THE ECLIPSE IPSE

A. Alderson

Rapporteur: Mr. G. Tomlinson

Slide 1

Good-afternoon Ladies and Gentlemen.

This afternoon I am going to talk about the Eclipse Programme which is an Alvey project.

The purpose of the Eclipse Programme is to develop an Integrated Project Support Environment built around a database. The total funding for this project is in excess of 7m. This gives an indication of the size of the undertaking. However, even with this degree of expenditure it is not possible to develop all of the tools required of a project support environment. The Programme has generally avoided work concerned with project management and concentrated on tools for the software development lifecycle in a configuration controlled environment.

Omitting a large area of concern to an IPSE has forced us to consider how to integrate existing tools and how to enable third parties to develop integrated tools. These considerations have had a considerable impact upon the direction of the project, leading us to base our work on PCTE, which I discussed earlier today, and leading us to develop a Public Tools Interface.

I will begin by outlining the scope of the Eclipse Programme, in terms of the constituents of the IPSE and its hardware environment, and in terms of the research being undertaken.

I will then discuss our concept of Integration which leads on to the Public Tools Interface. The PTI concerns itself with access to the database and with user interaction. I will discuss some of the concepts involved.

Finally, I will indicate the current status of the Programme.

slide 2

Before proceeding further, I will in good cinema tradition, give the credits. This is in case you fall asleep before the end.

There are six collaborators in the Eclipse Programme, three academic and three industrial. The academic partners are the University College of Wales, Aberystwyth; the University of Lancaster and the University of Strathclyde. These parties are responsible for the research aspects of the Programme, but they all contribute to the development of the IPSE itself in varying degrees.

The industrial partners are Learmonth and Burchett Management Systems who are responsible for the LSDM method (which may be known to you as SSADM, the method chosen by CCTA for use within UK Government projects), CAP, who are responsible for the Ada cross-development work and who make a strong contribution to the User Interfaces, and Software Sciences. Software Sciences are the prime contractor for Eclipse and are responsible for the production of the IPSE.

slide 3.1

Eclipse is a database based IPSE. The Eclipse development originally called for the IPSE to be based upon UNIX and to use SDS2 (a database produced by Software Sciences) and indeed the first demonstrated version of Eclipse is so constructed. However, we decided for the final implementation that PCTE should be the basis.

The decision is both commercial and technical. Three of the Eclipse partners, Software Sciences, CAP and Aberystwyth, are partners in the Sapphire project developing PCTE implementations. This shows the degree of our belief that PCTE is the best way forward for European interests.

Technically, PCTE offers a database system well suited to our needs, offers a user interface which we can build upon, provides an answer to the local distribution problem and gives a simple way to incorporate foreign tools. Using PCTE we can integrate any existing UNIX tool and any PCTE tool. The latter is important in view of the ESPRIT developments and we are in close contact with the PACT, SPINX and SPMMS projects.

slide 3.2

In order to present our view of database and user interface, and to encapsulate our view of integration, we provide our own Public Tools Interface. This is used by all developments within Eclipse.

We will be publishing this interface so that other tool builders may interface to Eclipse.

slide 3.3

The Eclipse Programme is developing three major tool sets for the LSDM method, for MASCOT3 (the new MoD method) and to support Ada. These three have been carefully chosen to address a large part of the development life cycle.

LSDM address the phases from requirement specification to logical system design, MASCOT3 address the phases to physical system design and the Ada facilities address implementation.

LSDM involves a series of data collection and data transformation steps using diagrams and tabular data. This culminates in database schema designs and process outlines. This logical system design has parallels with the starting point of MASCOT3, and we are considering the possibilities of automatic transformation between the outputs of LSDM and the inputs of MASCOT3. (We know that such a transformation is possible between the CORE method and MASCOT3. CORE has many parallels with LSDM.) MASCOT3 also uses diagrams and tabular data.

We are providing facilities to generate Ada source representing a MASCOT3 design into which the algorithms of the system may be inserted.

This souce code may be compiled using Eclipse's Ada cross-development facilities targetted to the INTEL 80286. Ada compilation based on version controlled program libraries, linking, down-line loading, remote test control and PROM burning are all provided.

slide 3.4

The Eclipse Programme is developing various other tools. For example, due to the heavy use made of diagrams by the various system development methods we have chosen to build a generic design editor which can be parameterised to cope with various diagram types.

We also hope that third parties will wish to integrate tools into Eclipse and we are discussing the possibility with other ALVEY projects.

slide 4

The academic partners are undertaking a wide ranging research programme.

In terms of effort the major topic is software reuse. The problem we have posed is that of cataloguing components so that they may be retrieved for reuse, and of reusing such components to produce prototype systems quickly. Related to this is the problem of describing the structure of systems we wish to build.

The methods research has two aspects. The first is the development of the generic design editor. The other aspect is a study of how sequences of methods can be created with automatic transformations between outputs and inputs.

The MMI work is studying means of developing effective user interfaces for complex software systems such as IPSE. I will give examples of what this has achieved later.

The distribution work is considering the problems of wide area communication between IPSES as would occur between prime and sub-contractors on a large project. Eclipse itself is an interesting example.

slide 5

The hardware architecture assumed for Eclipse is a homogeneous Ethernetwork of personnel workstations with bit-mapped graphics screens, local processing and local data storage. We are currently using SUN2 and SUN3 systems (using the GARNET prototype PCTE implementation) and a VAX 11/750 which has the major data storage capability and the gateway to other Eclipse systems.

This completes my rapid overview of the Eclipse Programme. I now wish to concentrate on some more detailed aspects of the IPSE itself.

slide 6

The first problem which confronts the IPSE designer is what do we mean by integrated. There are many possibilities.

One answer is the INTERLISP-type system. Here there is in effect only one tool. At any time any operation of the tool may be invoked and it has an action appropriate to the current status. In a sense this is the ultimate integration.

However it is difficult to add new facilities and impossible to use existing tools.

We chose a different notion of integrated. We decided to have separate tools which interacted with each other by sharing data structures. UNIX has this approach but its data sharing is primitive being just the sharing of byte strings. We wanted sharing of data structures.

We decided that integration should also mean consistency of interaction of all tools with the user. Obviously a design editor and an Ada compiler cannot look the same, but we did require the principle of minimum surprise to hold. Pressing the same button in any tool should have a predictably similar effect and two things that looked the same should have similar properties.

slide 7

To encapsulate these ideas we have defined a Public Tools Interface based upon PCTE.

All of the PCTE facilities are available but we require that tools use the Eclipse User Interface rather than the PCTE User Interface to ensure consistency of user interaction. We also require tools to use the Eclipse Two Tier Database rather than the PCTE OMS unless there is no alternative.

slide 8

The Eclipse Two Tier Database is an extension of the PCTE OMS concept.

In PCTE the Object is a database entity having attributes and relationships to other Objects, and having a Content. In PCTE a Content is just a UNIX file having no inherent structure.

Our extension is to apply database concepts to the Contents of Objects, so that a Content may have a schema. This is obviously valuable given that much of the data we have is structured (diagrams and tables) and we wish to apply both specific and generic tools to that data.

The data model we use in this second tier (within Contents) is based on entities, attributes and relationships as is the first tier as described by PCTE.

We allow relationships between second and first tier entities. This provides us with the ability to describe hierarchies of Objects of different types.

It could be argued that all of our entities could be described as first tier objects.

While this is theoretically possible, with current technology it would be unacceptably inefficient, since for many purposes we wish to consider an object as a single entity without concern for its precise Content (e.g. for version control, back-up).

slide 9

The second tier data model is similar to but not the same as PCTE. This may seem perverse but there are two major reasons.

The first is that we had an implementation, based on Carnegie-Mellon's IDL, of schema accessed contents; so cost was a consideration. The second is that we consider PCTE difficient in the data types it offers and we wished to simplify the construction of tools by offering sequences and enumerations.

slide 10

This left us with the problem of unifying PCTE and IDLE through a single interface.

This was eased by the shared concepts of entity, attribute and link.

We were also aware of the need to produce a concise interface. In Eclipse-V1 we developed a very wide interface giving very detailed control to the interface user. This had a number of adverse consequences.

First we had a lot of code because we had a lot of entry points. The fact that we used Ada in Eclipse-V1 did not help.

Second we had efficiency problems because we were using messages to access the database interface, and the detailed interaction implied many messages.

Third we found interface users used it very inefficiently. For example, when accessing three attributes from the same entity they would re-identify the entity before each attribute access. Tight budgets prevented much consultancy on use within the project so third party use would give great problems.

We have therefore attempted to introduce an interface where accesses can be optimised within the database and which has only few functions with little scope for alternative ways of doing things.

So for instance there is only one function which can obtain a value.

This function obtains the value of:

- any single valued attribute,

- any virtual attribute defined by the system (e.g. entity type-, permitted access level) rather than have separate functions,

- a single value by key from a bag,

- an iterator over a set or sequence.

Every function which takes an entity as a parameter may identify that entity by either:

- a pathname relative to the root, some reference object (e.g nome) or the current entity,

- the value of an iterator where the iterator is defined by a pathname with a pattern as the last element of the name. All entities matching the pattern may be accessed.

slide 11

The Eclipse User Interface is designed to utilise bit-mapped screens with both keyboard and mouse input. It offers multiple windows each of which may have multiple frames. (These may be text, graphics, control panel or message frames.) Windows may be iconised.

The interface handles the full range of interactions from text (tty) to graphics. The normal user interaction Eclipse is by means of control panels which are modelled on the instrument panels of hardware equipment.

The user has various interaction possibilities such as buttons, indicator lights, menus and signs. The latter are areas of the screen with a title. These can be used for both display and input of values

slide 12

This is an example of the Eclipse User Interface in use.

At the top is the Eclipse Status window showing system status information.

The main portion of the screen is occupied by the jsd tool window. This has two control panel frames with examples of menus, buttons and signs.

The lowest window is that for the shell tool. This is a TTY window.

At the lower left is the icon of the Eclipse Help tool.

slide 13

Our experience with Eclipse-V1 has shown that if we are to have consistency of interaction in different tools all of the following must be provided:

- Standards for presentation and interaction.

- An application interface assisting those standards. (In Eclipse we have a language, Format Definition Language, which expresses the physical presentation of the interface to the user).

- A Help system to give consistent treatment of user assistance.

- A message system to give consistent treatment of messages, prompts and confirmations.

- A consistent command syntax.

slide 14

The Eclipse Programme began in August 1984. The current and envisaged progress is as follows:

- Eclipse-V1 was demonstrated in Quarter 2 1986 at the Lancaster IPSE Conference and at the ALVEY Conference, where the User Interface attracted approving comment.

- The Eclipse PTI will be implemented by Quarter 4 1986, and will be available to third parties then. The specification will be available earlier.

- The MASCOT3 and Ada toolsets will be available by Quarter 2 1987.

- The LSDM toolset will be available by Quarter 3 1987.

Other results will be published as they become available. The Programme has published a number of papers already. The references are given in the appendix to this paper.

ECLIPSE PAPERS

1. A Alderson; M F Bott; M E Falla

   An Overview of Eclipse in Integrated Project Support Environments

   Ed. J McDermid, PPL, 1985

2. A Alderson; M F Bott; M E Falla

   Eclipse Object Management System

   IEE Journal of Software Engineering in Jan '86

3. A Alderson

   Configuration Management in Eclipse

   MILCOMP '85

4. M F Bott

   Software Support Environments in Commercial Data Processing

   To be published in: Data Processing 1986

5. R H Pierce

   Ada in the Eclipse Project Support Environment

   Proceeding of International Conference on Ada - Paris 1985

6. J Smart

   Man Machine Interface Management System for UNIX

   Uniforum 86 at Ananeim CA. 4-7 Feb 86

7. Describing Software Design Methodology

   I Sommerville; R Welland; S Beer

   To appear in the Computer Journal

8. The ECLIPSE System Structure Language

I Sommerville and K Thomson

19th Int. Conf on Systems Sciences, Honolulu, Jan 1986

9. A Software Components Catalogue

M Wood and I Sommerville

To appear in "Intelligent Information Retrieval Systems"

Ed. J A Campbell Published by Ellis Horwood

10. Eclipse: A Distributed Software Development Environment

D Hutchison and J Walpole

Software Engineering Journal, Vol 1, No 2 (March 1986)
pp 88 - 92


ECLIPSE


ALVEY

The University College of Wales, Aberystwyth
Coleg Prifysgol Cymru

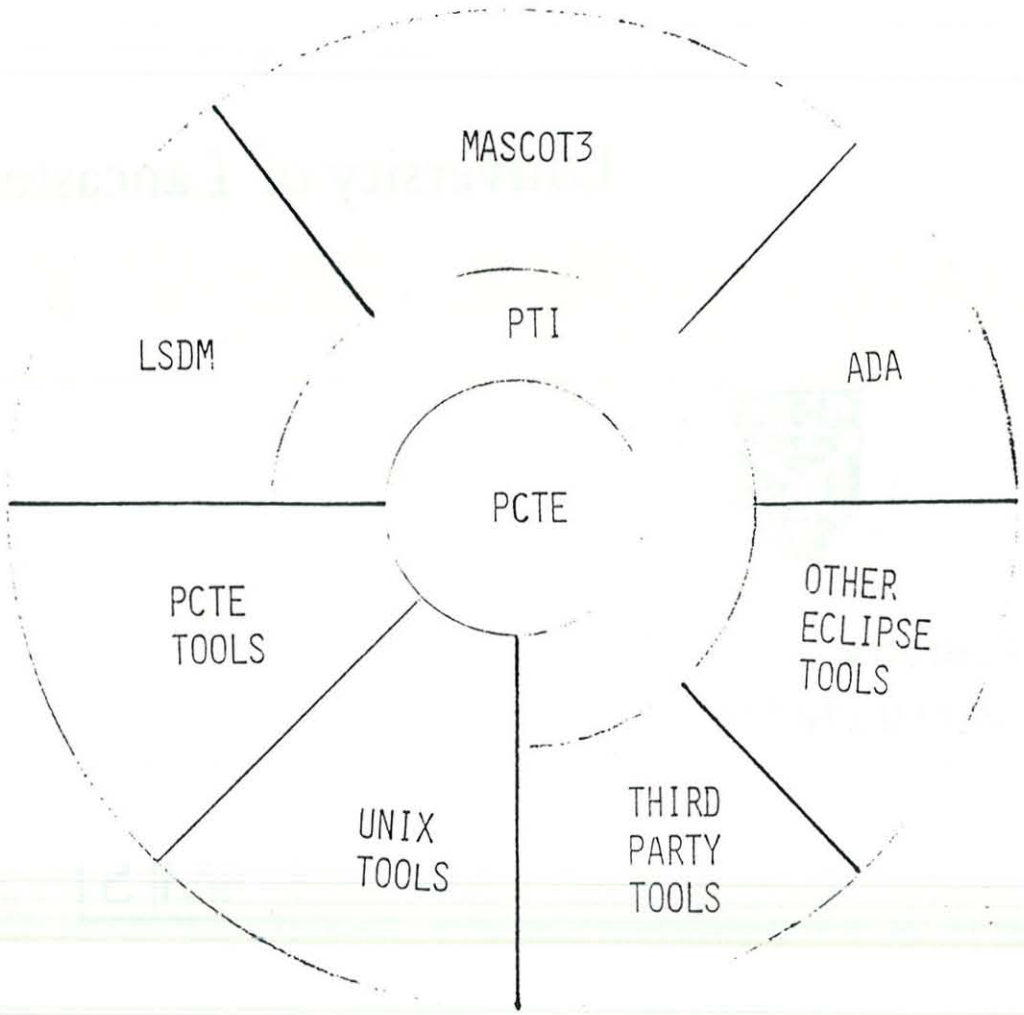
University of Lancaster

University
of Strathclyde

L B
M S

CAP

Software Sciences


MASCOT3
PTI
LSDM
ADA
PCTE
PCTE
TOOLS
OTHER
ECLIPSE
TOOLS
UNIX
TOOLS
THIRD
PARTY
TOOLS

ECLIPSE

MAJOR RESEARCH TOPICS

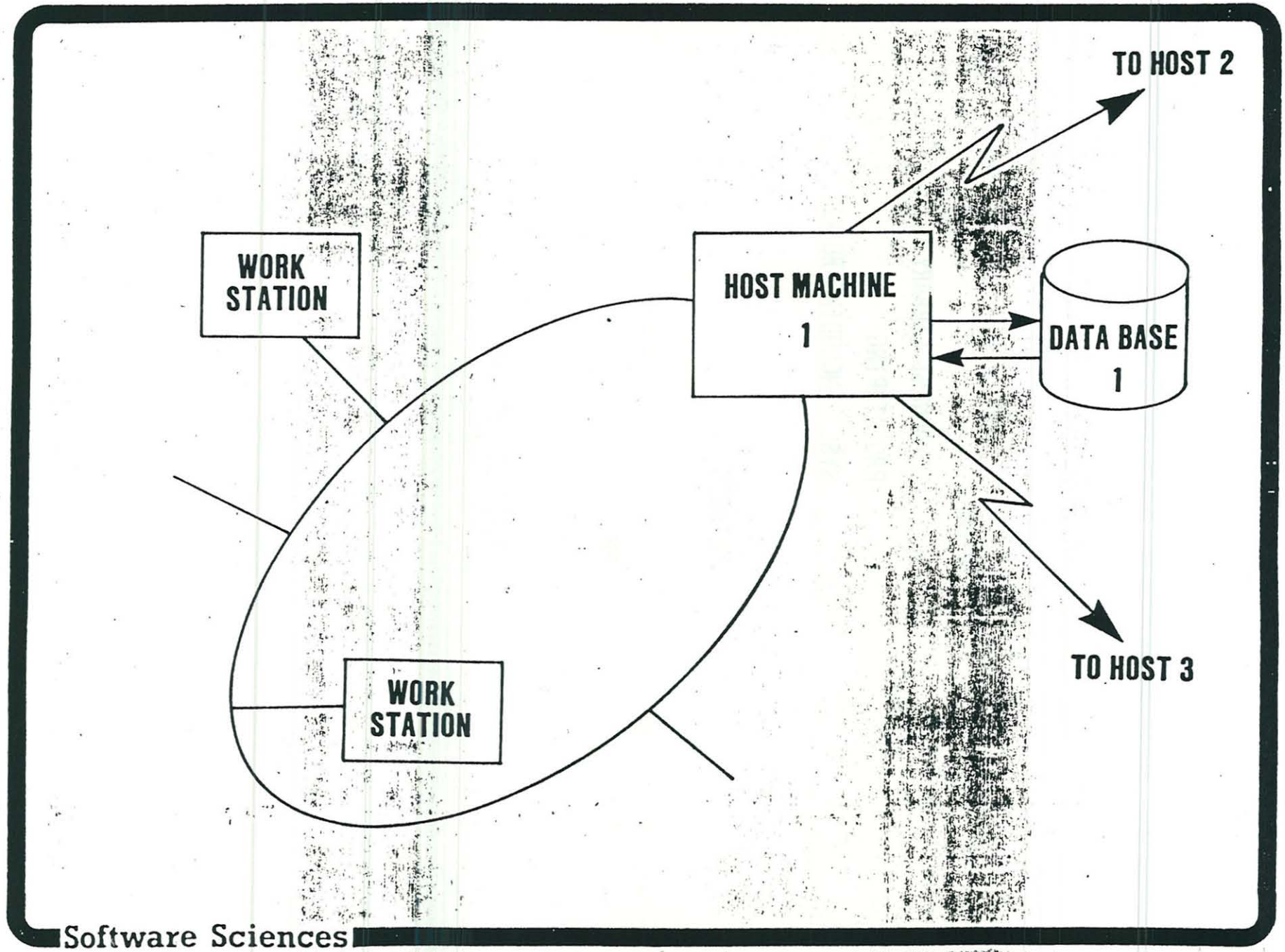
REUSE

- COMPONENT LIBRARY
- PROTOTYPING
- SYSL INCLUDING BUILD

METHODS

MMI

DISTRIBUTION (WAN)


TO HOST 2
WORK STATION
HOST MACHINE 1
DATA BASE 1
TO HOST 3
WORK STATION
Software Sciences

INTEGRATION

BY DATA

SHARING STRUCTURES

BY USER INTERFACE

CONSISTENT INTERACTION

ECLIPSE PTI

TWO TIER DATABASE

OMS

EXECUTION

COMMUNICATION

IPC

ACTIVITIES

DISTRIBUTION

UNIX

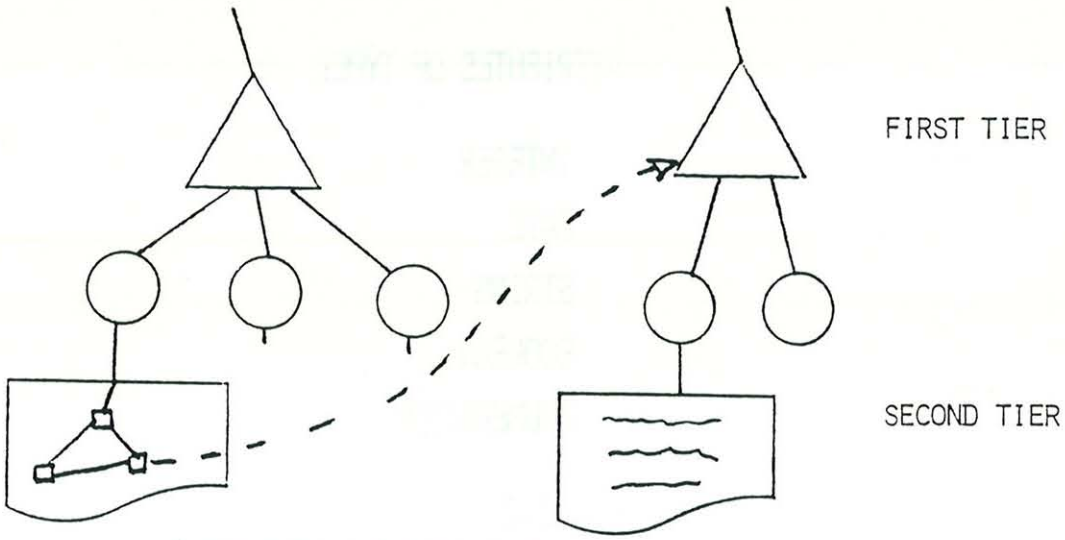
USER I/F ECLIPSE USER I/F

ECLIPSE PTI

TWO TIER DATABASE

AN EXTENSION OF THE OMS CONCEPT.

DATABASE MODEL APPLIED TO OBJECT CONTENT.



FIRST TIER DATABASE MODEL
IS PCTE OMS

ECLIPSE PTI

SECOND TIER DATABASE MODEL

IDLE (BASED ON IDL)

NODES (ENTITIES) WITH:

ATTRIBUTES OF TYPE:

INTEGER
DATE
STRING
BOOLEAN
ENUMERATION

LINKS TO :

LOCAL NODES
EXTERNAL NODES
PCTE OBJECTS

SEQUENCES

NODE TYPES MAY BELONG
TO CLASSES

ECLIPSE PTI

UNIFIED DATABASE MODEL

PCTE PLUS IDLE

ENTITIES WITH

ATTRIBUTES

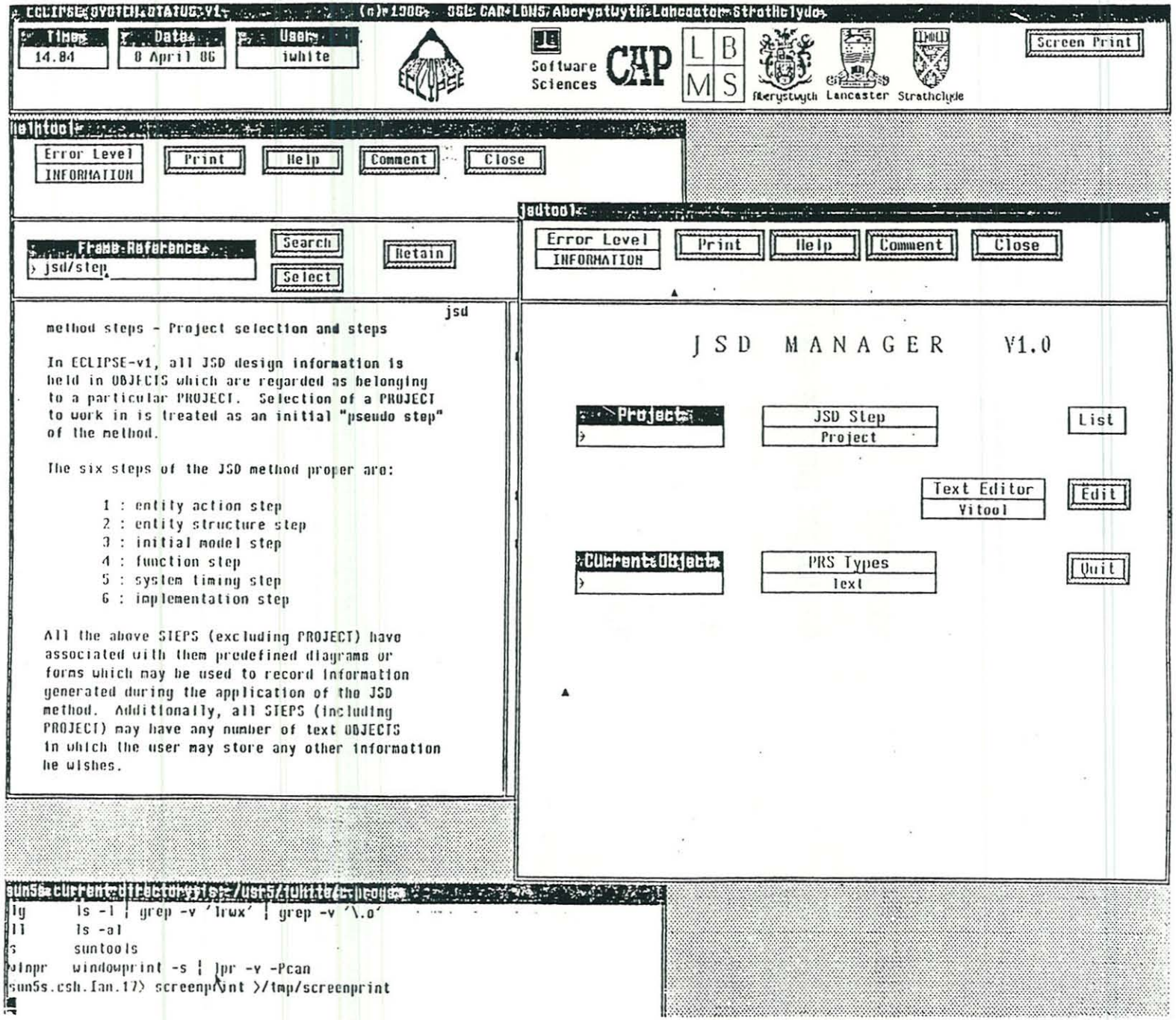
LINKS

VIRTUAL ATTRIBUTES
KEYED BAG ATTRIBUTES
DERIVED ATTRIBUTES

ECLIPSE PTI
USER INTERFACE

BIT-MAPPED SCREEN
KEYBOARD AND MOUSE
MULTIPLE WINDOWS
MULTIPLE FRAMES
ICONS

TTY
GRAPHICS
CONTROL PANEL
    BUTTONS
    LIGHTS
    MENUS
    SIGNS


Time
14.84
Date
8 April 86
User
iwhite
Software Sciences
CAP
LBMS
Aberystwyth
Lancaster
Strathclyde
Screen Print
Error Level
INFORMATION
Print
Help
Comment
Close
Frame Reference
jsd/step
Search
Select
Retain
jsd
method steps - Project selection and steps
In ECLIPSE-v1, all JSD design information is held in OBJECTS which are regarded as belonging to a particular PROJECT. Selection of a PROJECT to work in is treated as an initial "pseudo step" of the method.
The six steps of the JSD method proper are:
1 : entity action step
2 : entity structure step
3 : initial model step
4 : function step
5 : system timing step
6 : implementation step
All the above STEPS (excluding PROJECT) have associated with them predefined diagrams or forms which may be used to record information generated during the application of the JSD method. Additionally, all STEPS (including PROJECT) may have any number of text OBJECTS in which the user may store any other information he wishes.
JSD MANAGER V1.0
Project
JSD Step
Project
List
Text Editor
Vitool
Edit
Current Object
PRS Types
Text
Quit
lg ls -l | grep -v 'lrwx' | grep -v '\.o'
ll ls -al
s suntools
pinpr windowprint -s | lpr -v -Pcan
sun5s.csh.Jan.17> screenprint >/tmp/screenprint

ECLIPSE PTI

USER INTERFACE

HOUSE STYLE

APPLICATION INTERFACE (FDL)

HELP

MESSAGES

COMMANDS

PROGRESS

Q2 86

V1 DEMONSTRATED

Q4 86

V2 PTI AVAILABLE

Q2 87

V2 MASCOT TOOLSET

V2 ADA TOOLSET

Q3 87

V2 LSDM TOOLSET

DISCUSSION

Dr. Alderson was asked how the PCTE related to X/Open. He replied that this might become clearer later in his talk and one could look at the names of the collaborators.

Professor Randell asked if the PCTE sat on top of Unix, Dr. Alderson replied that it did not and was an extension to the Unix kernel.

Mr. Jackson asked whether one could add "content" to any PCTE object. Dr. Alderson replied that one could not since "content" is only a property of file type objects.

Professor Atkinson asked whether there was any way to discover on which processor an object was created. Dr. Alderson was not sure.

After Dr. Alderson had explained that composition was a property of the object category which enforced existence, in keeping with Bishop Berkeley's doctrine that things only exist if one is looking at them, Professor Randell commented that this was the best reference to Berkeley he had heard all week.

During Dr. Alderson's description of pathname interpretation Mr. Stroud remarked that it seemed to be the wrong way round. Fred.c was represented by an arity-many link called c from the current directory. He suggested it would be more natural to think of Fred as a composite object with links to its source code, documentation, test data etc. Professor Atkinson said it depended on which way you looked at it, i.e. whether a link was an attribute or vice-versa. In this case links could not have attributes. Dr. Alderson said that doing it this way made the schema definitions more compact since when you created a new program object you did not need to set up all the links. Mr. Stroud was not convinced, feeling that a program object should be some kind of pre-defined template or type.