O-Plan2: Modularity and Interfaces

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

Abstract

O-Plan2 is a command, planning and control architecture being developed at the Artificial Intelligence Applications Institute of the University of Edinburgh. It has an open modular structure intended to allow experimentation on or replacement of various components without the need to change the majority of the overall system.

This paper describes the modular structure of the system along with the internal and external interface languages which are being developed on the O-Plan2 project. In a number of cases, only very simple versions of the interfaces are supported in the current O-Plan2 system. However, even the early versions of such interfaces are proving useful to isolate functionality that may be generally required in a number of applications and across a number of different planning, scheduling and control systems.

1 Introduction

The O-Plan2 Project at the Artificial Intelligence Applications Institute of the University of Edinburgh is exploring a practical computer based environment to provide for specification, generation, interaction with, and execution of activity plans. O-Plan2 is intended to be a domain-independent general planning and control framework with the ability to embed detailed knowledge of the domain. See [1] for background reading and [3] for details of O-Plan1. The O-Plan2 system combines a number of techniques:

- A hierarchical planning system which can produce plans as partial orders on actions.
- An agenda-based control architecture in which each control cycle can post pending tasks during plan generation. These pending tasks are then picked up from the agenda and processed by appropriate handlers (*Knowledge Sources*).
- The notion of a "plan state" which is the data structure containing the emerging plan, the "flaws" remaining in it, and the information used in building the plan.
- Constraint posting and least commitment on object variables.
- Temporal and resource constraint handling using incremental algorithms which are sensitively applied only when constraints can alter.
- O-Plan2 is derived from the earlier Nonlin planner [4] from which it takes and extends the ideas of Goal Structure, Question Answering (Modal Truth Criterion) and typed conditions.
- We have extended Nonlin's style of task description language Task Formalism (TF).

O-Plan2 is aimed to be relevant to the following types of problems:

- project management for product introduction, systems engineering, construction, process flow for assembly, integration and verification, etc.
- planning and control of supply and distribution logistics.
- mission sequencing and control of space probes and satellites such as VOYAGER, ERS-1 etc.

2 The Scenario

- A user specifies a task that is to be performed through some suitable interface. We call this process *job assignment.*
- A *planner* plans and (if requested) arranges to execute the plan to perform the task specified.
- The *execution system* seeks to carry out the detailed tasks specified by the planner while working with a more detailed model of the execution environment.

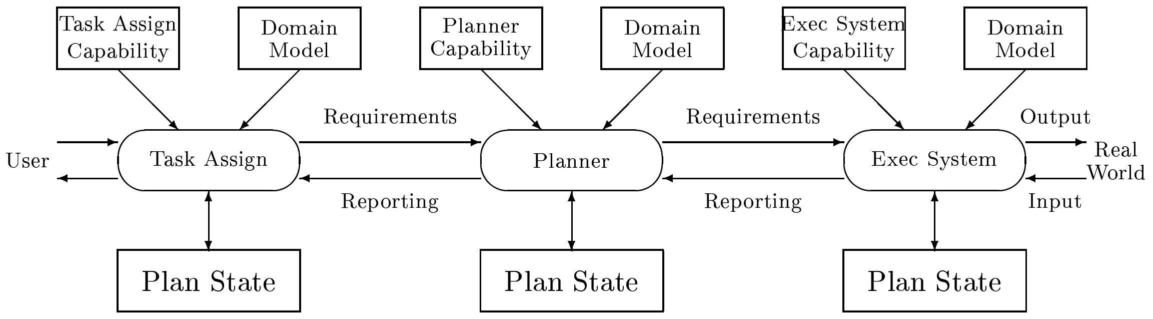


Figure 1: Communication between Central Planner and Ex. Agent

We have deliberately simplified our consideration to three agents with these different roles and with possible differences of requirements for user availability, processing capacity and real-time reaction to clarify the research objectives in our work. However, we believe that the ideas are relevant to the more general case of a *cooperative, hierarchical and distributed command, planning and control* environment.

A common representation is sought to include knowledge about the capabilities of the job assigner, the planner and the execution agent, and the information used to represent the requirements of the plan and the plan itself either with or without flaws (see Figure 1).

The current O-Plan2 system is able to operate both as a planner and a simple execution agent. The job assignment function is provided by a separate process which has a simple menu interface.

The planner components described in outline form in Figure 2 can be mapped to the system and process architecture detailed in Figure 3. Communication between the various processes and support modules in the system is shown in the latter figure.

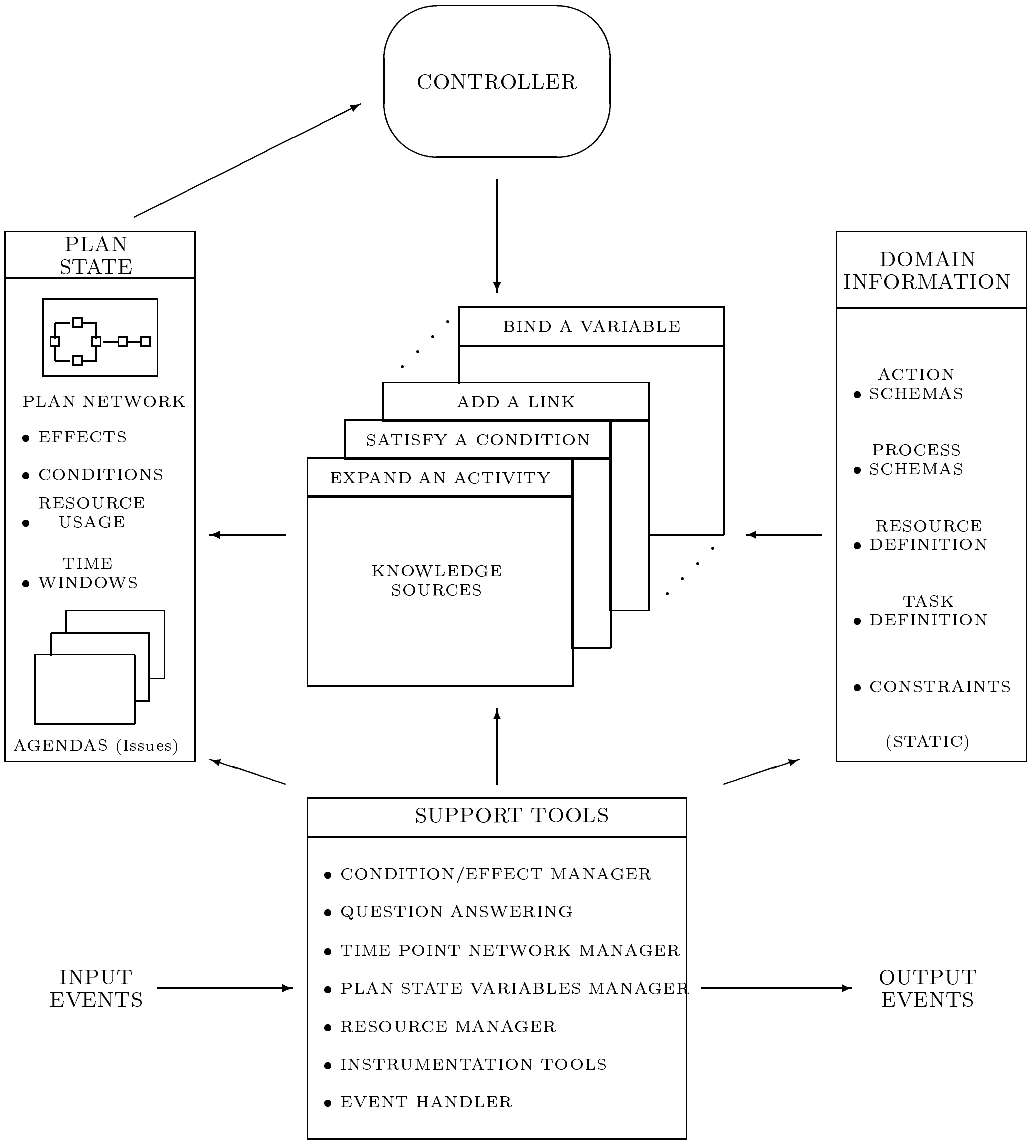


Figure 2: O-Plan2 Architecture

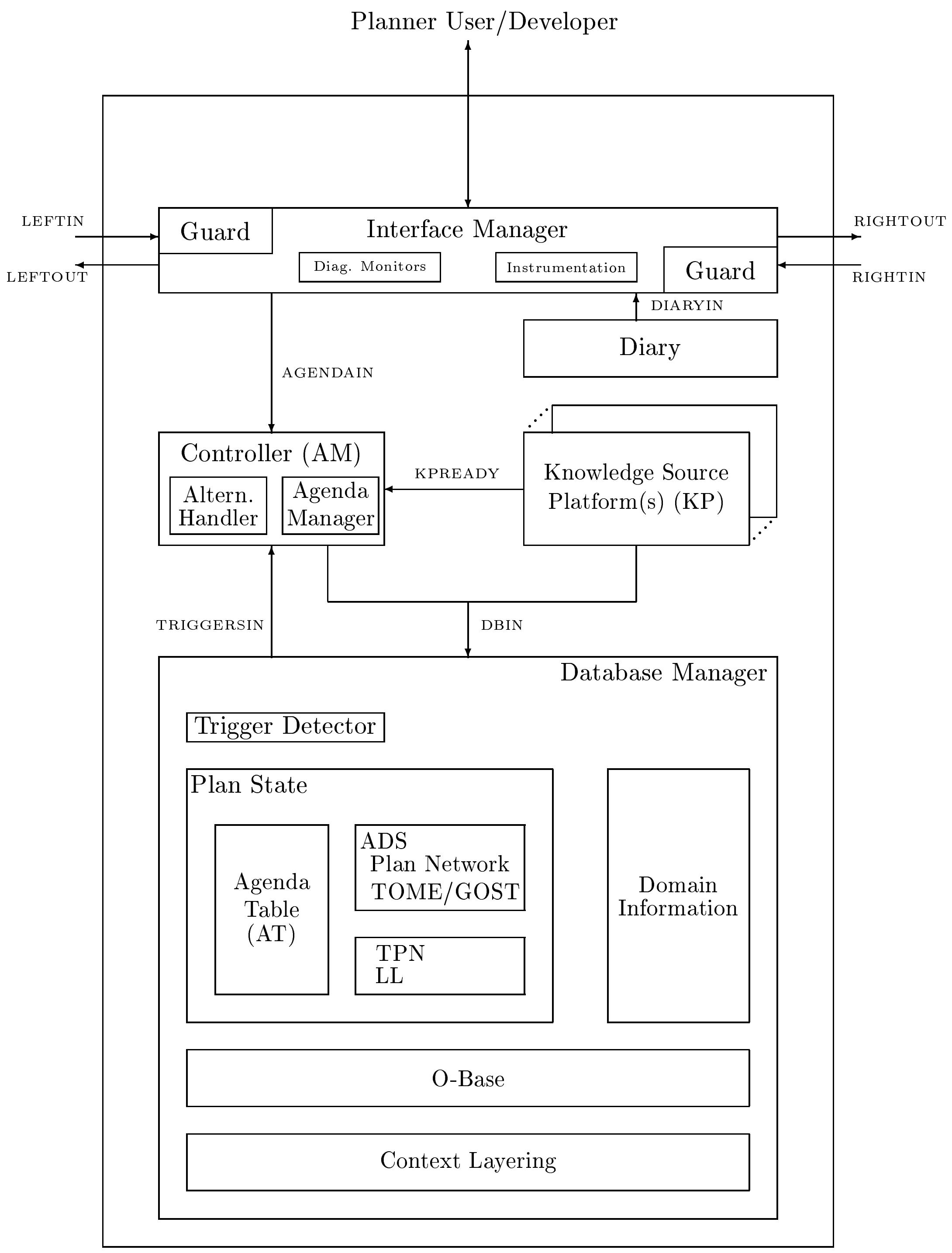
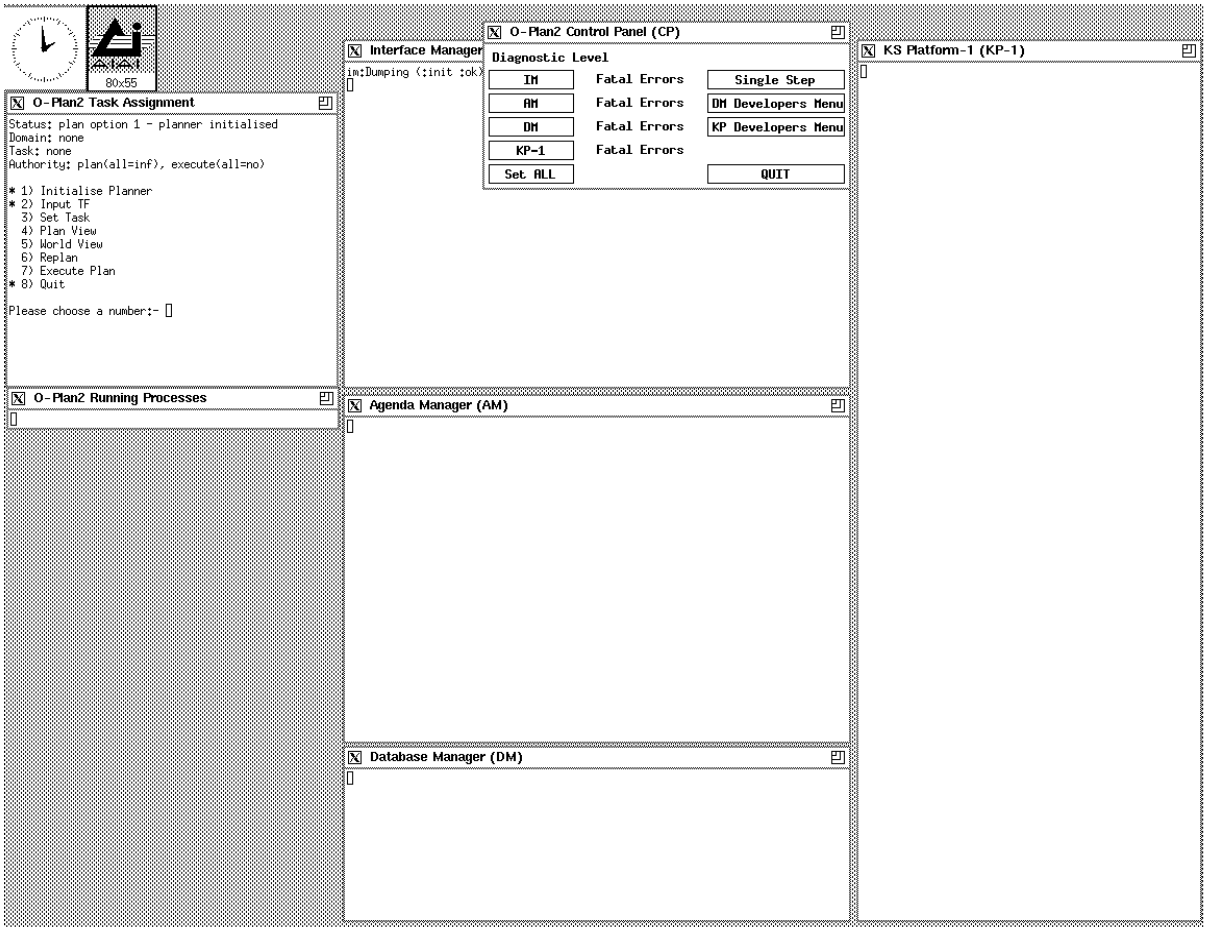


Figure 3: Internal Structure of the Current O-Plan2 Planner

3 Developer Interface

O-Plan2 is implemented in Common Lisp on Unix Workstations with an X-Windows interface. It is designed to be able to exploit multi-processors in future and thus has a clear separation of the various components (as shown in Figure 2). Each of these may be run on a separate processor and multiple *platforms* may be provided to allow for parallelism in knowledge source processing. A sample screen image as seen by the O-Plan2 developer or an interested technical user is shown in Figure 4.



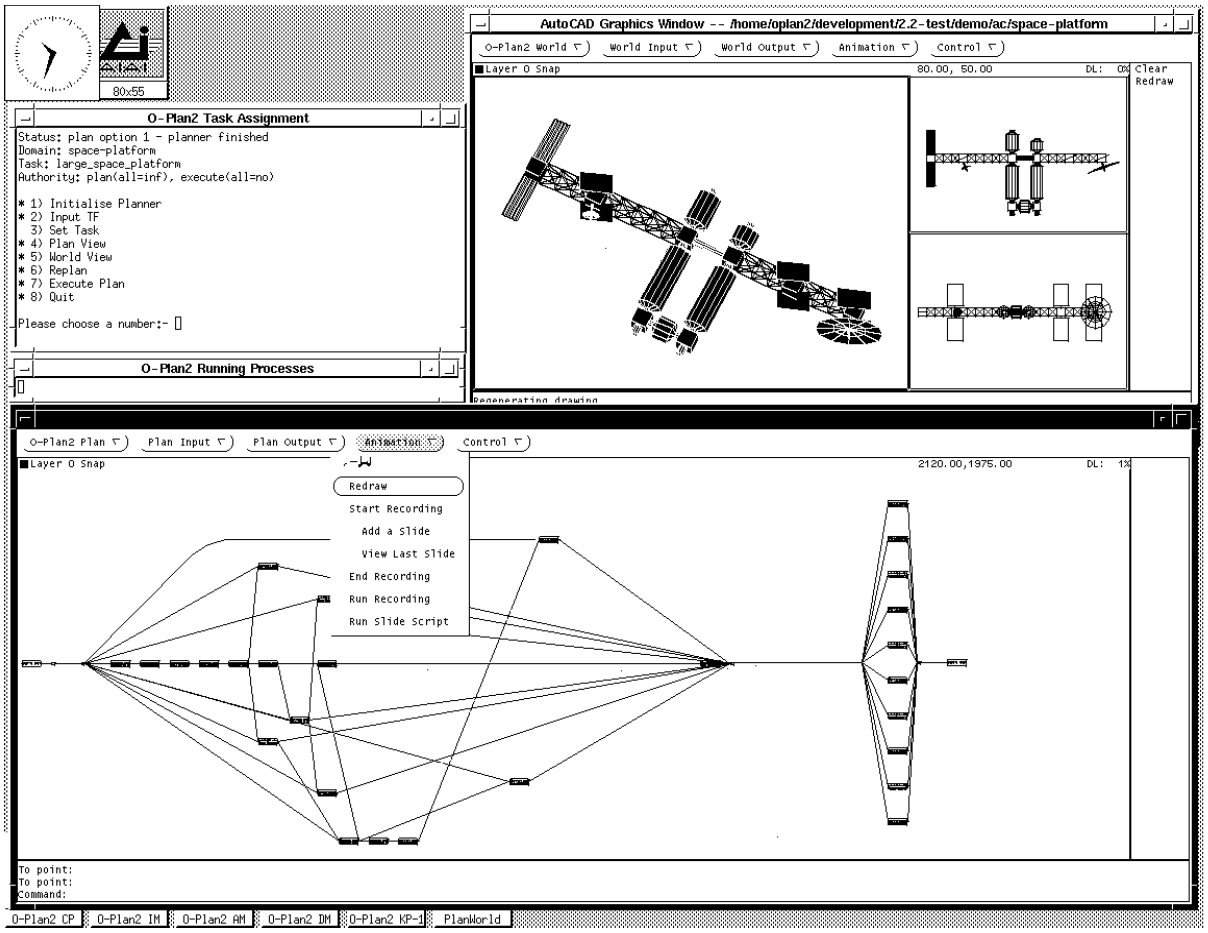
```
Status: plan option 1 - planner initialised
Domain: none
Task: none
Authority: plan(all=inf), execute(all=no)

* 1) Initialise Planner
* 2) Input TF
  3) Set Task
  4) Plan View
  5) World View
  6) Replan
  7) Execute Plan
* 8) Quit

Please choose a number:-
```

```
im:Dumping (:init :ok)
```

Figure 4: Example Developer Interface for the O-Plan2 Planning Agent

4 User Interface

AI planning systems are now being used in realistic applications by users who need to have a high level of graphical support to the planning operations they are being aided with. An interface to AutoCAD [2] has been built to show the type of User Interface we envisage (see Figure 5). The lower window shows a *Plan View* (such as showing the plan as a graph), and the upper right window shows a *World View* for simulations of the state of the world at points in the plan. The particular plan viewer and world viewer provided are declared to the system and the interfaces between these and the planner uses a defined interface to which various implementations can conform.



```
Status: plan option 1 - planner finished
Domain: space-platform
Task: large_space_platform
Authority: plan(all=inf), execute(all=no)

* 1) Initialise Planner
* 2) Input TF
  3) Set Task
* 4) Plan View
* 5) World View
* 6) Replan
* 7) Execute Plan
* 8) Quit

Please choose a number:- []
```

```
Regenerating drawing
```

```
To point:
To point:
Command:
```

Figure 5: Example Output of the AutoCAD-based User Interface

5 Modularity, Interfaces and Protocols

5.1 O-Plan2 Components

The O-Plan2 project has sought to identify modular components within an AI command, planning and control system and to provide clearly defined interfaces to these components and modules.

The main components are:

1. Domain Information - the information which describes an application domain and a tasks in that domain to the planner.
2. Plan State - the emerging plan to carry out identified tasks.
3. Knowledge Sources - the processing capabilities of the planner (*plan modification operators*).
4. Support Modules - functions which support the processing capabilities of the planner and its components.
5. Controller - the decision maker on the *order* in which processing is done.

5.2 Support Modules

Support modules are intended to to provide efficient support to a higher level where decisions are taken. They should not take any decision themselves. They are intended to provide complete information about the questions asked of them to the decision making level itself. The support modules normally act to manage information and constraints in the plan state. Examples of Support Modules in O-Plan2 include:

- Effect/Condition (TOME/GOST) Manager and Question Answering (QA) [7]
- Resource Utilisation Manager
- Time Point Network Manager [5]
- Object Instantiation (Plan State Variables) Manager
- Alternatives Manager
- Interface and Event Manager
- Instrumentation
- Monitors for output messages, etc.

A guideline for the provision of a good support module in O-Plan2 is the ability to specify the calling requirements for the module in a precise way (i.e. the *sensitivity rules* under which the support module should be called by a knowledge source or from a component of the architecture).

5.3 Protocols

In addition, a number of external interface specification and protocols for inter-module use have been established. Only first versions of these interfaces have been established at present, but we believe that further development and enhancement of the planner can take place through concentrating effort on the specification of these interfaces. This should greatly assist the process of integrating new work elsewhere into the planning framework too.

The protocols for regulating the processing conducted by a component of O-Plan2 are:

1. *Knowledge Source Protocol* for the ways in which a Knowledge Source is called by the Controller, can run and can return its results to the Controller and for the ways in which a Knowledge Source can access the current plan state via the Data Base Manager.
2. *KS_USER Protocol* for the ways in which the user (in the role of *Planner User*) can assist the planning system via a specially provided knowledge source.
3. *Inter-agent Communications Protocol* controls the way in which the Knowledge Sources operate and may use the Interface Manager's support routines which control the agent's input and output event channels.

5.4 Internal Support Facilities

The internal support provided within the planner to assist a System Developer and Knowledde Source writer includes:

1. *Knowledge Source Framework* (KSF) is a concept for the means by which information about a Knowledge Source can be provided to an agent. This will ensure that a suitable Knowledge Source Platform is chosen when a Knowledge Source is run inside an agent. It will also allow a model of the capabilities of other agents to be maintained. The KSF will also allow for triggers to be set up for releasing the Knowledge Source for (further) processing. It will allow a description of the parts of a plan state which can be read or altered by each stage within the knowledge source (to allow for effective planning of concurrent computation and data base locking in future).
2. *Agenda Trigger Language* gives a Knowledge Source writer the means by which a computation can be suspended and made to await some condition. The conditions could relate to information within the plan, for external events or for internally triggered Diary events. O-Plan currently provides a limited number of monitorable triggers of this kind, but we anticipate this being expanded significantly in future.
3. *Controller Priority Language* currently, the O-Plan2 Controller selects agenda entries based on a numerical priority which is simply a statically computed measure of the priority of outstanding agenda entries in a plan state. Our aim for the future is to provide a rule based controller which can make use of priority information provided in the form of rules in an O-Plan2 Controller Priority Language. This concept will allow us to clarify our ideas on

what informatio should govern controller ordering decisions. Domain information linking to generic Controller Language statements which can affect the controller decisions is likely to be considered as part of a link between Task Formalism (TF) and the operation of the Controller.

5.5 External Interfaces

The external interfaces provided by the planner are:

1. *Task Formalism* (TF) as the language in which an application domain and the tasks in it can be expressed to the planner.
2. *Plan Viewer User Interface* which allows for domain specific plan drawing and interaction to be provided.
3. *World Viewer User Interface* which allows for domain specific world state input and simulation facilities to be provided.
4. *External System Interface* provided by TF **compute conditions** [6] for ways in which external data bases, modelling systems, CAD packages, look-up tables, etc can be used and for ways in which these external systems can access plan information and provide qualifications on the continued validity of their results if appropriate.

6 Summary

This paper has presented an overview of the O-Plan2 system under development at the Artificial Intelligence Applications Institute of the University of Edinburgh. Aspects of the system concerned with separation of functionality within the system, internal and external interfaces have been addressed. The O-Plan2 system is starting to address the issue of what support is required to build an evolving and flexible architecture to support command, planning and control tasks.

Acknowledgements

The O-Plan2 project has been 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-C0081 (EOARD/88-0044) monitored by Dr Nort Fowler at the USAF Rome Laboratory. Additional resources for the O-Plan and O-Plan2 projects have been provided by the Artificial Intelligence Applications Institute through the EUROPA (Edinburgh University Research on Planning Architectures) Institute development project. Thanks are due to my colleagues on the O-Plan2 project - Dr Brian Drabble and Richard Kirby.

References

[1] Allen, J., Hendler, J. & Tate, A. Readings in Planning. *Morgan-Kaufmann* 1990.

[2] AutoDesk AutoCAD Reference Manual, 1989.

[3] Currie, K.W. & Tate, A. O-Plan: the Open Planning Architecture, *Artificial Intelligence* Vol 51, No. 1, Autumn 1991, North-Holland.

[4] Tate, A. Generating project networks. *In procs.* IJCAI-77*, 1977.*

[5] Drabble, B. & Kirby, R. Associating AI Planner Entities with an Underlying Time Point Network, Proceedings of the First European Workshop on Planning, St. Augustin, Germany, March 1991.

[6] Tate, A. (1984) *Planning and Condition Monitoring in a FMS*, Proceedings of the International Conference on Flexible Automation Systems, Institute of Electrical Engineers, London, UK.

[7] Tate, A. (1986) Goal Structure, Holding Periods and "Clouds", Proceedings of the Reasoning about Actions and Plans Workshop, Timberline Lodge, Oregon, USA. Eds, Georgeff, M.P. and Lansky, A. Published by Morgan Kaufmann.