Authority Management - Coordination between Task Assignment, Planning and Execution

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

Abstract

The O-Plan2 Open Planning Architecture allows for three separate agents for Task Assignment, Planning and Execution. Each O-Plan2 agent maintains a separate Plan State which consists of the agent's predictive plan and a set of outstanding agent processing requirements in the form of an Agenda. This agenda can represent plan flaws or unsatisfied requests from a superior agent, or still to be processed issues raised by a subordinate agent.

Previous work has defined a means to communicate task requirements, plans and partial plans and execution information between the three agents in the form of Plan Patches. Plan Patches may include additional agenda items destined for the receiving agent. An agent which accepts a Plan Patch uses the information in it to augment or modify its local Plan State and Agenda – which in turn induces processing.

The Task Assignment agent in O-Plan2 tells the planner and execution agents when they can create a plan for a nominated task and when a plan can be executed. This is done through a simple menu interface today. It is intended that O-Plan2 will support Authority Management in a more comprehensive and principled way in future. This document sets out background to how such Authority Management could operate.

An example application relating to Non-combatant Evacuation Operations (NEOs) as undertaken for rapid response planning in the US Joint Planning and Execution Community is provided. This is used to show how the proposed framework for the management of authority in coordinating planning and execution can work in practice.

1 O-Plan2 Background

O-Plan was initially conceived as a project to provide an environment 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.

O-Plan grew out of the experiences of other research into AI planning, particularly with Nonlin [22] and "blackboard" systems [17]. 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 assumed that the reader is familiar with these works as the bibliography does not cover all of them. The same volume [1] includes an introduction to the literature of AI planning. A description of the first O-Plan system (now referred to as O-Plan1) is provided in [6].

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 are the task 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 task planning or resource scheduling.
- to clarify the functions of components of a planning and control architecture.
- to draw on the O-Plan1 experience and to improve on it especially with respect to flow of control [24].
- to provide an improved version of the O-Plan 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.

O-Plan2 is incorporated within a blackboard-like framework; for efficiency reasons we have chosen an agenda driven architecture. Items on the agendas represent outstanding tasks to be performed during the planning process, and they relate directly to the set of *flaws* identified as existing within the emerging plan. A simple example of a *flaw* is that of a condition awaiting satisfaction, or an action requiring refinement to a lower level. A controller chooses on each processing cycle which flaw to operate on next.

The O-Plan2 system is more fully described in [25] and [26]. The O-Plan2 architecture has also been used as the basis for the TOSCA manufacturing scheduler [4].

2 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 (teleology) 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 targeted type represent an important class of industrial relevance.

3 Task Assignment, Planning and Execution in O-Plan2

Edinburgh research on planning and control architectures is aimed at building a practical prototype system which can generate plans and can reliably execute the plans in the face of simple plan failures. We are using our experiences in dealing with applications of AI planning techniques to practical projects to develop a planning system that closes the loop between planning and executing. There have been some successes with previous attempts at closing the loop [11], [13], [16], [27], but often the plans generated were rather limited and not very flexible. In general, the complexities of the individual tasks of plan representation, generation, execution monitoring and repair has led to research into each of these issues separately. In particular, there is now a mismatch between the scale and capabilities of plan representations proposed for real-time execution systems [14], [18] [20], and those that can be generated by today's AI planners. However, in most realistic domains the demand is for a system that can take a command request, generate a plan, execute it and react to simple failures of that plan, either by repairing it or by re-planning. Explicit knowledge about the structure of the plan, the contribution of the actions involved and the reasons for performing plan modifications at various stages of the plan construction process, provides us with much of the information required for dealing with plan failures. Such knowledge is also essential for further planning and re-planning by identifying generalisations or contingencies that can be introduced into the plan in order to avoid similar failures.

One of the simplifications most planners to date have made is to assume plans are constructed with full knowledge of the capabilities of the devices under their control. Thus, executing such plans involves the direct application of the activities within the plan by an execution agent which has no planning capability. Unfortunately, unforeseen events will occur causing failure of the current plan and a request for repair of the plan or re-planning directed at the planning system. Building into the execution agent some ability to repair plans and to perform further plan elaboration with a more detailed domain model would improve the problem solving performance of the execution agent, especially when it is remote from the central planning system.

3.1 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 *task 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 detail of 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 task assigner and to the planner.

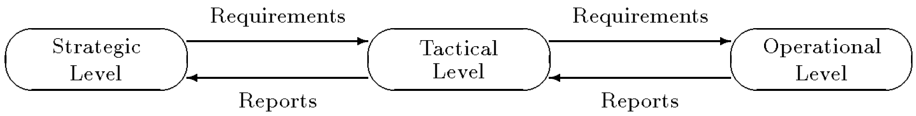


Figure 1: Communication between Task Assignment, Planning and Execution Agents

Figure 1 shows the relationship between the three levels. 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.

The execution agent executes the plan by choosing the appropriate activities to achieve the various sub-tasks within the plan, using its knowledge about the particular resources under its control. Thus, the central planner 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. The execution agent communicates with the real world by executing the activities within the plan and responding to failures fed back from the real world. 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.

3.2 Use of Dependencies

The use of dependencies within planning promises great benefits for the overall performance of a command, planning and control system particularly for plan representation, generation, execution and repair.

The notion of the teleology of a plan, which we call the Goal Structure [22], refers to the dependencies between the preconditions and postconditions of activities involved in the plan. Although, such dependencies have been shown to be useful for describing the internal structure of the plan and for monitoring its execution [13], [23], there has been no comprehensive discussion of their use in all aspects of plan generation, execution monitoring and plan repair. Intention based knowledge-rich plan representations of this type were used as the basis for the design of an Interactive Planning Assistant [2] [12] for the UK Alvey PLANIT Club. This allowed for browsing, explaining and monitoring of plans represented in a more useful form than that provided in conventional computer based planning support tools. More recently, O-Plan2 style plan representations were used within the OPTIMUM-AIV system [3] for spacecraft assembly, integration and verification at the European Space Agency in work conducted by a consortium of which AIAI was a part.

Early work on Decision Graphs [15] at Edinburgh has shown how the explicit recording of the decisions involved in the planning process could be used for suggesting where and how much re-planning should take place when unforeseen situations make the current plan fail. Some work to link these ideas with Nonlin was undertaken during the mid 1970's [7].

4 Representing and Communicating Plans

This section describes the representation of plans and agent plan states in O-Plan2 and the way in which parts of the plan state of one agent can be extracted and communicated (as a patch) to the plan state of another agent.

4.1 Plan States

One of the most important problems which needs to be addressed in any planning system is that of plan representation. An O-Plan2 agent's *plan state* holds a complete description of a plan at some level of abstraction. The plan state also contains a list of the current *flaws* in the plan. Such flaws could relate to abstract actions that still must be expanded before the plan is considered valid for passing on for execution, unsatisfied conditions, unresolved interactions, overcommitments of resource, time constraint faults, etc. The Plan State can thus stand alone from the control structure of the AI planner in that it can be saved and restored, passed to another agent, etc.

At any stage, a plan state represents an abstract view of a set of actual plans that could be generated within the constraints it contains. Alternative lower level actions, alternative action orderings and object selections, and so on are aggregated within a high level Plan State description.

Task Formalism (TF) (as used in Nonlin and O-Plan) is a declarative language for expressing action schemata, for describing task requests and for representing the final plan. It allows time and resource constraints in the domain to be modelled. The planner can take a plan state as a requirement (created by a TF Compiler from the user provided task specification in TF) and can use a library of action schemata or generic plan state fragments (themselves created by the TF Compiler from a domain description provided by the user) to transform the initial plan state into one considered suitable for termination. This final plan state could itself be decompiled back into a TF description if required.

Our design intention for O-Plan2 is that any plan state (not just the initial task) can be created from a TF description and vice versa. This was not fully achieved in the O-Plan1 prototype [6], but this remains our goal.

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 to 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 [14]) and may allow iteration, conditionals, etc.

We believe that the basic notions described above can serve us well as a basis for an attack on the problem of coordinated command, planning and execution in continuously operating domains. There must be a means incrementally to communicate plan related information between the agents involved with commanding, planning and executing plans - each of which will have their own level of model of the current command environment, plan and execution environment. We will explore the properties that we must seek from our basic notions in the following sections.

4.2 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 actions involved. Instead, all components must be considered to be operating independently and maintaining themselves in some stable mode where they are responsive to requests for action 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.

Our approach is to combine the ideas above to define an *Incremental Plan State* with three components:

- a plan patch,
- plan patch flaws as an agenda of processing requirements,
- plan patch attachment points.

Such Incremental Plan States are used for two way communication between the task assigner and the planner and between the planner and the execution agent. The O-Plan2 Plan State structures and flaw repertoire has been extended to cope, initially, with a dumb execution agent that can simply dispatch actions to be carried out and receive fault reports against a nominated set of conditions to be explicitly monitored (as described in [23]). In future research, the Plan State data structures and flaw repertoire will be extended again to cope with a semi-autonomous execution agent with some capability to further elaborate the Incremental Plan States and to deal locally with re-planning requirements [19].

We define a *Plan Patch* as a modified version of the type of Plan State used in O-Plan1. It has some similarity to an operator or action expansion schema given to an AI planning system in that it is an abstracted or high level representation of a part of the task that is required of the receiver using terminology relevant to the receiver's capabilities. This provides a simplified or black-box view of possibly quite detailed instructions needed to actually perform the action (possibly involving iterators and conditionals, etc). Complex execution agent representational and programming languages can be handled by using this abstracted view (e.g., [14], [18]). 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 [16].

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.

It can be seen that the arrangement above (mostly assumed to refer to the communication between a planner and execution agent) also reflects the communication that takes place between a task assigner and the planner in an O-Plan2 type AI planner. Requiring rather more effort is the investigation of suitable Plan Patch constructs to allow execution errors to be passed back to the planner or information to be passed back to the task assigner, but we believe that this is a realistic objective.

There is a need to communicate the points at which the Plan Patch should be attached into the full Plan State in the receiver. The sender and receiver will be operating asynchronously and one side must not make unreasonable assumptions about the internal state of the other.

Metric time is a simple means to give an attachment point as all agents can share the notion of a real-time clock. However, the use of metric time as an attachment point lacks flexibility. It gives the receiver little information about the real intentions behind the orderings placed on the components of the Plan Patch. It will, in some cases, be better to communicate plan patches relative to the Goal Structure [22] of the receiver or qualified in some other way to give the receiver more flexibility.

4.3 Plan Transactions

The overall architecture must ensure that an Incremental Plan State can be understood by the receiver and is accepted by it for processing. This means that all the following are understood by the receiver:

- plan patch description is clear,
- plan patch flaws can be handled by the receiver's Knowledge Sources,
- plan patch attachment points are understood.

It is important that the sender and receiver (whether they are the user and the AI planner, the planner and the execution agent, or one of the reverse paths) can coordinate to send and accept a proposed Incremental Plan State which the receiver must assimilate into its own Plan State. We propose to use *transaction processing* methods to ensure that such coordination is achieved.

We have created some specific flaw types and Knowledge Sources in the various components (task assignment, AI planner and execution agent) to handle the extraction and dispatch (as an Incremental Plan State) of a part of an internal Plan State in one component, and the editing of such an Incremental Plan State into the internal Plan State of the receiver. The "extraction" Knowledge Sources must be supplied with information on the Plan Patch description, flaw types and attachment points that the receiver will accept. This constitutes the primary source of information about the capabilities of the receiver that the sender has available and its representation will be an important part of the research.

Communication "guards" will ensure that the *a priori* criteria for acceptance of an Incremental Plan State for processing by the receiver's Knowledge Sources are checked as part of the Plan Transaction. It may also be the case that initial information about urgency will be able to be deduced from this acceptance check to prioritise the ordering of the new flaws with respect to the existing entries on the agenda in the receiver.

5 Example – NEO Missions

This section described an example application which is used to show how the proposed framework for the management of authority in coordinating planning and execution can work in practice.

Non-combatant Evacuation Operations (NEOs) are undertaken to provide rapid response to a variety of circumstances, including natural disasters, requiring the evacuation of civilians from trouble zones. NEO operations are often characterised by the need for rapid deployment of equipment and personnel, often involving multiple military and civil aid agencies, to ensure the timely availability of effective aid.

Crisis action planning procedures are used by the US Joint Planning and Execution Community for such circumstances [10]. This section establishes the terminology used for the planning and execution of Non-combatant Evacuation Operations (NEOs) in a form which is useful in addressing research issues at Edinburgh.

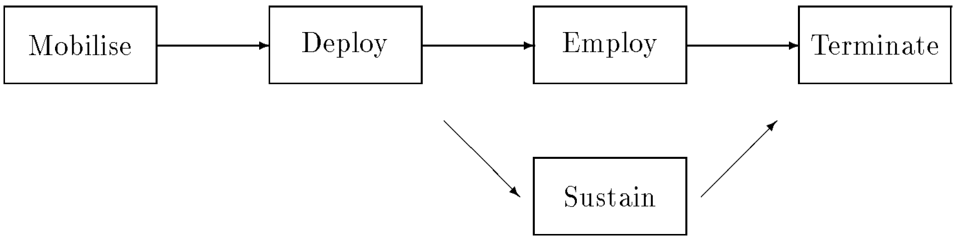
This simple evacuation operation shows the development of a plan for Non-combatant Evacuation Operation (NEO) from a hypothetical island named Pacifica. Though this scenario is completely fictious, the objectives, issues addressed, and underlying data are intended to be sufficiently realistic for the research. Publicly available United States Transportation Command Operations (USTRANSCOM) documents (see [8, 9]) were used as guides to determine some of the factors used in the examples.

5.1 NEO Mission Plan Options

It is customary to develop a small number of different *plan options* or *Courses of Action* (COAs) during the preparation for a NEO mission.

5.2 NEO Mission Phases

The generic *phases* of a NEO mission are:



In practice, phases overlap in time in a "ladder" fashion. For example, some employment can take place before all deployment is complete.

5.3 NEO Mission Levels

There are a number of levels of refinement of a plan for a NEO mission. These relate to the different levels of the COA development process:

1. Select mission type (e.g., scale of military operation).
2. Identify specific threats and locations.
3. Select employment operations, forces and destinations.
4. Add deployment actions for all units.
5. Add location information and compute main movement durations.
6. Add further airlift and sealift movements and durations.

Further refinement of the plan to add sustainment actions may then be done.

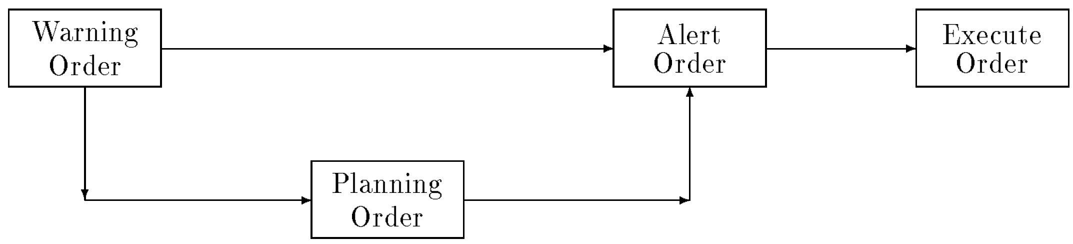
There are two levels of plan which can be stored and retrieved within the military planning community and are commonly referred to:

CONPLAN is a concept plan (developed down to level 3).

OPLAN is a detailed plan (developed down to level 6). It may be a development of a previous CONPLAN.

5.4 JCS Authority Orders

Authority to plan and execute plans is provided by the Joint Chiefs of Staff (JCS) to a Commander in Chief (CINC) with respect to a specific mission via a series of *orders*. See section 7 of [10] for more details. A simplified outline of the progression of orders is shown here. In practice, it is possible to omit the planning order in some circumstances and some earlier orders may be omitted by going straight to later orders with higher authority levels.



Authority management occurs via the issue of these orders and controls the flow between agencies and people responsible for task assignment, planning and execution in the US military planning and execution community.

In terms of planning and execution authority the orders have the following meaning:

	authority to plan to level	authority to execute mission phases
WARNING	CONPLAN	NONE
PLANNING	OPLAN	NONE
ALERT	OPLAN	MOBILISE
EXECUTE	OPLAN	ALL

Specific authority over planning levels and the external exposure of planning operations is important as even knowledge of the decision to create a plan to deal with some occurrence can have an impact on the occurrence itself (e.g., it may be possible to counter a threat by preparing a plan and making it public).

6 O-Plan2 Support for Authority Management

At the moment the Task Assignment agent in O-Plan2 tells the planner and execution agents when they can create a plan for a nominated task and when a plan can be executed. This is done through a simple menu interface today.

It is intended that O-Plan2 will support authority management in a more comprehensive and principled way in future. This section describes the way in which this is being done.

6.1 O-Plan2 Concept Extensions

O-Plan2 will support:

- the notion of separate *plan options* which are individually specified task requirements, plan environments and plan elaborations. The Task Assignment agent can create as many as required. The plan options may contain the same task[1] with different search options or may contain a different task and environmental assumptions. It is possible to have only one plan option as the minimum[2].
- the notion of plan *phases*. These are individually provided actions or events stated explicitly in the top level task description given by the Task Assignment agent. Greater precision of authority management is possible by specifying more explicit phases at the task level. It is possible to have only one "phase" in the task as the minimum[3].
- the notion of plan *levels*. Greater precision of authority management is possible by specifying more explicit levels in the domain Task Formalism (TF). It is possible to have only one "level" in the domain as the minimum.
- for each "phase", planning will only be done down to an authorised "level" at which point planning will suspend leaving appropriate agenda entries until deeper planning authorisation is given.
- execution will be separately authorised for each "phase".

The planner agent will only need to be able to refer to code numbers for plan options (1 upwards), phases (node numbers in the current plan option), and levels (0 upwards). Domain related names that are meaningful to the user may be associated with these numbers through the Task Assignment agent.

New Task Formalism forms and simple extensions to existing forms will support authority management in O-Plan2.

Changes of authority are possible via Task Assignment agent communication to the Planner agent. This is in the context of a current plan option and task provided previously. No TF for changing authorities is envisaged at this stage.

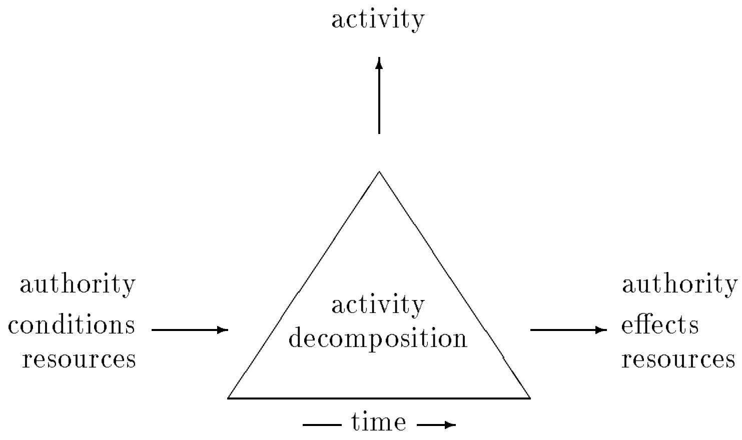
[1] Multiple conjunctive tasks in one scenario is also possible.
[2] Plan options may be established and explicitly switched between by the Task Assignment agent.
[3] In fact any sub-component of any task schema or other schema included by task expansion in a plan can be referred to as a "phase" within the O-Plan2 planner agent. This can be done by referring to its node number.

6.2 Task Formalism Extensions

The O-Plan2 team at Edinburgh are actively revising O-Plan2 Task Formalism (TF) and in particular are trying to simplify some of the notions and to relate them better to existing software engineering and systems engineering requirements capture and modelling languages and methods (like IDEF, CORE, HOOD, etc). This revision is incorporating facilities for authority management.

The main item in TF is a *schema* which describes an action and its decomposition to a lower level of detail. O-Plan2 allows for this same type of representation to be used for task descriptions, plans, partial plans, action schemas and other operators or primitive actions.

A TF schema reflects our "triangle" model of an activity. The vertical dimension reflects action (or plan or task) decomposition, the horizontal dimension reflects time. Inputs and Outputs are split into three principal categories (authority, effects/conditions[4] 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 action (or plan or task) decompositions, from the right to ask for actions (or plans or tasks) satisfying or providing the output requirement (a desired effect or "goal" to satisfy a condition), a required resource, or a needed authority. These two sides are used mostly by our planners to date. The third side can reflect triggering conditions for an action (or plan or task) and will be needed when improved independent processes (made up of unplanned events) are modelled (as in Drabble's Excalibur system prototype [11]).

[4] Plan Teleology – the Causal or Goal Structure of the plan

6.3 Task Assignment Agent User Interface

The Task Assignment agent of O-Plan2 will support authority management in a task setting framework. To establish an appropriate basis for future developments and allow for some initial internal support for authority management to be incorporated, the Task Assignment agent interface for version 2.1 will reflect the use of plan options, phases, levels and authority. For example:

```
Domain: pacifica
Status: plan option 1 - planning ...
Task: Operation_Blue_Lagoon
Authority: plan(all=inf), execute(all=no)
```

A HARDY-based[5] user interface to the Task Assignment agent which contains support for authority management is being designed and prototyped at Edinburgh.

7 Summary

This paper has introduced the requirement to clarify the explicit authorisation of planning and execution where separate command/task assignment, tactical planning and operational execution agents are involved. It is argued that this can improve coordination between such agents in a distributed environment. Clarity of specification of the assigned task and an understanding of the current authority to proceed to generate more detailed plans or to execute all or part of the plan can be a valuable aid to coordination.

The paper has described the need to identify a small number of explicit plan related concepts which can be referred to unambiguously between the various agents. These are: plan *options*, plan *phases* and plan *levels*. With these concepts, effective coordination at various levels of control can be achieved.

The work and the plan related concepts introduced have been related to an example application in military Non-combatant Evacuation Operations (NEOs).

[5]HARDY is a C++ based diagraming aid and hypermedia tool from AIAI.

References

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

[2] Alvey Directorate (1987) Alvey Grand Meeting of Community Clubs. Available through IEE, Savoy Place, London.

[3] 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.

[4] Beck, H.A. Constraint Monitoring in TOSCA, in *Proceedings of the 1992 AAAI Spring Symposium on Practical Approaches to Scheduling and Planning*, (eds. M.E.Drummond, M.Fox, A.Tate and M.Zweben), NASA Ames Research Center, AI Research Branch Technical Report FIA-92-17. Also available as an American Association of AI (AAAI) Technical Paper.

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

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

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

[8] Day, D. and McAlpin, S. Supplementary material describing USTRANSCOM transportation planning, Department of the Air Force, 1989.

[9] Department of the Air Force, HQ, USAF, Washington, DC 20330-5000, (May 1987), *Airlift Planning Factors (Military Airlift)*, Air Force Pamphlet 76-2.

[10] Department of Defense, Armed Forces Staff College (AFSC). *The Joint Staff Officer's Guide 1991.* AFSC Pub. 1.

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

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

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

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

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

[16] Malcolm, C. and Smithers, T. (1988) *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.

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

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

[19] Reece, G.A. (1992) *Reactive Execution in a Command, Planning and Control Environment*, Ph.D Dissertation Proposal, Department of AI Discussion Document, The University of Edinburgh.

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

[21] Sacerdoti, E. A structure for plans and behaviours. *Artificial Intelligence series, publisher. North Holland, 1977.*

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

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

[24] Tate, A. & Drabble, B. O-Plan2: Choice Ordering Mechanisms in an AI Planning Architecture in Proceedings 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.

[25] Tate, A., Drabble, B. & Kirby, R. Spacecraft Command and Control Using AI Planning Techniques - The O-Plan2 Project - Final Report, USAF Rome Laboratory Technical Report RL-TR-92-217. August 1992. Available as AIAI-TR-109, University of Edinburgh.

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

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

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