O-Plan

Installation Guide

Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom

July 13, 1995

Version 2.3

Acknowledgements

The O-Plan project began in 1984. Since that time the following people have participated: Colin Bell, Ken Currie, Jeff Dalton, Roberto Desimone, Brian Drabble, Mark Drummond, Anja Haman, Ken Johnson, Richard Kirby, Glen Reece, Arthur Seaton, Judith Secker, Austin Tate and Richard Tobin.

Prior to 1984, work on Interplan (1972–4) and Nonlin (1975–6) was funded by the UK Science and Engineering Research Council and provided technical input to the design of O-Plan.

From 1984 to 1988, the O-Plan project was funded by the UK Science and Engineering Research Council on grant numbers GR/C/59178 and GR/D/58987 (UK Alvey Programme project number IKBS/151). The work was also supported by a fellowship from SD-Scicon for Austin Tate from 1984 to 1985.

From 1989 to 1992, the O-Plan project was supported by the US Air Force Rome Laboratory through the Air Force Office of Scientific Research (AFOSR) and their European Office of Aerospace Research and Development by contract number F49620-89-C-0081 (EOARD/88-0044) monitored by Northrup Fowler III at the USAF Rome Laboratory.

From 1992 to 1995, the O-Plan project was supported by the ARPA/Rome Laboratory Knowledge Based Planning and Scheduling Initiative through the US Air Force Rome Laboratory through the Air Force Office of Scientific Research (AFOSR) and their European Office of Aerospace Research and Development by contract number F49620-92-C-0042 (EOARD/92-0001) monitored by Northrup Fowler III at the USAF Rome Laboratory.

Additional resources for the O-Plan and O-Plan projects have been provided by the Artificial Intelligence Applications Institute through the EUROPA (Edinburgh University Research on Planning Architectures) institute development project.

From 1989 to 1993, research on scheduling applications of the O-Plan architecture was funded by Hitachi Europe Ltd. From 1989 to 1992, the UK Science and Engineering Research Council (grant number GR/F36545 – UK Information Engineering Directorate project number IED 4/1/1320) funded a collaborative project with ICL, Imperial College and other partners in which the O-Plan architecture was used to guide the design and development of a planner with a flexible temporal logic representation of the plan state. A number of other research and development contracts placed with AIAI have led to research progress on the O-Plan prototype.

Contact Information

The O-Plan project team can be contacted as follows:

O-Plan Team
Artificial Intelligence Applications Institute
The University of Edinburgh
80, South Bridge
Edinburgh EH1 1HN
United Kingdom

Tel: (+44) 131 650 2732
Fax: (+44) 131 650 6513
Email: oplan@ed.ac.uk

Created: December 2, 1991 by Brian Drabble
Last Modified: July 13, 1995 (09:37) by Brian Drabble
Printed: July 13, 1995

Contents

1 Introduction

1.1 Contents of this guide

This guide is divided into the following sections:

1. **System Requirements**: this section describes the hardware and software you need to run the O-Plan system.
2. **Installing O-Plan**: this section describes the installation of the O-Plan software. The directions assume you have a reasonable understanding of your computer system and have a working knowledge of the UNIX operating system and its directory structure.
3. **Using O-Plan**: this section introduces the environment variables which O-Plan uses and the basic startup script for invoking the O-Plan system.

1.2 Text Conventions

This guide uses the following typeface conventions:

`cd demo`	Text you type from the keyboard. The [return] key is assumed to be pressed at the end of each command. For example, the UNIX command to move to the user's root directory is as follows % cd $home
`<name>`	Used to indicate a generic name which is normally replaced by a specific instance by the user
lowercase type	Used to specify path names, filenames and directory names. For example /project/oplan/demo/tf/pacifica-1.tf
UPPERCASE TYPE	Names of environment and system variables
italic text	Emphasised words and expressions Remember to *print* and *verify* the plan

1.3 Installation Steps

The installation involves more than just copying and "untarring" the contents of the tar image onto your machine. It involves the correct selection of hardware and the initial configuration of the O-Plan software. The steps to follow are outlined here:

1. Before installing O-Plan make operational the essential hardware listed in Section 2 "Systems Requirements".
2. Follow the instructions in Section 3 to read the contents of the release tar file into your computer system (new sub-directories will be generated as necessary) and to install the system so that it is ready to run.
3. Once installed the software needs to be configured for use. Follow the instructions in Section 4 to configure O-Plan to run both the demonstration and the users' own specific problems.

2 System Requirements

This section lists the essential hardware and software required to run the O-Plan system. Before installing the software check the list of essential hardware described below. The software systems listed must be operational before the O-Plan software can be installed.

2.1 Hardware Requirements

1. Medium to large scale workstation such as a Sun Microsystems Sun-4 or SPARCstation with a monochrome or colour monitor.
2. 32 megabytes of physical memory. O-Plan will run in less memory than this but the performance will be reduced due to increased paging to disc.
3. 50 megabytes of swap space. This is the minimum virtual memory in which the O-Plan system will run.
4. 15 megabytes of disk space for the source code, documentation and demonstrations.
5. A printer accessible via the UNIX `lpr` command which can print Postscript files.

O-Plan does not require specialised peripherals such a digitisers or plotters.

2.2 Software Requirements

The O-Plan system is written in a mixture of C and Common Lisp. Common Lisp is used for all development programming and separate C programs are used for menu and display manipulation. The software required is as follows:

1. UNIX version 4.1.1 or 4.1.3 of SunOS.
2. X11 window environment release 5.
3. Lucid Common Lisp (LCL) Version 4.1. There are several ways to set up LCL 4.1. O-Plan has been tested only in the non-dbcs version and is incompatible with the ANSI-packages module.

The O-Plan system also includes various user interface facilities, one of which allows the AutoCAD drawing package (Version 11 or later) to be used as a display interface. This package allows plans and the world state to be displayed on the screen. If the user wishes to use this interface then a copy of the AutoCAD package must be obtained.

3 Installing O-Plan

O-Plan for the Sun-4 and SPARCstation is released as a tar image and comprises source code, documentation and demonstration files. Each directory and sub-directory has its own README file to give details of its contents.

3.1 On Every System

At this point it is worth noting that if the user is unfamiliar with the process of extracting tar files from a tape/image into a directory (and where appropriate its sub-directories) then they should consult the UNIX *Command Reference Manual* or their own systems manager.

The install process will not succeed if:

1. the target directory is not owned by the user OR
2. the user does not have write access to the target directory

If either of the above cases are true then the user must:

1. Change the ownership of the target directory to the user carrying out the installation
2. Alter the access permissions on the target directory to allow write access

The following text describes how to install the O-Plan system and assumes the user has already gained access to the relevant directory. Additional notes on the installation procedure together with various pieces of advice are included in the file INSTALL which can be found in the source directory of release.

3.1.1 Installing from a Tar file

1. Enter the command to move to the target directory in which O-Plan is to be installed, for example, `/applications`.

   ```
   % cd /applications
   ```

2. Enter the following commands to make a directory called `oplan` and move into that directory.

   ```
   % mkdir oplan
   % cd oplan
   ```

3. Type the following command to extract the files from the tar image.

```
% zcat <tar_filename> | tar xvpf -
```

In both cases, as each file is extracted, its name and path is printed on the screen. After the extraction is complete the % prompt will re-appear on the screen.

The most common problems which are found when installing the tar image concern access privilege to directories. If the tar image fails to "untar" ensure that you own the directory into which the image is to be loaded into or you have read and write access privilege to the directory. To change the access privileges consult any UNIX user guide. If access privileges are correct then contact your local systems administrator for further advice.

After installation the files will be owned by the user who performed the installation.

3.1.2 Building the O-Plan System

After the installation procedure has been executed the following file structure should have been created.

```
oplan2
    |
    +--demo
    |   |
    |   +--ac
    |   +--tf
    |
    +--documentation
    |      |
    |      +--architecture-guide
    |      +--demonstration-guide
    |      +--handout
    |      +--implementation-guide
    |      +--information-package
    |      +--installation-guide
    |      +--introduction
    |      +--shared
    |      +--tf-manual
    |      +--user-guide
    |
    +--source
           |
           +--agenda-manager
           +--auto-tester
           +--common
           +--config
           +--control-panel
```

```
+--database-manager
|   |
|   +--tf-compiler
|
+--diary
+--doc
+--interface-manager
+--knowledge-sources
+--ks-platform
+--patch
+--profile
+--pw-viewers
+--scripts
+--single-process
+--support
+--task-assigner
+--test-results
+--world
+--xmenu
```

The most important subdirectory for the rest of this installation procedure is `source`. Everything needed to build O-Plan is in that subdirectory.

In outline, the procedure from this point is as follows:

1. Edit one line in `source/scripts/oplan`.
2. Edit one line in `source/Makefile`
3. Compile and build O-Plan.

These steps are described in detail below.

1. Edit `source/scripts/oplan`.

   This is the startup script which will be used to run the O-Plan system. The variable `base_dir` needs to be set to the directory into which the O-Plan system was read. Look for lines near the beginning the file that look something like this:

   ```
   # Edit the following line for your site:
   set base_dir=~oplan/development
   ```

   Edit the definition of `base_dir`. For example, if the correct directory name is `/applications/oplan`, edit the line to be

   ```
   base_dir=/applications/oplan
   ```

The "base_dir=" line in the distributed version of the oplan script may assign a different value than the one shown above; nonetheless, it is the line to edit. Suppose you have placed the O-plan files in /home/oplan/2.3. Edit the definition of "base_dir" to be:

```
base_dir=/home/oplan/2.3
```

Since the "oplan" script is a "sh" script, the base_dir name cannot use the tilde-notation (as in " oplan"). That's why it says "/home/oplan" and not " oplan" above. This is a change from some earlier versions.

Note that `source/scripts/oplan` will be copied to `bin/oplan` during the building of the system and it should be checked that `bin/oplan` is correct.

2. Edit `source/Makefile`.

   This Makefile will use Lucid Common Lisp (LCL) to build O-Plan. However, the command that invokes LCL is different on different systems, so the file must be edited to indicate what command to use. Look for two lines near the beginning of `source/Makefile` that look like this:

   ```
   # Edit the following line(s) to define commands for running Lisp.
   Lucid=lucid4.1+clos
   ```

   Edit the second of the two lines. For example, if the local command for running LCL is `lisp`, edit the second line to be:

   ```
   Lucid=lisp
   ```

3. Compile and build O-Plan.

   Before O-plan can be used, it is necessary to compile the source files, load them into Lisp, and save an executable image. It is also necessary to move some files from the `source` subdirectory to the `bin` and `lib` subdirectories. All of this can be accomplished by typing:

   ```
   cd source
   make all install
   ```

   If the `make` command completes successfully, type:

   ```
   cd ..
   ```

   You should now be back in the main O-Plan directory. At this point, check that the following files exist and have execute permission:

```
bin/oplan
bin/xmenu
bin/xuim

lib/lucid-image
```

You should also check that the following files exist:

```
lib/monitor.def
lib/oplan-planner-default.config
lib/oplan-planner-exec-default.config
```

After the Make procedure has been executed successfully two new top level directories will be created, `bin` and `lib`.

N.B. In future releases, the direct use of `make` may be replaced by an invocation of the script `source/scripts/oplan-install`. This will carry out the processes necessary to install O-Plan at the user's site.

3.2 Setting up for use of O-Plan

When the O-Plan system is invoked the `oplan` script will set a number of environment variables. A brief description of these environment variables is given below (a detailed description of these variables and their function is given in the User Guide). In most cases, it is not necessary for users to set any environment variables themselves. However, they may set some of the variables in order to change the defaults if that is desired.

- **setenv** OPLANDIR **<directory name>**
  The directory name should be the full name of the directory that contains the O-Plan system. This parameter is required.
- **setenv** OPLANTFDIR **$**OPLANDIR**/demo/tf**
  Specifies a directory in which TF files containing domain descriptions are held. This parameter is optional.
- **setenv** OPLANTMPDIR **$**HOME**/oplan-tmp**
  Specifies the directory into which temporary files generated by the O-Plan system will be written. This parameter is required.
- OPLANKSDIR
  This variable should not be set by users. The `oplan` command will print an error message and exit if OPLANKSDIR is defined. Older versions of O-Plan used this environment variable. This parameter is optional.
- **setenv** OPLANACADDIR **$**OPLANDIR**/demo/ac**
  Specifies the directory in which AutoCAD will look for drawing and other files for the

PlanWorld Viewers. Note that AutoCAD requires write access to this directory since it makes locking files in the same directory in which it looks for its saved drawings. This parameter is required if the AutoCAD interface is used.

- **setenv** OPLANEXECHOST <**host-name**>
  Specifies the host on the local network on which the external agent will run when the execute option is requested from the planner. The host should be different from that on which the planner itself is run (i.e., the execution agent will run on a second host). This parameter is optional.

The settable environment variables listed above should be set prior to any use of O-Plan by any user. They may be set in the user's shell startup files (in for example `.tcshrc`) or in any sensible way. Users may also add the directory name $OPLANDIR`/bin` to their path. This will allow O-Plan and its associated commands to be used from any directory.

4 Running O-Plan

After the installation procedure described in Section 3 is complete, O-Plan should be ready to run. As part of the O-Plan system a set of demonstration applications have been included in the release image. These allow a user to see how a domain is defined using the Task Formalism (TF) language and how O-Plan solves a task set by the user. The example applications are stored under the $OPLANDIR`/demo/tf` directory.

4.1 Bringing up the O-Plan System with the `oplan` script

The O-Plan system is invoked from the startup script `oplan` which is held in the $OPLANDIR`/bin` directory. There are several ways to arrange for this script to be accessible as a shell command.

1. Define a shell alias, for shells that provide aliases. For instance, `tcsh` users can add the following line to their home directory's `.tcshrc` file:

   ```
   alias oplan "/applications/oplan/bin/oplan"
   ```

   Note that you do not have to use the name `oplan` for the alias.
2. Alter your `PATH` to include the directory containing the script.
3. Move the script to a directory which is already on your `PATH`.

The O-Plan system is menu driven and requires the X Window System. Before running O-Plan:

1. If you are not already in X windows then execute the command to invoke it.
2. Ensure that the *required* environment variables have been set beforehand (normally this is done in the user's UNIX or shell startup file). Section 3.2. describes these variables and their expected values.
3. Define a command that will invoke O-Plan, as described above.

O-Plan can then be invoked by typing the command. For example:

```
% oplan
```

This will cause several new windows to appear on the screen. The O-Plan system is now ready to accept input from the user. The Task Assignment window allows the user to specify one of the applications from the demonstration set or a TF application which has been defined by the user. Full details of the different windows and input required can be found in the User Guide.

It is recommended at this point that the user read the User Guide and Demonstration Guide in order to run a particular demonstration to gain a better understanding of the O-Plan system. After running the demonstration the user may also wish to read the Task Formalism Manual which describes in detail the language used to define the application.