O-Plan2 Technical Paper

Knowledge Based Project Planning

Brian Drabble

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

February 28, 1995

ARPA-RL/O-Plan2/TP/13 Version 1

Contents

1 Introduction

Practitioners of project planning often promote the discipline as well understood, at the heart of good management, and extremely powerful. Despite this the subject is also an extremely active area for researchers in many disciplines, but for precisely the opposite reasons to those stated. For researchers in Artificial Intelligence, autonomous planning is one of the longest established topics of interest, and certainly one of the most difficult, so are the underlying principles and practices of project planning really so clear?

We take for granted the fact that every day we must plan, schedule, and control the activities which make up our lives. In light of this, then, it seems surprising that there is no general theory which seeks to explain our planning, scheduling and control behaviour. In the absence of an underlying theory can project management claim to be beyond its infancy?

This article outlines the well founded design issues to be considered when building an autonomous planning system.

2 A Definition

Discussion of "planning systems" tends to cover a wider range of topics than may be obvious from the title. For instance, many manufacturers interested in *planning* tend also to be interested in *scheduling* — a quite separate topic from the AI practitioner's viewpoint, yet inextricably linked in the real world since plans are only useful *if* they are executed, i.e. if the actions planned are *scheduled* and carried out. Qualifying "planning" by using the phrases *process planning* and *project planning* would seem to avoid any confusion about the application of planning, but at the risk of creating two approaches to the problem. Thankfully, the necessary planning techniques, as described in the AI literature, seem to apply to both.

2.1 Planning

A planning system is charged with producing a *plan* which is a possible solution to a specified problem. For example, if the goal of planning is to 'build a two storey house', then a plan is a solution if its execution will result in the building of the house, as required. A planning system is required to synthesize a correct plan.

This crisp definition of a planning system contains clues to the type of problem planning systems tackle, specifically (i) that the system synthesizes a solution, and (ii) that the search for a solution is 'goal' oriented. It is important to understand that *search* and *search control* are the dominant characteristics of the planning problem, and it is the vastness of the possible search space (the space of all possible search paths containing feasible and infeasible solutions) that makes the problem difficult.

2.2 The Plan Structure

What is a plan? Broadly speaking, it is a *structure* resembling the familiar PERT chart (a graph structure in which the nodes represent actions and the arcs represent precedence); knowledge based planning systems, however, require significantly richer detail to be embedded in this structure. As a minimum the plan produced will contain operator descriptions (templates), provided to the system for each domain of application; in simpler terms, a set of *actions*. In our house building examples an action may be to `(dig the foundations)`. These descriptions are drawn from the fuller set of descriptions which collectively represent the available knowledge of allowable operations in the domain. We shall look at other plan structure detail later.

Since the plan is a 'solution', there must be a corresponding problem — how is the problem described? It is defined as:

- A *problem* is a complete initial state of the application domain. In the house building domain this could comprise (i) a piece of land, (ii) a pile of bricks, and (iii) a sum of money.
- A *problem* is a partial goal state (sometimes referred to as a final state). The goal state is partial in that it need not describe all features of the final successful state: any state produced which includes at least the goals specified in the partial state will suffice, *e.g.* `(2 storey house built)`.

The job of the plan is then straightforward — it is the *transformation* between the initial state of the world and the final state. This transformation is achieved by application of the action descriptions as a form of modification operator, thereby incrementally effecting the overall transformation. This is quite easy to understand. Consider a PERT chart — the nodes are the action descriptions oordered with respect to each other by the topology of the underlying network (AI planning has generally preferred to use action-on-node representations). The network is the definition of the transformation. Though this definition of planning can be stated quite simply, it is more difficult to achieve in practice. A description of the problems involved in designing knowledge based planning systems can be found in [2].

3 The Components of a Planner

The simplest way to look at the task of a planning system is to study an example, and the O-Plan project[1, 5] conducted at Edinburgh University from 1984 onwards is a good and mature example (SIPE[6], designed and built by David Wilkins at SRI, is arguably the other best example of a mature system). O-Plan has a long pedigree; it inherited many features from the work on the Nonlin[4] system developed earlier at Edinburgh.

O-Plan combines a number of features and techniques:

- a hierarchical planning system which can produce plans as partial orders on actions;

- an agenda-based control architecture in which each control cycle can post pending tasks during plan generation. These pending tasks are then picked up from the agenda and processed by appropriate handlers (*Knowledge Sources*);
- the notion of a "plan state", which is the data structure containing the emerging plan, the "flaws" remaining in it, and the information used in building the plan;
- *constraint posting* and *least commitment* on object variables;
- temporal and resource constraint handling using incremental algorithms which are sensitively applied only when constraints alter;
- taken from the earlier Nonlin planner, and extended, the ideas of Goal Structure, Question Answering (Modal Truth Criterion) and typed conditions;
- also derived from Nonlin, a task description language called Task Formalism ((TF).

O-Plan2, the system currently under development at AIAI, is aimed to be relevant to the following types of problems:

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

3.1 The Operation of O-Plan2

O-Plan2 operates at 3 levels (see Figure 1). A user specifies a task that is to be performed through some suitable interface. We call this process *task assignment*, i.e. planning at the strategic level. A *planner* plans to perform the task specified, operating at the tactical level. The *execution system* seeks to carry out the detailed activities specified by the planner while working at the operational level with a more detailed model of the execution environment. The current O-Plan2 system is able to operate both as a planner and a simple execution agent. The task assignment function is provided by a separate process which has a simple menu interface.

The O-Plan2 project has sought to identify modular components within an AI command, planning and control system and to provide clearly defined interfaces to these components and modules. O-Plan2 is implemented in Common Lisp on Unix Workstations with an X-Windows interface. It is designed to be able to exploit distributed and multi-processor delivery systems in future.

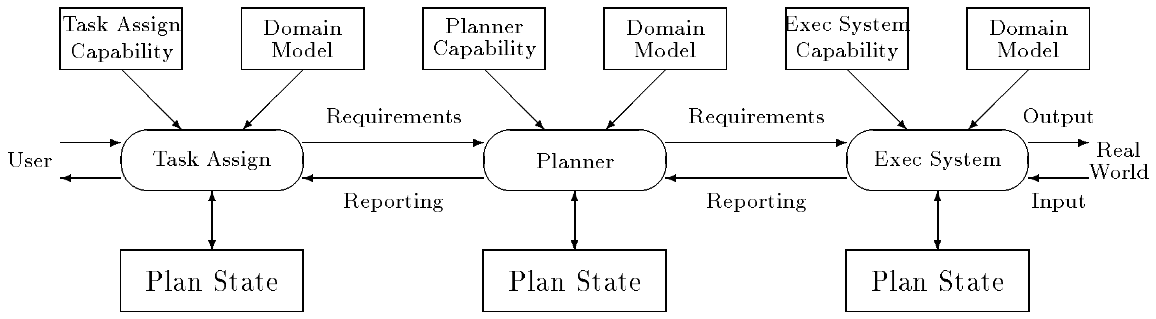


Figure 1: Communication between Strategic, Tactical and Operational Levels

The main components are:

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

3.2 The User Interface to O-Plan2

An interface has been built in AutoCAD to show the type of User Interface we envisage (see Figure 2). The window in the top left corner shows the Task Assignment menu and supports the management of authority to plan and execute plans for a given task. The lower window shows a *Plan View* (such as the plan shown as a graph), and the upper right window shows a *World View* for simulations of the state of the world at points in the plan. The particular plan viewer and world viewer provided are declared to the system; there is a definition of the interface between these and the planner, to which various implementations can conform. Most of the developer aspects of the planner interface are not shown to the normal user. In figure 2, the developer windows are shown in iconic form along the top edge of the screen.

Work is now underway for mid 1994 to show the O-Plan2 planner working with an enriched model of resources related to NEO evacuee transportation in the PRECiS [3] environment. A later demonstration in 1995 is intended to show how plans can be generated, their execution monitored with simple problem fixes applied in the PRECiS environment.

References

[1] **Currie, K.W. & Tate, B.A.** *O-Plan: the Open Planning Architecture.*

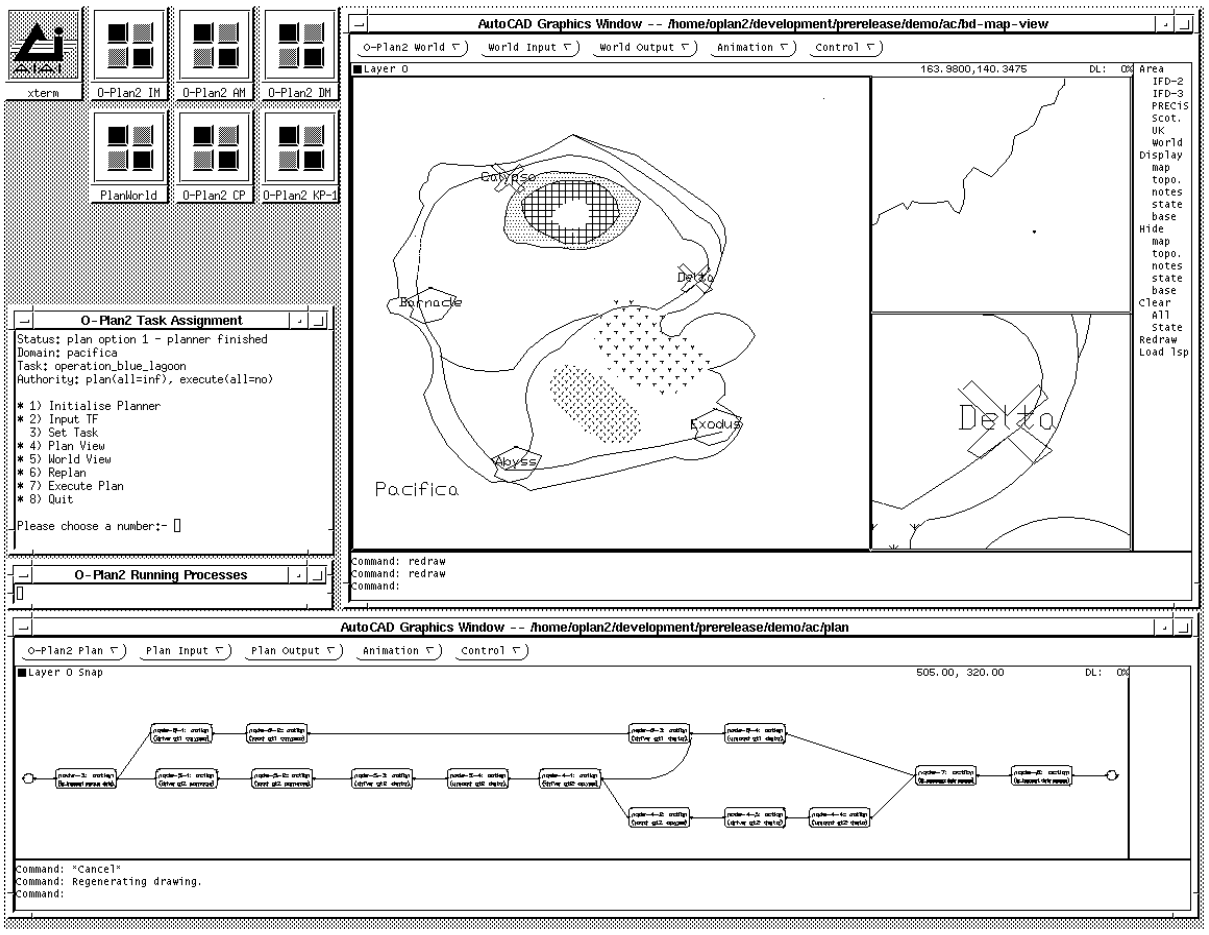
[2] **Currie K.W. and Drabble, B.**, *Knowledge Based Planning Systems: a Tour*, International Journal of Project Management, August 1992, Vol 10, No 3, pp131 - 136,Paper presented at the One day Symposium on Knowledge Based Systems in Project Management, 26th June 1991"

[3] **Reece, G.A., Tate, A., Brown, D. (Mitre) and Hoffman, M. (ISX)***The* PREC*i*S *Environment.* Paper presented at the ARPA-RL Planning Initiative Workshop at AAAI-93, Washington D.C. July 1993. Also available as University of Edinburgh, Artificial Intelligence Applications Institute Technical Report AIAI-TR-140.

[4] **Tate, A**. *Generating project networks.* In procs. of the International Joint Conference on Artificial Intelligence, 1977.

[5] **Tate, A, Drabble, B. and Kirby, R.B.** *O-Plan2: an Open Architecture for Command, Planning and Control* to appear in *Intelligent Scheduling*, (Eds. M. Fox and M. Zweben), Morgan Kaufmann, 1994. Also available as University of Edinburgh, Artificial Intelligence Applications Institute Technical Report AIAI-TR-136.

[6] **Wilkins, D**. *Domain Independent Planning: Representation and Plan Generation.* The International Journal of Artificial Intelligence, Vol. 22, 1984.



```
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 2: Example Screen for the AutoCAD-based PlanView User Interface