop. Gregor and Jim des Rivieres invited participants to produce language modification suggestions and then showed in real time how they could be implemented using the MOP. One example that was worked on in detail was an additional attribute of slots that would allow a class designer to specify on a per-slot basis which parts of a class should be displayed by a class browser. The solution involved getting the language to accept the new slot option as a natural part of defining a class. Other parts of the solution provided space to store the value of the attribute and ways to access it.

Gregor and Jim handed out two documents that are to be made into a book. The first is the current draft (11) of the MOP. The second is called "The Art of the Metaobject Protocol." It takes a subset of CLOS and develops a pseudo MOP for it. This process is used to make the reader understand the ideas behind a metalevel architecture in a very easy-to-read style.

The first part of the afternoon was taken up by a panel session of CLOS vendors which was chaired by Rick Dinitz. In preparation, a questionnaire had been distributed to both, participating vendors and workshop attendees. Here are some of the results from the questionnaire and the Workshop itself: What people wanted in terms of libraries and frameworks is support for knowledge bases, standard inferencing libraries, user interfaces, communications, database access, a way of sharing libraries and hypercard-like UI building facilities.

In terms of support environment people expressed a need for CLOS browsers capable, for instance, of showing large class hierarchies graphically. This implies multiple windows, scrolling, zooming, filtering, sorting, etc. Browsers should also be conscious of CLOS specifics, such as method specializers to help find the methods one really wants. There was also a desire for query-like capabilities over the class/method structures. Other wishes were for tracing facilities that are aware of the CLOS dispatch mechanisms, an object-oriented 'defsys' facility, ways to traverse data structures at run time and good delivery facilities.

Vendors represented were Lucid, Franz, and Symbolics. They explained some of their own findings of what users want and outlined their emphasis. The importance of the MOP was recognized by several vendors and the lack of a complete definition was recognized as a problem, but people agreed that the specification was not yet ready for freezing.

There was close to unanimous agreement on the importance of performance, particularly in generic function dispatch, instance creation and slot access. The need for easy interaction with other languages was stressed as well. The MOP, it was observed, should consider optimizations more explicitly. Representatives of both Lucid and Allegro said they were working on a kind of 'instance finalization' protocol, which is to be initiated just before an instance is garbage collected.

The final session was moderated by Andreas Paepcke and consisted of three presentations by Alan Gunderson, Jonathan Pierce and John Collins on their position papers.

The Workshop closed with a set of hungry and tired CLOS users and implementors staggering from the room.

Contact information

Andreas Paepcke
Hewlett-Packard Laboratories
1501 Page Mill Rd.
Palo Alto, Ca. 94025
paepcke@hplabs.hp.com