PlanWorld Viewers[1]

Austin Tate and Brian Drabble
Artificial Intelligence Applications Institute
The University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, UK

Abstract

The user interface to the O-Plan planning system seeks to differentiate the various *roles* played by users in systems which support command, planning and control functions. Appropriate support is offered to the Task Assigner, the planning specialist and the operational execution staff.

The planning role is supported by a user interface that provides different views of the plan structure. These can be technical or plan structure oriented views, or they may be more visualisation or world oriented views. We provide support to either view via an interface that supports the "plugging-in" of appropriate PlanWorld viewers which conform to a specified interface.

1 O-Plan – a Modular, Open Planning Architecture

The O-Plan 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-Plan 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 on AI planning systems. See [5] for details of the first version of the O-Plan planner which introduced an agenda-based architecture and the main system components. That paper also includes a chart showing how O-Plan relates to other planning systems. The second version of the O-Plan system adopted a multi-agent approach and situated the planner in a task requirement and plan execution setting. The multi-agent approach taken is described in greater detail in [20].

Figure 1 shows the communications between the 3 agents in the O-Plan architecture. A user specifies a task that is to be performed through some suitable interface. We call this process *task assignment*. A *planner* plans to perform the task specified. The *execution system* seeks to carry out the detailed actions specified by the planner while working with a more detailed model of the execution environment.

The O-Plan 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

[1]Sections 4 and 5 are based on material published in [19]. Section 3 is extracted from the Task Formalism Manual which is part of the O-Plan System Documentation Set. Sections 2 and 6 are extracted from the Architecture Guide of the O-Plan System Documentation Set.

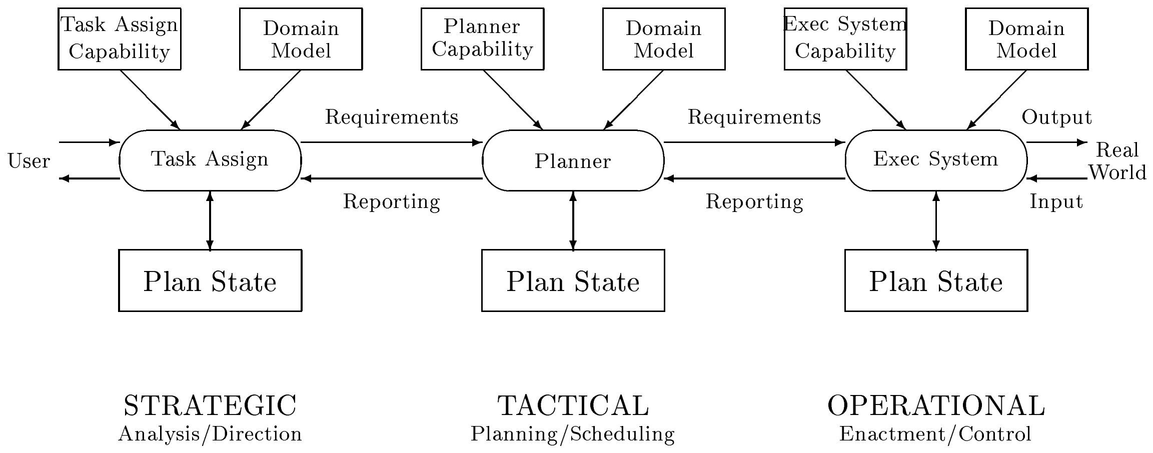


Figure 1: Communication between Strategic, Tactical and Operational Agents

- using opportunistic selection of the focus of attention on each problem solving cycle
- building information incrementally in "constraint managers", e.g.,
  - object/variable manager
  - time point network manager
  - effect/condition manager
  - resource utilisation manager
- using localised search to explore alternatives where advisable
- with global alternative re-orientation where necessary.

The O-Plan 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 background to this work is provided in [15]. The various components plug into "sockets" within the architectural framework. The sockets are specialised to ease the integration of particular types of component. See figure 2.

The various components of the agent architecture are:

PlanWorld Viewers – User interface, visualisation and presentation viewers for the plan – usually differentiated into technical *plan* views (charts, structure diagrams, etc.) and *world* views (simulations, animations, etc.).

Knowledge Sources – Functional components which can analyse, synthesise or modify plans.

Domain Library – A description of the domain and a library of possible actions.

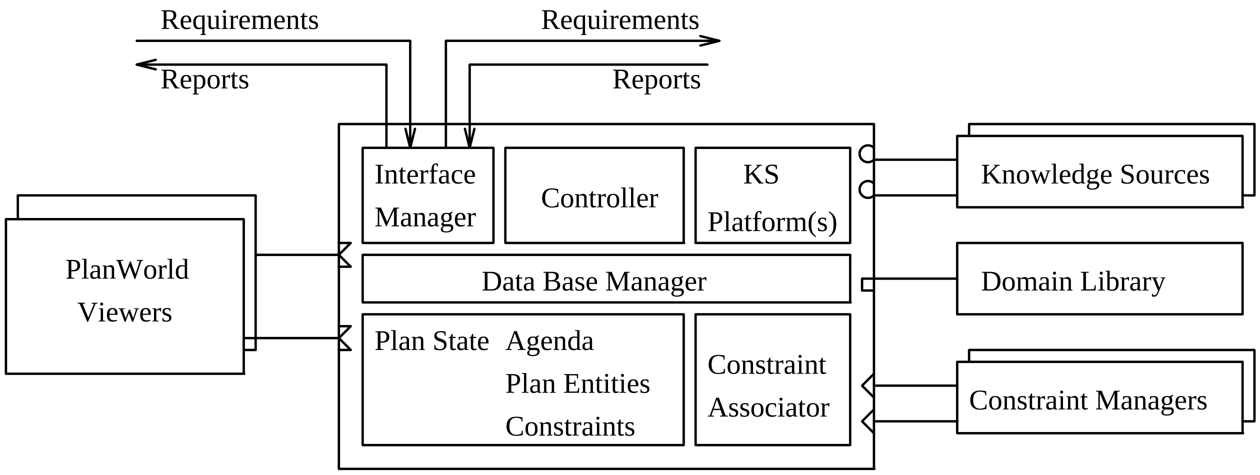


Figure 2: O-Plan Agent Architecture

Constraint Managers – Support modules which manage detailed constraints within a plan and seek to maintain as accurate a picture as possible of the feasibility of the current plan state with respect to the domain.

These plug-in components are orchestrated by an O-Plan agent kernel which carries out the tasks assigned to it via appropriate use of the Knowledge Sources and manages options being maintained within the agent's *Plan State*. The central control flow is as follows:

Interface Magager – Handles external events (requirements or reports) and, if they can be processed by the agent, posts them on the agent *Agenda*.

Controller – Chooses Agenda entries for processing by suitable Knowledge Sources

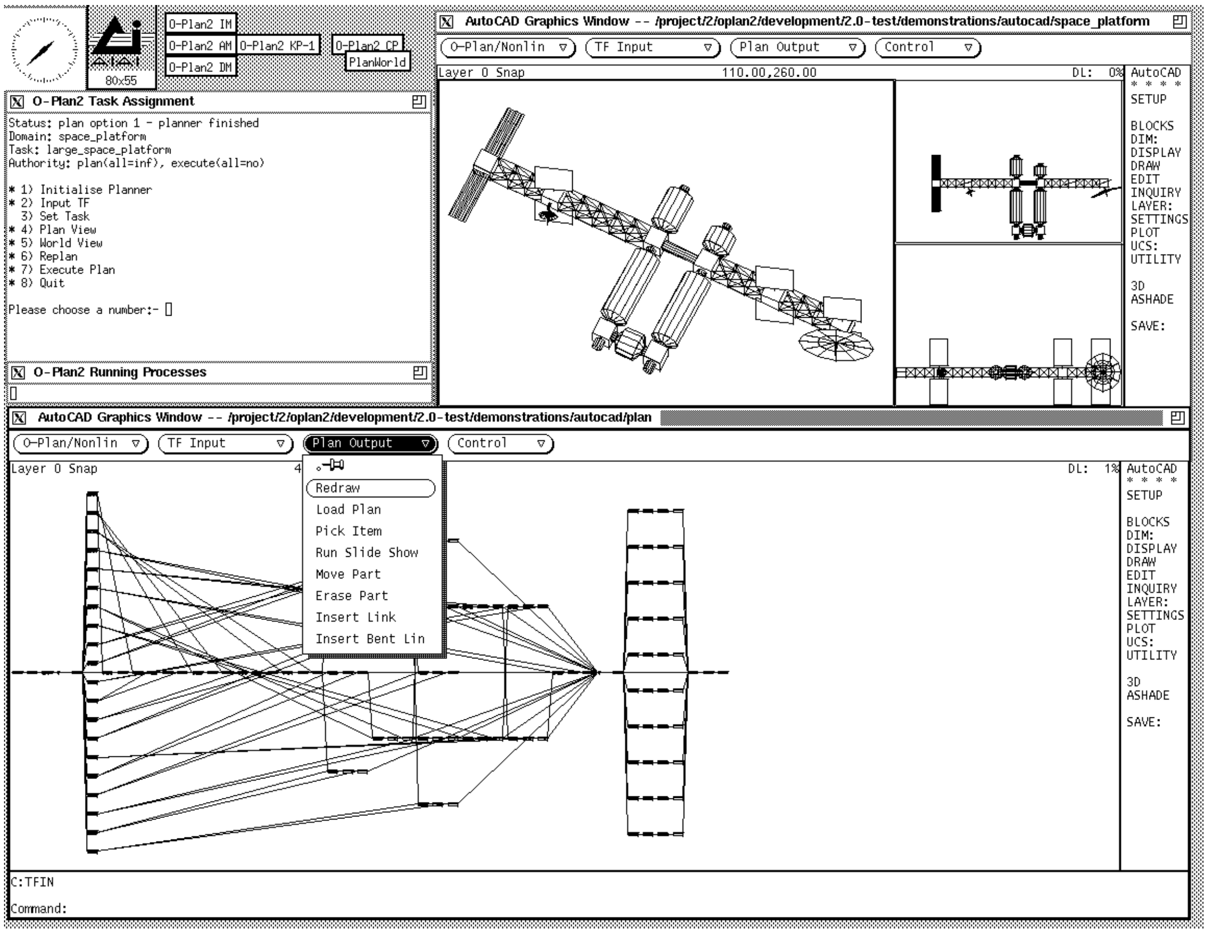
Knowledge Source Platform(s) – Chosen Knowledge Sources are run on an available and suitable Knowledge Source Platform.

Data Base Manager – Maintains the Plan State being manipulated by the agent and provides services to the Interface Manager, Controller and Knowledge Sources running on KS Platforms to allow this.

Constraint Associator Acts as a mediator between the Plan State maintained by the data base manager and the various Constraint Managers that are installed in the agent. It eases the management of interrelationships between entities and detailed constraints.

2 PlanWorld Viewer 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 being considered. 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-Plan 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.



```
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:- []
```

```
C:TFIN
Command:
```

Figure 3: PlanWorld User Interface for Space Platform Construction

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.

Some CAD packages provide facilities to enable tailored interfaces to be created to other packages. One such package is AutoCAD [2], [12] - though it is by no means unique in providing this facility. AutoCAD provides AutoLISP, a variant of the Lisp language, in which customised facilities may be provided [3], [13]. This is convenient for work in interfacing to AI systems as workers in the AI field are familiar with the Lisp language. However, the techniques employed would apply whatever the customisation language was.

We have built an interface to the Edinburgh AI planning systems which is based on AutoCAD. A complete example of the interface has been built for two different domains:

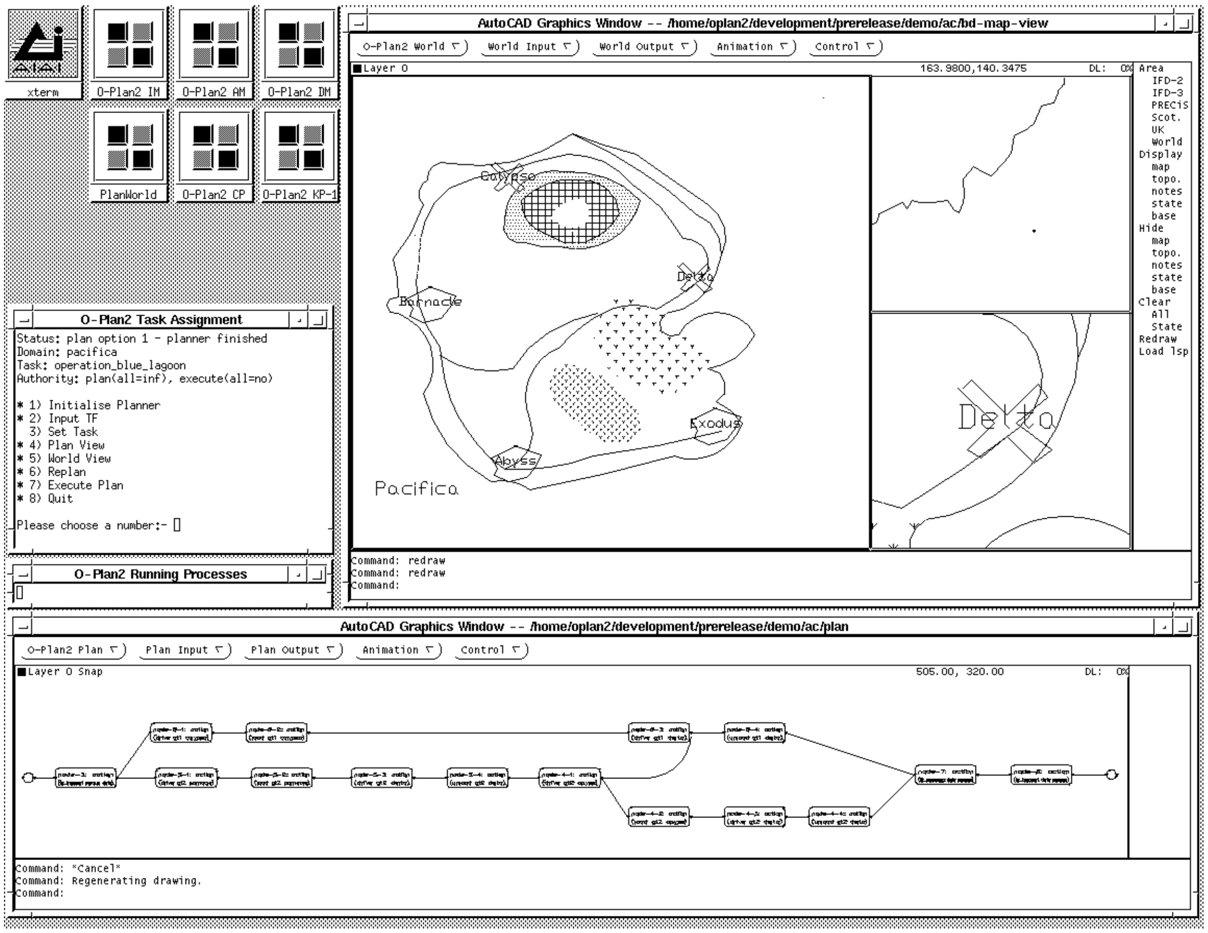
- Space Platform Building
  O-Plan Task Formalism has been written to allow the generation of plans to build various types of space platform with connectivity constraints on the modules and components. A sample screen image is shown in Figure 3.
- Non-combatant Evacuation Operation (NEOs)
  O-Plan Task Formalism has been written to model the evacuation of nationals from the mythical island of Pacifica in which unrest has broken out. A general use map-based World Viewer is used with this application. A sample screen image is shown in Figure 4.

A domain context display facility has been provided for both applications 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.

In the sample screen image of Figure 3, there are three main windows. The planner is accessible through the Task 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-Plan, this has been displayed using the *Load Plan* menu item 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 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-Plan, 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-Plan 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-Plan itself. The specific viewers to be used for a domain and the level of interface they can support for O-Plan use is described to O-Plan via the domain Task Formalism (TF). A small number of *viewer characteristics* can be stated. These are supported by O-Plan and a communications language is provided such that plan and world viewers can input to O-Plan and take output from it.

Sophisticated Plan and World Viewers could be used in future with O-Plan. We believe that time-phased tactical mapping displays of the type used in military logistics can be used as a World Viewer. We have also considered interfaces to a Virtual Reality environment we term PlanWorld-VR.



```
Status: plan option 1 - planner finished
Domain: pacifica
Task: operation_blue_lagoon
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:- []
```

```
Command: redraw
Command: redraw
Command:
```

```
Command: *Cancel*
Command: Regenerating drawing.
Command:
```

Figure 4: PlanWorld User Interface for Non-Combatant Evacuation Operations

3 O-Plan Interface to the PlanWorld Viewers

General purpose or domain-specific PlanWorld Viewers can be provided and used with O-Plan. O-Plan knows of a number of *Viewer Characteristics* which are used to ensure that interfacing between the Viewers and O-Plan is achieved in a modular and implementation independent fashion.

3.1 Plan Viewer

The characteristics possible for a plan viewer are as follows:

plan_output indicates that the plan viewer can accept output from the planner in the *O-Plan plan output format*. A simple textual presentation of this information is possible. Note that it is assumed that all plan viewers should have the **plan_output** feature available – it would be unhelpful of a plan viewer not to provide this feature at least in a simple form!

levels_output indicates that the plan viewer can show information about levels of a plan in a useful form.

resource_output indicates that the plan viewer can show information about resource usage perhaps in the form of gantt charts, capacity profiles, etc.

node_selection indicates that the plan viewer is able to give input to O-Plan showing nodes being pointed at in the last plan which was output. The node numbers given in that output will be passed for any node selected in the plan viewer by the user.

link_selection indicates that the plan viewer is able to give input to O-Plan showing links being pointed at in the last plan which was output. A pair of node numbers is produced by the plan viewer (relative to node numbers given in the last plan output) representing the end nodes of any link selected in the plan viewer by the user.

entity_detail indicates that the plan viewer can display detail of nominated entities.

tf_input indicates that the plan viewer can produce TF input in a legitimate format (for example, if tasks can be specified in the plan viewer by some means, or if actions, resource profiles, etc can be "drawn" and converted to legitimate TF). One way in which this can be done is by the provision of drawing aids for actions, links, conditions, effects, etc.

The *O-Plan plan output format* is introduced by the single word **plan** on one line followed by statements describing nodes. Nodes are introduced with the single word **node** on one line followed by a fixed number of lines as described below. A node statement is terminated with the single word **end_node** on a separate line. The plan output format is terminated by the single word **end_plan** on a separate line. Leading spaces and tab characters on any line may be ignored. Blank lines in the output may be ignored.

```
plan
   node
      <node_reference>
      ( [ <predecessor of begin_end> ... ] )
      ( [ <successor of begin_end> ... ] )
      ( [ <predecessor of end_end> ... ] )
      ( [ <successor of end_end> ... ] )
      <node_time_information>
      <node_type>
      <node_label>
   end_node
   ...
end_plan

<predecessor of begin_end> | <successor of begin_end> |
  <predecessor of end_end> | <successor of end_end> ::=
                                        <end> <node_reference>

<node_reference> ::= node-<integer>[-<integer> ...]

<node_label> ::= " [ <character> ... ] "

<node_time_information> ::= ( <earliest_begin_time>
                              <latest begin_time>
                              <earliest_end_time>
                              <latest_end_time>
                              <minimum_duration>
                              <maximum_duration> )

<earliest_begin_time> | <latest begin_time> |
  <earliest_end_time> | <latest_end_time>   |
  <minimum_duration>  | <maximum_duration>  ::= <integer>
```

It is useful to know that <node_reference>s easily show the expansion level at which a node was introduced into a plan. An example node number for a top level node such as the **finish** node of a plan is "node-2". A node which is at the third level might have a <node_reference> of "node-15-2-4".

If the plan viewer can call on a file of information to tailor its output, it is recommended that it contain entries in the following format (where this is possible).

```
<drawing_object_name> -> <associated_instructions_or_data>

<drawing_object_name> ::=   <action_or_event> <drawing pattern>
                          | <dummy_node_type>
```

```
<drawing_pattern> ::= <fully_instantiated_pattern> | <pattern_with_??>
```

<fully_instantiated_pattern> and <pattern_with_??> are patterns not containing match restrictions or variables.

The <associated_instructions_or_data> could hold icon filenames or drawing instructions, etc.

3.2 World Viewer

The characteristics possible for a world viewer are as follows:

snapshot indicates that the world viewer program can accept a sets of facts and statements about the world state in the form of the *O-Plan world output format* and can present this to the user. A simple textual presentation of this information is possible.

incremental indicates that it is possible to follow the initial startup of the program or any snapshot output (if that feature is available) with *changes* in the world state which the planner wishes to display. These are in the same format as the full snapshot *O-Plan world output format* but present only a partial description of a context in the plan.

tf_input indicates that the world viewer program can produce TF input in a legitimate format (for example, if tasks can be specified in the world viewer program by some means, or if initial information can be provided (e.g. an initial world state) and these can be converted to legitimate TF). One mechanism is to allow the drawing of objects directly in the domain (such as the features of a building or structure, or the placing of objects on a map) and to convert these to **initially** or **always** TF statements.

The user interface for O-Plan allows for facilities for context snapshot image saving (in a *pic*) and recording and playback of a series of such images (in *flicks*) to be provided. However, these will be provided and managed by the world viewer program and are thus not part of the definition of the world viewer system in TF.

The *O-Plan world output format* is introduced by the word **world** followed by a keyword **snapshot** or **increment** on one line followed by statements of the form shown on a single line with a line **end_world** being used to terminate the output.

```
world <world_view_type>
   <pattern> = <value>
   ...
end_world

<world_view_type> ::= snapshot | increment
```

If the world viewer program can call on a file of information to tailor its output, it is recommended that it contains entries in the following format (where this is possible).

```
<domain_statement> = <domain_value> -> <associated_instructions_or_data>

<domain_statement> | <domain_value> ::=  <fully_instantiated_pattern>
                                       | <pattern_with_?? >
```

The <associated_instructions_or_data> could hold drawing instructions, etc.

4 Using AutoCAD as a Basis for PlanWorld Viewers

This section gives details of the use of the AutoCAD package to provide example PlanWorld Viewers for the Edinburgh AI planners (Nonlin [14], Excalibur [6] and O-Plan [5],[20]). The range of ways to make use of a CAD package as a PlanWorld Viewer interface to an AI planner are described and details of the particular methods chosen for these experiments are given. Examples are provided using a simple space station assembly application.

4.1 AI Planners and CAD Systems

Artificial Intelligence (AI) planning systems attempt to take a description of the actions or operations which are possible in some application domain and then attempt to produce a plan to carry out some task, possibly within given constraints on time or resource usage. A number of AI planners produce their plans as a network of actions in a partial order. These output plans are similar to PERT networks used in project management systems.

Computer Aided Design (CAD) packages are readily available at low cost and can run on a range of personal computers and engineering workstations. They are well supported by their vendors; training is available and a wide range of text books supports their use by all levels of user. These packages provide a broad range of functions that can significantly enhance the simple graphical input and presentation interfaces already provided in AI planners. Features for printing, scaling, reorganisation of the image, editing, extraction of parts, annotation and presentation are all possible.

4.2 Edinburgh AI Planners

Edinburgh planning researchers have produced a number of prototype AI planners which can generate plans of action (mostly in the form of networks of actions) for some specified task in some application domain which can be described to the planner in an input language *Task Formalism (TF)*. These planners include Nonlin [14], Excalibur [6], and O-Plan [5],[20].

4.3 Graphical Interfaces

The Edinburgh planning work has included the production of a number of graphical interfaces to the various planners that have been built. The interfaces have been created as experiments to support a number of different types of user role with respect to a planner.

a) application domain and task definition by compiling graphical input of actions or tasks to the Edinburgh *Task Formalism (TF)* input language. Early work on this was performed by us [18] where we built a prototype Task Formalism (TF) Workstation on the Three Rivers/ICL PERQ computer to allow for graphical input and editing of actions and their sub-action expansions. Effects, conditions, resource usage and time constraints on the sub-activities could be specified. Some experimentation with the use of a requirements analysis methodology (based on CORE from SD-Scicon, [11]) to assist the user in reliably describing the domain was performed [21].

b) plan network drawing facilities have been provided in the O-Plan Graph Drawer [5]. This package is intended as a flexible and programmable graph output package which can draw a plan network at various levels of detail, use iconic images of actions, etc.

c) plan component selection facilities are provided in the O-Plan Graph Drawer to allow for the selection of a specific component such as an action. The design of the Graph Drawer and its interface to the client program (i.e. the planner) allows this selection to be fed back to the planner and some context specific action to take place. This action could be to create a pop-up window with a greater level of detail of the chosen component, to carry out some planner operation on the component (such as to treat this as a user request to expand an action to a lower level of detail), etc.

d) simulation of the plan by display of the state of the world model at some point in the plan is possible in most of the Edinburgh planners. The basic *Question Answering (QA)* or *Truth Criterion* routines in the planners [14] support the creation of a set of statements known about a selected point in the plan. This may either be printed in a text form, or it can be passed to a domain dependent package which can interpret the statements to produce a picture of the state of the world at the required point in the plan. There can be some ambiguity (due to actions still remaining unordered) in the statements produced and this needs to be taken into account in the drawing package provided. The design of the planner interfaces allows for a series of these single pictorial snapshots to be saved on file and replayed in the saved sequence as an animation of the plan being executed. To date only very simple domain dependent pictorial displays have been created for a block stacking domain [5] and to show the electrical wiring harness of a spacecraft being commanded [8].

4.4 Graphical Interaction – Four Basic Requirements

Following on from the perceived graphical interface requirements identified for the Edinburgh planners which are described above, the experiments with the AutoCAD interface has demonstrated the four styles of interaction and established basic mechanisms for performing each via AutoCAD.

a) create a schema or task definition graphically and input it to the planner

b) output a plan network

c) select a particular object (eg action node) in a schema or plan and pass its identity onto the AI planner

d) graphically depict the "state" of the world at some point in the plan.

They are not intended as finished pieces of work and will be revisited later in the various Edinburgh planning projects.

5 Experiments with the AutoCAD-based PlanWorld Viewers

Given the AutoCAD drawing environment described above, the facilities provided through the Viewer menu can be used as a graphical interface to the Edinburgh planners. Each of the four styles of graphical interaction has been experimented with and the experiments are described in the following sections.

5.1 Task Formalism Schema Input

It is possible to describe a Task Formalism schema using the features of the interface. A schema header can be inserted to give the titling information and comments associated with the schema. Then, nodes, dummies and links can be inserted, moved or erased until the appropriate sub-action network for the schema is correct. Conditions and effects on nodes can be included. Only limited space is provided for all annotations such as action, condition and effect patterns. However, any length of text can be used as the annotation and it is fitted into the space available. Normal AutoCAD package *zoom* facilities can be used to read text that is too small when first displayed.

Once the schema is in its final form, the *TF Out* menu item may be used. This creates a file which contains details of all the drawing components and screen locations in such a way that a straightforward conversion to the Task Formalism used by Edinburgh planners is possible.

The approach we have taken allows a task to be specified as a set of activities perhaps with some preordering constraints and/or a set of conditions that need to be achieved at certain points. Other researchers are investigating different domain specific means to give task information to AI planning systems. AutoCAD has been used to provide an interface to allow a building such as an office block to be laid out and then an interface has been created to allow the CAD system to create information which can be passed over to a planner [9].

5.2 Plan Network Output

The same drawing building blocks as are used for schema creation are used for displaying a plan generated by one of the Edinburgh planners.

The method chosen for drawing a plan is to create an AutoLISP function which when run displays the plan network. Part of the interface to the Edinburgh planners allows a routine to be called to display the current plan both in a text form to the screen and in a graphical form if a graphical interface is available. For the AutoCAD interface, the routine was modified to create a file which is the AutoLISP routine to display the network. Screen layout positions for the nodes and dummies in the plan is done with a simple depth first scan of the plan, ensuring that the plan links always flow from left to right. Some row adjustments are made to improve visual layout. An AutoCAD command to set the drawing limits in advance of any actual drawing is inserted to prevent a refresh of the screen if the drawing exceeds the default picture area.

5.3 Plan Network - Picking a Node for Interaction

The next type of graphical interaction demonstrated through the AutoCAD interface was intended to establish a basic mechanism for allowing the user to pick some component of a plan or schema with the mouse and for the identity of this component to be passed back to the AI planner or some other part of the total system. A simple AutoLISP procedure was written to allow the user to select an item. If no items were picked or more than one item was picked, the code seeks another selection. Once a single component has been identified, it is visually highlighted. The AutoCAD type of the selected component and any text attribute associated with the object is then extracted. In the experimental interface this is then printed to the screen. However, the AutoCAD facilities for calling the *shell* of the system in which AutoCAD is running can be used to call some other program and to pass it information about the selection made.

5.4 Simulation - Depicting the State of the World at some point in the Plan

The Edinburgh planners allow for the simulation of the plan by display of the state of the world model at some point in the plan. It is possible to include a domain dependent package which can interpret the statements to produce a picture of the state of the world at the required point in the plan. To date only very simple domain dependent pictorial displays have been hand crafted. During the experimentation with AutoCAD, a little work was performed to create domain displays for several domains including spacecraft command and control, house building and transportation planning as wel as the two domain described in this paper – the space station assembly task and the Non-combatant Evacuation Operations planning domain. Simple high level commands can be used to insert parts of a house, move components such as to rotate camera platforms or to indicate consumption of fuel on a spacecraft, or to show ship and supplies movements on a map.

It is clear that a general purpose CAD system could easily be adapted to create domain specific displays of the type assumed by the Edinburgh planning systems' simulation interfaces. Other general purpose interfaces, based for example on maps, could also be interfaced with suitable adaptor code.

6 O-Plan User Roles

User interaction with O-Plan can occur for a variety of purposes. Various *roles* of an user interacting with O-Plan are defined and are supported in different ways within the system. We consider the identification of the different roles to be an useful aid to guide future user interface support development.

6.1 Domain Expert Role

A single user responsible for defining the bounds on the application area for which the system will act. The domain expert user may directly or indirectly specify O-Plan Task Formalism to define the domain information which the planner will use.

6.2 Domain Specialist Role

One or more domain specialists may define information at a more detailed level within the framework established by the domain expert. Once again, the domain specialist may directly or indirectly specify O-Plan Task Formalism to provide the detailed domain information which the planner will use.

6.3 Task Assignment User Role

The command user interacts only with the Task Assignment Agent to provide user requirements or commands. This user is responsible for the selection of the task which the system will try to carry out. The current system provides a menu which allows for a domain to be selected and for a choice to be made from the task schemas within the Task Formalism for that domain. Future management of alternative plan options, plan analysis support and the provision of authority to plan or execute the plan are to be supported at this level.

6.4 Planner User Role

The planner user is the user responsible for ensuring that a suitable plan is generated to carry out the given task. This may involve the selection of alternatives, the restriction of options open to the planner and browsing on the emerging and final plan to ensure it meets the task requirements set by the task assignment user. Since the planner user can perform decision making in the planner agent, the planner user is supported by a knowledge source called KS-USER. This knowledge source can be added to the agenda for the current plan state on demand (via an user request). Since the KS-USER knowledge source normally has high priority, it will normally be called as soon as possible. The KS-USER knowledge source activation has access to the current plan state to allow for decisions on user intervention to depend on the contents of the current plan state.

6.5 Execution System Watch/Modify Role

The user may interact with the execution system to watch the state of execution of the plan and perhaps even to modify the behaviour of the execution system.

6.6 World Operative

Any users who are required to carry out activities in the world (acting as an *effector*) or who report aspects of the environment (acting as a *sensor*).

6.7 World Interventionist

If a world simulation is being used to demonstrate the O-Plan execution system, an user may be given facilities to intervene in the world simulation to cause events to happen and problems to occur such that execution of plans in uncertain situations can be tested.

6.8 User Support to Controller Role

The user may assist an O-Plan agent's controller to decide which knowledge source to dispatch to a waiting knowledge source platform or to decide on when to direct a running knowledge source to stop at a stage boundary.

6.9 User Support to Alternatives Handler

The user may assist an O-Plan agent's Alternatives Handler to decide which alternative to select when one is needed or to suggest an alternative is tried rather than continuing with the current plan state.

6.10 System Developer Role

The system developer has access to the diagnostic interface of the system running within each agent. This is supported by the Developer Diagnostic Interface of each O-Plan agent. The behaviour of this interface can be set and modified via a Control Panel which allows for the setting of levels of diagnostics using buttons, etc.

6.11 System Builder

The O-Plan Agent Architecture is intended to be sufficiently flexible to allow a system builder to create a system with defined behaviour. To this end, it is possible to have radically different plan state data structures, knowledge sources, domain information and controller strategies. For example, the O-Plan Architecture already has been used to provide a Manufacturing Scheduling System which uses a resource orientated representation for the plan state rather than the action orientated plan representation in the O-Plan Planner. This scheduler, called TOSCA (The Open SCheduling Architecture) [4], also has different knowledge sources than those used in the O-Plan Planner.

7 Future Development of the O-Plan User Interface

This paper has documented the work done to date on the user interface to the O-Plan planning agent – the PlanWorld Viewers. It also showed the careful separation of user roles for the various ways in which users can interact with the planning agent and with the other agents and components of the overall O-Plan Command, Planning and Control Architecture.

Work to date on O-Plan has principally focussed on the planning agent and variations of the execution agent (e.g., a Reactive Execution Agent worked on by Reece [10]). The interactions between these two agents has also been of principal importance.

More recently, work has begun on an improved basis for modelling tasks, plans and activities which is based on a general model of these as constraints on behaviour – the <I-N-OVA> Constraint Model of Plans [17]. We are starting to investigate a general model for interaction between system components, agents and users based on the mutual communication of such constraints on activity as a metaphor for mixed initiative planning [16]. Work in this area includes improved characterisation of the value of one plan over another using domain-related characteristics and features [7]. Emphasis will therefore shift to the user interface in the Task Assignment agent of O-Plan and its interface to the Planning agent.

Acknowledgements

O-Plan is an on-going project at Edinburgh. Current O-Plan work is supported by the US Advanced Research Projects Agency (ARPA) and the US Air Force Rome Laboratory Planning Initiative acting through the Air Force Office of Scientific Research (AFOSR) under contract F49620-92-C-0042. The project is monitored by Dr. Northrup Fowler III at Rome Laboratory. The United States Government is authorised to reproduce and distribute reprints of the paper for government purposes notwithstanding any copyright notation hereon.

References

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

[2] AutoDesk AutoCAD Reference Manual, 1989.

[3] AutoDesk AutoLISP Reference Manual, 1989.

[4] Beck, H., TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints, Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, pages 138-149, (eds. Kooij, C., MacConaill, P.A., and Bastos, J.), 1993.

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

[6] Drabble, B., Excalibur: A Program for Planning and Reasoning with Processes, *Artificial Intelligence*, Vol. 62 No. 1, pp. 1-40, 1993.

[7] Gil, Y., Tate, A. and Hoffman, M., Domain-Specific Criteria to Direct and Evaluate Planning Systems, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. M.Burstein), Morgan-Kaufmann, 1994.

[8] Drummond, M.E., Currie, K.W. and Tate, A. (1988) O-Plan meets T-SAT: first results from the application of an AI planner to spacecraft mission sequencing, AIAI-PR-27.

[9] Ito, K., Ueno, Y., levitt, R.E. and Darwiche, A. (1990) Linking Knowledge-Based Systems to CAD Design Data with an Object-Oriented Building Product Model, Research Report, Center for Integrated Facility Engineering, Stanford University, Ca.

[10] Reece, G.A. and Tate, A., Synthesizing Protection Monitors from Causal Structure, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), AAAI Press, Chicago, USA, 1994.

[11] SD-Scicon plc (1981) Controlled Requirements Specification - Seminar Notes, SD-Scicon plc, Camberley, UK.

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

[13] Smith, J. and Gesner, R. (1989) Inside AutoLISP, New Riders Publishing Cp., Thousand Oaks, Ca.

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

[15] Tate, A., The Emergence of "Standard" Planning and Scheduling System Components, in *Current Trends in AI Planning*, (eds. Backström, C. & Sandewall, E.), IOS Press, 1993.

[16] Tate, A., Mixed Initiative Planning in O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop at Tucson, Arizona, USA, (ed. M. Burstein), Morgan-Kaufmann, 1994.

[17] Tate, A. Characterising Plans as a Set of Constraints - the <I-N-OVA> Model - a Framework for Comparative Analysis, to appear in Special Issue on "Evaluation of Plans, Planners, and Planning Agents", ACM SIGART Bulletin Vol. 6 No. 1, January 1995.

[18] Tate, A. and Currie, K.W. (1985) The O-Plan Task Formalism Workstation, UK Alvey Planning SIG, Sunningdale, UK, January 1985. Also available as AIAI-TR-7.

[19] Tate, A. and Drabble, B., Using a CAD System as an Interface to an AI Planner, Proceedings of the Workshop on Artificial Intelligence and Knowledge-Based Systems for Space, ESTEC, European Space Agency, Noordwijk, The Netherlands, May 1991, ESA WPP-025, Volume 1.

[20] Tate, A., Drabble, B. and Kirby, R., O-Plan2: an Open Architecture for Command, Planning and Control, in Intelligent Scheduling, (eds, M.Zweben and M.S.Fox), Morgan Kaufmann Publishers, Palo Alto, CA., USA, 1994.

[21] Wilson, A.C.M. (1984) Information for Planning, M.Sc. Thesis, Department of Artificial Intelligence, University of Edinburgh.