Key Concepts in the O-Plan2 Knowledge Based Plan Representation

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

Abstract

Knowledge based plan representations have been developed over several decades of AI planning research. They can support a rich model of processes, tasks, plans, resources and agents. These representations can be used for purposes other than plan generation.

The key concepts within the Edinburgh O-Plan and O-Plan2 plan representation are described. The paper addresses the use of the concepts as a basis for the design of two applications: the Interactive Planning Assistant for the UK Alvey Programme PLANIT Club and the OPTIMUM-AIV system for spacecraft assembly, integration and verification.

1 Introduction

There is an increasing use of Artificial Intelligence and Knowledge Based Systems techniques for the generation of plans and schedules. However, whether using traditional computer support in these areas or whether using some of the early AI systems which can support these tasks, there are problems in making use of *predictive* plans and schedules which are difficult to alter and keep up to date as requirements or circumstances change. There is a demand for improved models of processes, plans and schedules in which intentions and alternatives can be captured and which can provide support to *react* to changing circumstances. There is growing interest in the use of such “knowledge rich” plan representations to augment the capabilities of project management, process planning and job shop scheduling systems.

Plan representations have been developed over several decades of AI planning research [1]. They can support a rich model of processes, tasks, plans, resources and agents. These representations can be used for purposes other than plan generation.

The paper will first describe the contributions to “Knowledge Rich” plan representations from the O-Plan [6] and O-Plan2 [21] research. This research on planning and control architectures has been aimed at building a practical prototype system which can generate plans to meet given task requirements and can reliably execute the plans in the face of plan failures. We are now using our experience of "knowledge-rich" plan representations to augment the plan modelling and analysis capabilities that can be provided and which can work alongside a range of existing systems and tools (such as used today for project management, decision support, process enactment, etc.).

The paper will then describe the PLANIT and OPTIMUM-AIV systems. They used plan representations based on those used within the Edinburgh Nonlin [16], O-Plan (now referred to as O-Plan1) and O-Plan2 systems. This includes some key concepts of hierarchical plans, rich activity and resource models, the use of Goal Structure to capture the intentions behind plan steps, and a language in which to express the activity and process models.

2 Key Concepts in the O-Plan2 Plan Representation

This section describes some of the key contributions from work on the O-Plan2 plan representation (and its predecessors Nonlin and O-Plan1) which were used as a basis for systems like PLANIT and OPTIMUM-AIV.

2.1 Three levels – Strategic, Tactical and Operational

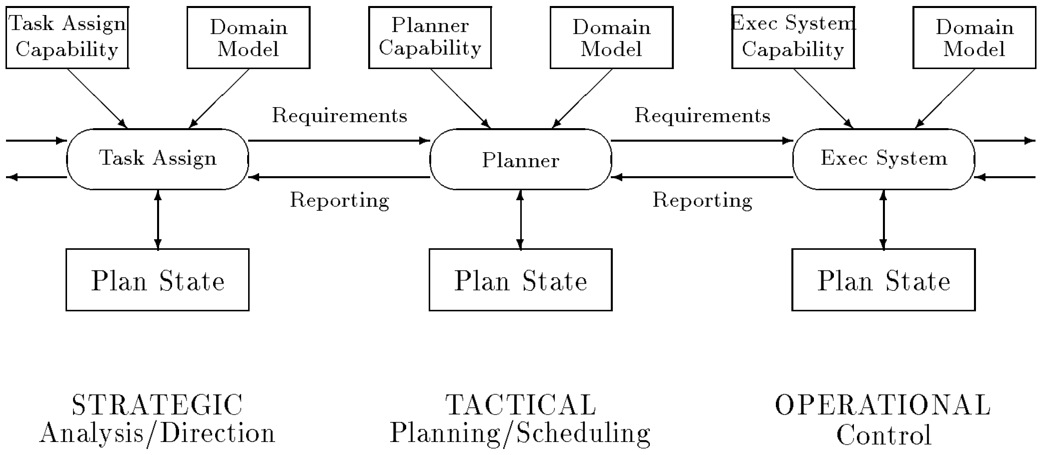

Figure 1: Communication between Task Assignment, Planning and Execution Levels

O-Plan2 deals with command (task assignment), planning and control (plan execution) activities. We are interested in three levels at which plans are represented (see Figure 1). We have deliberately simplified our consideration to three levels with different roles and with possible differences of requirements for skilled user availability, processing capacity and real-time reaction to clarify the presentation of our work. However, the three levels relate to corresponding organisational or functional boundaries in many organisations involved in planning and control. The levels considered are:

1. A user analyses the problem being faced and sets direction by specifying an objective or set of objectives using some suitable interface. We call this process *task assignment*.
2. A *planner* creates a plan to perform the task specified. The planner reasons about time and resources involved in specific instances of generic plans. The planner has knowledge of the general capabilities of an execution system and its enactment capabilities but does not need to know about the detail of how the activities will actually be performed.
3. The *execution system* seeks to carry out the plan as specified by the planner (using that plan as a set of constraints on how it can behave) while working with a more detailed model of the execution environment than is available to the task assigner and to the planner.

These three levels correspond to different types of problem solving capabilities or knowledge: analysis and direction capabilities at the strategic level; the capability to synthesise designs and plans at the tactical level; and control skills at the operational level.

The three levels can communicate to respond to changing requirements, changes in the environment or plan failure. In the O-Plan2 system, the three levels are reflected in separate computational *agents* – the task assignment agent, the planner agent and the execution agent.

We are exploring a common representation for the input/output requirements and capabilities of the planner and plan execution agents. This supports the communication between a user assigning a task or tasks, a planner and an execution system situated in the environment in which the plan is being executed. The common representation includes knowledge about the capabilities of the planning and execution agents, the requirements of the plan and the plan itself.

2.2 Plan State with Agenda

An O-Plan2 *plan state* represents an abstract view of a set of actual plan elaborations that exist within the constraints it contains. Alternative lower level activities, alternative activity orderings, alternative object selections, and so on are aggregated within a high level plan state description. Lower levels can use the flexibility allowed within a plan state while carrying out their own role.

The plan state also contains a list of the current *flaws* or *pending processing requirements* in the plan. These are kept on an *agenda*. Such agenda items could relate to abstract activities that still must be expanded before the plan is considered valid for passing on for execution, unsatisfied conditions, unresolved interactions, overcommitments of resources, time constraint violations, etc. The plan state can thus stand alone from the control structure of the planning and execution agents in that it can be saved and restored, passed to another agent, etc.

In practice, the O-Plan2 architecture is designed for operation in an environment where the ultimate aim of termination will not be achieved. There will be new command requests arriving and earlier ones being modified, parts of plans will be under execution as other parts are being elaborated, execution faults are being handled, etc.

2.3 Knowledge Sources – Plan Modification Operators

The plan state cannot contain arbitrary data elements. An O-Plan2 agent is made up of code that can interpret the plan state data structure and interpret the lists of flaws in such a way that it can select from amongst its computational capabilities and its library of domain specific information to seek to transform the current Plan State it is given into something that is desired by the overall architecture. This is to reduce the list of outstanding agenda entries in the plan state. The O-Plan2 architecture associates a Knowledge Source or *Plan Modification Operator* with each agenda entry type that can be processed [5]. The processing capabilities in the Knowledge Sources are called to handle each agenda entry.

2.4 Plan Patches

The requirement for asynchronously operating planners and execution agents (and indeed users and the real world) means that it is not appropriate to consider that a plan requirement is set, passed on for elaboration to the planner and then communicated to a waiting execution agent which will seek to perform the activities involved. Instead, all components must be considered to be operating already and maintaining themselves in some stable mode where they are responsive to requests for activity from the other components. For example, the execution agent may have quite elaborate local mechanisms and instructions to enable it to maintain a device (say a spacecraft or a manufacturing cell) in a safe, healthy, responsive state. The task then is to communicate some change that is requested from one component to another and to insert an appropriate alteration in the receiver such that the tasks required are carried out.

In effect a *delegation* of part of a plan must take place from some superior to some subordinate agent.

We define a *Plan Patch* as a modified version of the type of plan state used in O-Plan2. Communication between the Task Assigner, Planner and Execution System is via *Plan Patches* which share the same representation as plans [19]. A plan patch is an abstracted or high level representation of a part of the task that is required of the receiver and contains items relevant to the receiver's capabilities. This provides a simplified or black-box view of possibly quite detailed instructions needed to actually perform the activity (possibly involving iterators and conditionals, etc). Complex execution agent representational and programming languages [11], [13] could be handled by using this abstracted view. For example, reliable task achieving *behaviours* which included contingencies and safe state paths to deal with unforeseen events could be hidden from the planner by communication in terms of a simplified and more robust model of the execution operations [12].

Outstanding flaws in the plan patch are communicated along with the patch itself. However, these flaws must be those that can be handled by the receiver. Agent communication "guards" are used to ensure that messages accepted by an agent are understandable by that agent in terms of its capabilities. Plan patches also allow execution errors to be passed back to the planner or information to be passed back to the user.

2.5 Triangle Model of Activity

The O-Plan2 team at Edinburgh are working to simplify some of the notions from AI planning and to relate them better to existing systems engineering requirements capture and modelling languages and methods (like IDEF, CORE, HOOD, etc.).

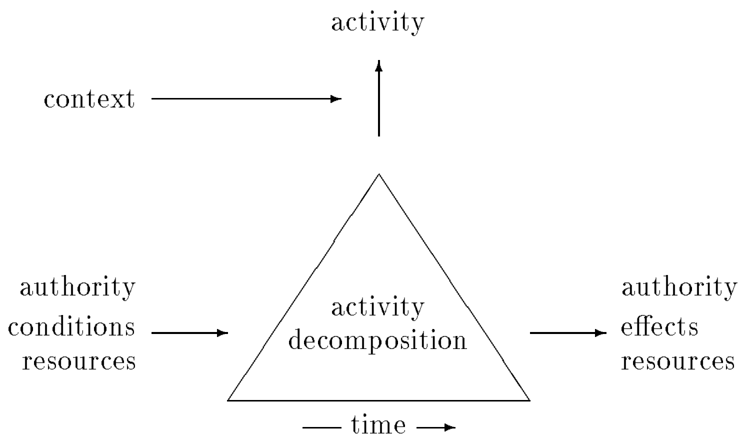

Figure 2: Triangle Model of Activity

This work is reflected in our "triangle" model of an activity (see figure 2). The vertical dimension reflects action decomposition, the horizontal dimension reflects time. Inputs and outputs are split into three principal categories (authority, teleology and resources). Arbitrarily complex modelling is possible in all dimensions. "Types" are used to further differentiate the inputs and outputs, and their semantics.

"Entry" to the model can be from any of the three points in the triangle model: from the top vertex to ask for activity expansions or decompositions, from the right to ask for activities satisfying or providing the output requirement (authority, goal or resource). These two sides are used mostly by AI planners to date. The third side from the left can reflect non-intended triggering conditions for an action and will be needed when improved independent processes are modelled.

The "intentions" or "rationale" behind the use of a particular activity can be related to the features of this triangle model. Normally causality or teleology via the pre-conditions/post-conditions has been used in AI planners for many years to record the plan rationale. In the richer model now in use in O-Plan2, rationale in terms of resource usage and supply or authority requirements and provision may also be stated. This makes it possible to use a uniform approach to the modelling of authority, product flow and resource requirements.

2.6 Intentions within Plans

The “intentions” or “rationale” behind the use of a particular activity in a process or plan can be related to the features of this triangle model. For some time, plan causality or teleology represented in the conditions/effects of activities has been used in AI planners to record plan rationale (or Goal Structure [16] as we call it at Edinburgh). But in the richer model now in use in O-Plan2, for example, it is also possible to differentiate the purpose of the inclusion of an activity in a plan as being to provide resources (i.e. this activity has been included in a plan only to provide a resource, its post-conditions may be a side effect for this *particular* use). The same applies to authority provision [20].

2.7 Constraint Management within Plans

Time constraints, resource usage, object selection and condition/effect causal constraints are handled as an integral part of the overall O-Plan2 system structure by treating specialised constraint management as supporting the core decision making components in the architecture. Specialised (and separable) representations and handlers for the various types of constraint can therefore be employed to deal with time, resources, object and causal constraints within a plan state.

Minimum/maximum bound pairs and preference information is maintained for constrained numerical items such as time windows on activities, resource levels, etc.

2.8 Domain Modelling in Task Formalism (TF)

Domain representation for planning attempts to capture the detailed description of permissible activities or operations within an application area, including information about how conditions imposed on the use of these activities should be satisfied, and their effects on the domain if the activities are used. This richness of required information has led to the specification and development of a high level domain description language called Task Formalism, or more conveniently TF. TF originated in the Nonlin planning system [16] but has been refined and extended for domain descriptions within the O-Plan1 and O-Plan2 planning systems developed at AIAI.

TF is *not* intended as the normal mode of interaction with the user describing a domain. It is an *intermediate language* which fits between a supportive (graphical) user interface and the planner. TF can be considered to be the target language for a helpful domain writer’s support tool. TF has also been designed to allow a useful level of compile time checking to be performed.

TF is used to give an overall hierarchical description of an application area by specifying the activities within the domain and in particular their more detailed representation as a set of sub-activities with ordering constraints imposed. Plans are generated by choosing suitable “expansions” for activities (by refining it to a more detailed level) in the plan and including the relevant set of more detailed sub-activities described therein. Ordering constraints are then satisfied to ensure that asserted effects of some activities satisfy, and continue to satisfy, conditions on the use of other activities. Other constraints, such as a time window for the activity or resource usage required, are also included in the description. These descriptions of activities form the main structure within TF - the *schema*. Schemas are also used in a completely uniform manner to describe *tasks*, set to the planning system, in the same formalism. Other TF structures hold global information and heuristic information about preferences of choices to be made during planning.

The O-Plan2 design allows for different plan state representations in the different agents. Task Formalism is particularly suited to the representation of a plan state within the planner agent and, hence, to act as a basis for communication to the planner's superior (task assignment) and subordinate (execution system) agents. The actual plan state inside the task assignment and execution system agents is likely to differ from that within the planner. For example, the execution system may be based on more procedural representations as are found in languages like PRS (the Procedural Reasoning System [11]) and may allow iteration, conditionals, etc.

3 PLANIT - Interactive Planner's Aid

The UK Alvey Programme's PLANIT Community Club produced a prototype planners' aid called the Interactive Planners' Assistant (IPA) during 1986-7 [8],[15].

In the PLANIT IPA, rich plan representations were *used* without plans being actually generated. Flexible plan representations provided assistance for project management (interfaced to the Metier ARTEMIS system), process planning (interfaced to a Jaguar Cars' process planner) and job shop scheduling (interfaced to the UK Atomic Energy Authority's WASP scheduler). PLANIT could help the user to browse on a plan, monitor its execution and make single step modifications to it as necessary, taking into account knowledge of resources, agent capabilities, how the original plan was constructed and what the aims of the plan were.

The PLANIT IPA could perform "single step plan modifications" to allow a user to monitor the actual execution of a plan and to investigate options and the consequences of changes or decisions. Active support to providing valid options for user choice was given by the system. The IPA made use of AI plan representations which could capture intentions, rich resource models, dependencies between tasks, plan alternatives, domain capabilities and task descriptions. The plan representation used was based on Edinburgh work on the AI planners Nonlin [16] and O-Plan1 [6].

The PLANIT Club involved 29 organisations in the UK who worked with support from the UK government's Alvey Programme of research into Intelligent Knowledge Based Systems during 1986 and 1987. AIAI provided the core rationale and design for the IPA – the main prototype tool produced during the Club programme. A paper [18] gives details of our experience of participation in the PLANIT Club.

3.1 Application – An Enterprise for Oil Tanker Truck Production

The PLANIT IPA was applied to show how an enterprise could use knowledge based plans to coordinate the production of an oil tanker truck. This showed the relationship between:

- *process plans* for the production of parts of a tanker. In the IPA demonstration several alternative available methods of producing the tanker oil vessel "end cap" were included (by spinning, by pressing or by rolling and welding).
- *project plans* to show the various steps involved in configuring and making the truck including resource needs and time scales.
- *job shop schedules* for the part of the enterprise concerned with producing the end caps.

This demonstration application showed how the IPA could be used to monitor the execution of the project plan for a specific truck. During the building of the truck, shop floor production difficulties with the chosen end cap production method led to an alternative process being proposed which allowed the project plan to be altered to avoid a bottleneck in the job shop. This went beyond the typical support in project management and job shop scheduling tools, which would have focussed on altering times or resources to avoid problems rather than questioning the plan logic or the actual process alternatives currently built into the plan.

3.2 Use of Knowledge Based Plan Features

PLANIT made use of a number of plan representation features described in section 2.

1. **Separation of Domain, Task and Plan Knowledge.**

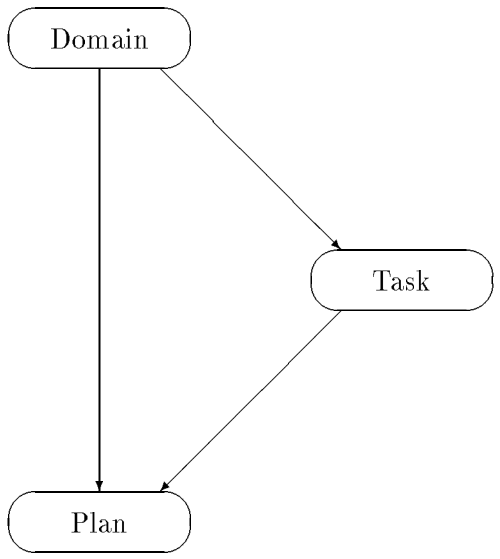

Figure 3: Dependencies between Domain, Task and Plan Knowledge Partitions

Figure 3 illustrates the dependency relationships between Domain, Task and Plan knowledge. Tasks and Plans are both based upon the entities in the Domain model. Plans also are elaborations of a specific Task.

- *domain* knowledge, describes "fixed" things like facilities, organisational relationships, procedures, systems, products and the types of resource available. This knowledge is likely to be highly reusable for many different requirements.
- *task* knowledge, describes the objectives such as the goal or goals which the plan is designed to achieve, the activity to be carried out, the *actual* resources available, the time available, etc.
- *plan* knowledge, describes a particular way (currently under exploration) in which the specified task objectives can be achieved in the current domain.

2. **Separation of plan constraint management into specialist Activity, Resource and Product Managers.**

   In the PLANIT IPA, any change of plan had to meet constraints on activities, resources and products. A number of specialised constraint managers ensured that their own set of constraints were met.

3. **Use of Goal Structure to explain the functions and intentions of a plan.**

   The IPA could inform the user of the intention of steps in a plan and of their relationship with other steps in the plan and with resource usage.

4. **Description of Planning and Re-planning Capabilities.**

   Knowledge of how to manipulate plans was held as a number of *Plan Modification Operators* (PMOs) through which plan state changes could be made. These PMOs were of two types:

   - *Analysis and planning* knowledge, describing ways of critiquing, optimising or changing the current plan.
   - *Execution* knowledge, describing ways in which the current plan can be translated into action in the current domain with the current task objectives.

5. **Ability to do single step plan modification.**

   Since the IPA had plan analysis and plan modification capabilities, it was able to suggest valid changes to the plan and inform the user of the consequences of proposed changes to the plan, thus allowing what-if questions to be answered. Valid Plan Modification Operators could be used to "single-step" through proposed plan changes.

6. **Ability to browse plan states.**

   The IPA provided the ability to browse temporal or logical states in a simulation of the plan. This is useful for critiquing and optimising plans and can provide useful communication back to the person responsible for the task or plan at an appropriate level.

These features combined to allow the PLANIT IPA to act as a flexible tool to support a user who needed to monitor processes, plans and schedules in actual use and to support a user in making changes to a plan during its use. This type of active support to *using* plans, is an area where many existing computer based tools are weak.

4 OPTIMUM-AIV

Optimum-AIV [3],[4] is a more recent example of the use of flexible plan representations in a project management domain alongside Metier's ARTEMIS project support tools. The plan causal structure is represented to give support to replanning when problems occur.

Optimum-AIV is able to support the generation and execution of plans for spacecraft assembly, integration and verification. The system was built by a consortium including AIAI for the European Space Agency (ESA). The project followed on from earlier work for ESA on the PlanERS-1 planning system [10] to support mission planning and operations for the ERS-1 Earth Resources Spacecraft.

The system shows how plan representations from AI research have been used to link to and improve upon the traditional sources of project management information in use for such tasks. Optimum-AIV could be used alongside Metier's ARTEMIS project management tool which is already in use within ESA for spacecraft assembly, integration and verification work.

4.1 Application – Assembly, Integration and Verification of Spacecraft

The assembly, integration and testing of complex spacecraft is a demanding activity, requiring careful project management. It involves the coordination of many resources, facilities, equipment, test capabilities, etc. Spacecraft components are brought together from suppliers and scientific establishments for environmental, electrical, communications and other tests. The results of the tests conducted will establish whether corrective actions need to be taken to remedy any problems found during the process. Careful records must be kept of the tests and the actions taken. Deadlines, e.g. for meeting launch windows for the spacecraft or for use of expensive testing facilities, must be respected.

It is therefore essential that a flexible plan is maintained which is responsive to the outcomes of tests and to changing delivery or resource availability dates. AI based plan representations were used to augment the existing project management aids at ESA for spacecraft AIV during the OPTIMUM-AIV project.

4.2 Use of Knowledge Based Plan Features

Optimum-AIV made use of a number of plan representation features described in section 2.

1. **Hierarchical Description of Activities and Plans.**

Activities that may be performed in the domain being modelled could be described in a hierarchical fashion. Plans also were hierarchically described. This allowed information to be presented at an appropriate level of detail for the use to which it was to be put.

2. **Search for and Re-use of Plans.**

   AI oriented plan generation and search methods were employed. Plan libraries storing generic plans could be maintained to allow previous experience to be recalled and tailored to new situations.

3. **Plan Critiquing.**

   Changes made to plans could be criticised or critiqued to ensure that constraints and intentions were not compromised as changes were made to them.

4. **Constraint Management.**

   Time feasibility windows were maintained for activities in the plan and for resource usage.

5. **Capture of Intentions in Plans.**

   The capture of intentions within plans using Goal Structure was an aid to explanation or justification of parts of the plan and to facilitate flexible re-planning proposals.

6. **Test Recovery Plan Patches.**

   Test recovery plan patches (called "Test Failures Solutions") were provided to enable a plan to be brought back on track after spacecraft component test failures occurred. The test recovery plan fragments were represented in the same form as activity descriptions and overall plans.

These features combined to allow OPTIMUM-AIV to support a project engineer in initially creating and then monitoring the execution of an AIV plan. The system provided active and flexible support to the consideration of options when tests failed to ensure the plan was brought back on track.

The OPTIMUM-AIV system is now in use on projects within the European space industry [14].

5 Summary

AI based plan representations have found productive use in systems which do not necessarily involve only plan generation. The flexible representation of information about activities, tasks and plans allows dependency and intention knowledge to be captured such that intelligent support can be provided for plan analysis, use, monitoring and change. The key concepts within the O-Plan2 plan representation and their contribution to such productive uses have been described.

Acknowledgements

O-Plan2 is an on-going project at Edinburgh. I am grateful to Brian Drabble and Jeff Dalton on the O-Plan2 project and other collegues who have worked on the University of Edinburgh planning research projects for their input to the work described here. Current O-Plan2 work is supported by the US Advanced Research Projects Agency (ARPA) and the US Air Force Rome Laboratory acting through the Air Force Office of Scientific Research (AFSC) under contract F49620-92-C-0042. The United States Government is authorised to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon.

PLANIT concepts were developed by a team at Edinburgh including Mark Drummond and John Fraser. OPTIMUM-AIV was a project for the European Space Agency involving a consortium with AIAI (Jussi Stader in particular), CRI, Matra Espace and Progrespace.

References

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

[2] Alvey Directorate, *Alvey Grand Meeting of Community Clubs*, available through IEEE, Savoy Place, London, 1987.

[3] Aarup, M., Arentoft, M.M., Parrod, Y., Stader, J., Stokes, I. & Vadon, H., *OPTIMUM-AIV: A Knowledge Based Planning and Scheduling System for Spacecraft AIV*, in Knowledge Based Scheduling, (eds. Fox, M. & Zweben, M.), Morgan Kaufmann, 1994.

[4] Arentoft, M.M., Parrod, Y., Stader, J., Stokes, I. & Vadon, H., *OPTIMUM-AIV: A Planning and Scheduling System for Spacecraft AIV*, Telematics and Informatics Vol. 8, No. 4, pp. 239-252, Pergamon Press, 1991.

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

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

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

[8] Drummond, M.E., & Tate, A., *PLANIT Interactive Planners' Assistant - Rationale and Future Directions*, AIAI-TR-108, AIAI, University of Edinburgh, 1992.

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

[10] Fuchs, J.J., Gasquet, B., Olalainty, B., & Currie, K.W., *Plan-ERS1: An Expert System for Generating Spacecraft Mission Plans*, Proceedings of the First International Conference on Expert Planning Systems, Brighton, UK. Available from IEE, London, 1990.

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

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

[13] Nilsson, N.J., *Action Networks*, Proceedings of the Rochester Planning Workshop, October 1988.

[14] Parrod, Y. and Valera, S., *OPTIMUM-AIV: A Planning Tool for Spacecraft AIV*, in *Preparing for the Future*, ESA's Technology Programme Quarterly, Vol. 3 No. 3, European Space Agency, September 1993.

[15] PLANIT Club, *PLANIT Club Final Report* published on behalf of the PLANIT Club by Systems Designers plc, Fleet, Hampshire, UK, document ref. C03209, 1987.

[16] Tate, A., *Generating Project Networks*, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-77), Cambridge, Mass. USA, 1977.

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

[18] Tate, A., *Research to Product - the Community Club Experience*, invited keynote paper to Artificial Intelligence '87, Sydney, Australia, in *Artificial Intelligence Developments and Applications*, (eds. J.S.Gero and R.Stanton), pp. 5-16, North-Holland, Amsterdam, 1987.

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

[20] Tate, A. *Authority Management - Coordination between Planning, Scheduling and Control*, Workshop on Knowledge-based Production Planning, Scheduling and Control at the International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France, August 1993.

[21] Tate, A., Drabble, B. & Kirby, R., *O-Plan2: an Open Architecture for Command, Planning and Control*, in Knowledge Based Scheduling, (eds. M.Fox, M. and M.Zweben), Morgan Kaufmann, 1994.