O-Plan2: an Open Architecture for Command, Planning and Control

Austin Tate, Brian Drabble and Richard Kirby

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

1 Introduction

O-Plan2 (the Open Planning Architecture) provides a generic domain independent computational architecture suitable for command, planning and execution applications. The main contribution of the O-Plan2 research has been a complete vision of a modular and flexible planning and control system incorporating artificial intelligence methods.

This paper describes the O-Plan2 agent oriented architecture and describes the communication which takes place between planning and execution monitoring agents built upon the architecture. Separate modules of such a system are identified along with internal and external interface specifications that form a part of the design.

Time constraints, resource usage, object selection and condition/effect causal constraints are handled as an integral part of the overall system structure by treating specialised constraint management as supporting the core decision making components in the architecture. A close coupling of planning and time or resource scheduling is therefore possible within a system employing an activity based plan representation.

2 History and Technical Influences

O-Plan grew out of the experiences of other research into AI planning, particularly with Nonlin [28] and "blackboard" systems [20]. The *Readings in Planning* volume [1] includes a taxonomy of earlier planning systems which places O-Plan in relation to the influences on its design. It is assume that the reader is familiar with these works as the references do not include them all. The same volume [1] includes an introduction to the literature of AI planning.

The main AI planning techniques which have been used or extended in O-Plan are:

- A hierarchical planning system which can produce plans as partial orders on actions (as suggested by Sacerdoti [23]), though O-Plan is flexible concerning the order in which parts of the plan at different levels are expanded.
- 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 (HEARSAY-II [16] and OPM [15] uses the term *Knowledge Source* for these handlers).

- 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. This is similar to the work of McDermott [19].
- Constraint posting and least commitment on object variables as seen in MOLGEN [35].
- Temporal and resource constraint handling, shown to be valuable in realistic domains by Deviser [36], has been extended to provide a powerful search space pruning method. The algorithms for this are incremental versions of Operational Research (OR) methods. O-Plan has integrated ideas from OR and AI in a coherent and constructive manner.
- O-Plan is derived from the earlier Nonlin planner [28] from which we have taken and extended the ideas of Goal Structure, Question Answering (QA) and typed preconditions.
- We have maintained Nonlin's style of domain and task description language (Task Formalism or TF) and extended it for O-Plan2.

2.1 O-Plan1

The main effort on the first O-Plan project (now referred to as O-Plan1) was concentrated in the area of plan generation. The work on O-Plan1 is documented in a paper in the *Artificial Intelligence Journal* [5]. One theme of the O-Plan1 research was search domain knowledge based space control in an AI planner. The outputs of that work gave a better understanding of the requirements of planning methods, improved heuristics and techniques for search space control, and a demonstration system embodying the results in an appropriate framework and representational scheme.

O-Plan1 sought to build an open architecture for an AI planning system. It was our aim to build a system in which it was possible to experiment with and integrate developing ideas. Further, the system was to able to be tailored to suit particular applications. Time and resource constraints were handled to restrict search while still working within an activity based plan representation.

2.2 O-Plan2

The O-Plan2 project began in 1989 and had the following new objectives:

- to consider a simple "three agent" view of the environment for the research to clarify thinking on the roles of the user(s), architecture and system. The three agents being the job assignment agent, the planning agent and the execution agent.
- to explore the thesis that communication of capabilities and information between the three agents could be in the form of *plan patches* which in their turn are in the same form as the domain information descriptions, the task description and the plan representation used within the planner and the other two agents.

- to investigate a single architecture that could support all three agent types and which could support different plan representations and agent capability descriptions to allow for work in activity planning or resource scheduling.
- to clarify the functions of components of a planning and control architecture.
- to draw on the earlier Edinburgh planning experience in O-Plan1 [5] and to improve on it especially with respect to flow of control [32].
- to provide an improved version of the O-Plan1 system suitable for use outside of Edinburgh within Common Lisp, X-Windows and UNIX.
- to provide a design suited to use on parallel processing systems in future.

This paper gives an overview of the O-Plan2 architecture and its use in a prototype planning system. Further details of the system are available in [33].

3 Characterisation of O-Plan2

The O-Plan2 approach to command, planning, scheduling and control can be characterised as follows:

- successive refinement/repair of a complete but flawed plan or schedule
- least commitment approach
- using opportunistic selection of the focus of attention on each problem solving cycle
- building information incrementally in "constraint managers", e.g.
  - effect/condition manager
  - resource utilisation manager
  - time point network manager
  - object/variable manager
- using localised search to explore alternatives where advisable
- with global alternative re-orientation where necessary.

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 such as Voyager, ERS-1, etc.

These applications fit midway between the large scale manufacturing scheduling problems found in some industries (where there are often few inter-operation constraints) and the complex *puzzles* dealt with by very flexible logic based tools. However, the problems of the target type represent an important class of industrial, scientific and engineering relevance.

4 Communication in Command, Planning and Control

4.1 The Scenario

The scenario we are investigating is as follows:

- 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 planner has knowledge of the general capabilities of a semi-autonomous execution system but does not need to know about the actual activities that execute the actions required to carry out the desired task.
- 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 than is available to the job assigner and to the planner.

The central planner therefore communicates a general plan to achieve a particular task, and responds to failures fed back from the execution agent which are in the form of flaws in the plan. Such failures may be due to the inappropriateness of a particular activity, or because the desired effect of an activity was not achieved due to an unforeseen event. The reason for the failure dictates whether the same activity should be re-applied, replaced with other activities or whether re-planning should take place.

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.

4.2 A Common Representation for Communication between Agents

We have been exploring a common representation to support the communication between a user, requesting the plan, and the real world, in which the plan is being executed. Such communication may take place either directly through a planner or indirectly via a central planner and a dumb or semi-autonomous execution agent.

The common representation includes knowledge about the capabilities of the planner and execution agent, the requirements of the plan and the plan itself either with or without flaws

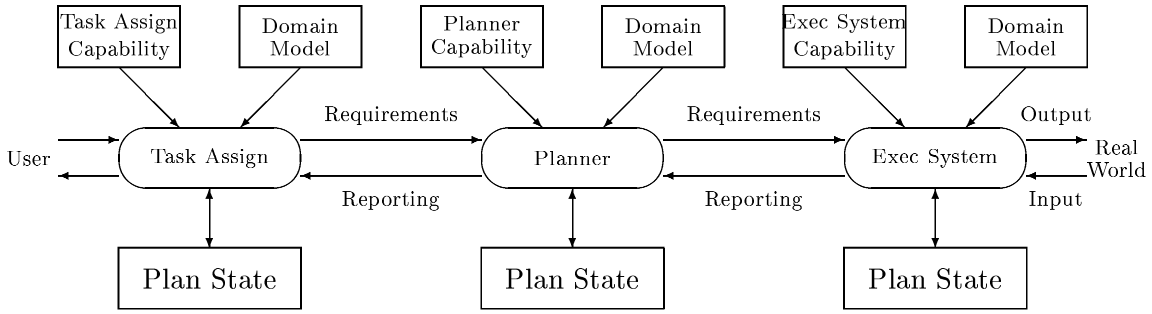

Figure 1: Communication between Central Planner and Ex. Agent

(see Figure 1). Thus, a planner will respond to the requirements of a user. Based on the knowledge of its own capabilities and that of the execution environment, it will generate a plan. This plan may then be executed directly in the real world, or, indirectly via an execution agent. The execution agent executes this plan in the real world and monitors the execution, responding to failures in one of two ways. If it does not have knowledge of its own capabilities, it simply returns knowledge of the failure to the central planner and awaits a revised plan to be sent. In this case, the execution agent is dumb. If it does have knowledge of its own capabilities, it may attempt to repair the plan and then continue with execution. On the other hand, if a repair is beyond the capabilities of the execution agent, then this knowledge is fed back to the central planner and again a revised plan is expected. In this case, the execution agent is semi-autonomous. When failures during the application of the plan are fed back to the planner, these may be acted upon by it and a repair of the plan made or total re-planning instigated. This may, in turn, involve the user in reformulating the task requirement. A revised or new plan is then executed. Finally, success of the execution or partial execution of the plan is fed back to the user.

The communication of task, plan and execution information between agents is in the form of *plan patches* since it is assumed that each agent is operating asynchronously with its own plan state and model of the environment. Further details are given in [31].

5 O-Plan2 Architecture

This section describes the O-Plan2 architecture and describes the major modules which make up the system. An agenda based architecture has been used as the central feature of the system and the design approach. Within this framework there has been consideration of choice enumeration, choice ordering, choice making and choice processing. This is important as it allows us to begin to justifiably isolate functionality which can be described in terms of:

- triggering mechanisms — *i.e.* what causes the mechanism to be activated.
- decision making roles — precisely what type of decision can be made.

- implications for search — has the search space been pruned, restricted or further constrained as far as possible.
- decision ordering — in what order should we choose between the alternative decisions possible.
- choice ordering — for a decision to be made, which of the open choices should we adopt.

The main components of an O-Plan2 agent are:

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

A generalised picture of the architecture illustrated with the components to specialise the architecture to be a planning agent is shown in Figure 2. Further details of each component follows in subsequent sections. In these sections, illustrations of the contents of the main components are made by referring to the parts of a planning agent.

5.1 Domain Information

Domain descriptions are supplied to O-Plan2 in a language called Task Formalism (TF). This is compiled into the internal data structures to be used during planning. A TF description includes details of:

1. activities and events which can be performed or occur in the domain.
2. information about the environment and the objects in it.
3. task descriptions which describe the planning requirements.

TF is the means through which a domain writer or domain expert can supply the domain specific information to the O-Plan2 system, which itself is domain *independent*. O-Plan2 embodies search space pruning mechanisms using this domain information (strong search methods) and will fall back on other weak methods, if these fail. TF is the mechanism that enables the user of the system to supply domain dependent knowledge to assist the system in its search.

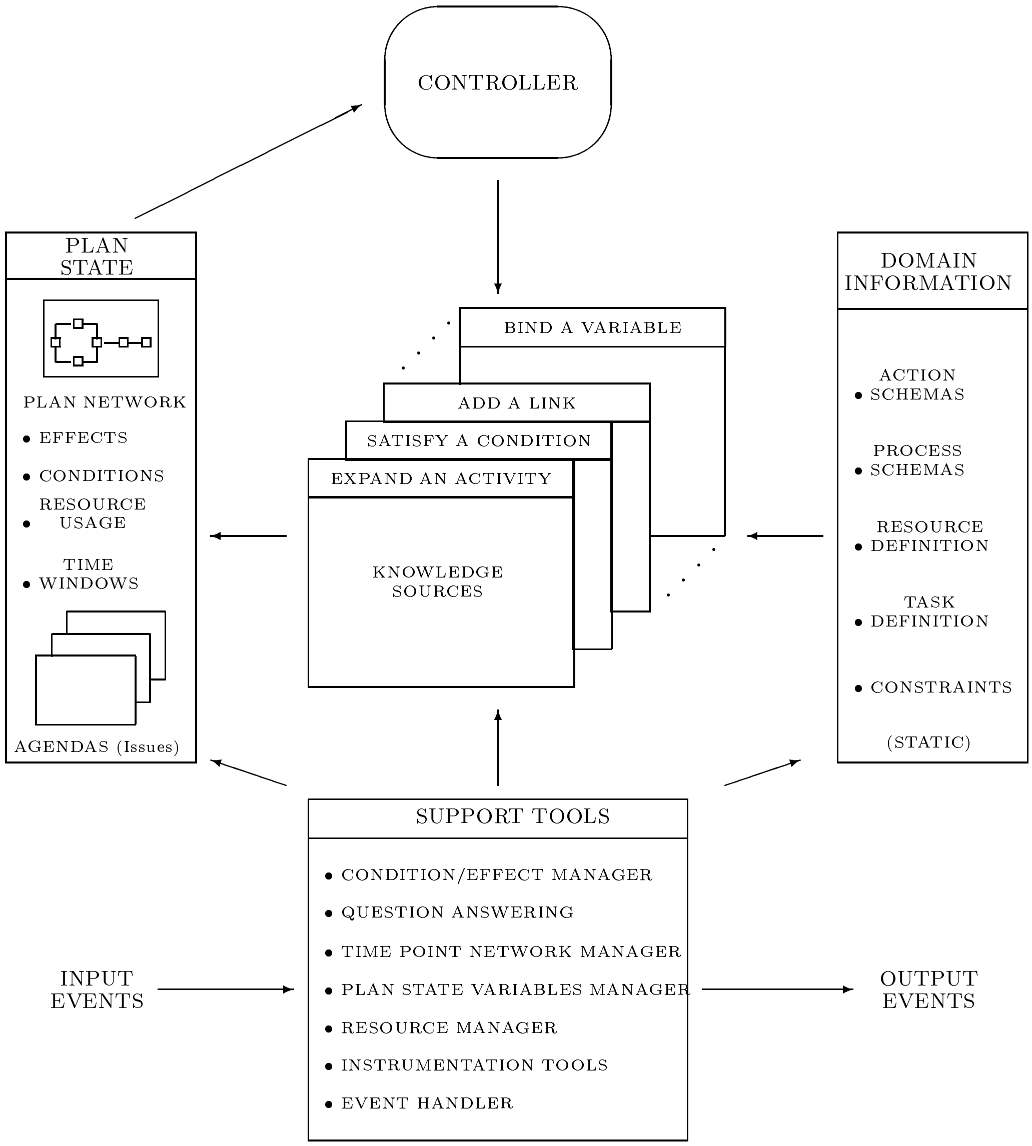

Figure 2: O-Plan2 Architecture

5.2 Plan State

In contrast to the relatively static information outlined above, the plan state (on the left of Figure 2) is the dynamic data structure used during planning and houses the emerging plan. There are a number of components to this structure, the principal ones being:

- the plan network itself. This is based on a partial order of activities, as originally suggested in the NOAH planner [23]. In O-Plan2 the plan information is concentrated in the "Associated Data Structure" (ADS). The ADS contains node and link structures noting temporal and resource information, plan information, etc.
- the plan rationale. As in Nonlin and O-Plan1, the system keeps explicit information to "explain" why the plan is built the way it is. This rationale is called the Goal Structure (GOST) and, along with the Table of Multiple Effects (TOME), provides an efficient data structure for the condition achievement support module used in O-Plan2 (Question Answerer – QA – *c.f.* Chapman's Modal Truth Criteria [3]).
- the agenda. O-Plan2 starts with a complete plan, but one which is "flawed", hence preventing the plan from being capable of execution. The nature of the flaws present will be varied, from actions which are at a higher level than that which the executing agent can operate, to linkages necessary in the plan to resolve conflict. Some agenda entries can represent potentially beneficial, but not yet processed, information. The agenda is the repository for this "pending" information which must be processed in order to attain an executable plan.

The plan state is a self-contained snapshot of the state of the planning system at a particular point in time in the plan generation process. It contains all the state of the system hence the generation process can be suspended and this single structure rolled back at a later point in time to allow resumption of the search[1].

5.3 Knowledge Sources

These are the computational capabilities associated with the processing of the flaws contained in the plan and they embody the planning knowledge of the system. There are as many Knowledge Sources (KS) as there are flaw types, including the interface to the user wishing to exert an influence on the plan generation process. A KS can draw on domain information (*e.g.* the use of an action schema for purposes of expansion) to process a flaw, and in turn they can add structure to any part of the plan state (*e.g.* adding ordering links to the plan, inserting new effects or further populating the agenda with flaws).

[1]Assuming that the Task Formalism and the Knowledge Sources used on re-start are the same "static" information used previously.

5.4 Support Modules

In order to efficiently support the main planning functionality in O-Plan2 there are a number of support modules separated out from the core of the planner. These modules have carefully designed functional interfaces in order that we can both build the planner in a piecewise fashion, and in particular that we can experiment with and easily integrate new implementations of the modules. The modularity is possible only through the experience gained in earlier planning projects where support function requirements were carefully separated out from the general problem solving and decision making demands of the system.

Support modules are intended to provide efficient support to the higher level Knowledge Sources 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. Some support modules act as *constraint managers* for a sub-set of the plan state information.

The support modules include the following:

- Question-Answerer (QA) is the process at the heart of O-Plan2's condition satisfaction procedure. It can establish whether a proposition is true or not at a particular point in the plan. The answer it returns may be (i) a categorical "yes", (ii) a categorical "no", or (iii) a "maybe", in which case QA will supply an alternative set (structured as an and/or tree) of strategies which a Knowledge Source can choose from in order to ensure the truth of the proposition. The QA procedure makes use of the information managed by the time point network and condition/effect constraint management components (see below) to filter the answers provided to the decision making level above.
- Time Point Network Manager (TPN) to manage metric and relative time constraints in a plan.
- TOME and GOST Manager (TGM) to manage the causal structure (conditions and effects which satisfy them) in a plan.
- Plan State Variable Manager to manage partially bound objects in the plan.
- Resource Utilisation Manager to monitor and manage the use of resources in a plan.
- Instrumentation and Diagnostics routines. O-Plan2 has a set of routines which allow the developer to set and alter levels of diagnostic reporting within the system. These can range from full trace information to fatal errors only. The instrumentation routines allow performance characteristics to be gathered while the system is running. Information such as how often a routine is accessed, time taken to process an agenda entry, etc. can be gathered.

5.5 Controller

Holding this loosely coupled framework together is the Controller acting on the agenda. Items on the agenda (the flaws) have a context dependent priority which the Controller can re-compute, and which allows for the opportunism required to drive plan generation. Agenda entries can be triggered by specific plan state changes or other events, such as the binding of a variable, the satisfaction of a condition, the occurrence of an external event, a reminder from an internal agent diary, etc.

The controller also provides the framework to activate Knowledge Sources on Knowledge Source Platforms and to give them appropriate access to domain and plan information. Further details of the choice ordering mechanisms in O-Plan2 is given in [32].

The controller provides facilities for managing alternative plan states for internal search within an O-Plan2 agent where this is feasible.

6 Process Structure of the O-Plan2 Implementation

The current O-Plan2 prototype 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 abstract architecture described in Figure 2 can be mapped to the system and process architecture detailed in Figure 3 which shows the specialisation of the architecture to the O-Plan2 planner agent. Communication between the various processes and managers in the system is shown. Each entry within the Figure is explained later in this section.

The basic processing cycle of O-Plan2 (as illustrated by the planner agent) is as follows:

1. An event is received by the Event Manager which resides within the Interface Manager (IM) process. The IM is in direct contact with all other processes of the architecture through the Module Communication Channel (MCC)[2]. Support modules allow the developer to change levels of diagnostics and to set up instrumentation checks on the planner. The Event Manager has two Guards, one on the left input channel (from the job assigner) and one on the right input channel (from the execution system). The input channels themselves are separated into priority levels.

   The guards verify and if necessary reject events which are not relevant to the system. The guards use knowledge of the system's capabilities derived from the Knowledge Sources and domain model (TF) currently loaded into the system.

2. If the event is approved by the guard then it is passed to the Controller/Agenda Manager (AM) which assigns it the necessary triggers and Knowledge Source activation entry. The entry (now referred to as an Agenda Entry) is then passed to the Database Manager (DM) to await triggering. The entry is placed in the Agenda Table (AT) monitored by the Trigger Detector (TD).

[2] The MCC is not shown in Figure 3 to simplify the diagram.

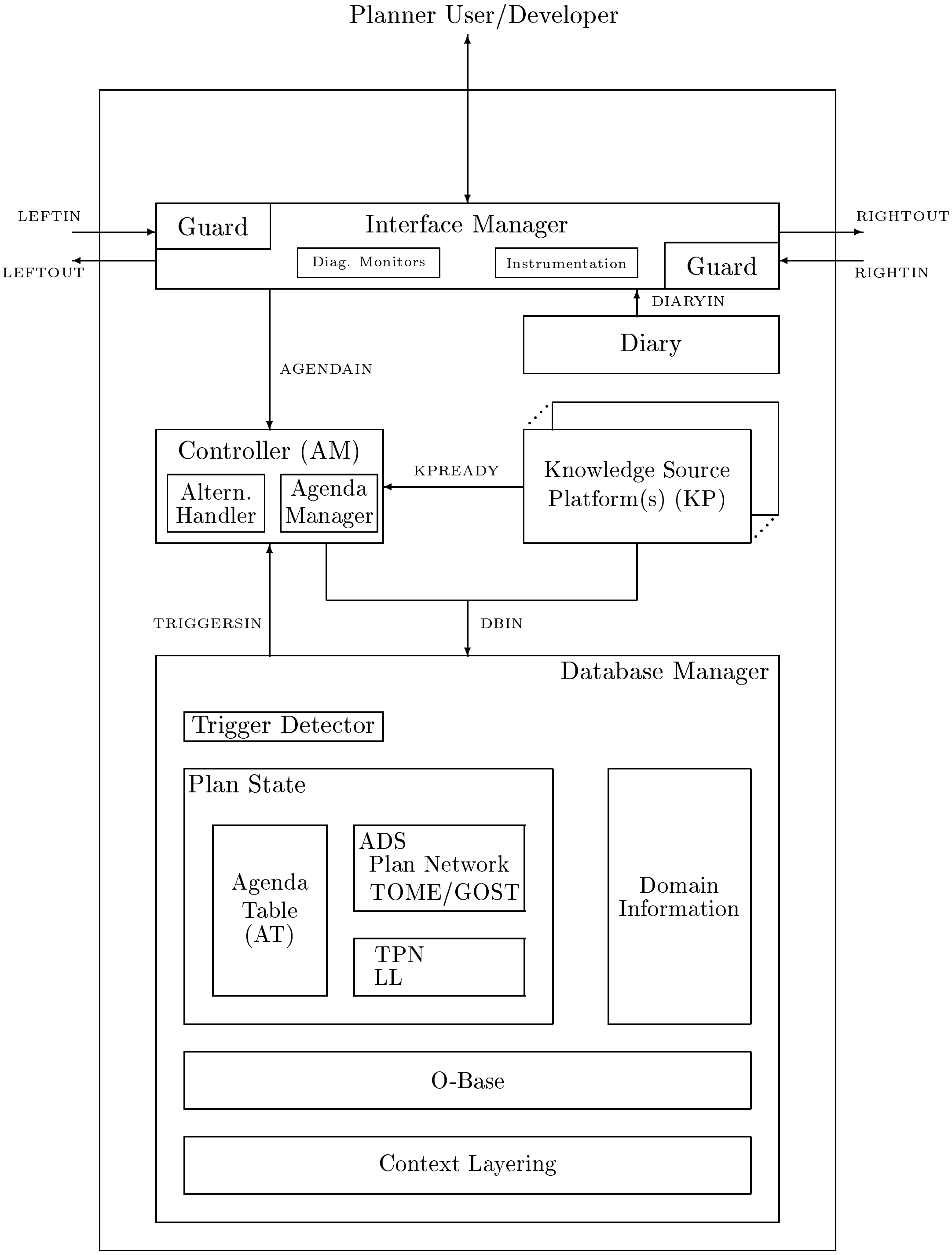

Figure 3: Internal Structure of the Current O-Plan2 Planner

3. When triggered, the Trigger Detector informs the Agenda Manager and may *cache* a copy of the triggered agenda entry in the Agenda Manager. The order of entries on the triggered agenda is constantly updated as new agenda entries are added or triggers on waiting agenda entries become invalid. A trigger can become invalid due to its triggering condition ceasing to hold.

   Knowledge Sources can use the Diary Manager functions to assist them to perform their task. The Diary Manager (DIARY) is responsible for handling triggers associated with a given time. For example, send an action for execution at a specific time or trigger a regular event.

   Eventually the agenda entry is selected for processing by the Controller/Agenda Manager.

4. The Controller/Agenda Manager assigns an available Knowledge Source Platform (KP) which can run the pre-nominated Knowledge Source on the triggered agenda entry.

5. When a Knowledge Source Platform has been allocated, if it does not already contain the nominated Knowledge Source, the Platform may request the body of the Knowledge Source from the Database Manager, in order to process the agenda entry. Knowledge Sources may be preloaded on the Platform so this request is not necessary in all cases. Some Platforms may be best suited to run particular Knowledge Sources, hence the system will not store all Knowledge Sources at all Platforms. The Knowledge Source Platforms will eventually have their own local libraries of Knowledge Sources. Locking down of a specific real time Knowledge Source to a dedicated Platform is allowed for in the design.

6. A protocol (called the Knowledge Source Protocol) and an access key are used to control communication between the Controller/Agenda Manager and a Knowledge Source running on a Platform. This controls the processing which the Knowledge Source can do and the access it has to the current plan state via the Database Manager (DM).

   A Knowledge Source can terminate with none, one or multiple alternative results through interaction with the Controller via the protocol. The Controller uses an Alternatives Manager Support Module to actually manage any alternatives it is provided with and to seek alternatives when no results are returned by a Knowledge Source. A Knowledge Source can also be asked to terminate at suitable internal "stage" boundaries by the Controller[3].

The internal details of the Database Manager (DM) will depend upon the particular representation chosen for the Plan State. In Figure 3 the internal details of the Database Manager relate to the O-Plan2 planner. Here there is a separation of the Associated Data Structure (ADS) level which describes the plan network, the Table of Multiple Effects (TOME) and the Goal Structure (GOST) from the lower level time constraint management done via the Time Point Network (TPN) and its associated metric time point list called the Landmark Line (LL) and the underlying resource constraint management (done via a Resource Utilisation Manager).

[3] O-Plan2 Knowledge Sources can comprise a number of separate stages where suspension of processing can occur at any stage boundary.

7 O-Plan2 Planner

The O-Plan2 planner agent has been the main focus of our work to date. The following sections describe the ways in which the generic O-Plan2 architecture has been specialised for this planner.

7.1 Plan State

The planning agent plan state holds information about decisions taken during planning and information about decisions which are still to be made (in the form of an agenda).

7.1.1 Plan Network

The Associated Data Structure (ADS) provides the *plan entities* which define the plan as a set of activity and event nodes with ordering information in the form of links as necessary to define the partial order relationships between these elements. The end points of these plan entities are associated with a lower level Time Point Network (TPN). Effects, conditions, time windows and resource utilisation information is also attached to the nodes at the ADS level.

Time windows play an important part in O-Plan2 in two ways: firstly as a means of recording time limits on the start and finish of an action and on its duration and delays between actions, and secondly during the planning phase itself as a means of pruning the potential search space if temporal validity is threatened.

Time windows in O-Plan2 are maintained as **min/max** pairs, specifying the upper and lower bounds known at the time. Such bounds may be symbolically defined, but O-Plan2 maintains a numerical pair of bounds for all such numerical values. In fact, a third entry is associated with such numerical bounds[4]. This third entry is a *projected* value (which could be a simple number or a more complex function, data structure, etc.) used by the planner for heuristic estimation, search control and other purposes. The numerical outer bounds on time windows which are maintained by the Time Point Network Manager are used in the QA process at the heart of the planner and, if there are tight time constraints on a plan, they can effectively prune valid responses for ways to satisfy conditions or correct for interactions between conditions and effects.

7.1.2 TOME and GOST

The Table of Multiple Effects (TOME) holds statements of form:

```
fn(arg1 arg2 ...) = value at time-point
```

The Goal Structure (GOST) holds statements of form:

[4]All numerical values in O-Plan2 are held as triples: minimum, maximum, and projected values.

```
<condition-type> fn(arg1 arg2 ...) = <value> at <time-point>
                                     from <contributor-list>

where <contributor-list> is a set of pairs of format:
(<time-point> . <method-of-satisfaction-of-condition>)
```

In the current implementation, effects and conditions are kept in a simple pattern directed lookup table as in Nonlin [28]. The O-Plan1 *Clouds* mechanism [30] for efficiently manipulating large numbers of effects and their relationship to supporting conditions will be used in O-Plan2 in due course.

7.1.3 Plan State Objects and Variables

O-Plan2 can keep restrictions on plan state objects without necessarily insisting that a definite binding is chosen as soon as the object is introduced to the Plan State. Plan State Variables can be used in effects, conditions, etc.

7.1.4 Resource Utilisation Table

The Resource Utilisation Table holds statements of form:

```
set/+/- resource(<resource-name> <qualifier> ...) = <value>
                                                 at <time-point>
```

The statement declares that the particular resource is set to a specific value or changed by being incremented or decremented by the given value at the indicated time point. There can be uncertainty in one or both of the value and the time point which are held as **min/max** pairs.

Task Formalism resource usage specifications on actions are used to ensure that resource usage in a plan stays within the bounds indicated. There are two types of resource usage statements in TF. One gives a *specification* of the **overall** limitation on resource usage for an activity (over the total time that the activity and any expansion of it can span). The other type describes actual resource *utilisation* **at** points in the expansion of an action. It must be possible (within the min/max flexibility in the actual resource usage statements) for a point in the min/max range of the sum of the resource usage statements to be within the overall specification given. The Resource Utilisation Table is used to manage the actual resource utilisation **at** points in the plan.

7.2 Planning Knowledge Sources

The O-Plan2 architecture is specialised into a planning agent by including a number of Knowledge Sources which can alter the Plan State in various ways. The planning Knowledge

Sources provide a collection of *plan modification operators* which define the functionality of the planning agent beyond its default O-Plan2 architecture properties (essentially limited to initialisation and communication capabilities by default).

The planning Knowledge Sources in the current version of the O-Plan2 planner includes:

- KS_SET_TASK a Knowledge Source to set up an initial plan state corresponding to the task request from the job assignment agent.
- KS_EXPAND a Knowledge Source to expand a high level activity to lower levels of detail.
- KS_CONDITION a Knowledge Source to ensure that certain types of condition are satisfied. This is normally posted by a higher level KS_EXPAND.
- KS_ACHIEVE a Knowledge Source initiated by KS_EXPAND to achieve conditions possibly by inserting new activities into the plan.
- KS_OR a Knowledge Source to select one of a set of possible alternative linkings and plan state variable bindings. The set of alternative linkings and bindings will have been created by other Knowledge Sources (such as KS_CONDITION) earlier – normally as a result of a Question Answerer (QA) call.
- KS_BIND a Knowledge Source used to select a binding for a plan state variable in circumstances where alternative possible bindings remain possible.
- KS_USER a Knowledge Source activated at the request of the user acting in the role of supporting the planning process. This is used at present to provide a menu to browse on the plan state and potentially to alter the priority of some choices.
- KS_POISON_STATE a Knowledge Source used to deal with a statement by another Knowledge Source that the plan state is inconsistent in some way or cannot lead to a valid plan (as far as that Knowledge Source is aware).

In addition, the default Knowledge Sources available in any O-Plan2 agent are present and are as follows:

- KS_INIT Initialise the agent.
- KS_COMPILE Alter the Knowledge Source (*agent capability*) Library of an O-Plan2 agent by providing new or amended Knowledge Sources (described in a *Knowledge Source Framework* language). In the current implementation of O-Plan2, this cannot be done dynamically.
- KS_DOMAIN Call the Domain Information (normally TF) compiler to alter the Domain Information available to the agent.
- KS_EXTRACT_RIGHT Extract a plan patch for passing to the subordinate agent to the 'right' of this agent - i.e the execution agent.

- KS_EXTRACT_LEFT Extract a plan patch for passing to the superior agent to the 'left' of this agent - i.e the job assignment agent.
- KS_PATCH Merges a plan patch from an input event channel into the current plan state.

7.3 Use of Constraint Managers to Maintain Plan Information

The O-Plan2 planner uses a number of *constraint managers* to maintain information about a plan while it is being generated. The information can then be utilised to prune search (where plans are found to be invalid as a result of propagating the constraints managed by these managers), to restrict the range of valid answers provided by the Question Answerer (QA) procedure in the planner, or to order search alternatives according to some heuristic priority. The constraint managers are provided as a collection of *support modules* which can be called by Knowledge Sources to maintain specialised aspects of the information in a plan or to answer queries based upon this information.

7.3.1 Time Point Network Manager (TPNM)

O-Plan2 uses a point based temporal representation with range constraints between time points and with the possibility of specifying range constraints relative to a fixed time point (time zero). This provides the capability of specifying relative and metric time constraints on time points. The functional interface to the Time Point Network (TPN), as seen by the Associated Data Structure (ADS) has no dependence on a particular representation of the plan state. Further details are given in [8].

The points held in the TPN may be indirectly associated with actions, links and events, with the association being made at the Associated Data Structure level. The points are numbered to give an index with a constant retrieval time for any number of points. This structure allows points to be retrieved and compared through a suitable module interface and with a minimum of overhead. The interface reflects the *functionality* required of the TPN, and hides the detail. This ensures that we have no absolute reliance on points as a necessary underlying representation. Time points whose upper and lower values has converged to a single value are inserted into a time ordered Landmark Line (LL). This allows the planner to quickly check the order of certain points within the plan. The TPN and LL are maintained by the Time Point Network Manager (TPNM). As well as its use in the O-Plan2 activity orientated planner, the current TPNM has also been applied to large resource allocation scheduling problems in the TOSCA scheduler [2] where the number of time points was in excess of 5000 and the number of temporal constraints exceeded 3000.

7.3.2 TOME/GOST Manager (TGM)

The conflict free addition of effects and conditions into the plan is achieved through the TGM, which relies in turn on support from the Question Answerer (QA) module which suggests resolutions for potential conflicts. The resolutions proposed are sensitive to metric time constraints as managed by the Time Point Network Manager.

7.3.3 Resource Utilisation Management (RUM)

O-Plan2 uses a Resource Utilisation Manager to monitor resource levels and utilisation. Resources are divided into different types such as:

1. Consumable: these are resources which as "consumed" by actions within the plan. For example: bricks, fuel, money, etc.

2. Re-usable: these are resources which are used and then returned to a common "pool". For example, robots, workmen, lorries, etc.

Consumable resources can be subcategorised as *strictly consumed* or may be *producable* in some way. Substitutability of resources one for the other is also possible. Some may have a single way mapping such as money for fuel and some can be two way mappings such as money for travellers' cheques. Producable and substitutable resources are difficult to deal with because they *increase* the amount of choice available within a plan and thus *open up* the search space.

The current O-Plan2 Resource Utilisation Manager uses the same scheme for strictly consumable resources as in the original O-Plan1. However, a new scheme based on the maintenance of optimistic and pessimistic resource profiles with resource usage events and activities tied to changes in the profiles is now under study.

7.3.4 Plan State Variables Manager (PSVM)

The Plan State Variable Manager is responsible for maintaining the consistency of restrictions on plan objects during plan generation. O-Plan2 adopts a least commitment approach to object handling in that variables are only bound as and when necessary. The Plan State Variables Manager within the Database Manager (DM) maintains an explicit "model" of the current set of plan object restrictions and seeks to ensure that a possible instantiation of the object is possible at all times.

When a Plan State Variable (PSV) is created by the planner the Plan State Variables Manager creates a plan state variable name (PSVN), plan state variable body (PSVB) and a range list from which a value must be found. For example, the variable could be the colour of a spacecraft's camera filter which could be taken from the range (**red green blue yellow opaque**). A plan state variable must have an enumerable type and thus cannot be, for example, a real number. The PSVB holds the **not-sames** and **constraint-lists** and may be pointed to by one or more PSVNs. This allows easier updating as new constraints are added and PSVB's are made the same. Two or more PSVB's can be collapsed into a single PSVB if all of the constraints are compatible. i.e. the **not-sames** and **constraints-list**. A PSVN pointing to a collapsed PSVB is then redirected to point at the remaining PSVB. This scheme allows triggers to be placed on the binding of PSV's (e.g. do not bind until the choice set is less than 3) and allows variables which are creating bottlenecks to be identified and if necessary further restricted or bound.

7.4 Other Support Modules in O-Plan2

As well as the managers referred to above, a number of other support routines are available for call by the Knowledge Sources of O-Plan2. The main such support mechanisms which have been built into the current O-Plan2 Planner include:

- **Question Answerer (QA)**
  The Question-Answering module is the core of the planner and must be both efficient and able to account for both metric and relative time constraints. QA supports the planner to satisfy and maintain conditions in the plan in a conflict free fashion, suggesting remedies where possible for any interactions detected. The QA procedure makes use of the constraint managers to reduce the number of legal answers it provides.

- **Graph Operations Processor (GOP)**
  The GOP provides efficient answers to ordering related questions within the main plan (represented by a graph). GOP works with metric time ordered and relative or partially ordered activities in the graph.

- **Contexts**
  All data within the O-Plan2 plan state can be "context layered" to provide support for alternatives management and context based reasoning. An efficient, structure sharing support module provides the ability to context layer any data structure accessor and updator function in Lisp. This is particularly useful for the underlying content addressable database in the system: O-Base.

- **O-Base**
  This database support module supports storage and retrieval of entity/relationship data with value *in context*. This model allows for retrieval of partially specified items in the database.

In addition, there are support modules providing support for the User Interface, Diagnostics, Instrumentation, etc.

7.5 Alternatives Manager

There is an additional support module capability in O-Plan2 which is utilised by the Controller. This provides handling of alternative plan states within an O-Plan2 agent.

If a Knowledge Source finds that it has alternatives ways to achieve its task, and it finds that it cannot represent all those alternatives in some way within a single plan state, then the Controller provides support to allow the alternatives that are generated to be managed. This is done by the Knowledge Source telling the Controller about all alternatives but one favoured one and asking for permission to continue to process this. This reflects the O-Plan2 search strategy of *local best, then global best.* A support routine is provided to allow a Knowledge Source to inform the Controller of all alternatives but the selected one.

A Knowledge Source which cannot achieve its task or which decides that the current plan state is illegal and cannot be used to generate a valid plan may terminate and tell the Controller to poison the plan state. In the current version of O-Plan2, this will normally initiate consideration of alternative plan states by a dialogue between the Controller and the alternatives manager. A new current plan state will be selected and become visible to new Knowledge Source activations. Concurrently running Knowledge Sources working on the old (poisoned) plan state will be terminated as soon as possible as their efforts will be wasted.

As well as having the existing system's option to explore alternative plan states, future versions of O-Plan2 will consider ways to *unpoison* a plan state by running a nominated *poison handler* associated with the Knowledge Source that poisoned the plan state or with the *reason* for the plan state poison. This is important as we envisage O-Plan2 being used in continuous environments where alternative plan states will become invalid.

7.6 Implementation as Separate Processes

In the current UNIX and Common Lisp based implementation of O-Plan2 the main managers and Knowledge Source Platforms are implemented as separate processes. One advantage of this approach is that Knowledge Sources can be run in parallel with one another, and that external events can be processed by the Interface Manager (the manager in charge of all interaction, diagnostic handling and instrumentation) as they occur. The agent latency or reaction time performance of the system is measured by the time taken to move an incoming event through the agenda triggering mechanism to a waiting Knowledge Source Platform. The cycle time performance of the system is measured by the time taken to move an agenda entry posted by one Knowledge Source through the triggering mechanism to run on a waiting Knowledge Source Platform.

8 O-Plan2 User Interface

8.1 Planner 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. In the past, our AI planners have provided custom built graphical interfaces embedded in the specialist programming environments in which the planners have been implemented. It is now important to provide interfaces to AI planners that are more easily used and understood by a broader range of users. We have characterised the user interface to O-Plan2 as being based on two *views* supported for the user. The first is a *Plan View* which is used for interaction with a user in planning entity terms (such as the use of PERT-charts, Gantt charts, resource profiles, etc). The second is the *World View* which presents a domain orientated view or simulation of what could happen or is happening in terms of world state.

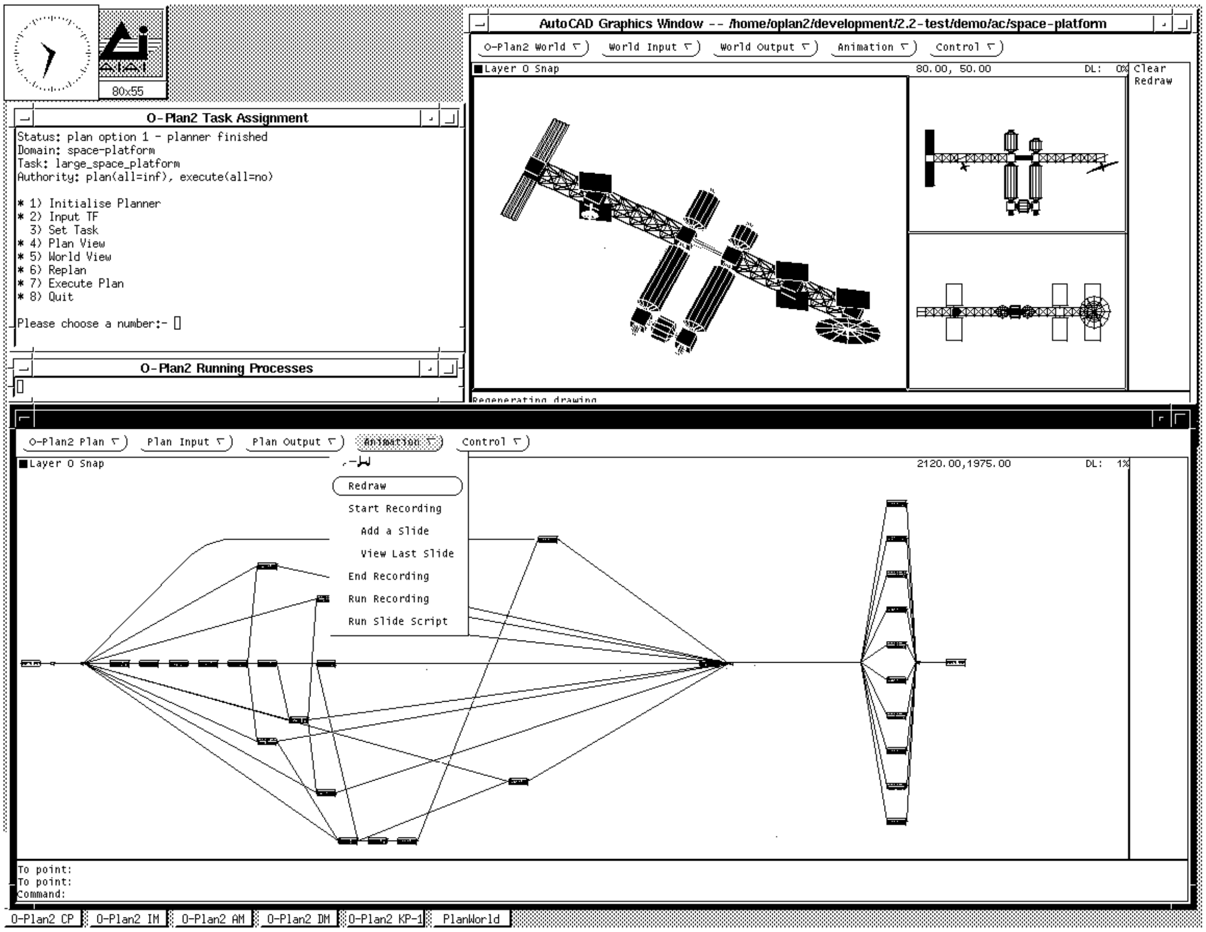

Figure 4: Example Output of the AutoCAD-based User Interface

Computer Aided Design (CAD) packages available on a wide range of microcomputers and engineering workstations are in widespread use and will probably be known to potential planning system users already or will be in use somewhere in their organisations. There could be benefits to providing an interface to an AI planner through widely available CAD packages so that the time to learn an interface is reduced and a range of additional facilities can be provided without additional effort by the implementors of AI planners.

We have built an interface to the Edinburgh AI planning systems which is based on AutoCAD [26]. A complete example of the use of the interface has been built for a space platform building application. O-Plan2 Task Formalism has been written to allow the generation of plans to build various types of space platform with connectivity constraints on the modules and components. A domain context display facility has been provided through the use of AutoLISP. This allows the state of the world following the execution of any action to be visualised through AutoCAD. Means to record and replay visual simulation sequences for plan execution are provided.

A sample screen image is included in Figure 4. There are three main windows. The planner is accessible through the Job Assignment window to the top left hand corner which is showing the main user menu. The planner is being used on a space station assembly task and has just been used to get a resulting plan network. In the *Plan View* supported by O-Plan2, this has been displayed in the large AutoCAD window along the bottom of the screen. Via interaction with the menu in the AutoCAD window, the planner has been informed that the user is interested in the world context at a particular point in the plan - the selected node is highlighted in the main plan display. In the *World View* supported by O-Plan2, the planner has then provided output which can be visualised by a suitable domain specific interpreter. This is shown in the window to the top right hand corner of the screen where plan, elevation and perspective images of the space station are simultaneously displayed.

The O-Plan2 Plan View and World View support mechanisms are designed to retain independence of the actual implementations for the viewers themselves. This allows widely available tools like AutoCAD to be employed where appropriate, but also allows text based or domain specific viewers to be interfaced without change to O-Plan2 itself. The specific viewers to be used for a domain and the level of interface they can support for O-Plan2 use is described to O-Plan2 via the domain Task Formalism (TF). A small number of *viewer characteristics* can be stated. These are supported by O-Plan2 and a communications language is provided such that plan and world viewers can input to O-Plan2 and take output from it.

8.2 System Developer Interface

When O-Plan2 is being used by a developer, it is usual to have a number of windows active to show the processing going on in the major components of the planner. There is a small window acting as the job assignment agent with its main O-Plan2 menu. There are then separate windows for the Interface Manager (IM) – through which the user can communicate with other processes and through which diagnostic and instrumentation levels can be changed. The Agenda Manager/Controller (AM), the Database Manager (DM) and the Knowledge Source Platform(s) (KP) then have their own windows. Further pop-up windows are provided when viewing the plan state graphically or when getting detail of parts of the plan, etc.

A sample developer screen image is shown in Figure 5.

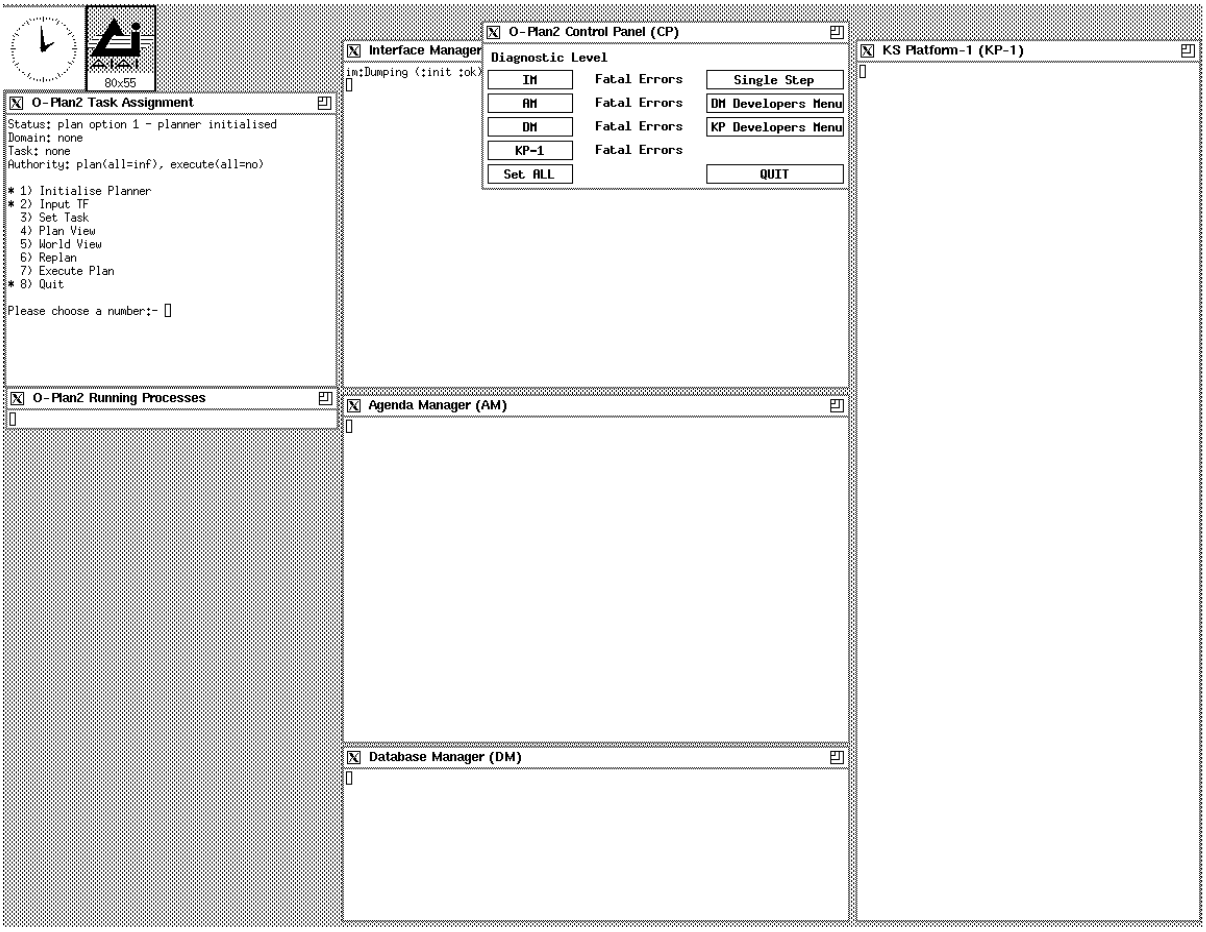

Figure 5: Example Developer Interface for the O-Plan2 Planning Agent

9 Applications

The O-Plan2 prototype has been tested on a number of simple, but realistic, domains as well as on puzzles intended to test specific features.

Block Stacking. A set of *puzzle* problems used to test effect/condition interaction and goal handling in O-Plan2.

House Building. A "standard" domain for tests of the Edinburgh planners with a number of variants to test specific features. The aim is to construct a project plan to build a house with certain requirements.

Space Station Assembly. This application shows the development of a plan for the construction of one of a number of different Space Platforms. Platforms are constructed from a series of joints, trusses, pressurised modules, solar panels, radiators and antennas. This example has been included to demonstrate the AutoCAD user interface which has been constructed for O-Plan2.

Satellite Control. This application shows the development of a plan for the control of a simple satellite we have called EUSAT (Edinburgh University Satellite) which is based on the University of Surrey's successful UOSAT-II. The O-Plan2 planning agent has been demonstrated generating a plan for operation of the spacecraft for one day by generating the actual on-board computer Diary commands and was able to pass it to an O-Plan2 based execution agent for simulated dispatch and monitoring to take place.

10 Related Projects

O-Plan2 is one of several projects at Edinburgh grouped under the title of EUROPA (Edinburgh University Research into Open Planning Architectures). The combined research of these projects cover issues in Knowledge Based Planning and Scheduling and are anchored around the two main, long term research projects of O-Plan2 and TOSCA (The Open SCheduling Architecture [2]). O-Plan2 has concentrated on an activity based plan state with good time and resource constraint handling for this base. TOSCA is a variant of the same ideas applied to the area of operations management in the factory (job shop) environment. TOSCA employs appropriate Knowledge Sources for its domain of application (e.g. resource assignment, bottleneck analysis) which operate on an emerging schedule state, similar to the notion of the plan state mentioned above. There is a good measure of overlap between the techniques used on these projects, particularly with respect to time and resource handling. Our aim is to develop designs and architectures suited to both activity planning and scheduling problems and to develop as much common ground as is possible. O-Plan2 plays a key role in this plan.

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. Northrup Fowler III at the USAF Rome Laboratory. Additional resources for the O-Plan work has been provided by the Artificial Intelligence Applications Institute through the EUROPA (Edinburgh University Research on Planning Architectures) Institute development project.

References

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

[2] Beck, H.A. *TOSCA: The Open SCheduling Architecture*, Papers of the AAAI Spring Symposium on "Practical Approaches to Scheduling and Planning", Stanford, CA, USA, March 1992.

[3] Chapman, D. *Planning for conjunctive goals.* Artificial Intelligence Vol. 32, pp. 333-377, 1987.

[4] Currie, K.W. & Tate, A. (1985) *O-Plan: Control in the Open Planning Architecture*, Proceedings of the BCS Expert Systems 85 Conference, Warwick, UK, Cambridge University Press.

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

[6] Daniel, L. (1983) *Planning and Operations Research* in Artificial Intelligence: Tools, Techniques and Applications (eds. O'Shea and Eisenstadt), Harper and Row, New York.

[7] Drabble, B. *Planning and reasoning with processes*, in Procs. of the 8th Workshop of the Alvey Planning SIG, The Institute of Electrical Engineers, November, 1988. Full paper to appear in Artificial Intelligence, 1992.

[8] Drabble, B. & Kirby, R.B. *Associating* AI *planner entities with an underlying time point network*, in Procs. of the European Workshop on Planning, pp. 27-38, St. Augustin, Germany, March 1991. *Lecture Notes in* AI *No. 522*, Springer-Verlag.

[9] Drabble, B. & Tate, A., *Using a CAD system as an interface to an AI Planner*, European Space Agency Conference of Space Telerobotics, European Space Agency, 1991, Noordwijk, Holland.

[10] Drummond, M. & Currie, K. *Exploiting temporal coherence in nonlinear plan construction*, in Procs. of IJCAI-89, Detroit.

[11] Drummond, M.E., Currie, K.W. & Tate, A. (1988) *O-Plan meets T-SAT: First results from the application of an AI Planner to spacecraft mission sequencing*, AIAI-PR-27, AIAI, University of Edinburgh.

[12] Fikes, R.E., Hart, P.E. & Nilsson, N.J. (1972) *Learning and Executing Generalized Robot Plans*, Artificial Intelligence Vol. 3.

[13] Georgeff, M. P. & A. L. Lansky (1986) *Procedural Knowledge*, in Proceedings of the IEEE, Special Issue on Knowledge Representation, Vol. 74, pp. 1383-1398.

[14] Hayes, P.J. (1975) *A representation for robot plans*, in Procs. of the International Joint Conference on Artificial Intelligence (IJCAI-75), Tbilisi, USSR.

[15] Hayes-Roth, B. & Hayes-Roth, F. *A cognitive model of planning*, Cognitive Science, pp. 275-310, 1979.

[16] Lesser, V. & Erman, L. *A retrospective view of the Hearsay-II architecture*, in Procs. of the International Joint Conference on Artificial Intelligence (IJCAI-77), pp. 27-35, 1977

[17] Liu, B., Ph.D Thesis, *Knowledge Based Scheduling*, Edinburgh University, 1988.

[18] Malcolm, C. & Smithers, T. (1988) *Programming Assembly Robots in terms of Task Achieving Behavioural Modules: First Experimental Results*, in Procs. of the Second Workshop on Manipulators, Sensors and Steps towards Mobility as part of the International Advanced Robotics Programme, Salford, UK.

[19] McDermott, D.V. *A Temporal Logic for Reasoning about Processes and Plans*, Cognitive Science, 6, pp. 101-155, 1978.

[20] Nii, P. *The blackboard model of problem solving*, AI Magazine Vol.7 No. 2 & 3. 1986.

[21] Nilsson, N.J. (1988) *Action Networks*, in Procs. of the Rochester Planning Workshop, October 1988.

[22] Rosenschein, S.J., & Kaelbling, L.P. (1987) *The Synthesis of Digital Machines with Provable Epistemic Properties*, SRI AI Center Technical Note 412.

[23] Sacerdoti, E. *A structure for plans and behaviours*, Artificial Intelligence Series, North Holland, 1977.

[24] Sadeh, N. & Fox, M.S., *Preference Propagation in Temporal/Capacity Constraint Graphs*, Computer Science Dept, Carnegie-Mellon University, 1988, Technical Report CMU-CS-88-193.

[25] Smith, S., Fox, M. & Ow, P.S., *Constructing and maintaining detailed production plans: Investigations into the development of knowledge based factory scheduling systems*, AI Magazine, 1986, Vol 7, No.4

[26] Smith, J. & Gesner, R. (1989) *Inside AutoCAD*, New Riders Publishing Cp., Thousand Oaks, Ca.

[27] Sridharan, N. *Practical Planning Systems*, in Procs. of the Rochester Planning Workshop, AFOSR, 1988.

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

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

[30] Tate, A. (1986) *Goal Structure, Holding Periods and "Clouds"*, in Procs. of the Reasoning about Actions and Plans Workshop, Timberline Lodge, Oregon, USA. (eds, Georgeff, M.P. & Lansky, A.), published by Morgan Kaufmann.

[31] Tate, A. (1989) *Coordinating the Activities of a Planner and an Execution Agent*, in in Procs. of the Second NASA Conference on Space Telerobotics, (eds. G.Rodriguez & H.Seraji), JPL Publication 89-7 Vol. 1 pp. 385-393, Jet Propulsion Laboratory, February 1989.

[32] Tate, A. (1990) *O-Plan2: Choice Ordering Mechanisms in an AI Planning Architecture*, in Procs. of the 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control, San Diego, California, USA on 5-8 November 1990, published by Morgan-Kaufmann. Also updated with B.Drabble as AIAI-TR-86, AIAI, University of Edinburgh.

[33] Tate, A., Drabble, B. & Kirby, R.B. *Spacecraft Command and Control using AI Planning Techniques - The O-Plan2 Project - Final Report*, USAF/AFOSR contract no. F49620-89-C0081. Technical Report from Rome Laboratory, Griffiss AFB, NY 13441-5700. Also available as AIAI-TR-109, AIAI, University of Edinburgh.

[34] Tecknowledge, *S.1 Product Description*, Tecknowledge Inc., 525 University Avenue, Palo Alto, CA 94301, 1988.

[35] Stefik, M. *Planning with constraints*, Artificial Intelligence, Vol. 16, pp. 111-140, 1981.

[36] Vere, S. Planning in time: windows and durations for activities and goals. IEEE *Transactions on Pattern Analysis and Machine Intelligence Vol. 5, 1981.*

[37] Wilkins, D.E. (1985) *Recovering from execution errors in SIPE*, Computational Intelligence Vol. 1 pp. 33-45.

[38] Wilkins, D.E. *Practical Planning*, Morgan Kaufman, 1988.