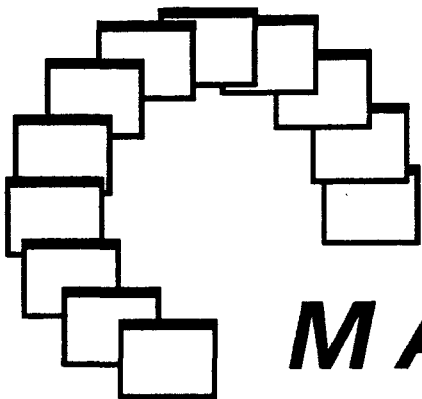




Volume 3, Number 1

January 1987

MASTERSCOPE

Letter from the Editor

As the new Technical Editor, I'd like to take this opportunity to describe the changes that you're going to notice in *Masterscope* starting with this issue.

First, a little history. At AAAI '86 in Philadelphia I expressed concern to various people from Xerox AIS that *Masterscope* issues were so few and far between. The response was that since *Masterscope* was a user newsletter they would really appreciate the users getting more involved in it's production, so I volunteered. Much to my surprise, "getting more involved in it's production" turned out to be the job of Technical Editor, but that's what I get for volunteering ... besides, Xerox AIS didn't completely leave me out in the cold, they've provided me with a very capable Managing Editor in the form of Paula Burke at AIS in Pasadena. Xerox has also agreed to fund *Masterscope* for one to two years, at which point it will become independent. It's turned out to be a lot of fun, too, even if the readers of the INFO-1100 mailing list on Arpanet are fed up with getting requests for articles.

What do the changes mean to you? Beginning with this issue, *Masterscope* will be produced on a quarterly basis -- January, April, July, and October. There's a down side to more issues in that we will require more articles from the users. What I'd like to see is your articles describing how you're using the machines, what you like about the hardware and the environment, descriptions of useful techniques and applications that you've written. Just about anything relating to the Xerox Artificial Intelligence Environment; Interlisp-D, LOOPS, Xerox Quintus Prolog or your local users' group activities.

An important feature of having more user involvement with the newsletter is that we have more input in what goes in it. I'd like to see the newsletter become another tool to help users get work done. Maybe you have found the answer to someone else's problem or you might be able to help someone else with their problem, so your tips for other users, nifty things you've learned, or questions you have are important. If you've got suggestions on new columns or articles you'd like to see, please send them in.

Also, let me know if you like what we're doing. See the section at the end of this issue on how to submit articles.

With this issue we have added a new column from Xerox AIS management. The format of the column will vary with each issue and will highlight products and services and will also include interviews with the Xerox AIS Marketing and Operations Managers. This issue features an article by Marty Raim, Manager of AIS Customer Support describing the new Software Maintenance policy. The April column will describe major differences between Interlisp-D and Xerox Common Lisp. Looking ahead to July, we're planning an interview section. More about the subject and interviewee in the April issue.

Masterscope can easily become a primary method of communication between users as well as a good way of letting the folks within Xerox know what the user community would like. However, I can't make this happen by myself; I need the participation and support of each of you, the users of XAIS products.

Arun Welch
Masterscope Technical Editor

Table of Contents

Letter From the Editor	1
Interlisp-D Articles	
Knowledge-Based Interface to Process Simulation	3
FlagMan86: An Interfacing Spreadsheet	8
Wizard's Shelf	11
Xerox Quintus Prolog Article	
Multiparadigm Programming:: Integrating XQP into LOOPS	18
LOOPS Article	
Computer Communication Network Simulation	20
News and Announcements From Xerox AIS	
AIS Software Maintenance Agreement	22
LOOPS Workshop	24
Users' Group Activities	25
Educational Services Class Schedule	25
Submission Information	26
ARPANET Information	27

INTERLISP - D

Knowledge-Based Interface to Process Simulation

Roar Fjellheim, Einar Dehli, Ulf Wretling, Kjell Tangen, Per-Olav Opdahl

and Odd-Wiking Rahlff

Computas Expert Systems

P.O. Box 410, 1322 Hovik, Norway

1. Introduction

Designing and using simulation models for complex industrial process plants (e.g. oil refineries) is a demanding and time-consuming task, requiring extensive knowledge of both the plant and the simulation system. While the basic principles and tools for process flow sheeting (calculation of steady-state heat and material balances in chemical processes) are well understood (See Figure 1.), there is a definite need for improved man-machine interfaces and support systems for exploiting the powerful flow sheet simulator programs now available.

This article describes a project that develops an intelligent frontend for the PROCESSTM flow sheet program (See Figure 2.). The KIPS front-end (Knowledge-Based Interface to Process Simulation) is being implemented using Loops (See Figure 3.) on the Xerox workstation. The project is conducted under a contract with Statoil, Den norske stats oljeselskap A/S.

While the standard interface to PROCESS is through a text editor and an elaborate command file, the work pattern with KIPS is as follows. The engineer with some initial design ideas starts by building a flow sheet on the graphics screen of the workstation. The design work is supported by knowledge embedded in the KIPS database, which contains basic building blocks and model (sub)assemblies, as well as design rules. As soon as the model has been constructed in an interactive session, the corresponding input file is automatically generated and sent to PROCESS for execution. The advantages are:

- The knowledge involved in designing a model is shared between the engineer and the system. KIPS may serve as a repository for good design practise, and for accumulating experience.
- The engineer is freed from manual handling of drawings, preparation of detailed input files, etc., and may concentrate on high-level issues of flow sheet design instead. The user interacts with the system in terms of familiar process symbols.

2. System Structure

The four major components of KIPS, as shown in Figure 1, are:

1. The objective of the system is to support design of process models, which are input from, and displayed to the user graphically as process flow diagrams, and converted to text form for PROCESS.

2. The process models are created and manipulated through a number of interactive design functions. Foremost among these are a graphical editor for flow diagrams and a PROCESS input file generator.
3. The design work is supported by knowledge bases, where PROCESS concepts and previously acquired model components, design rules, etc. are encoded.
4. The user gains access to the remainder of the system through a uniform user interface, using the familiar window/mouse/icon approach.

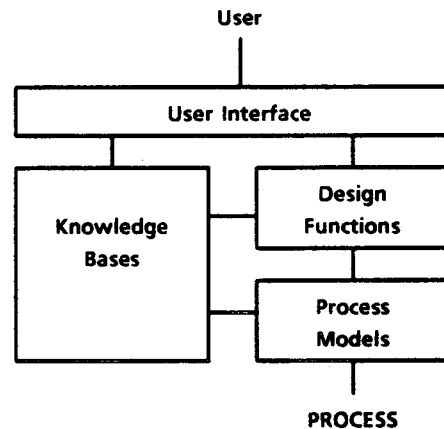


Figure 1. KIPS Main Modules

The knowledge base is a major focus for this work. Constructing a PROCESS model is largely seen as copying pieces of information from the knowledge base into the evolving model, then modifying the pieces to suit the current needs. This instantiation of general descriptions to particular instances is accompanied by activation of design rules for checking the validity of user actions, providing advice upon request, or automatically filling in parts of the model.

3. Knowledge Bases

The backbone of the knowledge bases in KIPS is a LOOPS class hierarchy defining model components, augmented with rule sets as described in more detail below. Figure 2 depicts a central fragment of the class structure:

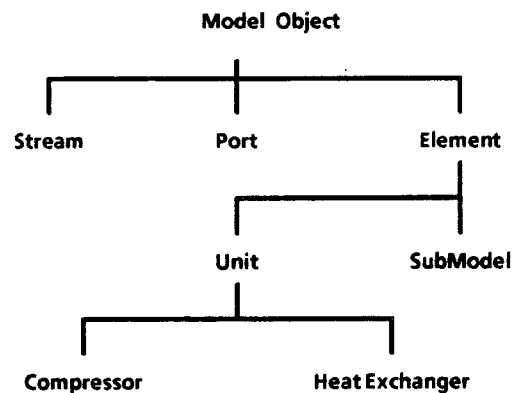


Figure 2. Model Object Taxonomy

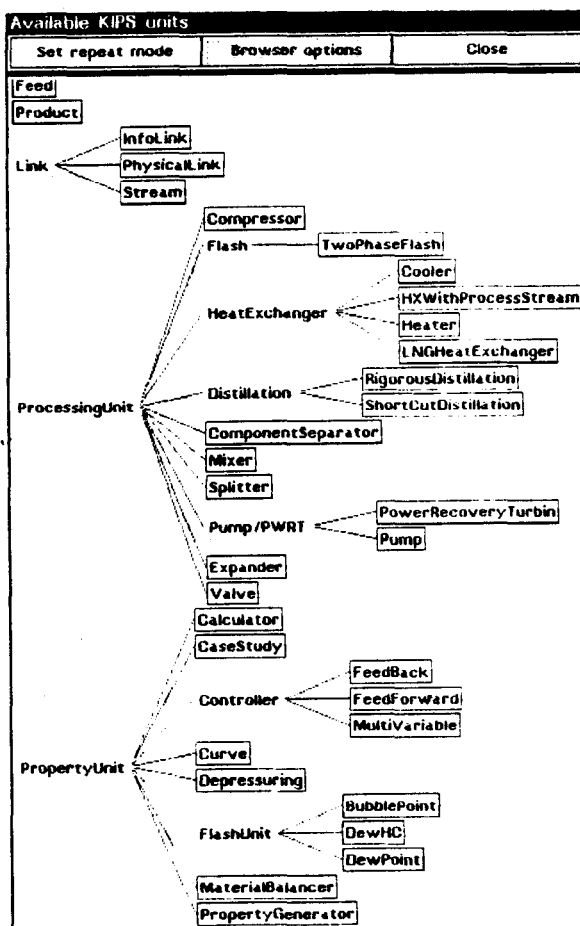
A model contains ModelObjects. A ModelObject is either an Element, a Port or a Stream. The Element is a general class, with subclasses SubModel (i.e. a submodel with its own internal structure), or a basic Unit, of which Compressor, HeatExchanger, etc. are specific subclasses. The units (e.g. Compressor) may spawn further subclass definitions, corresponding to particular ways of using these units (based on experience). Figure 3 shows some of the currently defined PROCESS units. The inclusion of both SubModel and Unit as building-blocks in models means that fully recursive structures are possible.

Each unit has a number of parameters that can be specified to PROCESS. Each parameter is described by an instance variable having a number of properties:

type	parameter type: number, text, reference to unit, etc.
default	default parameter value
unit	the engineering unit in which a numerical parameter is given
range	minimum/maximum values
values	list of legal values
reason	why does the parameter have its current value (a text)
etc.	

A specification of a unit is defined as a particular way of using that unit, minimally defined by partitioning the set of parameters for the unit into required, optional, special, fixed and unused parameters. The specification of a unit can be changed during a session, and the system will change its behavior with respect to that unit correspondingly.

Figure 3. PROCESS Unit Taxonomy



We will now describe a common framework for constructing (usually small) rule sets associated with particular parameters. Each unit parameter description may have associated rule set methods and Lisp expressions of the following kinds:

check Method and/or Lisp expressions for checking validity of a proposed new value.
 advice Method or Lisp expression for advising the user on how to select a new value.

Checks are activated as a side-effect of an attempted update of a parameter value, using the active value mechanisms in LOOPS. If the proposed value is possible but extreme, a warning message is given.

Advice must be explicitly requested by the user.

The check and advice rule sets use a number of specially designed functions and methods that provide easy access to information in any part of the model.

Given the facilities described above, *knowledge programming* in KIPS takes two principal forms:

- Refinements of the classification hierarchy, e.g., defining subclasses of an existing unit class to capture different idiomatic uses of that unit.
- Definition of rule sets associated with particular parameters. In the KIPS system, there are many rule sets, each typically containing less than ten rules.

The maintenance of the knowledge base is entrusted a knowledge base administrator (who may at the same time be a user). Figure 4 illustrates the fundamental work cycle of KIPS: Users build process models by instantiating knowledge base entities, while the knowledge base administrator inputs basic PROCESS definitions, as well as generalizes particular models (model elements) into the knowledge base.

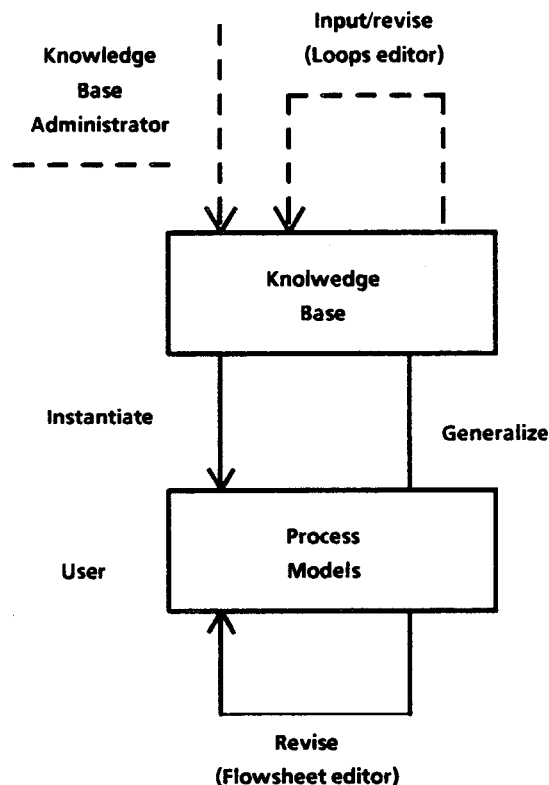


Figure 4. Knowledge Instantiation/Generalization

4. User interface

When a user starts up the KIPS system, the screen contains a number of small icons representing knowledge-bases and flow sheets. By pointing at an icon with the mouse and clicking a mouse button, the icon opens to reveal its contents. Knowledge bases are displayed as LOOPS classification graphs, while flow sheets are diagrams closely resembling those used by process engineers.

A new unit or submodel is placed in a flow sheet by pointing to its name in a knowledge base, and then pointing to a location in the flow sheet where the element should be placed. The unit is moved around by animation. Parameters may be filled in (via *pop-up* menus), and the element connected to other flow sheet elements by pointing. The rule sets come into action during this process, as described earlier. Finally, a menu attached to the flow sheet may be activated for generating an input file for PROCESS or producing hardcopy output.

Hierarchical design is a most important aspect of design processes, and is supported by the general model structure shown in the previous section. Functions for creating submodels, zoom in on the internal structure of submodels, etc. are available to the user. We believe this is a fundamental requirement for any system purporting to support design in a complex domain, such as simulation modeling.

5. Current Status

In early May 1986, a prototype of KIPS is in use by the research and development department at Statoil, mainly for demonstration and testing purposes.

- The graphical flow sheet editor is operational but further enhancements are under development.
- PROCESS units regarded as relevant for offshore simulations have been incorporated in the knowledge base by "tapping" experienced process engineers.
- PROCESS is a sequential simulation program and a precedence ordering of the units is suggested to the user, using Gundersen's algorithm [5].
- A PROCESS input file is automatically generated and transmitted to a host computer (an IBM main frame) and the simulation results are transmitted back to the Xerox workstation.

6. Further work

A major motivation behind the KIPS project is finding ways of including design knowledge in the intelligent interface to PROCESS. This is clearly a vast topic, and we have chosen to approach it in an incremental fashion.

One way of viewing the organization of process modeling knowledge is related to the hierarchical structure of models:

Process structuring: On the highest level, the overall structure of the process is selected. Relevant constraints are the goals of the process (e.g. in terms of external feed/product properties), previous design experience, safety standards, etc.

Submodel/unit selection: Given the overall model structure, the next step in the refinement is to choose particular submodels/units to fill the structure.

Parameter selection: Finally, the parameter values of the individual units must be set so as to achieve the desired role of the unit in the overall process.

Logically, the design should proceed top-down, and the reasons for decisions at different levels should be maintained for later project phases. Unfortunately, this is seldom carried out in practise, thereby losing a major source for making informed model revisions.

Another possible direction for the KIPS project is to extend the present concentration on the model design aspect to also include other phases of simulation projects:

- Planning (sequences of) simulation experiments to achieve given project goals.
- Qualitative/coarse simulation in the front end to weed out first-line problems.
- Fault diagnosis of simulation if the results are wrong or non-satisfactory.
- Evaluation, interpretation and presentation of the simulation results.

References

- [1] A.W. Westerberg, et al. : Process Flow Sheetting, Cambridge University Press, 1979.
- [2] N.F. Brannock, et al. : PROCESSM Simulation Program, A Comprehensive Flow Sheetting Tool for Chemical Engineers, Computers & Chemical Engineering, Vol. 3, pp. 329-352, 1979.
- [3] Daniel G. Bobrow, et al.: The LOOPS Manual, Xerox Corporation, 1983.
- [4] M. Stefik, et al.: Knowledge Programming in LOOPS, AI Magazine, Fall 1983.
- [5] T.Gundersen & T Hertzberg: Partitioning and Tearing of Networks - Applied to Process Flow Sheetting, Model, Identification and Control, 1983, PP. 139 - 165.

FlagMan86 : An Inferencing Spreadsheet

E.D. Anderson
Expert Systems Laboratory
Office of Interdisciplinary Programs
Georgia Institute of Technology
Atlanta, Georgia 30332
(404) 894-2423

The Expert Systems Laboratory was established in mid 1986 at the Georgia Institute of Technology in Atlanta to apply current Artificial Intelligence (AI) technologies to solving commercial problems, typically in office settings.

The Laboratory contains a variety of engineering workstations (Sun-3, Masscomp, MicroVax-II, and an IBM/AT), as well as a TI Explorer and the Xerox 1186 AI Workstation. The 1186 is of particular interest because of its PC/DOS emulation board and window in which standard business software applications may be run. The Lotus Development Corporation spreadsheet products (1-2-3 and Symphony) are of special concern to the laboratory staff.

In the current business environment, microcomputer supported spreadsheets, such as those provided with the Lotus 1-2-3 and Symphony software packages, enable novice users to complete powerful numeric processing without detailed relational database skills. This phenomenon has lead boardroom level executives to handle problems "personally," ranging from the detailed manipulation of budget proposals to the analysis of capital expenditures. It has also empowered the mid-level manager with opportunities to do detailed cost accounting, marketing analyses, project management, and related data and analysis-intensive tasks on a low-cost microcomputer without traditional data processing support.

FlagMan86 is designed as a demonstration of general expert systems capabilities using the combined analysis features of a Lotus 1-2-3 spreadsheet embedded in the Interlisp-D environment. A "Spreadsheet Scanner Button" is also used to accelerate the data transfer between the spreadsheet and the underlying expert system. The name - FlagMan - is based on the concept of an active agent watching the project manager as he or she designs a new software engineering project schedule and costs out the related manpower and computer services. The FlagMan knows key ratios drawn from similar project histories and "raises the flag" on any ratios within the planned project which may be "out of line" with prior experiences. FlagMan is part of the Georgia Tech Information Management System (GTIMS) designed to provide decision support to Tech project managers in both the planning and monitoring stages of contract software development.

The original concepts behind the inferencing spreadsheet were explored in 1984 on a PC-class microcomputer using Lotus 1-2-3 and a small lisp interpreter (lQLisp) in high memory over the spreadsheets for project management and monitoring. Numeric data within the sheet were first transformed by Lotus macros into Lisp S-expressions and then written to a transfer file. Next, control jumped to the Lisp in high memory for inferencing using a simple forward-chaining engine. Throughout, design emphasis has been placed on the knowledge base itself. This knowledge base represents an evolving policy instrument of the Georgia Tech Research Institute (GTRI) and as such is under constant refinements and challenges.

In 1985, with growing institutional interest in expert systems applied to generic business problems (including project management), the inference engines were expanded into both forward and backward chaining modes using an XT-class microcomputer with 640KB of dynamic memory and Golden Common Lisp. In this implementation, the Lotus 1-2-3 spreadsheet was opened in high memory above the larger Lisp interpreter.

It is important to note that the original knowledge base of approximately 50 rules was ported intact between the two micro-Lisp environments by first porting the inference engines between Lisp dialects and then "popping" the knowledge base between the environments virtually unchanged. The new engines accepted the old rule syntax without further alterations. This set the stage for the next quick portage of the knowledge base from Golden Common Lisp on the microcomputers onto the Xerox 1186 AI Workstation through the PC/DOS emulation window.

In early 1986, the same inference engines were ported using the Programmer's Assistant and MasterScope into the Koto Release of the Interlisp-D environment. The original knowledge bases, preserving original keystrokes, were then both read from the original disks using PCE file utilities, and were also shift-snapped through the PC/DOS Emulation window. Next a Spreadsheet Scanner was implemented, with the technical assistance of the Atlanta Xerox Artificial Intelligence staff, to pull numeric data from the face of the spreadsheet under the control of a mouse-sensitive button. The

button attaches itself to the upper lefthand corner of the Lotus 1-2-3 spreadsheet. (See the attached bitmap at the end of this article.)

A Rules-Interpreter window is bound to the right side of the sheet. Here the captured numerics are first re-displayed in list format and then an "Understand" function applying a series of "Consider" sub-functions transforms the information into symbolic form -- short phrases in common business English. A typical transformation might be from \$45,000 to (This project has a fairly scant funding level.)

Research into common business English is only beginning in cooperation with the Georgia Tech Department of English (COSALS). Here the attempt is to write knowledge-base rules in such a manner as to be both understandable and operational in a business environment. An example from Flagman86 states "Plan is OK but can be improved. Sponsor should be asked to extend the time frame by three months before project contract can be signed." This technical approach places an emphasis on maximum use of the expressive power of symbolic computation on an AI workstation and its ability to communicate directly with human users on their own terms. The numbers in the embedded spreadsheet begin literally to speak English to the expert system user in real-time on the Xerox 1186.

The common business English facts derived from the current numbers of the spreadsheet are then passed as a "facts-stream" to one of the two inference engines for processing. The resulting analysis may be sent to either the right side window where rule firings appear one at a time, or to a "FastTalk" window immediately above the spreadsheet where the inference process runs to completion presenting the final analysis in an uninterrupted flow.

Current interest in the spreadsheet inferencing capabilities of the Xerox 1186 AI workstation are leading us to investigate another application beyond software project management -- the auditing of travel expenses in an office setting. Here the techniques built up gradually with the micro-based FlagMan and the AI Workstation version of Flagman86 are leading to rapid prototyping of a new spreadsheet inferencing system in a matter of months, rather than years.

To begin the design of a knowledge base for a new office, a travel expense summary spreadsheet is first opened on the 1186. Records of traveller expenses have already been entered in designated cells of the spreadsheet using a PC-class microcomputer and then sent to the audit staff in hardcopy format. Their interview-derived comments are blended with the policy statement of top management concerning current travel practices. Finally, using the 1186, the rules derived from these policies and interview findings are entered into the knowledge base underlying the spreadsheet. Now the 1186 supported inferencing spreadsheet is ready for designer testing, audit or review, and employer feedback for insights into the planning of future trips and the management of audit exceptions that may have been discovered.

The cost-effectiveness of the new travel advisor application is based on the re-use of the previously developed inference engines and knowledge base structures. On the Xerox 1186 AI workstation the PC/DOS Emulation window with the Spread Sheet Scanner Button are proving to be key factors in the rapid development cycle of the new inferencing spreadsheet and in the extension of the application commercially.

WIZARD'S SHELF

Tips, Tweaks and Miscellaneous Information

This section is devoted to interesting pieces of code, patches to bugs, and requests for the like. It's also for software questions that users want to ask other users. While I've tried to test most of the submitted code, there's no guarantee that it will work on your system. Some of the tips might be unorthodox, or cause your system to crash.

The "rules" for this section are as follows:

1. If you submit software or hardware patches, don't submit things that are copyrighted, untested, or that you know will cause systems to behave strangely.
2. If you are able to answer any of the questions that someone has, then feel free to send me the answer, and I'll feed it to the requester, and include it in the next issue.
3. Neither Xerox, nor I, nor the original authors take any responsibility for what these bits of code do to your system (I have to put this in, obviously). The understanding, however, is that people who submit things for this section will make an attempt to debug them, since they would like others to do the same...
4. If you've got questions you'd like included in the next issue, send them to me. See instructions at the end of the newsletter on how to send them to me. If you've got software that you'd like to make available to other people, this is also the place for it. Send a short description of the software, along with instructions for obtaining it. Note that this isn't a form of free advertising, but rather a way of distributing knowledge among users, so I'm going to have to restrict this to public domain software (software that you only charge for distribution costs, not for software costs).

❑ **Inspect within DEdit** From Nigel Gilbert

For those like me who would like to add an "Inspect" item to the DEdit menu, this is a recipe for doing it. The code which follows also installs Henry Thompson's DEDITK patch.

You need a function, DEditInspect:

```
(DEditInspect
[LAMBDA NIL                                     (*gng: 21-Nov-85 10:39)
  (* * DEDIT Method for inspecting the value of the selected
    form. Called on buttoning Inspect in DEDIT EditOps)
  (SETCURSOR DEFAULTCURSOR)
  (CLEARW PROMPTWINDOW)
  (LET ((it (TOPSELECTION)))
    (if [SETQ it (NLSETQ (EVAL (CAR it]
```

```

      then (INSPECT (CAR it))
    else (printout PROMPTWINDOW T "Cant: Has no value" T]))

```

and filecoms, DEDITINSPECTCOMS:

```

((FNS DEditInspect)
 (FILES DEDITK.DCOM)
 (P (* ensure that the Inspect menu item is not added twice)
  (OR (ASSOC (QUOTE Inspect)
            MYD)
       (EV MYD (INSERT (Inspect DEditInspect)
                        BEFORE Break))))
 (DEDITK)))

```

Then MAKEFILE(DEEDITINSPECT) and LOAD(DEEDITINSPECT).

Buttoning Inspect on the DEDIT menu will inspect the value of the selected item, or show an error message in the Prompt window.

❑ Printing bitmaps in the lower left corner

From Chris Lane, Stanford University

To print your bitmap in the lower left hand corner (or anywhere on the page), you might try the following:

```

(SETQ STREAM (OPENIMAGESTREAM '{CORE}BITMAP.IP 'INTERPRESS))
  (BITBLT (WHICHW) NIL NIL STREAM) (* Or any BITMAP ... )
  (SEND.FILE.TO.PRINTER (CLOSEF STREAM) printer)

```

This works on Interpress and Impress printers and should work for Press printers.

❑ CTRL-U patch

From Matt Heffron, Beckman Instruments

Since KOTO has de-implemented the feature where typing a CONTROL-U while typing caused the input to be given to the editor (EDITE) before being returned, and has made CONTROL-U do the same thing as CONTROL-Q (delete current line), I tried to find a work-around. It turns out that the global variable CTRLUFLG is the indicator for TTYIN to call EDITE on the input expression, so

```

(INTERRUPTCHAR (CHARCODE ↑U) 'CTRLUFLG)

```

makes ↑U ALWAYS set the CTRLUFLG, even if typed while under another reading function (e.g. RSTRING). So, it turns out that \INSIDE.TTYIN is T while in TTYIN, so the interrupt can be changed. First define the function:

```

(DEFINEQ [CTRLUINTERUPT NIL
  (DECLARE (GLOBALVARS CTRLUFLG)
    (USEDFREE \INSIDE.TTYIN))
  (COND ((AND (BOUNDP (QUOTE \INSIDE.TTYIN))
              \INSIDE.TTYIN))

```

```
(SETQ CTRLUFLG T))
```

then set the interruptchar:

```
(INTERUPCHAR (CHARCODE ↑U) '(CTRLUINTERUP))
```

this will make the ↑U interrupt effective ONLY when under TTYIN.

❑ Instruction and Init File

From Larry Masinter, PARC

Anyone teach courses in a university setting using 1108's or 1186's. Also, undergraduate or technical seminars. I will summarize the responses and send the summary to all who contribute.

Also, INIT files, what's in 'em? Favorite hacks, must-have LispUsers, etc.

[Ed. Note: If we get lots of neat init files, I'll put a synopsis into the next issue. It's also a good idea to set up a pool of instructional material, since the learning curve for Interlisp can be high.]

❑ 2-Button Mouse Fix

From Arun Welch, Ohio State

The 2-button optical mice have a problem in that the buttons travel too deep into the body of the mouse and then break. This is a known problem, mentioned in ISD Field Service Bulletin # 65017, Jan 1986 (you probably don't receive these, as they are hardware service bulletins, but your Service Rep. does). There are a couple of ways to fix it, the official way, the slightly official, and the unorthodox but still works way.

1. Official way: Call up your Service Rep., and ask him to bring you 2 plastic standoff's, part number PN 14P80522, and have him install them as described in the bulletin.
2. Slightly less official, but still official: Order the above parts, and install them yourself as follows:
 - a. Unscrew the mouse cover.
 - b. Using 2-sided foam tape, position the plastic standoff's so that they will block the mouse button travel. This is approximately 1/2 cm to the left of the little notch, and 1/2 cm to the left of the right border.
 - c. Screw top back on.
3. Unorthodox:
 - a. Unscrew the mouse cover.
 - b. Using a reasonably sharp object (swiss army knife in my case), cut a chunk of an office eraser so that it will prevent the mouse button from traveling too far (this will probably take a couple iterations).
 - c. Similar to (b) above, but substitute chunks of eraser for plastic standoffs.

What prompted this was the fact that my mouse just broke, and I couldn't be bothered with calling out the Service Rep. to fix it. Obviously, I take no responsibilities for anything that the above would do to any warranty/service agreements you might have with Xerox. In my case, the machine is under

self-maintenance anyway, and a new mouse costs \$320 (though the body is probably less). As long as you don't use a conductive material for the standoff's there really doesn't seem to be much damage you could do.

❑ FTP Protocol Preference

From Christopher Schmidt, Stanford University

If you use servers that support both PUP LEAF and TCPFTP and you use both protocols, but would like to control Lisp's preference, you may find this hack handy.

If you prefer LEAF to TCPFTP you'd load the patch and evaluate

```
(SET.FDEV.PRIORITY '(LEAF TCPFTP)).
```

Normally SET.FDEV.PRIORITY will prompt you to mouseconfirm each host as its device is deleted from the protocol-instantiated cache. You can suppress this interaction by supplying a second argument of T.

Filed on SUMEX-AIM.Stanford.edu in

<LISPUSERS.KOTO>FDEVPRORITYPATCH.LSP and .DCOM. Compiling requires FDEV record defined in EXPORTS.ALL or <LISPUSERS.KOTO>KOTO.RECORDS.

P.S. Another way to achieve this goal is to register all your hosts with different PUP and IP hostnames.

```
(RPAQQ FDEVPRORITYPATCHCOMS ((FNS SET.FDEV.PRIORITY)
                                (GLOBALVARS \FILEDEVICES)))
(DEFINEQ (SET.FDEV.PRIORITY
          (LAMBDA (DEVICENAMES QUIETFLG)                                (* cvs "23-Jul-86
22:48")
```

(* This function arranges the order of FDEVs on \FILEDEVICES so that protocols will be preferred in the order given. E.g. (SET.FDEV.PRIORITY (QUOTE (LEAF TCPFTP))) will cause hosts supporting both protocols to be accessed via LEAF in preference to TCPFTP.)

(* The first loop removes all FDEVs for hosts instantiated for any of the given device types.) (* This code presumes that the name of the OPENFILE method for a device includes the name of the type of device.)

```
(for NAME in DEVICENAMES
do (for FDEV in \FILEDEVICES when (STRPOS NAME (fetch OPENFILE
                                                    of FDEV))
do (if (OR QUIETFLG (MOUSECONFIRM (CONCAT "Delete FDEV for " NAME
                                           " device "
                                           (fetch DEVICENAME
                                           of FDEV)
                                           " ?"))
        NIL PROMPTWINDOW T))
then (BREAKCONNECTION (PACK* NAME '#LEAF))
      (\REMOVEDEVICE FDEV))))
```

(* The second loop iterates over the requested devices deleting and then adding each. Each addition goes onto the head of \FILEDEVICES so we process the devices in "reverse" order.)

```
(bind FDEV for NAME in (REVERSE DEVICENAMES)
  when (SETQ FDEV (for FDEV in \FILEDEVICES
    thereis (EQ NAME (fetch DEVICENAME of FDEV))))
  do (\REMOVEDEVICE FDEV)
  (\DEFINEDEVICE NAME FDEV))

(* Finally, return the names of
  all devices in their new order.)
(for FDEV in \FILEDEVICES collect (fetch DEVICENAME of FDEV))) )
```

❑ Saving Graphs

From Anna Zilberg, Xerox AIS

If you created a graph that has either {FONTCLASS} or {FONTDESCRIPTOR} datatypes as its NODEFONT values, you cannot save it.

When you SHOWGRAPH of a loaded file it will break on FONTCREATE. Saving the graph as UGLYVARS won't help. These datatypes for NODEFONT are created via one of the following:

1. Using LAYOUTSEXPR to create a graph or adding new nodes in EDIT mode of SHOWGRAPH while DEFAULT.GRAPH.NODEFONT is set to NIL (initial setting).
2. If DEFAULT.GRAPH.NODEFONT is set using (FONTCREATE) and NODEFONT was not specified explicitly.
3. You used LAYOUTGRAPH and specified NODEFONTs using (FONTCREATE).

Workarounds for this problem are:

1. Preventative
 - a. Either (SETQ DEFAULT.GRAPH.NODEFONT '(Family Size Face)) which covers cases 1 and 2, or
 - b. Specify NODEFONT argument to NODECREATE as '(Family Size Face) - which covers case 3.
2. Fix for existing graphs:

Evaluate:

```
(for NODE in (fetch GRAPHNODES of graphname) do (replace NODEFONT of NODE
  with '(Family Size Face))).
```

After that you may do SHOWGRAPH on the "repaired" graph, save it again etc.

❑ Eliminating "Window Image" When Printing Windows

From Josh Stern, Xerox AIS

The function HARDCOPYW, which prints out windows, contains a line of code that looks like:

```
(PRINTER.BITMAPFILE (OR FILE (PRINTER.SCRATCH.FILE)) PRINTERTYPE
```

```
BITMAP SCALEFACTOR REGION ROTATION "Window Image"))
```

Even though "Window Image" is a constant, we can ADVISE the code to change it, without modifying the source! Will Snow of Software Support gave us the following:

```
(ADVISE '(PRINTER.BITMAPFILE IN HARDCOPYW) '(DEFAULTWINDOWNAME))
  (DEFINEQ (DEFAULTWINDOWNAME NIL (SETQ TITLE "MY WINDOW")
    (* needless to say, you can change this to anything you want)))
```

This works because ADVISE wraps around its target, either BEFORE or AFTER it is invoked. The default is BEFORE - since Will's code specified neither, that's what we got. All you have to know is the name of the arguments, as ?= shows them to you.

You can DF an ADVISE'd function to see what condition it is in. In our case, when we do the ADVISE shown above, the value returned is

```
(PRINTER.BITMAPFILE-IN-HARDCOPYW), so that's what you DF:
(DF 'PRINTER.BITMAPFILE-IN-HARDCOPYW)
```

Try it! You can (UNADVISE '(PRINTER.BITMAPFILE IN HARDCOPYW)) to restore things to normal.

[Ed. Note. --- This also is a pretty good example of using ADVISE!]

❑ Meltdown

From Larry Masinter, PARC

This function is an IDLEFN, which makes the screen look like someone took a blowtorch to it.

```
(DEFINEQ (MELTDOWN
  (LAMBDA (W)
    (BITBLT (WINDOWPROP W (QUOTE IMAGECOVERED)) NIL NIL W)
    (INVERTW W)
    (LET ((WID (WINDOWPROP W (QUOTE WIDTH)))
          (HGT (WINDOWPROP W (QUOTE HEIGHT))) MAXW MAXH X Y)
      (SETQ MAXX (DIFFERENCE WID 200))
      (SETQ MAXY (DIFFERENCE HGT 200))
      (do (BITBLT W (SETQ X (RAND 0 MAXX))
                  (SETQ Y (RAND MAXY))
                  W
                  (PLUS X (RAND -1 1))
                  (PLUS Y (RAND -1 1))
                  200 200)
          (BLOCK))))))
```

Then

```
(LISTPUT IDLE.PROFILE 'DISPLAYFN 'MELTDOWN)
```

□ TeX DVI Program

From David Neves and Terry J. Killpack

Does anyone have a TeX dvi view program for the 1100's?

□ Patch for Courier Server, From LispUsers

From Christopher Lane

Since the LispUsers packages for Koto are finally out, I thought I'd re-broadcast the following patch file which we still find necessary if you wish to use the Interlisp-D Courier Server under the current Koto release. Without the patch you will get a "Host not responding..." error when you try to open a Courier connection (or even echo for that matter) to another D-machine.

The contents of patch file:

```
(FILECREATED " 6-Feb-86 14:07:34"
  {XDE:MCS:STANFORD}<LISPUSERS>KOTO>LOOKUPSERVERPATCH.;1)
  (PRETTYCOMPRINT LOOKUPSERVERPATCHCOMS)
  (RPAQQ LOOKUPSERVERPATCHCOMS [(ADVISE LOOKUP.NS.SERVER)
    (P (UNMARKASCHANGED 'LOOKUP.NS.SERVER))
    (PUTPROPS LOOKUP.NS.SERVER READVICE
      [NIL (AFTER NIL (\PUTBASE !VALUE 5 0))
        (READVICE LOOKUP.NS.SERVER)
        (UNMARKASCHANGED 'LOOKUP.NS.SERVER)
      ]
    )
  ]
  STOP
```

XEROX QUINTUS PROLOG

Multiparadigm Programming: Integrating XQP into LOOPS

Tim Koschmann
Xerox AI Systems
Des Plaines, Illinois

LOOPS with its objects, active values, programmer's interface and gauges is highly regarded as a prototyping environment. Implementing knowledge-based applications in LOOPS, however, is sometimes less than convenient because LOOPS lacks elaborate facilities for performing tasks based on deductive retrieval. Prolog has built-in facilities for doing deductive retrieval but lacks the kinds of features found in LOOPS for supporting object hierarchies, displaying data and capturing user input.

In the Xerox AI Environment, applications can combine object-centered and logic-based paradigms within a common project. This proves to be quite useful because, given their respective strengths and weaknesses, LOOPS and PROLOG serve to complement each other very well. Logic programming, as provided in Xerox Quintus Prolog, can be used in place of rule-based programming in LOOPS.

As example of an application, consider a typical expert system designed to be used as a diagnostic consultant. The system interface could be implemented rapidly using gauges and/or browsers in LOOPS. The knowledge base could be represented as Prolog facts and rules. Development of a backward chaining inference engine would be a trivial undertaking in Prolog. The "working memory," that is, the part of the system that models the entities being reasoned about, could be implemented using LOOPS objects. Examples exist of systems with architectures similar to the one described above.

Building the Sysout. In order to use LOOPS and Xerox Quintus Prolog together, a sysout must be built that contains both systems. Starting from a XQP 1.5 Release Sysout for the 1109 or using the 1186 Beta Release Sysout, one must load the appropriate packages to support LOOPS. This is done in the usual fashion using the LOADLOOPS function. After loading LOADLOOPS, the variable LOOPSFILES can be modified to prevent the packages that support the LOOPS rule language (e.g., LOOPSRULES, LOOPSRULESC, LOOPSRULESD, and LOOPSRULESP) from being loaded when LOADLOOPS is executed. This helps to keep the size of the sysout down. The full sysout requires about 6Mb before Prolog expansion.

Accessing LOOPS Objects. The Prolog programmer needs just three predicates in order to access and manipulate LOOPS objects. The predicates are as follows:

```
get_value(Object,Slot,Value).
put_value(Object,Slot,New_Value).
```

```
send__message(Object,Selector,Result,ArgList).
```

The `get__value` predicate obviously retrieves the contents of an object's slot, `put__value` will stuff a slot and `send__message` will direct a message to a LOOPS object. A fourth predicate called `get__object` will recover a LOOPS object given its name. These and some related predicates can be found in the file `LOOPSACCESS.PL`, which can be requested from the XQP Support Group. As an example of the use of these predicates consider the following procedure for determining whether an instance with name `ObjectName` is a member of the class with name `ClassName`:

```
member__class(ObjectName,ClassName) :-
  get__object(ObjectName,Object),
  get__object(ClassName,Class),
  send__message(Object,'InstOf!',Result,[Class]),
  !,
  Class = Result.
```

The two `get__object` terms fetch the LOOPS objects associated with the `ObjectName` and the `ClassName`. The message sent has the same effect as issuing the following command in Lisp:

```
(__Object InstOf! Class)
```

As described in the LOOPS documentation, this call will return the class object if `Object` is a member of `Class` and `NIL` otherwise. The procedure therefore succeeds if the result returned by the message-sending call is equivalent to the class. The `cut` is used to make the procedure determinant.

Invoking the Prolog Rule Set. The means by which a Prolog database can be accessed from a Lisp program is described in the Xerox Quintus Prolog System-Dependent Features Manual (section 1.7). The form of the call is:

```
(PROLOG predicate__name predicatearguments WAITFLG).
```

When replacing LOOPS rule sets with calls to Prolog, a function call of the above form can be used in place of the message-sending call that invokes the LOOPS rule set.

LOOPS

Computer Communication Network Simulation

Gary J. Minden

Telecommunication and Information Sciences Laboratory
The University of Kansas
Lawrence, Kansas

The Telecommunication and Information Sciences Laboratory (TISL) at the University of Kansas is extensively involved in the modeling and simulation of communications system. Such systems range from modeling modems communicating over phone lines to high-speed modems over satellite channels to fiber optic systems. TISL has used a number of software packages for link simulations and more recently, has developed a new Block Oriented System Simulator (BOSS) for communication system simulation at the link (channel) level.

During the late Summer of 1985 a need arose to implement a prototype computer communication network simulator within a three month period. Our goals for the simulator were:

- To develop a network simulation structure
- To develop a protocol design and test tool
- To provide an interactive design environment, and
- To demonstrate a packet switched network simulator.

Earlier simulators required the network designer to describe the network in a general purpose textual, programming-like language. However, if you ask a designer to describe a network, he invariably draws a picture. Therefore, we felt a graphical approach was natural for the designer. We chose to implement the prototype simulator in the Interlisp-D/LOOPS programming environment because we had already begun initial work in LOOPS. The interactive nature of the programming environment and the expected productivity of Interlisp-D/LOOPS were well suited to the task.

The computer communication network simulator that we implemented models packet-switched computer networks. We separated the simulator into three areas: model graphics, and control following standard practice. Four LOOPS classes model the behavior of general communication devices, a routing switch, a communication link, and a host computer. The routing switch implements either a flooding or fixed-routing algorithm. In parallel with the four communication classes are four graphic classes that provide a means to define and display the network. Finally, there are five classes that provide overall control and structure to the computer network. These classes include a packet handler to define and allocate data packets, a task master to manage the discrete event simulation, a graphic window to display

the entire network and provide menu support, a matrix viewer to display and edit matrices, and a network operator to combine the above into a single entity. The classes in the prototype simulator were structured in a manner to be useful in further programming. The prototype simulator can handle networks of 15 to 20 nodes and approximately 45 links.

The designer uses the network simulator by opening a network window. The designer places switches, hosts, and communication links using pop-up menus and the mouse. We use the Interlisp-D inspector to change the characteristics of the communication instances. The most important function of a network switch is routing packets to their destination. Routing is defined by the system identifying the current switch, the current destination and indicating possible next switches. The designer selects one or more "next switches" for the packet, each hopefully closer to the destination. Once the network is defined, the designer can specify disturbances (a link breaking or a transient in traffic) for the simulation. Secondary routes from the switch to the destination make the network robust with respect to these disturbances. The simulator is then initialized and the simulation started.

The simulator is a standard discrete event simulator. During initialization each communication instance is called upon to initialize its own future events. This normally consists of scheduling a check of the queue associated with the instance, perhaps doing some work and scheduling a new event in the future. Event scheduling is handled by a class called the TaskMaster. The TaskMaster schedules events based on simulation time. An event consists of time, a function to execute (usually " $\leftarrow$ ") an instance and an argument list. When an event is scheduled, it is placed in an ordered queue based on time. The TaskMaster simply removes the first event from the queue, sets the current simulation time from the event time and executes the specified function.

While a simulation is executing, the designer sees activity at each switch and link in the network window. As queue lengths change at each switch and link, a vertical bar in the diagram indicates the current length of each queue. This is implemented with standard LOOPS active values. While this provides immediate feedback to the designer, it significantly slows the simulation.

The object-oriented structure of the simulator makes it simple to evaluate alternative approaches to network design. To evaluate a different network algorithm, it is only necessary to add or modify a method within a communication class. The network then takes on this new behavior. We are presently working on techniques to make this easier for network designers.

The prototype simulator has served its purpose as a prototype and we are presently designing a comprehensive network simulation tool.

NEWS AND ANNOUNCEMENTS

FROM XEROX AIS

Xerox AIS Support Maintenance Agreement

Marty Raim, Manager
Xerox AIS Customer Support

As you know, as of November 15, 1986 Xerox Software Support services are available only to customers who have purchased an annual Software Maintenance Agreement. Services are provided under the terms of the 90 day system warranty and include:

- Software Updates
- Major Software Releases
- Toll-Free Hotline Support

Contact your local Xerox AIS Sales Representative if you have not yet purchased your Software Maintenance Agreement.

The goal of Customer Support is to sustain user satisfaction with AIS products. This is an ongoing, information-intensive campaign to provide accurate information to the Interlisp-D and Xerox Quintus Prolog user community. Support services are provided under the terms of the 90 day system warranty, and thereafter by annually renewing a Software Maintenance Agreement.

- We endeavor to support the customer in his use of our equipment to the extent that his expectations, at time of purchase, are realized if not surpassed.
- We provide a quality check on all new products by administering -- and participating in -- an aggressive beta test.
- We support our customers by providing information. We are concerned with the accuracy of that information, its appropriateness to the customer's needs, and the timeliness with which it is delivered.

Contacting Support

To contact Customer Support, call toll-free in the continental United States (800) 228-5325, or in California (800) 824-6449. The Hotline is open every business day from 7:30 a.m. to 4:30 p.m. PST.

Customer Support can also be reached via electronic mail. Our net address is AISupport@Xerox.com. We will soon be inaugurating an MCI Mail link, which will enable any customer to correspond with us directly.

Problem reports and questions can be submitted in writing to the following address: AISupport, Xerox Artificial Intelligence Systems, Mail Stop 432, P.O. Box 7018, 250 North Halstead Street, Pasadena Ca 91109.

Please specify your Support Identification Number each time you contact Customer Support. If you don't know your number, simply contact the Hotline, specify your name and affiliation, and a number will be provided. Callers without a valid Software Maintenance Agreement will receive a 30 day grace period during which Hotline support will be provided.

When you reach us over the Hotline with a question or a problem, we will endeavor to provide an answer "on-the-spot". If your question requires lengthy analysis, it will be assigned a unique 9-digit code and queued for processing by the first available support person. You may contact Customer Support at any time to inquire about the status of your problem, simply by providing the 9-digit code. We strive to solve most customer problems within one business day, though no guarantees are given. If your problem requires extended analysis by our engineering staff, response could be delayed by a week or more.

Currently Supported Products

- The Intermezzo release of Interlisp-D on the Xerox 1108, 1109, 1132, 1100
- Xerox Quintus Prolog on the Xerox 1109 and 1186
- TCP/IP client software on the workstation
- Xerox NS client/server software
- RS232 support
- Busmaster on the Xerox 1109

Customer Services

- Programming advice

We provide answers to technical questions about Interlisp-D and Xerox Quintus Prolog. Customer questions fall into three categories: (1) Advice on how to use existing functionality; (2) Advice on how to achieve a particular effect using documented (and occasionally undocumented) functionality; (3) Advice on algorithm design to achieve a required level of performance.

- Operational advice

We provide answers to non-technical questions involving workstation operation: (1) How to partition a workstation to meet different requirements; (2) How to use system tools to install software from various media; (3) How to boot the workstation; (4) How to initialize a partition for a specific site/individual profile.

- Error recovery

We advise users how to recover from "soft" error; we teach users how to diagnose bugs they've introduced into their software; we walk users through major Interlisp-D tools, such as the Editor, Inspector, and Break package, in an effort to create self-sufficiency.

- Communicating with other systems

We respond to users having problems communicating with other systems using TCP, Pup, and XNS protocols; we provide detailed consultation for users who rely on RS232-based software for file transfer, printing, and chat.

- Bug reporting

We accept customer bug reports and subject them to analysis and verification. We submit bug reports to the AR database, and try to supply workarounds to customers whose productivity has been impaired.

- Crisis intervention

We respond to users whose systems have "crashed," threatening to destroy programs and data. We provide techniques for resuscitating dead systems, scavenging broken file systems, and restoring corrupted disks.

- Enhancement requests

We forward user requests for new system features to Development

Our goal is to provide the software services you need to use your Xerox AI Workstation productively and to its best advantage. We welcome your comments and suggestions.

LOOPS WORKSHOPS

Understanding the principles of object-based programming can be difficult. Our minds have become conditioned to think about problems using the more conventional paradigm of procedure-based programming. In order to help people overcome this difficulty, the Xerox System Analyst have developed a seminar called "The Loops Workshop." The one day workshop provides a basic understanding of object-oriented programming, why one might use it in place of conventional programming methodology, and how to recognize problems appropriate for the object-based approach.

During the course of the day, a prototype expert system is built with the workshop attendees providing input as to the topic, nature and knowledge base for the system. By the end of the day, attendees have had the experience of helping to design, create, and grow an expert system using the tools provided by LOOPS, one of the most powerful expert system building shells available.

Xerox AI Analysts offer these workshops free of charge in all Xerox AIS regional offices or they can be held at customer sites. The average workshop has a dozen attendees, though workshops have been conducted for as few as six and as many as twenty people.

If you are interested in attending a LOOPS workshop or having one conducted at your place of business, please contact your local Xerox AI Sales Representative.

USERS' GROUP ACTIVITIES

On September 29th and 30th Beth Derrough and Andy MacRae from the Midwest Area brought together the Midwest AI Users Group for a 1-1/2 day meeting to discuss their AI projects. The participants attend workshops on "LOOPS, Building Your Own Browser," "Prolog & LOOPS Integration," and "Graphics and Sketch." Dr. Larry Masinter, PARC researcher, talked about Common Lisp and Common Loops. Additionally, Whit Haney discussed Xerox AI strategy and shared his insight on Xerox' corporate commitment to AI's continued success. Gary Aslin spoke at the dinner, stressing the importance Xerox places on customer feedback, partnering and successful development of AI systems.

The meeting, which had a \$175 registration fee attached, was well attended by individuals from private industry and universities. Gary Berosik of Honeywell was elected President of the Midwest AI Users Group and Don Herkimer of Martin Marietta Aerospace was elected Secretary.

Contact your Xerox AIS Sales Representative for information on Users' Group Meetings in your area.

EDUCATIONAL SERVICES

Xerox Artificial Intelligence Systems offers Interlisp-D, LOOPS and Xerox Quintus Prolog classes. Previously the Lisp I and Lisp II classes were each one week long and could be taken separately. Starting with the March classes the Lisp I and Lisp II will be combined into a single two week class. The second week of Lisp I is now entitled Lisp IB and registration requires the consent of an instructor. LOOPS and Prolog each remain as one week classes.

Below is the schedule for the first half of 1987.

DATE (week of)	CLASS	LOCATION
19 Jan	LISP I: Introduction to Interlisp-D	Pasadena, CA
26 Jan	LISP II: Intermediate Interlisp-D Programming	Pasadena, CA
2 Feb	LOOPS I: Programming in LOOPS	Pasadena, CA
23 March	LISP I: Introduction to Interlisp-D	Pasadena, CA
30 March	LISP IB: Intermediate Interlisp-D Programming	Pasadena, CA
4 May	LOOPS I: Programming in LOOPS	Pasadena, CA
11 May	Prolog I: Introduction to Xerox Quintus Prolog	Pasadena, CA
1 June	LISP I: Introduction to Interlisp-D	Pasadena, CA
8 June	LISP IB: Intermediate Interlisp-D Programming	Pasadena, CA

To coincide with the release of Xerox Common Lisp, we will offer a 3 1/2 day Conversion Seminar to help experienced programmers use the new XAIE. The seminar will cover Xerox Common Lisp, the new aspects of the programming environment, and the conversion of programs written in Interlisp-D to XCL. Contact your Xerox AIS Sales Representative or the Xerox Educational Services Coordinator, at extension 2676 at the toll free numbers for exact dates of the Conversion Seminars.

SUBMISSION TO NEWSLETTER

Masterscope is published quarterly by Xerox Artificial Intelligence Systems, 250 North Halstead Street, Pasadena, CA 91109. Opinions expressed herein are those of the authors and do not necessarily reflect Xerox Company Policy.

Masterscope is intended as a communications forum, providing interaction between customers/users of Xerox AI products and between the users and Xerox AIS Management. Please send comments, submissions for future *Masterscope* issue, and additions or corrections to the mailing list to:

Via network:

Welch@Ohio-State.ARPA
Welch@Ohio-State.CSNet
{ihnp4,cbosgd}!osu-eddie!welch

or:

AINewsletter ↑ .pasa@Xerox

Via US Mail:

Arun Welch
Laboratory for AI Research
Computer Information and Science Department
Ohio State University
2036 Neil Avenue Mall
Columbus, Ohio 43210-1277

Xerox AIS Software Support

Toll Free Numbers

(800)228-5325 Outside California

(800)824-6449 Inside California

ARPANet Info-1100 and Bug-1100 Mailing Lists

Another avenue of communication available to the users of Xerox 1100 workstations is the ARPANet Info-1100 and Bug-1100 mailing lists. (Note: As of January 1987 Info-1100 and Bug-1100 have been combined. The combined list is Info-1100 with Bug-1100 an alias of Info-1100.) This discussion list focus on Xerox 1100 workstations (1100, 1108, 1121, 1132, 1185, 1186) and protocols. The purpose of Info-1100 is to stimulate communication and sharing between computer science research groups that are using or are interested in these machines.

Info-1100 is an "information and bug channel" where messages and announcements of available LispUsers packages can be made or for queries such as, "Does anybody know of a PUP file server implementation for the Sinclair ZX-80?." Readers can also warn others of Interlisp-D bugs, suggest workarounds or solicit help or discussion.

Xerox PARC, SIS, and AIS people are included in the distribution list to facilitate communication about new developments, bugs, performance issues, etc.. Additionally, Info-1100 is automatically forwarded to AISupport.pasa@Xerox, where bugs are noted and acted upon if appropriate.

Archives of messages are kept on SUMEX-AIM in files:

<BBOARD>INFO-1100.TXT <BBOARD>BUG-1100.TXT

Some net-contributed LispUsers packages, updates, and patches are kept on SUMEX-AIM in files:

<LISPUSERS.INTERMEZZO> <LISPUSERS.KOTO> etc.

Requests to be added to or deleted from these lists, problems, questions, etc. should be sent to:

Info-1100-Request@SUMEX-AIM.STANFORD.EDU

Coordinator: Christopher Schmidt <SCHMIDT@SUMEX-AIM.STANFORD.EDU>

Masterscope
Xerox Artificial Intelligence Systems
250 North Halstead Street
P.O. Box 7018
Pasadena, CA 91109-7018