RL-TR-95-235
Final Technical Report
December 1995

AN ENGINEER'S APPROACH TO THE APPLICATION OF KNOWLEDGE BASED PLANNING AND SCHEDULING TECHNIQUES TO LOGISTICS

University of Edinburgh

Sponsored by Rome Laboratory and
Advanced Research Projects Agency
ARPA Order No. 8599

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be releasable to the general public, including foreign nations.

RL-TR-95- 235 has been reviewed and is approved for publication.

APPROVED:

NORTHRUP FOWLER III, Ph.D.
Project Engineer

FOR THE COMMANDER:

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/ (C3C), Rome NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

AN ENGINEER'S APPROACH TO THE APPLICATION OF KNOWLEDGE BASED PLANNING AND SCHEDULING TECHNIQUES TO LOGISTICS

Austin Tate
Brian Drabble
Jeff Dalton

Contractor: University of Edinburgh
Contract Number: F49620-92-C-0042
Effective Date of Contract: 15 July 1992
Contract Expiration Date: 14 July 1995
Short Title of Work: An Engineer's Approach to Knowledge Based Planning
Period of Work Covered: Jul 92 - Jul 95

Principal Investigator: Austin Tate
Phone: 0131-650-2732

RL Project Engineer: Northrup Fowler III, Ph.D.
Phone: (315) 330 -3011

This research was jointly sponsored by the Advanced Research Projects of the Department of Defense and by Rome Laboratory and was monitored by Northrup Fowler III, Ph.D., RL/C3C, 525 Brooks Rd, Rome NY 13441-4505.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1995	Final Jul 92 - Jul 95

4. TITLE AND SUBTITLE
AN ENGINEER'S APPROACH TO THE APPLICATION OF KNOWLEDGE BASED PLANNING AND SCHEDULING TECHNIQUES TO LOGISTICS

5. FUNDING NUMBERS
C - F49620-92-C-0042
PE - 61101E
PR - H599
TA - 00
WU - 01

6. AUTHOR(S)

Austin Tate, Brian Drabble, and Jeff Dalton

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

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

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

Rome Laboratory/C3C
525 Brooks Rd
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-95-235

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Northrup Fowler III, Ph.D./C3C/(315) 330-3011

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The O-Plan research has achieved a clearer understanding of the components necessary in a flexible planning system and has shown how such components can be combined in a systems integration architecture. The work has determined improved ways to restrict search in a planner by using the knowledge available from modeling an application domain, and it has developed a better characterization of plans as sets of activity constraints, opening up many possibilities for richer distributed, cooperative, and mixed-initiative planning systems in the future. The project has created a prototype implementation and has demonstrated it on a class of realistic applications.

14. SUBJECT TERMS

Planning, Artificial intelligence

15. NUMBER OF PAGES
342

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

O-Plan is a command, planning and control architecture with an open modular structure intended to allow experimentation on, or replacement of, various components. The research is seeking to determine which functions are generally required in a number of application areas and across a number of different command, planning, scheduling and control systems.

O-Plan aims to demonstrate how a planner, situated in a task assignment and plan execution (command and control) environment, and using extensive domain knowledge, can allow for flexible, distributed, collaborative, and mixed-initiative planning. The research is seeking to verify this total systems approach by studying a simplified three-level model with separable task assignment, plan generation and plan execution agents.

O-Plan has been applied to logistics tasks that require flexible response in changing situations.

The O-Plan research has achieved a clearer understanding of the components necessary in a flexible planning system, and has shown how such components can be combined in an open systems integration architecture. The work has determined improved ways in which the knowledge available from modelling an application domain can be used effectively to restrict search in a planner. An improved characterisation of a plan as a set of constraints on activity opens up many possibilities for richer distributed, cooperative and mixed-initiative planning systems in the future. The project has created a prototype implementation which has been demonstrated on a class of realistic applications.

Acknowledgements

The O-Plan project began in 1984. Since that time the following people have participated: Colin Bell, Ken Currie, Jeff Dalton, Roberto Desimone, Brian Drabble, Mark Drummond, Anja Haman, Ken Johnson, Richard Kirby, Arthur Seaton, Judith Secker, Glen Reece, Austin Tate and Richard Tobin.

Prior to 1984, work on Interplan (1972-4) and Nonlin (1975-6) was funded by the UK Science and Engineering Research Council.

From 1984 to 1988, the O-Plan project was funded by the Science and Engineering Research Council on grant numbers GR/C/59178 and GR/D/58987 (UK Alvey Programme project number IKBS/151). The work was also supported by a fellowship from SD-Scicon for Austin Tate from 1984 to 1985.

From 1989 to 1992, research on scheduling applications of the O-Plan architecture was funded by Hitachi Europe Ltd. A number of other research and development contracts placed with AIAI have led to research progress on O-Plan.

From 1989 to 1995, the O-Plan project has been supported by the US Air Force Rome Laboratory through the Air Force Office of Scientific Research (AFOSR) and their European Office of Aerospace Research and Development by contract numbers F49620-89-C0081 (EOARD/88-0044) and F49620-92-C-0042 (EOARD/92-0001) monitored by Northrup Fowler III at the USAF Rome Laboratory. The projects have been part of the ARPA/Rome Laboratory Planning Initiative (ARPI).

Additional resources for the O-Plan projects have been provided by the Artificial Intelligence Applications Institute through the EUROPA (Edinburgh University Research on Planning Architectures) Institute development project.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

This research was jointly sponsored by the Advanced Research Projects Agency of the Department of Defense and by the U.S. Air Force's Rome Laboratory, and was monitored by Northrup Fowler III, RL (C3C), 525 Brooks Rd., Griffiss AFB, NY 13441-4505, USA.

The United States Government is authorised to reproduce and distribute reprints of this paper and each of the appendices for government purposes notwithstanding any copyright notation hereon.

Abbreviations

The following abbreviations are used within the report. This section serves as a reminder of their meaning wherever the context is not clear.

ADS Associated Data Structure – the level of data structure in O-Plan at which a plan is represented. This is "associated" with an underlying Time Point Network (TPN).

AM O-Plan Agenda Manager – one of the main processes of the O-Plan system and the main part of the "Controller" which decides on what can be processed next in an O-Plan agent.

ARPA Advanced Research Projects Agency – earlier called DARPA, the Defense Advanced Research Projects Agency.

ARPI ARPA/Rome Laboratory Planning Initiative – the Knowledge-based Planning and Scheduling Initiative research and development programme.

CPE Common Prototyping Environment – a shared framework of tools and domain information used within the ARPI.

COA Course of Action – military terminology for a particular plan option for soem given task and assuming certain constraints.

DM O-Plan Database Manager – one of the main processes of the O-Plan system which manages the plan state and gives access to it on behalf of other modules.

GOP Graph Operation Processor – a support routine in O-Plan used to manipulate information in graphs or networks, e.g., in the Time Point Network (TPN).

GOST Goal Structure Table – used to hold conditions associated with a plan and their method of satisfaction.

IFD Integrated Feasibility Demonstrator – used to demonstrate ARPI technologies on military relevant problems.

IM O-Plan Interface Manager – one of the main processes of the O-Plan system which manages inter-module, inter-agent and user communications.

<I-N-OVA> Issues, Nodes, Orderings, Variables, Auxiliary Constraints Model – used to represent constraints on activity or plans.

KP O-Plan Knowledge Source Platform – one of the main processes of the O-Plan system on which Knowledge Sources can be run.

KS Knowledge Source – a computational capability in O-Plan.

KSF Knowledge Source Framework – a proposed language for describing an agent's capabilities (it's Knowledge Sources).

MTC Modal Truth Criterion – another name adopted by other researchers for a process similar to Question Answering (QA).

NEO Non-combatant Evacuation Operations – military operations to evacuate civilians from a danger zone.

PMO Plan Modification Operator – a term used to describe the abstract operation of O-Plan in which partially-specified plans are modified by "Operators" during the search for a solution to a given task. PMOs correspond to Knowledge Sources in O-Plan.

PSV Plan State Variable – an object in a plan which is not fully defined.

PSVM Plan State Variables Manager – the Constraint Manager in O-Plan which looks after Plan State Variables (PSVs).

PRECis Planning, Reactive Execution and Constraint Satisfaction domain – an experimental application domain to allow demonstration and evaluation of systems for planning, scheduling, constraint satisfaction and reactive plan execution. This domain involved NEOs from the fictional island of Pacifica.

QA Question Answering – the O-Plan support routine

which finds the ways in which a plan condition can be satisfied.

REA Reactive Execution Agent – an agent designed to support the execution of plans where reaction to changing circumstances is required.

RUE Resource Utilisation Entry – the form of constraint information looked after by the Resource Utilisation Manager (RUM).

RUM Resource Utilisation Manager – a constraint manager which looks after resource constraint information.

TIE Technology Integration Experiment – an experiment to join together two or more technologies from the ARPI to evaluate some given objective.

TF Task Formalism – the domain description language for the O-Plan planner.

TGM TOME/GOST Manager – the Constraint Manager in O-Plan which looks after effects and conditions.

TOME Table Of Multiple Effects – used to hold effects associated with a plan.

TPN Time Point Network – used to hold time points associated with a plan and constraints between these time points.

TPNM Time Point Network Manager – the Constraint Manager in O-Plan which builds and looks after the TPN.

Contents

List of Figures

1 Summary

The O-Plan research and development project is seeking to identify re-usable modules and interfaces within planning systems which will enable such systems to be tailored or extended quickly to meet new requirements. A common framework for representing and reasoning about plans based on the manipulation of constraints underlies the model used by the architecture. Within this framework, rich models of an application domain can be provided to inform the planner when creating or adapting plans for actual use.

A number of important foundations have been laid for flexible planning work in the future. They are:

- A view of the planner as *situated* in the context of task assignment, plan execution and change.
- A simple abstract architecture based on an agenda of "issues" from which items can be selected for processing. The processing takes place on an available computational platform (human or machine), with the appropriate functional capabilities described as knowledge sources.

  This architecture allows for independent progress to be made in a number of important areas for successful planning systems, including search control and opportunism, planner capability description, and system resource scheduling.
- A structure that allows separate (often specialised) handlers for different types of constraint to be included, so that the results become effective overall constraints on the operation of a planner.
- Ways to use domain knowledge, where possible, to constrain the search of a planner.
- The common model of activity, tasks and plans based on a set of constraints – the <I-N-OVA> constraint model.

  A common model can in turn support systems integration and open up collaboration and distribution opportunities.
- Symmetric interaction by system components and users. Both are seen as manipulating the same set of constraints.
- An approach to the user interface of a planner, based on Plan and World Views.

The O-Plan planner is general purpose and applies to a wide variety of important application areas. Its current application to military logistics planning tasks is described.

A number of publications resulting from the O-Plan project are attached as appendices. These have been chosen to give more details of the principal contributions of the work. The attached papers are described at the start of the appendices, and their relationship to work described in this paper will be highlighted throughout the text by "[**See Appendix ...**]".

2 O-Plan – the 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 the specification, generation and execution of activity plans, and for interaction with such 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 [2] for background reading on AI planning systems. See [9] 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. This multi-agent approach is described in greater detail in [42] **[See Appendix A]**. The benefits of viewing the planner as situated in a command and control framework are described in [18] **[See Appendix C]**.

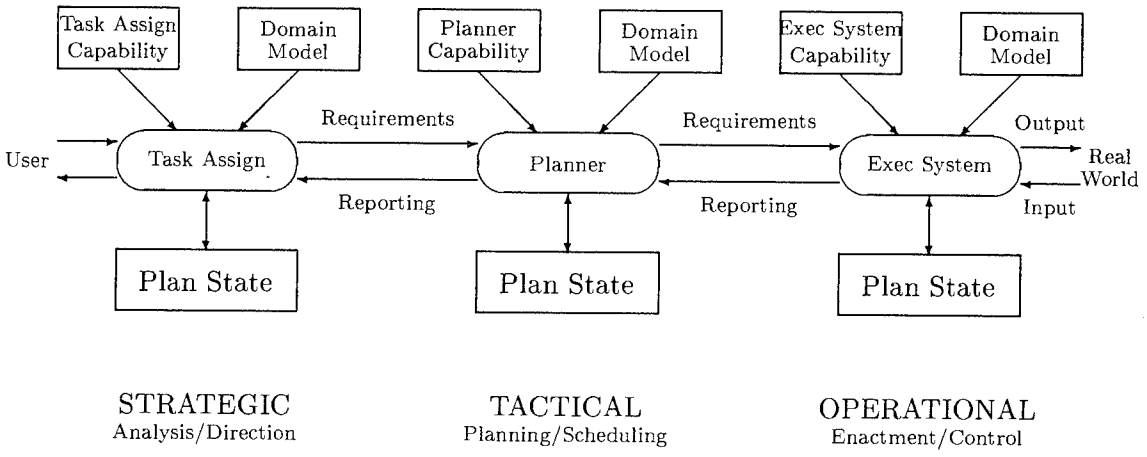


Figure 1: Communication between Strategic, Tactical and Operational Agents

Figure 1 shows the communications between the three agents in the O-Plan architecture[1]. A user specifies a task that is to be performed through some suitable interface. We call this process *task assignment.* A *planner* constructs a plan that would 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 activities of the three agents may be more or less concurrent.

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

- successive refinement/repair of a complete plan or schedule which contains an agenda of

[1]This simplified view of the environment within which a planner operates helps to clarify the O-Plan research objectives. It is sufficient to ensure that the tasking and execution environments are represented.

outstanding issues;

- a least commitment approach;
- opportunistic selection of the focus of attention on each problem solving cycle;
- incremental tightening of constraints on the plan, performed by "constraint managers", e.g.,
  - time point network manager,
  - object/variable manager,
  - effect/condition manager,
  - resource utilisation manager;
- localised search to explore alternatives where advisable;
- 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. The background to this work is provided in [33] **[See Appendix B]**. 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.

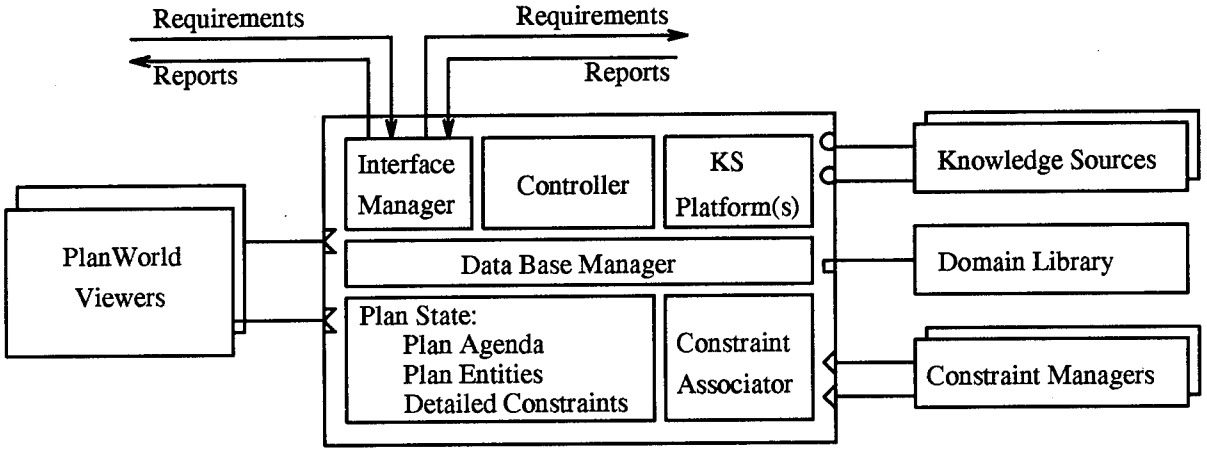


Figure 2: O-Plan Agent Architecture

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 including a library of possible actions.

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

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 Manager – 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 and provides services to the Interface Manager, Controller and Knowledge Sources.

Constraint Associator Acts as a mediator between changes to the Plan State made by the Data Base Manager and the activities of the various Constraint Managers that are installed in the agent. It eases the management of interrelationships between the main plan entities and detailed constraints.

3 A Situated Planner – Coordinating Task Assignment, Planning and Plan Execution

The O-Plan project has identified the need for AI planners to be viewed as situated agents, where planning is one of a number of tasks involved in dealing with the whole problem of task assignment, planning, execution and control. While the planner deals with the plan generation aspect of the problem, other agents may deal with task elicitation, plan analysis, reactive execution, plan repair, etc. Each of these systems has its own perspective on the planning problem and each is capable of communicating in a way which allows other systems to assimilate new information into their perspective of the problem. Within such a collection of agents, a situated planner takes task assignments from a superior agent and creates a plan or further elaborates it before passing it to the subordinate execution support agents for further processing or enactment.

In many domains such as manufacturing, construction assembly, logistics, spacecraft control, etc. the planner needs to deal with changes occurring in two very different ways:

1. **Change of Task and Requirements:**
   The task set to the planner may change or be refined as the plan is being generated, requiring the planner to:
   - alter its focus, e.g., `plan to move the 82nd airborne now rather than later`,
   - choose alternative methods, e.g., `move the 82nd airborne by sea rather than air`,
   - abandon the task altogether, e.g., `abandon the deployment task and return the 82nd airborne to their home base`.
2. **Change in the Environment:**
   Events may occur in the domain which require the plan to be repaired by the insertion of new constraints or activities. In some cases the failure may be so severe that the entire plan needs to be abandoned and an alternative found.

The reason for taking this situated view is that planners should not be considered as functioning in isolation. In addition to being able to communicate about the overall task being performed, the planner ought to be able to interact closely with the environment in which it is placed. This allows more knowledge about the tasking and execution environment to be used during planning and replanning.

3.1 Planner to Plan-Execution System

The O-Plan architecture has been designed to support the creation of situated agents, and work to date has concentrated on building generative planning agents and execution agents, with links between them. The results of this research have been used in a number of systems that have drawn on the O-Plan work. For example, the Optimum-AIV [1] system, developed

for Assembly, Integration and Verification of spacecraft at the European Space Agency, and now in use for Ariane Launcher preparations, uses concepts from O-Plan's plan representation to support the repair of plans to deal with test failures.

As part of the O-Plan research, an associated Ph.D student project explored the creation of a reactive execution agent within the O-Plan agent architecture [27]. This work also showed the value of using the plan intentions captured in Goal Structure to support effective reactive execution and re-planning [29] [**See Appendix H**].

3.2 Task Assigner to Planner

In many domains the problems of command, task setting, planning, plan analysis and plan enactment have been compartmentalised, leading to many systems having an inability to assimilate new information into existing plan options. In particular, the problem of dealing with task assignment and its link to the generative planner has been neglected by planning researchers. Future research in the O-Plan project aims to address this area and in particular the problem of allowing different situated agents to maintain their own perspective on the planning problem while at the same time allowing plans to be communicated between them. This will make it possible to communicate and use commands, plans, and tasks with improved precision, timeliness and level of detail between a number of situated agents. The O-Plan research has already addressed two key issues of the task assignment problem:

- **Plan Quality**:
  The task assigner needs to analyse the quality of the plans being generated and to provide feedback and direction concerning the options and plans which should be explored further. Joint work with USC/ISI to link O-Plan to their EXPECT system [21] has shown that plans can be generated and analysed to provide valuable feedback to human planners.
- **Role of Authority**:
  The activities of the various situated agents need to be coordinated, and authority management is viewed as one way in which this can be done [32]. For example, in plan generation, it may be necessary to be given authority to work on certain options and to have direction on the level of detail to which a plan should be developed. In plan enactment, it is important to identify (and possibly name) which phases of the plans can be executed and which parts should be held back for further approval.

These two aspects are elaborated further in the next two subsections.

3.3 Integrating Plan Quality Considerations into Planning

Current AI planners can generate a solution that satisfies the requirements they are given. Some planners provide facilities to control the quality of the solution to be returned, by using evaluation functions or search-control rules. However, they do not usually integrate plan quality considerations across several plans. In addition, their plan representations may not reflect the

plan quality criteria that are necessary in practice. Often, the quality criteria that human expert planners consider:

- are highly dependent on the situation and the scenario at hand (some criteria may be more important if there is a certain deadline, or new criteria may need to be considered if new considerations arise), and
- include complex factors and tradeoffs that are often not represented by an automatic planner.

Research on plan analysis has concentrated on addressing two issues:

- to provide a tool – EXPECT [21] – which allows human planners to define criteria for plan quality and preferences among alternative plans and options.
- to operationalise these criteria to guide a generative planner in proposing better quality plans [14],[15],[22] **[See Appendix G]**.

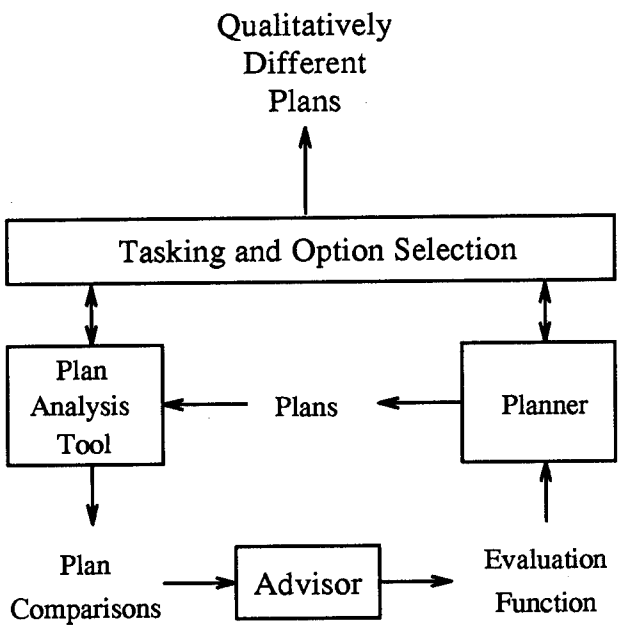


Figure 3: Combining a Planner and a Plan Analysis Tool

An approach is being investigated which combines the O-Plan planner with the EXPECT knowledge-based plan analysis system. Figure 3 describes the way in which it is proposed that O-Plan and EXPECT can be linked and the way in which plans and analysis information flows. Using these two systems, it may be possible to build an interface between the planner and the user that provides the following functionality:

- support to the user in defining criteria for evaluating plan quality through a knowledge acquisition tool,
- evaluation of the quality of plans proposed by the planner,
- justifications for judgements of plan quality,
- guidance to the planner in its search for suitable plans.

To date the O-Plan system is able to generate plans and communicate them to the EXPECT system for evaluation. Work is continuing to expand the interface between EXPECT and O-Plan to strengthen the support for users in specifying, comparing and refining the constraints on a range of different plan options, at the task assignment level of a planning support environment, and to allow this information to be used directly by O-Plan in guiding it in its search for a good solution.

3.4 The Role of Authority for a Situated Planning Agent

At the moment, the Task Assignment agent in O-Plan informs 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. It is intended that O-Plan will support authority management in a more comprehensive and principled way in future [32] [**See Appendix F**]. The O-Plan research has identified the need to support:

- Plan *options*: 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 with different search options or may contain a different task and environmental assumptions. It is possible to have only one plan option.
- Plan *phases*: individually provided actions or events stated explicitly in the top level task description given by the Task Assignment agent. More precise authority management is possible by specifying more explicit phases at the task level. It is possible to have only one phase in a task.
- Plan *levels*: specified degrees of detail to which plans can be produced. More precise authority management is possible by specifying more explicit levels in the O-Plan domain description language, Task Formalism (TF). It is possible to have only one level in a domain.

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.

It is anticipated that the Task Assignment agent of O-Plan will need to support such authority management capabilities. To establish an appropriate basis for future developments, and allow for some initial internal support for authority management to be incorporated, the current

release of O-Plan has a simple authority scheme and reports this at the head of the Task Assignment agent menu shown here:

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

This reports that the system is planning for task `Operation_Blue_Lagoon` in the domain `pacifica` and that it is currently planning within `plan option 1`. It is authorised to plan to any level of detail (infinity) for all phases (`plan all=inf`) but is not yet authorised to execute any actions (`execute all=no`).

A prototype HARDY-based[2] user interface for the Task Assignment agent has been created and connected to O-Plan.

[2]HARDY is a diagramming aid and hypermedia tool from AIAI.

4 Using Domain Knowledge in Planning

O-Plan has the ability to use domain knowledge about time constraints, resource requirements and other types of knowledge to restrict the range of plans being considered as feasible solutions to the tasks specified. The O-Plan research programme has studied a number of mechanisms for using such knowledge to prune or prioritise search. These include using temporal constraints [4],[16], resource constraints [17], temporal coherence of conditions [19], and Goal Structure condition type information [30],[31].

4.1 An Approach to Incorporating Constraint Management into a Planner

The O-Plan research has studied ways to enable specialised and efficient constraint handling methods to be used to manage the detailed constraints within a plan, constraints such as those on action ordering, action times, and resource use. The main, higher-level entities in plans (such as activities) are represented separately from constraints, in an *Associated Data Structure* (ADS). Separating the main plan entities from the detail of the lower-level constraints allows a more modular interface to be provided.

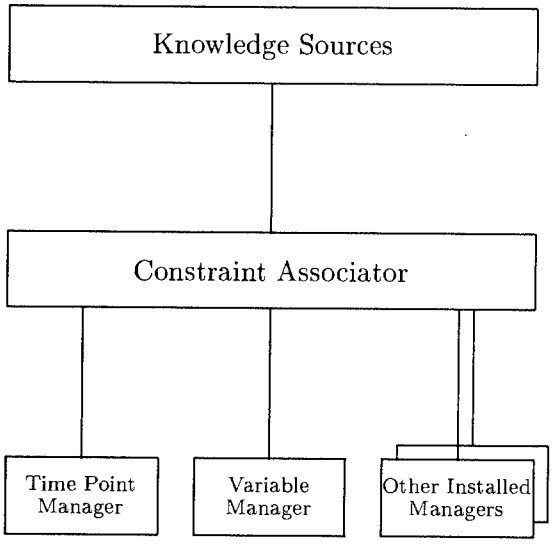


Figure 4: Associator to Mediate between Knowledge Sources and Constraint Managers

To improve the modularity of the Issue Handlers (Knowledge Sources), which must maintain detailed constraints on the main entities being manipulated, the architecture includes a *Constraint Associator* as shown in Figure 4 [35] **[See Appendix K]**. The interface to this component allows

for the handling of various types of constraint through plug-in constraint handlers[3].

Experience of writing Knowledge Sources for O-Plan and other systems has shown that it can be difficult to preserve the modularity of the code while still taking into account all aspects of detailed constraint propagation. This is particularly so for those constraints involving time and variables in a plan. Time and variables are involved in many of the manipulations performed by knowledge sources (not surprisingly in a generic temporal-activity planner). Time and variables are also often parts of other constraint descriptions for resources, Goal Structure, etc. The framework for constraint handling in O-Plan via the Constraint Associator therefore separates out the Temporal and Variable Constraint managers from any others which may be installed into O-Plan.

Being certain that there are time and variable managers installed, with a necessary minimum level of capability, allows simplifying assumptions to be made when writing Knowledge Sources in O-Plan. Knowledge Sources use the Constraint Associator to make all changes to the detailed constraints within a plan. The Constraint Associator makes appropriate calls to the detailed constraint managers installed and can cope with a range of specialised implementations of constraint handlers. All constraints can be noted, even in cases where a handler is not available[4].

The Constraint Associator maintains a uniformity of interface to the Knowledge Sources which provides some key benefits [35],[39]. The Knowledge Sources, Constraint Associator and all installed Constraint Managers may make use of a *Minimal Plan Ontology* for their communications. This provides a small set of descriptive terms about plan features and entities which may be used for communication between the components. This minimal plan ontology includes the notions of time points, the "before" ordering relation on time points, variables, and equality and inequality relations on variables. Constraints Manager responses can be:

1. **yes**, the constraint added or changes made to the constraint are valid and the changes are now under management;
2. **no**, the change could not be made to the constraint given the current constraints under management; or
3. **maybe**, if certain changes (e.g., addition of ordering links or further variable bindings) are made.

This interface is similar to the Question Answering mechanism used to establish the value of world conditions at a point in a partially-ordered plan representation [31]. That was itself a basis for the formalisation of the Modal Truth Criterion (MTC) by Chapman [7]. Such a Truth Criterion is at the heart of many current planning systems. Therefore, a common style of interface to establish the validity of constraints at points in a plan is being maintained by the O-Plan Constraint Associator in cooperation with the Constraint Managers installed into it.

The Constraint Associator can also identify potential *cross-constraint* relationships and deal with them autonomously without the Knowledge Source writer having to handle possible knock-

[3]The current implementation does not yet make full use of this simplifying framework.

[4]O-Plan already has the concept of an *Other Constraint* type in which notes of further constraints can be kept in a plan.

on effects. For example, this means that a change to time constraints which may affect the current resource constraints will be identified and passed on to a resource constraint manager if one is installed. The Constraint Associator is also designed such that it can combine the results of a number of constraint manager calls and can return a single more tightly constrained set of changes to the plan state if necessary.

By using this approach to incorporating constraint management into a planner, it is possible to plug in diversified and specialised constraint handlers suited to their specific purposes. For example, a specialised spatial constraint manager using 3-D reasoning methods could be incorporated without major changes to the system design.

The following sections describe the main constraints employed by O-Plan and the managers responsible for them.

4.2 Time Constraints

O-Plan supports relative and metric time constraints for time points in actions, tasks and plans. The O-Plan constraint manager responsible for such constraints uses a Time Point Network (TPN) to support its operation and hence is called the Time Point Net Manager (TPNM) [16]. Each time point is constrained by the network to have an upper and lower bound on its temporal distance from other points in the network and from time zero.

The time points held in the TPN are indirectly linked to actions and events in a plan: the Associated Data Structure (ADS). This ensures that the TPN and the ADS can be independently changed. Moreover, the functional interface to the TPN does not reveal the underlying representation, so that a different way of handling time constraints could be substituted.

In addition to its use in the O-Plan activity-orientated planner, the current TPNM has been applied to large resource-allocation scheduling problems in the TOSCA scheduler [3], where the number of time points was in excess of 5000 and the number of temporal constraints exceeded 3000. The TOSCA scheduler was itself based on the O-Plan architecture, making use of a different Associated Data Structure based on resource reservation periods, rather than actions as in the planner.

4.3 Object/Variable Constraints

During the planning process a number of objects, and variables representing objects, can be introduced into a plan. O-Plan uses a rich model of constraints to handle the interactions and dependencies among the different objects and variables, including co-designation (equality), non-codesignation (inequality), scalar (set membership), and numeric range constraints.

Plan State Variables (PSVs) are created by the planner as necessary when the plan refers to an object that has not yet been identified. The Plan State Variables Manager (PSVM) is the O-Plan Constraint Manager responsible for maintaining the network of plan state variable constraints introduced into the plan.

When a PSV is created, it has stored with it a list of constraints. As more of the plan is

developed, further constraints may be added. Dependencies can arise between different plan state variables and these are of two forms:

- **Same**: the variables must have the same value. (It follows from this that the constraints on the variables can be conjoined.)
- **Not-same**: the variables cannot have the same value.

In addition, each plan state variable has:

- A **type**: the set of domain objects from which the variable's value must be chosen.
- Value **constraints**: conditions the variable's value must satisfy, e.g., that its `size` be `large` and its `colour` `green`.

For example, plan state variable ?v1 may be constrained to be of type `movable-object`, `green` and `large`, not the same as ?v2, and the same as ?v14 and ?v8.

As with other O-Plan Constraint Managers, the responses that the PSVM can give are:

1. **yes**, the change (e.g., creating a new variable with given constraints, changing a variable's constraints) is valid, and the changes are now under management;
2. **no**, the change could not be made given the current constraints under management; or
3. **maybe**, if certain changes (e.g., addition of ordering links or further variable bindings) are made.

4.4 Resource Constraints

O-Plan uses a Resource Utilisation Manager (RUM) to manage the detailed resource constraints within a plan. The RUM can handle a number of different resource types and can reason about how resource levels change during the generation of a plan. Domain knowledge about different types of resources allows the planner:

1. to check that resource usage demands can be met from the resources available at any time;
2. to provide heuristic estimates of the quality of a plan as it is generated; and
3. to provide suggestions (if possible) on the repair of a failed plan should resource usage be the problem (reduce resource levels, produce more of the resource earlier, move actions back or forward in time, etc.)

There are two major resource types supported by the RUM: consumable resources and reusable resources. Consumable resources are ones which are consumed during the life of a plan, e.g.,

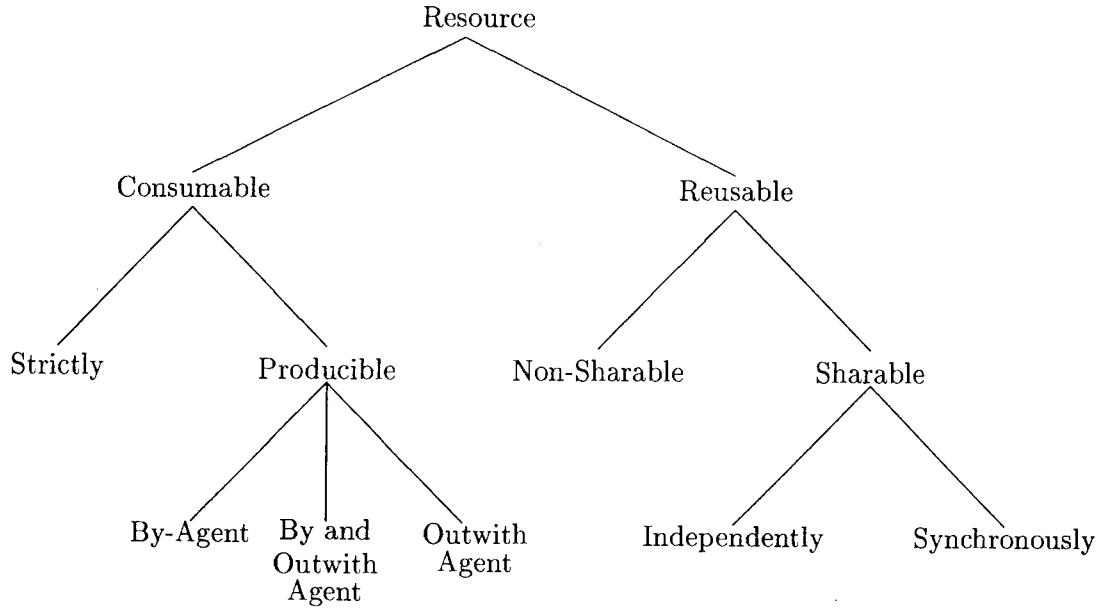


Figure 5: Example Hierarchy of Resource Types

fuel, money, ammunition, etc. Reusable resources are ones which can be allocated to a plan for it to use and then possibly be returned (de-allocated) for re-use, e.g., trucks, manpower, runways, etc. Each of these can be further subdivided as shown in Figure 5.

A design for a sophisticated resource reasoning capability has been created for O-Plan [17] [**See Appendix E**], and a subset of this is provided by the current implementation. It is the function of the RUM to check on the levels of resources being used a certain points in the plan. The RUM is informed of resources level changes from the main planning level by means of Resource Utilisation Entries (RUE's). A RUE can effect a resource in one of five different ways:

1. **Set** a resource level to be a particular value (or within a particular range), for example to top up a fuel tank to its maximum capacity.
2. **Allocate** a certain amount of resource, i.e., reduce the amount of resource remaining as available from that point within the plan.
3. **Deallocate** a certain amount of resource back to a common pool, i.e., increase the amount of resource available from that point in the plan.
4. **Consume** a certain amount of resource.
5. **Produce** a certain amount of "new" resource.

The RUM's primary function is to manage the current set of RUEs which are part of the Plan State. It must signal to the caller when there is an inconsistent set of such RUEs. This is similar

to the way that the Time Point Network Manager and other O-Plan Constraint Managers operate. The procedural interface to the RUM is the same as that of other constraint managers in that it can return any of three kinds of result:

- **yes**, the RUE can be added without any adverse impact;
- **no**, the RUE cannot be added, given the current set of resource constraints; or
- **maybe**, the RUE can be added so long as the indicated problems are handled, i.e., further temporal and/or object variable constraints are added to the plan.

This allows us to define a *Resource Criterion* [17] which is similar to the Question Answering mechanism used to establish world conditions in O-Plan.

4.5 Goal Structure and Condition Types

A lesson learned in the expert systems and knowledge-based systems field is that it is important to make maximum use of domain knowledge where it is available in order to address many real problems. One powerful means of using domain knowledge to restrict and guide search in a planner is to recognise explicit precondition types, as introduced into Interplan [30] and Nonlin [31] and subsequently used in other systems such as Deviser [43], SIPE–2 [44], and O-Plan [9],[42].

An explicit account of the *Goal Structure* or *teleology* of a plan can be kept in these systems. This records the causal relationships between actions in the plan and can show the intentions of the domain writer or planner in satisfying conditions on actions. In some circumstances, such domain knowledge can be used to prune the search of a planner. The information is provided to the planner via a planner's domain description language (e.g., Task Formalism – TF – in Nonlin and O-Plan). The domain writer takes the responsibility for a deliberate pruning of the search space or for providing preferences via condition types. This caused us to adopt the term *knowledge-based planning* to describe our work.

Nonlin and O-Plan TF extends the notion of a precondition on an action and mates it with a "process-oriented" view of action descriptions. A TF schema description specifies a method by which some higher level action can be performed (or higher level goal achieved). Each schema is thought of as provided by its own "manager". The schema introduces lower level actions under the direction of its manager and uses that manager's own resources. The schema may say that some specific sub-action is included in order to set up for some later sub-action as part of the overall task. In TF, such internally satisfied requirements in actions are specified as **supervised** conditions. The manager also relies on other (normally external) agents to perform tasks that are their own responsibilities, but affect the ability of this manager to do the task. These are given as **unsupervised** conditions. There are other conditions which the manager may wish to impose on the applicability of particular solutions (e.g., don't try this method for house building if the building is over five stories tall). These are termed **only_use_if** conditions in O-Plan.

A detailed description of the use of condition types to inform search in an AI planner is provided in [37] **[See Appendix D]**. That paper also compares the use of condition types in O-Plan with a number of other planners.

5 <I-N-OVA> – Manipulating Plans as a Set of Constraints

The <I-N-OVA>[5] (*Issues – Nodes – Orderings/Variables/Auxiliary*) Model is a way to represent plans as a set of constraints [38] [**See Appendix J**]. By having a clear description of the different components within a plan, the model allows plans to be manipulated and used separately from the environments in which they are generated.

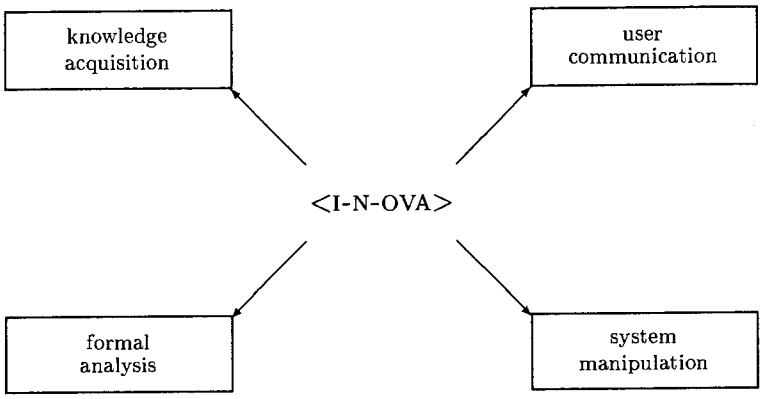


Figure 6: <I-N-OVA> Supports a Number of Requirements

As shown in figure 6, the <I-N-OVA> constraint model underlying plans is intended to support a number of different uses of plan representations:

- automatic manipulation of plans and to act as an ontology to underpin such use.
- human communication about plans.
- principled and reliable acquisition of plan information.
- formal reasoning about plans.

These cover both formal and practical requirements and encompass the needs of both human and computer-based planning systems.

Our aim is to characterise the plan representation used within O-Plan and to more closely relate this work to emerging formal analyses of plans and planning. This synergy of practical and formal approaches can stretch the formal methods to cover realistic plan representations, as needed for real problem solving, and can improve the analysis that is possible for production planning systems.

A plan is represented as a set of constraints which together limit the behaviour that is desired when the plan is executed. Work on O-Plan and other practical planners has identified different

[5] <I-N-OVA> is pronounced as in "Innovate".

entities in the plan which are conveniently grouped into three types of constraint. The set of constraints describes the possible plan elaborations that can be reached or generated as shown in figure 7.

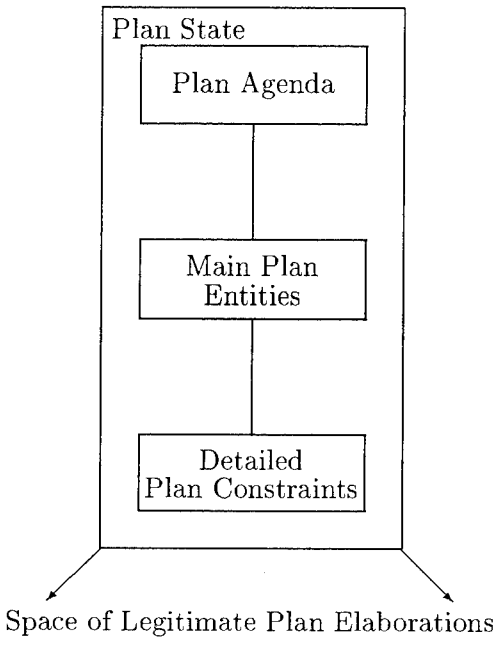


Figure 7: Various Plan Constraints Define a Space of Plan Elaborations

The three types of constraint in a plan are:

1. Plan Agenda – a set of "Issues" that must be addressed and thus define a set of "implied constraints" on legitimate future states. The agenda implies the pending or future constraints that will be added to the plan as a result of handling unsatisfied requirements, dealing with aspects of plan analysis and critiquing, etc. The agenda is a "to-do" list which can be used to decide what plan modifications should be made to a plan by a planner (user or system).

2. Main Plan Entities or Plan Node Constraints – the main plan entities related to external communication of a plan. They describe a set of external names associated to time points. In an activity planner, the nodes are usually the actions in the plan associated with their begin and end time points. In a resource-centred scheduler, nodes may be the resource reservations made against the available resources with a begin and end time point for the reservation period.

3. Detailed Plan Constraints – specialised constraints on the plan associated with the Main Plan Entities. Work on the O-Plan planner has identified the desirability of distinguishing two special types of detailed constraint and categorising all others as "auxiliary":

- Ordering or Temporal Constraints (such as temporal relationships between the nodes or metric time properties).
- Variable Constraints (co-designation and non-co-designation constraints on plan objects in particular).

Auxiliary Constraints are other detailed constraints related to input (pre-), output (post-) and protection conditions, and to resources, authority requirements, spatial constraints, etc. The auxiliary constraints are grouped into 4 categories:

- Authority Constraints
- World Condition/Effect Constraints
- Resource Constraints
- Other Constraints

Ordering and Variable constraints are highlighted since they may form part of other detailed constraints in a temporal reasoning domain such as occurs in planning and scheduling problems. That is, an auxiliary constraint may themselves involve ordering or variable constraints. Knowing that these constraints have such "cross-associations" has been found to simplify the design of constraint handling mechanisms and to ease implementation issues [33],[35]. It has also proved to be helpful in formalising planners and their plan representations (e.g., [23],[24]).

Auxiliary Constraints may be expressed as occurring at a time point ("point constraints") or across a range of points ("range constraints"). Point constraints can be used to express input and output constraints on nodes and other constraints that can be expressed at a single time point. Range constraints relate to two or more time points and can be used to express protection intervals, etc.

There is a deliberate and direct mapping of the model of plans and activity used within O-Plan and the <I-N-OVA> Constraint Model of Plans to existing structured analysis and diagramming methods such as IDEF, R-Charts, etc. [34] **[See Appendix I]**. Other researchers have also recognised the value of merging AI representation concepts with structured analysis and diagramming techniques for systems requirements modelling (e.g., [5],[25]).

6 Abstract View of the O-Plan Control Flow

The O-Plan research described in previous sections has allowed us to simplify previous descriptions of the O-Plan architecture, and in particular to present a simplified abstraction of the core working of an O-Plan agent.

O-Plan operates on a workflow principle, being driven by an agenda of "issues". A simple abstraction of this is shown in (figure 8).

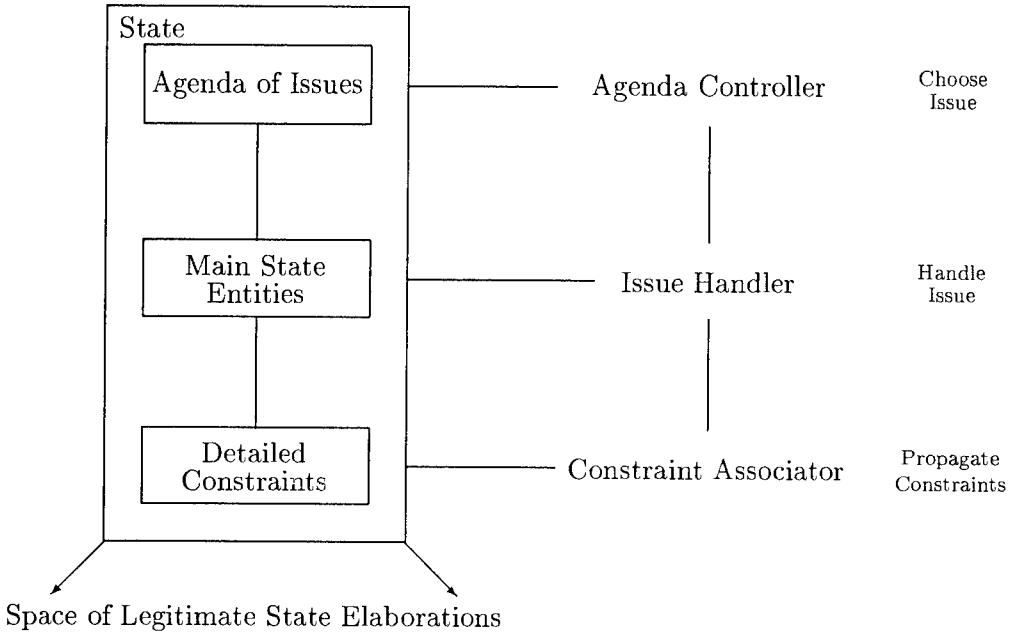


Figure 8: Framework of Components in the O-Plan Agenda-based System

O-Plan refines a "current state". It maintains one or more *options* within the state in which the previous alternative decisions that can be taken restrict the space of state elaborations which can be reached from that point[6]. The system needs to know what outstanding processing requirements exist in the state (shown in figure 8 as the *Agenda of Issues*). These represent the implied constraints on valid future states. One (normally) of these outstanding processing requirements is chosen to be worked upon next (by the *Agenda or Option Controller*). This calls up processing capabilities (or *Issue Handlers*) within the system which can make decisions and modify the State. The modifications can be in terms of definite changes to entities in the state or by noting further processing requirements (as a result of state analysis and critiquing, etc).

We have found it to be useful to separate the entities representing the decisions already made during processing into a high level representing the *Main State Entities* shared across all system

[6]State constraint relaxation may also be possible to increase the space of state elaborations in some cases.

components and known to various parts of the system, and more *Detailed Constraints* which form specialised areas of the representation of the state. These lower level more compartmentalised parts can represent specialised constraints within the state such as time, resource, spatial and other constraints. This separation can assist in the identification of modularity within the system.

7 Working with the User

An interface to AutoCAD has been built to show the type of User Interface we envisage. This is called the PlanWorld Viewer Interface [40] [**See Appendix L**]. Figure 9 shows an example screen using this interface. The window in the top left corner shows the Task Assignment menu and supports the management of authority [32] to plan and execute plans for a given task. The lower window shows a *Plan View* (showing the plan as a graph or as gantt charts), and the upper right window shows a *World View* for visualisation or 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 and the interfaces between these and the planner uses a defined interface to which various implementations can conform. O-Plan has been interfaced to a number of Plan and World Viewers including PostScript pre-viewers for plan networks, process modelling tools, map-based interfaces and tools to create animation sequences of possible plan execution. The developer interface to O-Plan is not shown to the normal user.

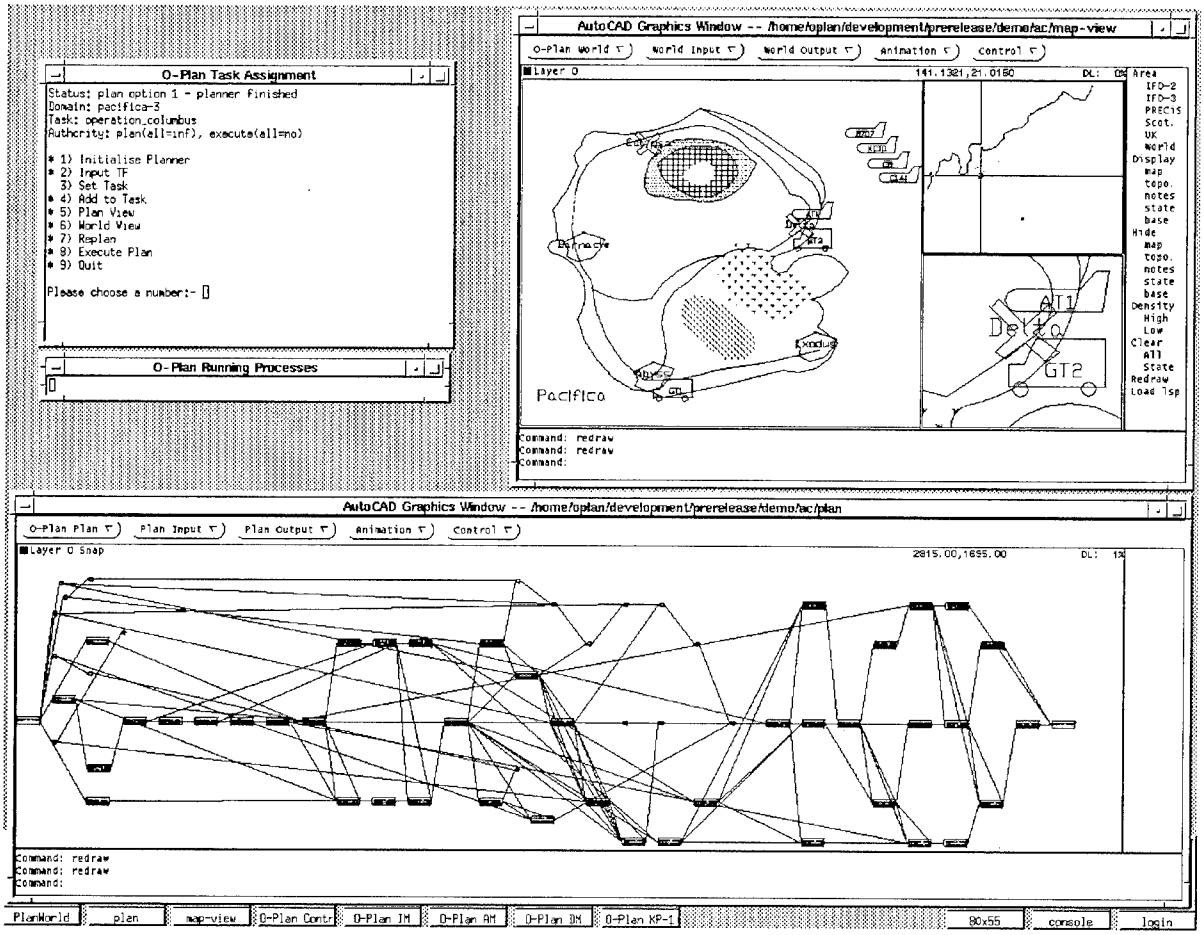


Figure 9: Example Output of the PlanWorld Viewer User Interface

Recent work on the O-Plan user interface has focussed on the representation and management of constraints in planning, particularly in order to simplify some aspects of the user's role in

the architecture and to act as a mechanism for user/system mixed initiative planning [36] [**See Appendix M**].

8 Logistics Applications

8.1 Target Applications for O-Plan

O-Plan 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 platforms in the future.

O-Plan is intended 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.

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

The architecture itself has wider applicability. For example, it has been used as the basis for the design of the TOSCA manufacturing scheduler in a project for Hitachi [3].

8.2 Crisis Action Planning

The application emphasis of the O-Plan project has been to aid in the definition, generation and support of the military crisis action planning process. There are six phases identified in reponding to a crisis as shown in figure 10.

Phase 1	Situation Development
Phase 2	Crisis Assessment
Phase 3	COA Development: O-Plan provides support in the development of COAs and in estimating the feasibility of the generated COAs. This is the main contribution of the project.
Phase 4	COA Selection: O-Plan provides support in the refinement and presentation of COAs.
Phase 5	Execution Planning
Phase 6	Execution

Figure 10: Crisis Action Planning Phases

The O-Plan research principally addresses phases three through six. AIAI has also worked with a number of groups on representations of plans which can be used to communicate across the different phases and agents involved, across the whole of the crisis planning process.

The requirements for task statement and plan generation by O-Plan in crisis action planning have been tested in the PRECiS domain [28] **[See Appendix N]** and in a simplified version of an Integrated Feasibility Demonstration scenario (IFD-2) from the ARPA/Rome Laboratory Planning Initiative [20]. These test domains allow for realistic and military relevant scenarios and issues to be addressed in a setting suitable for research and development.

Crisis action planning calls for plans that are flexible, robust and responsive to changing task requirements and to changes in the operational situation. Current planning aids are too inflexible. The aim of the O-Plan project is to show how a planner using extensive domain knowledge and situated in a task assignment (command) and execution (control) environment can allow for better flexible, distributed, collaborative and mixed initiative planning.

Current military planning systems usually allow only one COA to be fully thought through, and any alternatives are seen as poor relations. This is due to the fixed-step nature of the process, which is not currently viewed as an iterative process in which several sources of knowledge and techniques (e.g., planning, scheduling, tasking, resourcing, repairing) can be brought in as and when required. A more flexible planning framework may allow military planners to be freed from a step-by-step approach and to consider more options and constraints where appropriate within the planning process.

8.3 The PRECiS/Pacifica Domain

The principal development of O-Plan has been motivated by applications related to logistics, transportation planning/scheduling problems and Non-combatant Evacuation Operations (NEOs). The testbed provided by the PRECiS (Planning, Reactive Execution and Constraint Satisfaction) environment defines the data and hypothetical background for logistics planning and reacting scenarios and has been used for demonstration and evaluation purposes within the project.

The definition of the PRECiS environment has drawn on work by several people: Brown at Mitre Corporation to describe a realistic NEO scenario for the Planning Initiative's Integrated Feasibility Demonstration (IFD-3) [20]; Reece and Tate to define an openly accessible fictional environment based on the island of Pacifica [26], suitable for enabling technology researchers interested in planning and reactive execution of plans; and Hoffman and Burnard at ISX Corporation to produce a cut-down demonstration scenario suitable for transportation scheduling research experiments within the ARPA/Rome Laboratory Planning and Scheduling Initiative. The results have been provided in a publicly available document [28] and in other materials.

Four primary needs of the ARPA/Rome Laboratory Planning and Scheduling Initiative are met by the PRECiS environment:

1. Realistic scenarios can be explored from the data provided in the environment for COA generation, case based reasoning, transportation scheduling and the reactive execution of plans.

2. Requirements of "tier-1" enabling researchers are sufficiently met by the data in order for them to pursue their individual research programmes.

3. Entities in the environment are hypothetical and do not reflect actual peoples and locations, yet are realistic in the types of data that would normally be available.

4. The scenario and domain descriptions are not confidential or military critical. They can be openly demonstrated and publications can be based upon them. This is important for enabling researchers.

Work on the PRECIS environment and the Pacifica island model has continued. Map viewers and simulators are now available for demonstration and evaluation purposes.

8.4 Demonstration of O-Plan Generating Courses of Action for NEOs

The aim of this section is to describe the evaluation experiments conducted as part of the O-Plan project and to show how each one can be related to the categorisation of experiment types defined in the ARPI Evaluation Handbook [8]. The main demonstration domain has been Non-combatant Evacuation Operation (NEO) planning in the PRECIS environment.

A number of planned demonstrations were conducted at the end of each year of the project. In addition to these planned demonstrations a further demonstration was conducted to link the O-Plan system with a plan analysis tool (USC/ISI's EXPECT system) to show the value of being able to analyse plans and to provide feedback to the planner on how to improve the quality of the plans being generated.

The following items describe some of the experiments carried out in each of main categories of the ARPI handbook: *Programmatic*, *Demonstration* and *Scientific*. Each section describes the aims of the experiment, an overview of the approach and the results obtained. A full list of the experiments and their results can be found in [13] [**See Appendix O**].

Year 1 – 1993: Generation of Plans from the IFD-2 Scenario

The was a demonstration experiment using the IFD-2 SOCAP Tunisian scenario run on SIPE-2 [6]. From the start of the experiment it was recognised that SIPE-2 was a more developed system than O-Plan and as such this could only be an approximation to IFD-2. However, using the Task Formalism (TF) (O-Plan's domain input language) then supported within O-Plan Version 2.1 it was possible to encode the SOCAP domain and to identify a number of shortcomings in O-Plan TF [10, 11]. The schema library for this domain contained 63 schemas which defined alternative missions, deployment and employment plans, sea and airlift resources, etc. The Courses of Action (COAs) generated contained an average of 150 actions and were developed in approximately 40 seconds. O-Plan was able to generate plans in the SOCAP domain for two tasks:

- **Task 1**: "Deter three threats"
  The task requires a plan to deter one army, one air force and one navy threat by specified dates. The threats are forces which have crossed the protected border.

- **Task 2**: "Deter three threats and counter a further nine"
  The task requires a plan to deter the same three threats as well as countering a further nine threats: three army, navy and air force respectively. These nine forces are threatening to cross the border but have not yet done so.

The outcome of the experiment was that we identified a problem of incorrect constraint posting and as a result very large search spaces were being generated. The reason for the large search spaces was the combinatorics of the domain mainly arising from the codesignation of cross constraints involving time and resources. The research on the O-Plan architecture had identified the need for improved handling of these types of constraint but it was not reflected in the implementation. Similar problems were found in the earlier Nonlin system in a domain investigating the problem of Replenishment At Sea (RAS) [41]. In the RAS problem ships need to be moved from one battle group to another while others are sent for replenishment. Again the problem was one of selecting a particular ship too early rather than waiting until further constraints could be identified and posted.

Off line analysis of the problem showed that the problem could be solved with little or no search being involved. For example, many of the forces which could be chosen for a particular mission were similar and consequently the planner should have left the decision over which force to use until it was forced upon it, i.e., developing the force's employment plan.

Year 2 – 1994: Use of a Rich Resource Model in an Activity Planner Framework

This was a demonstration experiment and was designed to show the ways in which a rich model of domain resources, e.g., trucks, aeroplanes, fuel, runways, etc, could be encoded and used within an activity planner. As part of the preparation for the demonstration a study was carried out into the different types of resources present in planning domains and into previous planning approaches to resource reasoning [12]. The results of this study were twofold.

1. It became possible to identify the type of resource reasoning support which should be possible with an activity planning framework.
2. It resulted in the design of a flexible Resource Utilisation Manager (RUM) for use in an activity planner such as O-Plan and SIPE-2.

The support provided by the new RUM design would allow a range of resources types to be represented and manipulated and went beyond those types supported to date in other systems. The demonstration successfully showed that plans could be generated for a number of different resource constrained tasks specified in the PRECIS domain. The schema library for this domain contained 20 schemas which defined alternative evacuation methods, e.g., trucks or helicopters, fuel supplies, transport aircraft, etc. The COAs generated contained an average of 30 actions and were developed in approximately 40 seconds.

A number of techniques were explored and validated which showed how resources could be defined and manipulated using a range of methods. These methods made explicit use of O-Plan's Resource Utilisation Manager to track consumable resources and O-Plan's TOME and

GOST Manager to track reusable/sharable resources. Whilst these method allow the same breadth of coverage as was expected with the new RUM design, they do not have the same level of flexibility and support. In tasks where the resources were limited, e.g., small amounts of diesel fuel, the system was able to use knowledge of resources to rule out certain options as being impossible. In tasks where the choices were more extensive, e.g., use several transport types with no temporal restrictions, the system was still able to find a solution in an acceptable period of time.

Year 3 – 1995: Coordinated Command, Planning and Control

This was a demonstration experiment which showed O-Plan solving a number of tasks from a command, planning and control scenario. The aims of the demonstration were to show:

- O-Plan reacting to changes in the environment and identifying those parts of the plan which were now threatened by these changes.
- O-Plan reacting to changes in the overall task by integrating new plan requirements into the plan.

In both these cases the changes were to be made to an ongoing and executing plan.

The types of changes explored in this demonstration include failures of trucks due to blown engines and tyres and the inclusion of new objectives, e.g., pick up an extra group of evacuees. The PRECIS domain used for the demonstration has been deliberately simplified to allow a number of different aspects to be explored while keeping the plan to a manageable size. This is for viewing purposes only so that the user could follow what was happening in the demonstration. However, while being a simplification, the types of problem encountered and the solutions proposed by the planner are of relevance to military crisis action planning. Larger and more complex plans are available in other Pacifica domains. The schema library for this domain contained 12 schemas which defined alternative evacuation methods, e.g., trucks or helicopters, fuel supplies, transport aircraft, etc. The COAs generated contained an average of 20 actions and were developed in approximately 40-60 seconds. 4 different repair plans were used in the demonstration and they were as follows:

- To repair a blown engine on a ground transport.
  - The engine can only be fixed by a repair crew which is dispatched from Delta with a tow truck. The transport is then towed to Delta for repairs. The evacuees remain with truck while it is being towed.
  - The failure of the transport occurs in a time critical situation and there is insufficient time to tow the broken transport to Delta. The evacuees are moved from the broken ground transport by helicopter to Delta and the transport is abandoned.
  - This is similar to the previous repair plan except that the evacuees are moved by another ground transport instead of by helicopter.

- To repair a blown tyre on a ground transport
  - The driver of the ground transport can fix the tyre by the side of the road. The effect of the repair action is to delay the ground transport by a fixed amount of time.

Closely allied to the third year O-Plan demonstration, an associated Ph.D student project by Glen Reece showed the link between a proactive planner and a more comprehensive reactive execution agent [27] based on the O-Plan architecture. This agent has been used to reactively modify plans in response to operational demands in a simulation of the Pacifica island in the context of a NEO.

Linking of O-Plan and the EXPECT Plan Analysis Tool

This was a demonstration experiment conducted with USC/ISI in which the O-Plan system was linked with their EXPECT plan analysis tool [14],[15]. The Tunisian scenario used for IFD-2 was chosen for the evaluation domain. The schema library for this domain contained 63 schemas which defined alternative missions, deployment and employment plans, sea and airlift resources, etc. The Courses of Action (COAs) generated contained an average of 150 actions and were developed in approximately 40 seconds. The different COAs were generated using alternative mission profiles and force packages.

	COA 1	COA 2	COA 3	COA4
AIRPORTS				
- number of airports	1	1	1	2
- sorties per hour	315	315	315	480
- sq. ft. aircraft parking	2M	2M	2M	3M
SEAPORTS				
- number of seaports	1	1	1	2
- number of piers	6	6	6	15
- number of berths	6	6	6	16
- max. vessel size in ft.	600	600	600	765
- number of oil facilities	1	1	1	3
CLOSURE DATE	C + 29	C + 22	C + 23	C + 23
LOGISTICS PERSONNEL	1154	5360	5396	7362
LINES OF COMMUNICATION				
- number of locations	1	5	7	6
- max. distance in miles	20	99	140	120
- air and sea?	yes	yes	yes	yes

Figure 11: EXPECT's Evaluation of Several Alternative Plans Generated by O-Plan

EXPECT allows military planners to analyse these alternative COAs generated by O-Plan against a number of user defined domain evaluation criteria and to create an evaluation matrix for a number of chosen COAs. From the analysis, military planners are able to identify aspects of the COAs which are acceptable, e.g., low number of support personnel and those which are not, e.g., a closure date greater than 29 days. An EXPECT evaluation matrix from a series of different COAs generated by O-Plan for a logistics scenario is shown in Figure 11. This information

could then be used to impose addition requirements on the planning system to provide a better quality solution.

8.5 Bringing O-Plan Technology into Productive Use

The O-Plan system has been developed to be as modular as possible, with open interfaces to allow easy integration with the work of other ARPA/Rome Laboratory Planning Initiative participants. This has led to discussions with several groups within the Initiative and an exchange of ideas and research results. We see one of our contributions within the Initiative as providing a common framework in which the specialised contributions of different groups can be explored. We have also passed results to other ARPA programs under the ARPA Knowledge Sharing effort, e.g., to define the context handling facilities in the LOOM system. The widespread publication of the results of the project is the main way in which the project seeks to disseminate its results to the technical community.

The transition path to eventual productive use for O-Plan, and the concepts in O-Plan, is through a series of releases to the Common Prototyping Environment (CPE) of the ARPA/Rome Laboratory Planning Initiative [20] and through involvement in Technology Integration Experiments (TIEs) with other participants [6]. The transition should then involve the integration of aspects of the demonstrated technology into Integrated Feasibility Demonstrations (IFDs) within the Initiative.

O-Plan has been released in three annual versions to the ARPI CPE and through that has been made available to a number of sites. The latest version at the end of the project is O-Plan release 2.3.

A Technology Integration Experiment (TIE) has been conducted with USC/ISI to look at linking plan generation and plan evaluation (described in section 3.3). This is central to the O-Plan project's aim of situating planning in a task assignment and execution setting. It is a topic that is vital for successful military crisis planning and response. Extension of this joint work is now proposed.

The O-Plan project has also begun discussions to establish ways in which O-Plan could be incorporated into an Integrated Feasibility Demonstration (IFD) for air campaign planning within the ARPI.

9 Conclusions

The O-Plan research has achieved a clearer understanding of the components necessary in a flexible planning system and has shown how such components can be combined in a systems integration architecture. The work has determined improved ways to restrict search in a planner by using the knowledge available from modelling an application domain, and it has developed a better characterisation of plans as sets of activity constraints, opening up many possibilities for richer distributed, cooperative and mixed-initiative planning systems in the future. The project has created a prototype implementation and demonstrated it on a class of realistic applications.

References

[1] Aarup, M., Arentoft, M.M., Parrod, Y., Stokes. I., Vadon, H. and Stader, J. Optimum-AIV: A Knowledge-Based Planning and Scheduling System for Spacecraft AIV, in *Intelligent Scheduling* (eds. Zweben, M. and Fox, M.S.), Published by Morgan Kaufmann Inc, San Francisco, USA, 1994.

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

[3] 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.

[4] Bell, C.E. and Tate, A., Using Temporal Constraints to Restrict Search in a Planner, Paper presented to the Third UK Planning SIG Workshop, Sunningdale, Oxon, UK. Proceedings published by the Institution of Electrical Engineers, London, January 1985.

[5] Borgida, A., Greenspan, S. and Mylopoulos, J., Knowledge Representation as the Basis for Requirements Specifications, *IEEE Computer Magazine*, Special Issue on Requirements Engineering Environments, April 1985.

[6] Burstein, M.H., Schantz, R., Bienkowski, M.A., desJardins, M.E. and Smith, S.F., The Common Prototyping Environment – A Framework for Software Technology Integration, Evaluation and Transition, *IEEE Expert: Intelligent Systems and their Applications*, Vol. 10, No. 1, pp. 17-26, February 1995, IEEE Computer Society.

[7] Chapman, D. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333-377, 1991.

[8] Cohen, P., Dean, T., Gil, Y., Ginsberg, M. and Hoebel, L. Handbook of Evaluation for the ARPA/Rome Laboratory Planning Initiative, February, 1994.

[9] Currie, K.W. and Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence* 52(1), pp. 49-86, North-Holland, 1991.

[10] Drabble, B., Conversion of SIPE-2 Domain Descriptions to O-Plan Task Formalism, O-Plan Technical Report ARPA-RL/O-Plan/TR/1, March 1993.

[11] Drabble, B., Applying O-Plan2 to Military Logistics Planning, O-Plan Technical Report ARPA-RL/O-Plan/TR/9, September 1993.

[12] Drabble, B. *Comparison of the CAMPS system with O-Plan2: Architecture and Reasoning Capabilities*, O-Plan Technical Report ARPA-RL/O-Plan2/TR/12 Version 1, January 1995.

[13] Drabble, B. and Tate A. Applying O-Plan to the NEO Scenarios, O-Plan Technical Report ARPA-RL/O-Plan/TR/23,July 1995.

[14] Drabble, B. and Gil, Y. Acquiring Criteria for Plan Quality Control, Proceedings of the AAAI Spring Symposium workshop on Integrated Planning Applications, June 1995. Stanford Univerisity, CA, USA. Published by the American Association for Artificial Intelligence, Menlo Park, California.

[15] Drabble, B. and Gil, Y. Yes, but why is that plan better?, Proceedings of the International Conference on Artificial Intelligence in the Petroleum Industry, September 1995, Lillehammer, Norway.

[16] Drabble, B. and Kirby, R., Associating AI Planner Entities with an Underlying Time Point Network, Proceedings of the First European Workshop on Planning (EWSP-91), Springer-Verlag Lecture Notes in Artificial Intelligence No 522, 1991.

[17] Drabble, B. and Tate, A., The Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), AAAI Press, Chicago, USA, June 1994.

[18] Drabble, B. and Tate, A., O-Plan: A Situated Planning Agent, Proceedings of the Third European Workshop on Planning (EWSP'95), Assisi, Italy, September, 1995.

[19] Drummond, M. and Currie, K. Exploiting Temporal Coherence in Nonlinear Plan Construction, Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-89, Detroit, USA, 1989.

[20] Fowler, N., Cross, S.E. and Owens, C. The ARPA-Rome Knowledge-Based Planning and Scheduling Initiative, *IEEE Expert: Intelligent Systems and their Applications*, Vol. 10, No. 1, pp. 4-9, February 1995, IEEE Computer Society.

[21] Gil, Y. Knowledge Refinement in a Reflective Architecture, Proceedings of the Twelfth National Conference on Artificial Intelligence, Seattle, WA, USA. August 1994. Published by AAAI Press/The MIT Press Menlo Park, CA, USA.

[22] 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. Burstein, M.), Morgan-Kaufmann, 1994.

[23] Kambhampati, S., Design Tradeoffs in Partial Order Planning, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), Chicago, IL., USA, 1994.

[24] Kambhampati, S., Comparing Partial Order Planning and Task Reduction Planning: a Preliminary Report, Proceedings of the Workshop on Comparative Analysis of AI Planning Systems, AAAI-94, Seattle, USA, 1994.

[25] Ramesh, B. and Dhar, V., Representing and Maintaining Process Knowledge for Large-Scale Systems Development, IEEE Expert, pp. 54-59, April 1994.

[26] Reece, G.A. and Tate, A. The Pacifica NEO Scenario, Technical Paper ARPA-RL/O-Plan/TP/3, March 1993.

[27] Reece, G.A., Characterization and Design of Competent Rational Execution Agents for Use in Dynamic Environments, Ph.D Thesis, Department of Artificial Intelligence, University of Edinburgh, November 1994.

[28] Reece, G.A., Tate, A., Brown, D. and Hoffman, M., The PRECiS 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.

[29] Reece, G.A. and Tate, A., Synthesizing Protection Monitors from Causal Structure, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), pp. 146-152, AAAI Press, Chicago, USA, 1994.

[30] Tate, A., Using Goal Structure to Direct Search in a Problem Solver. Ph.D. Thesis, University of Edinburgh, 1975.

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

[32] 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, 1993.

[33] 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.

[34] Tate, A., Putting Knowledge Rich Process Representations to Use, *IOPener – The Journal of the IOPT Club for the Introduction of Process Technology*, Vol. 2 No. 3 pp 12-14, March 1994, UK Introduction of Process Technology (IOPT) Club, c/o Praxis Ltd, UK.

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

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

[37] Tate, A., Drabble, B. and Dalton, J., The Use of Condition Types to Restrict Search in an AI Planner, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, August 1994.

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

[39] Tate, A., Integrating Constraint Management into an AI Planner, *Artificial Intelligence in Engineering*, Vol. 9, No. 3, pp 221-228, 1995.

[40] Tate, A. and Drabble, B., PlanWorld Viewers, Proceedings of the 14th Workshop of the UK Planning and Scheduling Special Interest Group, Colchester, UK, November 1995.

[41] Tate, A. and Whiter, A., Multiple Resource Constraints and an Application to a Naval Planning Problem, Proceedings of the First Conference on Artificial Intelligence Applications, pp. 410-416, AAAI, Denver, Colorado, USA, December 1984.

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

[43] Vere, S. Planning in Time: Windows and Durations for Activities and Goals, *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 5, 1981.

[44] Wilkins, D. *Practical Planning*, Morgan-Kaufmann, 1988.

Appendices

Figure 12 shows how the attached papers in the appendices relate to the main themes of the O-Plan research.

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

Provides an overview of the O-Plan architecture with its task assignment, planning and execution agents, giving information about the aims of the work and its applications.

APPENDIX B: 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.

Provides an overview of the module specifications, interfaces and protocols used within the O-Plan architecture.

APPENDIX C: Drabble, B. and Tate, A., O-Plan: A Situated Planning Agent, Proceedings of the Third European Workshop on Planning (EWSP'95), Assisi, Italy, September, 1995.

This APPENDIX explains the importance of exploiting the task assignment and execution framework within which a planner is situated. The benefits of being able to use a rich model of this environment are explained.

APPENDIX D: Tate, A., Drabble, B. and Dalton, J., The Use of Condition Types to Restrict Search in an AI Planner, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, August 1994.

O-Plan can make use of domain knowledge of various kinds to restrict its search for plans. It can thus be applied to larger problems than would otherwise be the case. This paper describes one strong contribution of the O-Plan research to finding ways to encode domain knowledge in forms which can be used by a planner.

APPENDIX E: Drabble, B. and Tate, A., The Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), AAAI Press, Chicago, USA, June 1994.

Accounting for resource availability is an important requirement when planning. This paper describes the novel mechanisms used within O-Plan for managing resources using incremental algorithms. The generic interface to such constraint managers within O-Plan is also described.

APPENDIX F: 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, 1993.

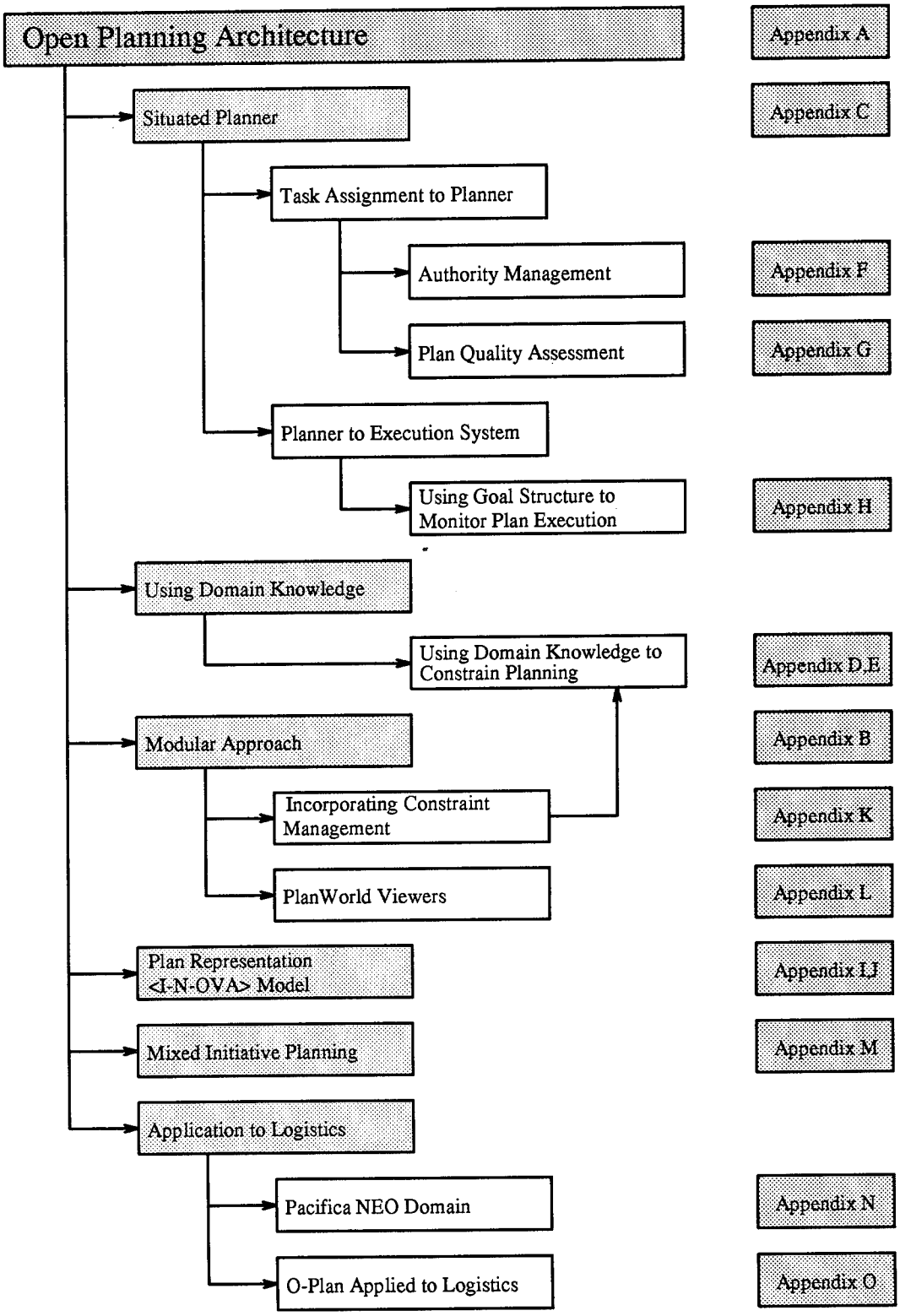


Figure 12: Road Map to Attached Papers in the the Appendices

In a cooperative planning environment, a planner should not be considered to work in isolation, simply producing plans. This paper describes early work on modelling the role of authority and delegation within a command, planning and control environment in such a way that it can be used to effectively coordinate planning activities.

APPENDIX G: 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. Burstein, M.), Morgan-Kaufmann, 1994.

Producing useful, effective plans requires improved information about the quality of plans and about the features of plans from which quality can be determined. This paper explores the general factors that may be used to analyse differences between plans and examines a specific domain of military Non-combatant Evacuation Operations (NEOs) to provide examples of domain criteria used to assess plan quality.

APPENDIX H: 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.

O-Plan produces plans that can be executed. Work in the O-Plan project on a Reactive Execution Agent was performed on a linked Ph.D. This paper describes the use for plan execution support of knowledge embedded in O-Plan plans that captures the rationale, or Goal Structure, of the plan steps. It is shown that reactive plan change support can be provided using such knowledge.

APPENDIX I: Tate, A., Putting Knowledge Rich Process Representations to Use, *IOPener – The Journal of the IOPT Club for the Introduction of Process Technology*, Vol. 2 No. 3 pp 12-14, March 1994, UK Introduction of Process Technology (IOPT) Club, c/o Praxis Ltd, UK.

The knowledge rich plan structures used within O-Plan are themselves of value in other contexts. This paper describes how the O-Plan plan model can support improved process modelling, analysis and workflow in organisations.

APPENDIX J: Tate, A. Characterising Plans as a Set of Constraints – the <I-N-OVA> Model – a Framework for Comparative Analysis, Special Issue on Evaluation of Plans, Planners, and Planning Agents, *ACM SIGART Bulletin* Vol. 6 No. 1, January 1995.

In order to promote convergence of work in software engineering, process management, AI planning and formal mathematical work on planning, a model of plans as a set of constraints on behaviour or activity is being explored by the O-Plan project. This paper describes the <I-N-OVA> constraint model employed and relates it to other work.

APPENDIX K: Tate, A., Reasoning with Constraints in O-Plan2, Extended version, containing a number of additional sections, of a paper in the Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (M.Burstein, ed.), Tucson, Arizona, USA, Morgan Kaufmann, 1994.

The O-Plan research has simplified the way in which detailed constraints within plans can be managed, and has introduced a way in which constraint managers could be plugged into a planner using a well defined protocol. This paper describes the approach adopted.

APPENDIX L: Tate, A. and Drabble, B., PlanWorld Viewers, Proceedings of the 14th Workshop of the UK Planning and Scheduling Special Interest Group, Colchester, UK, November 1995.

The user interface to O-Plan makes use of plug-in viewers which can support technical plan views (e.g., workflows and charts) or domain related world views (e.g., maps and animations). This paper describes the ways in which a user can interact with O-Plan and describes the PlanWorld Viewer interface.

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

Planning is not done as an isolated activity. It relies on cooperative work between many people and systems. This paper describes O-Plan's approach to mixed initiative and cooperative planning. It is based on the mutual process of placing constraints on behaviour.

APPENDIX N: Reece, G.A., Tate, A., Brown, D. and Hoffman, M., The PRECiS 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.

O-Plan research and the prototype implementation have been applied to specific logistics problems related to military Non-combatant Evacuation Operations (NEOs). This paper describes a non-confidential demonstration and test environment and example scenarios in which the requirements for O-Plan were established and demonstrations provided. It involves NEOs from the fictional island of Pacifica.

APPENDIX O: Drabble, B., Tate, A. and Dalton, J. Applying O-Plan to the NEO Scenarios. O-Plan Technical Report, Artificial Intelligence Applications Institute, University of Edinburgh, July 1995.

This paper describes and evaluates the application of O-Plan to the PRECiS/Pacifica NEO Scenarios.

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

Austin Tate, Brian Drabble and Richard Kirby

Appears in *Intelligent Scheduling*, (eds. Zweben. M. and Fox, M.S.), pp213-240.
Publisher Morgan Kaufmann, Palo Alto, California, 94303, USA, 1994.

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

Austin Tate, Brian Drabble and Richard Kirby

1 Introduction

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

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

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

2 History and Technical Influences

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

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

- A hierarchical planning system which can produce plans as partial orders on actions (as suggested by Sacerdoti [23]), though O-Plan is flexible concerning the order in which parts of the plan at different levels are expanded.
- An agenda-based control architecture in which each control cycle can post pending tasks during plan generation. These pending tasks are then picked up from the agenda and processed by appropriate handlers (HEARSAY-II [16] and OPM [15] uses the term *Knowledge Source* for these handlers).
- The notion of a "plan state" which is the data structure containing the emerging plan, the "flaws" remaining in it, and the information used in building the plan. This is similar to the work of McDermott [19].

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

2.1 O-Plan1

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

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

2.2 O-Plan2

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

- to consider a simple "three agent" view of the environment for the research to clarify thinking on the roles of the user(s), architecture and system. The three agents being the job assignment agent, the planning agent and the execution agent.
- to explore the thesis that communication of capabilities and information between the three agents could be in the form of *plan patches* which in their turn are in the same form as the domain information descriptions, the task description and the plan representation used within the planner and the other two agents.
- to investigate a single architecture that could support all three agent types and which could support different plan representations and agent capability descriptions to allow for work in activity planning or resource scheduling.
- to clarify the functions of components of a planning and control architecture.

- to draw on the earlier Edinburgh planning experience in O-Plan1 [5] and to improve on it especially with respect to flow of control [32].
- to provide an improved version of the O-Plan1 system suitable for use outside of Edinburgh within Common Lisp, X-Windows and UNIX.
- to provide a design suited to use on parallel processing systems in future.

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

3 Characterisation of O-Plan2

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

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

O-Plan2 is aimed to be relevant to the following types of problems:

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

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

4 Communication in Command, Planning and Control

4.1 The Scenario

The scenario we are investigating is as follows:

- A user specifies a task that is to be performed through some suitable interface. We call this process *job assignment.*
- A *planner* plans and (if requested) arranges to execute the plan to perform the task specified. The planner has knowledge of the general capabilities of a semi-autonomous execution system but does not need to know about the actual activities that execute the actions required to carry out the desired task.
- The *execution system* seeks to carry out the detailed tasks specified by the planner while working with a more detailed model of the execution environment than is available to the job assigner and to the planner.

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

We have deliberately simplified our consideration to three agents with these different roles and with possible differences of requirements for user availability, processing capacity and real-time reaction to clarify the research objectives in our work.

4.2 A Common Representation for Communication between Agents

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

The common representation includes knowledge about the capabilities of the planner and execution agent, the requirements of the plan and the plan itself either with or without flaws (see Figure 1). Thus, a planner will respond to the requirements of a user. Based on the knowledge of its own capabilities and that of the execution environment, it will generate a plan. This plan may then be executed directly in the real world, or, indirectly via an execution agent. The execution agent executes this plan in the real world and monitors the execution, responding to failures in one of two ways. If it does not have knowledge of its own capabilities, it simply returns knowledge of the failure to the central planner and awaits a revised plan to be sent. In this case, the execution agent is dumb. If it does have knowledge of its own capabilities, it may attempt to repair the plan and then continue with execution. On the other hand, if

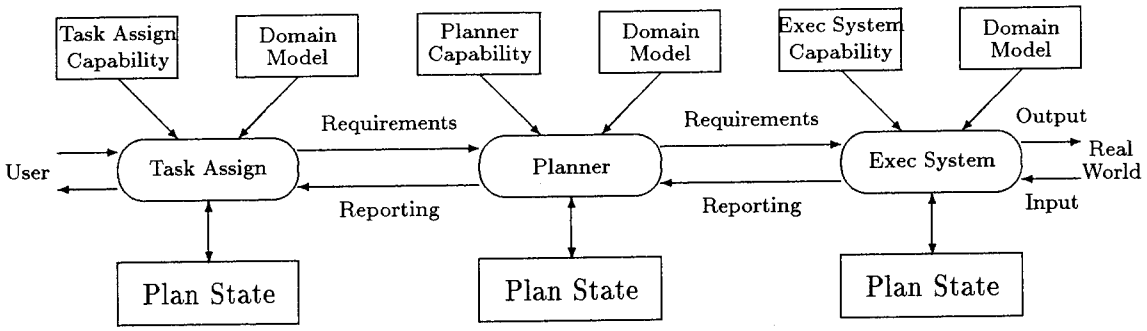


Figure 1: Communication between Central Planner and Ex. Agent

a repair is beyond the capabilities of the execution agent, then this knowledge is fed back to the central planner and again a revised plan is expected. In this case, the execution agent is semi-autonomous. When failures during the application of the plan are fed back to the planner, these may be acted upon by it and a repair of the plan made or total re-planning instigated. This may, in turn, involve the user in reformulating the task requirement. A revised or new plan is then executed. Finally, success of the execution or partial execution of the plan is fed back to the user.

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

5 O-Plan2 Architecture

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

- triggering mechanisms — *i.e.* what causes the mechanism to be activated.
- decision making roles — precisely what type of decision can be made.
- implications for search — has the search space been pruned, restricted or further constrained as far as possible.
- decision ordering — in what order should we choose between the alternative decisions possible.
- choice ordering — for a decision to be made, which of the open choices should we adopt.

The main components of an O-Plan2 agent are:

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

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

5.1 Domain Information

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

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

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

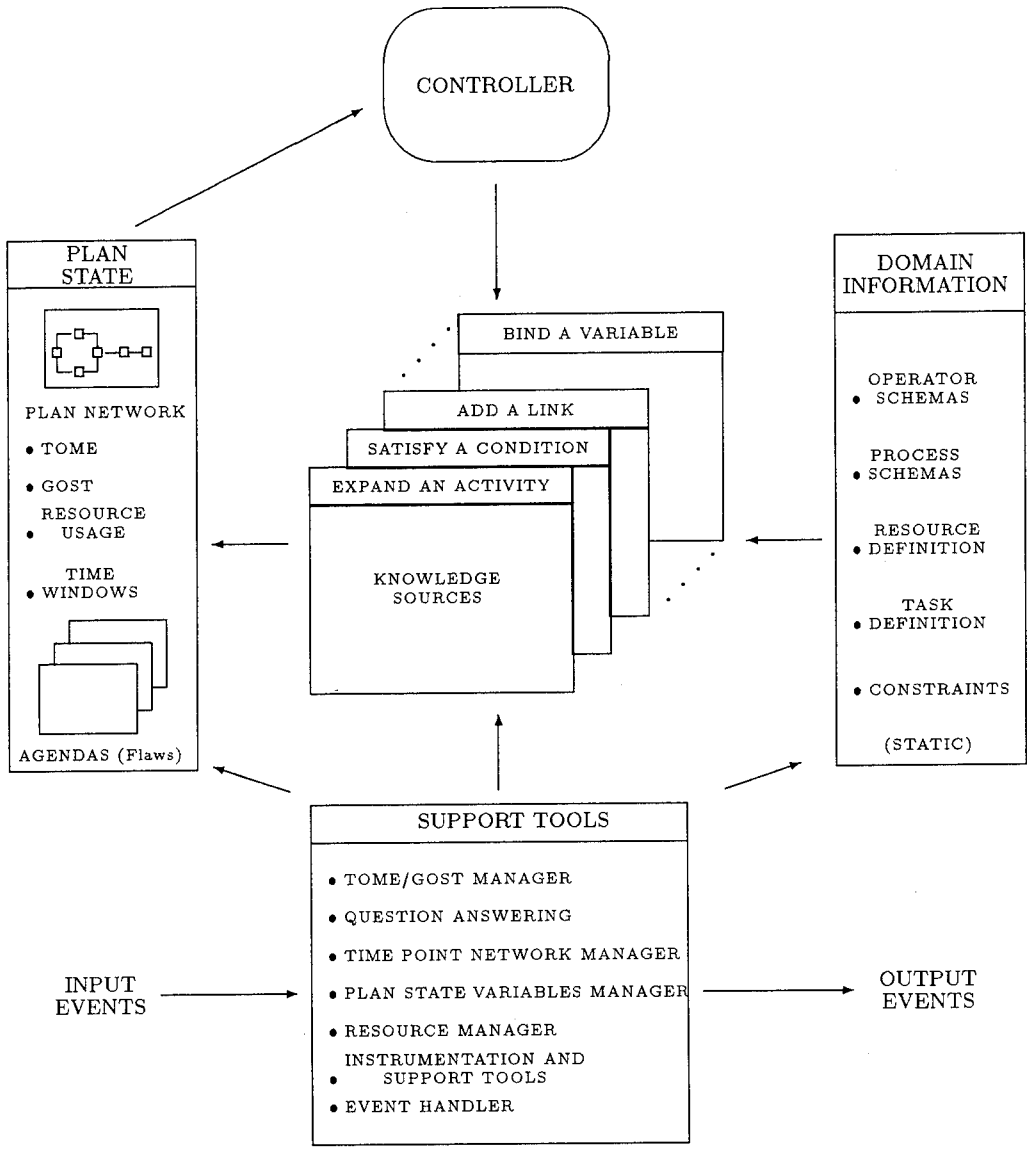


Figure 2: O-Plan2 Architecture

5.2 Plan State

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

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

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

5.3 Knowledge Sources

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

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

5.4 Support Modules

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

Support modules are intended to provide efficient support to the higher level Knowledge Sources where decisions are taken. They should not take any decision themselves. They are intended to provide complete information about the questions asked of them to the decision making level itself. Some support modules act as *constraint managers* for a sub-set of the plan state information.

The support modules include the following:

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

5.5 Controller

Holding this loosely coupled framework together is the Controller acting on the agenda. Items on the agenda (the flaws) have a context dependent priority which the Controller can re-compute,

and which allows for the opportunism required to drive plan generation. Agenda entries can be triggered by specific plan state changes or other events, such as the binding of a variable, the satisfaction of a condition, the occurrence of an external event, a reminder from an internal agent diary, etc.

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

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

6 Process Structure of the O-Plan2 Implementation

The current O-Plan2 prototype system is able to operate both as a planner and a simple execution agent. The job assignment function is provided by a separate process which has a simple menu interface.

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

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

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

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

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

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

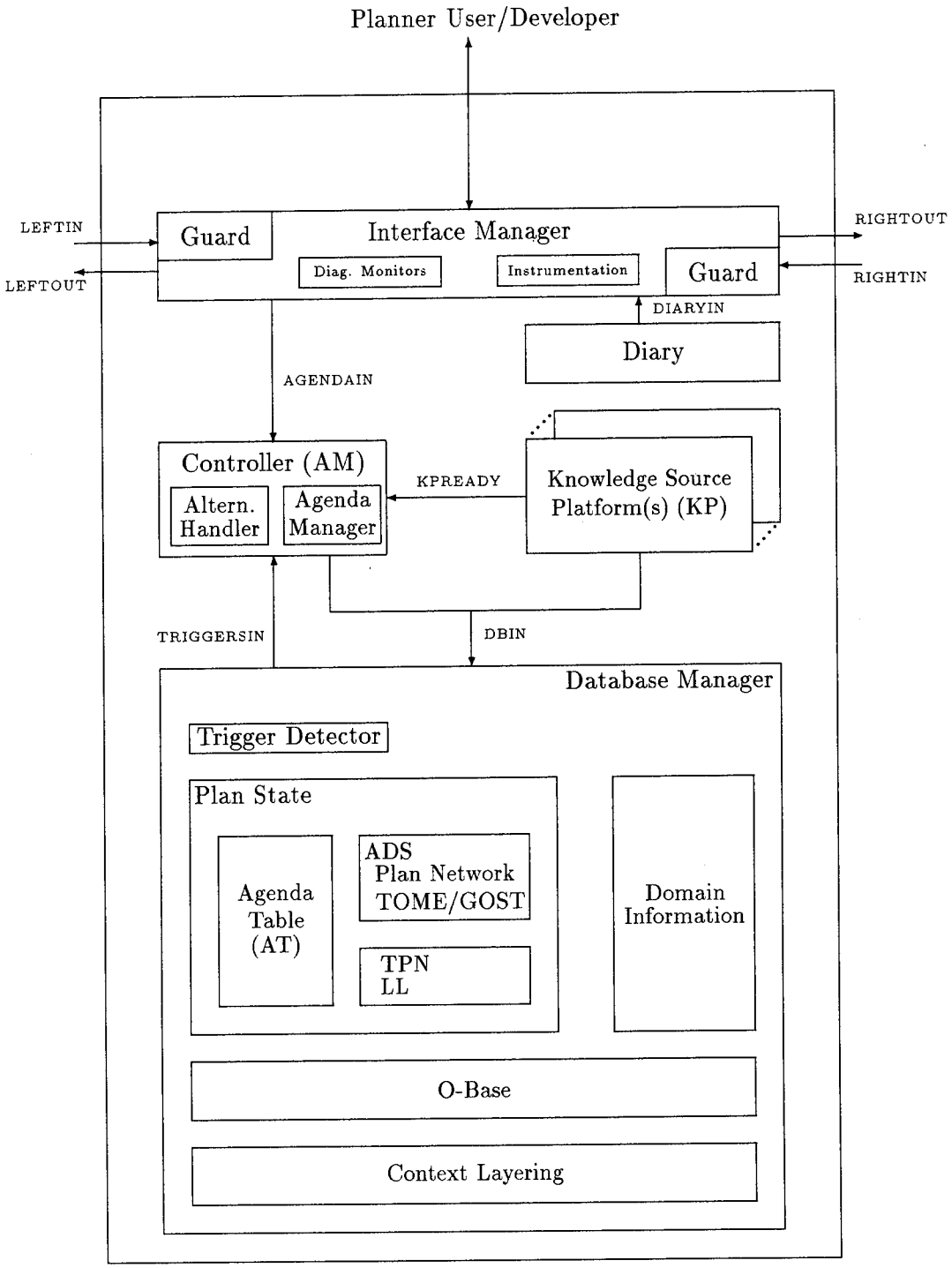


Figure 3: Internal Structure of the Current O-Plan2 Planner

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

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

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

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

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

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

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

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

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

7 O-Plan2 Planner

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

7.1 Plan State

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

7.1.1 Plan Network

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

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

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

7.1.2 TOME and GOST

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

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

The Goal Structure (GOST) holds statements of form:

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

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

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

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

7.1.3 Plan State Objects and Variables

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

7.1.4 Resource Utilisation Table

The Resource Utilisation Table holds statements of form:

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

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

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

7.2 Planning Knowledge Sources

The O-Plan2 architecture is specialised into a planning agent by including a number of Knowledge Sources which can alter the Plan State in various ways. The planning Knowledge Sources provide a collection of *plan modification operators* which define the functionality of the planning

agent beyond its default O-Plan2 architecture properties (essentially limited to initialisation and communication capabilities by default).

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

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

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

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

7.3 Use of Constraint Managers to Maintain Plan Information

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

7.3.1 Time Point Network Manager (TPNM)

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

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

7.3.2 TOME/GOST Manager (TGM)

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

7.3.3 Resource Utilisation Management (RUM)

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

1. Consumable: these are resources which as "consumed" by actions within the plan. For example: bricks, fuel, money, etc.
2. Re-usable: these are resources which are used and then returned to a common "pool". For example, robots, workmen, lorries, etc.

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

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

7.3.4 Plan State Variables Manager (PSVM)

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

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

7.4 Other Support Modules in O-Plan2

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

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

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

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

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

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

7.5 Alternatives Manager

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

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

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

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

7.6 Implementation as Separate Processes

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

8 O-Plan2 User Interface

8.1 Planner User Interface

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

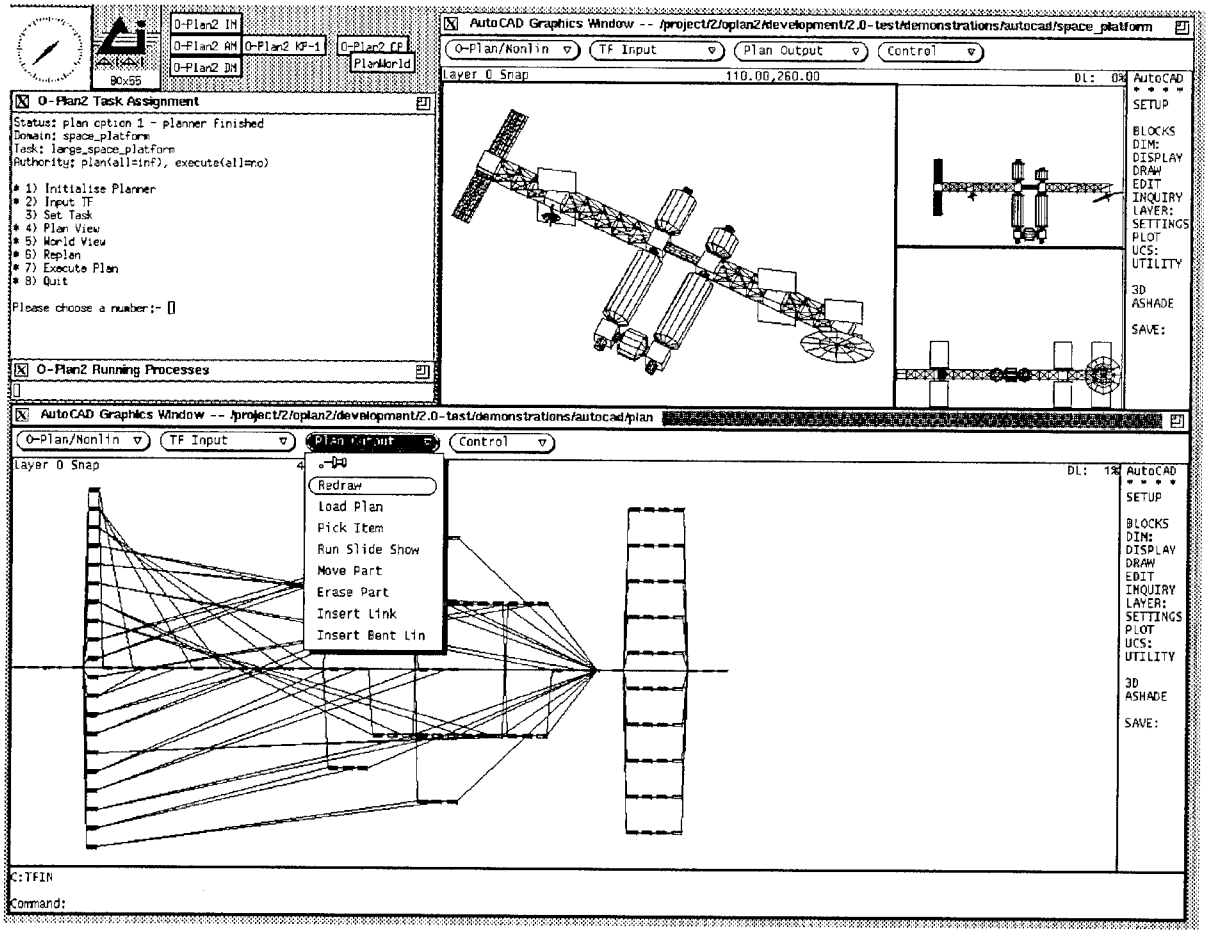


Figure 4: Example Output of the AutoCAD-based User Interface

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

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

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

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

8.2 System Developer Interface

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

A sample developer screen image is shown in Figure 5.

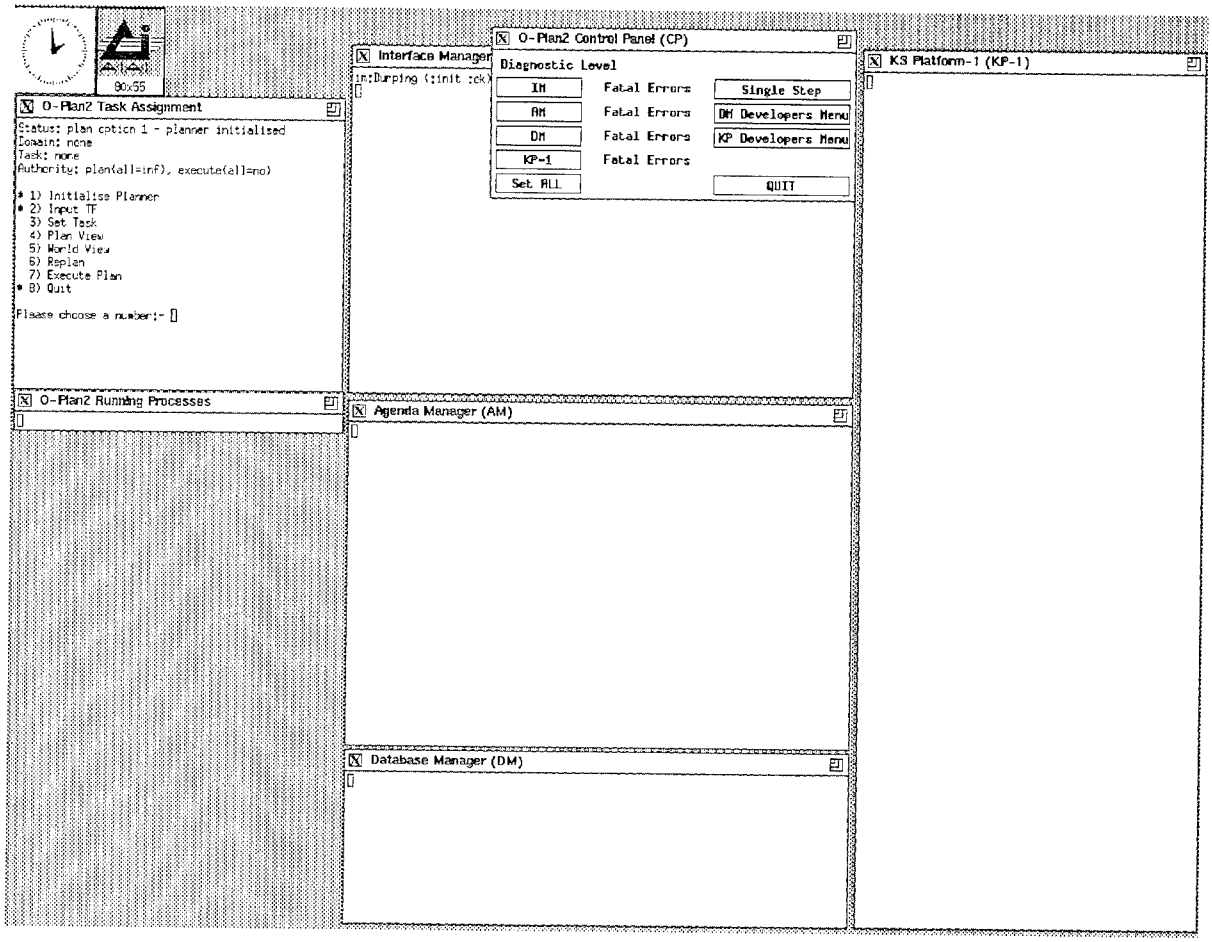


Figure 5: Example Developer Interface for the O-Plan2 Planning Agent

9 Applications

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

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

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

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

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

10 Related Projects

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

Acknowledgements

The O-Plan2 project has been supported by the US Air Force Rome Laboratory through the Air Force Office of Scientific Research (AFOSR) and their European Office of Aerospace Research and Development by contract number F49620-89-C0081 (EOARD/88-0044) monitored by Dr. Northrup Fowler III at the USAF Rome Laboratory. Additional resources for the O-Plan work has been provided by the Artificial Intelligence Applications Institute through the EUROPA (Edinburgh University Research on Planning Architectures) Institute development project.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Appendix B – The Emergence of "Standard" Planning and Scheduling System Components – Open Planning and Scheduling Architectures

Austin Tate

Appears in *Current Trends in AI Planning*, (eds. Backström. C. & Sandewall. E.), pp14-32, IOS Press, Amsterdam, 1993.

The Emergence of "Standard" Planning and Scheduling System Components – Open Planning and Scheduling Architectures

Austin Tate

Abstract

As realistic systems are built and commercial toolkits are created for planning and scheduling applications, it becomes increasingly important that modularisation of these "standard" components is attempted. This can lead to re-usability, embeddability and improved implementation provision.

One early example of a standardised component is the Truth Criterion condition establishment algorithm which is now found at the heart of most activity based planning systems. The modularisation of this algorithm has led to ann explosion of further development, empirical and theoretical study of this component. The provision of powerful constraint management languages and tools could lead to a rapid expansion of the benefits to be gained by identifying more standard components that can be combined and re-used in planning and scheduling systems.

O-Plan2 is a command, planning and control architecture which has an open modular structure intended to allow experimentation on or replacement of various components. This paper describes the modular structure of the system along with the internal and external interface languages and protocols which are being developed on the O-Plan2 project. The research is seeking to isolate functionality that may be generally required in a number of applications and across a number of different planning, scheduling and control systems.

This paper is intended to further discussions on the identification of suitable "standard" re-usable components in planning and scheduling systems.

1 Introduction

Three decades of planning and scheduling research in artificial intelligence has led to a number of "standard" approaches and components which are at the heart of the majority of systems. As realistic systems are built and commercial toolkits are created for planning and scheduling applications, it becomes increasingly important that modularisation of these "standard" components is attempted. This can lead to re-usability, embeddability and improved implementation provision.

One early example of a standardised component is the Truth Criterion condition establishment algorithm. Such a Truth Criterion is now found at the heart of most activity based planning systems. The modularisation of the capability to establish the truth of a given statement at a given point in a partially ordered network of time points in a partial plan has led to an explosion of further development, empirical and theoretical study of this component.

In order to benefit from advances in various technologies and to allow improved implementations of components to be used, we need to be able to recognise separable functions and capabilities within our planners and schedulers. By separating the processing capabilities at the *architecture* level of a planner or scheduler from the *plan or schedule representation* we can begin to address modularity issues of this kind.

Moves to provide powerful constraint management languages and tools could lead to a rapid expansion of the benefits to be gained by identifying more standard components that can be combined and re-used in planning and scheduling systems. This can allow time network management, management of the persistence of facts over time, resource management and other such constraint management by separate components provided by someone other than the original developer or integrator.

O-Plan [9] and O-Plan2 [19] are successors to Nonlin. O-Plan2 is a command, planning and control architecture being developed at the Artificial Intelligence Applications Institute of the University of Edinburgh. It has an open modular structure intended to allow experimentation on or replacement of various components without the need to change the majority of the overall system. This paper describes the modular structure of the system along with the internal and external interface languages which are being developed on the O-Plan2 project. In a number of cases, only very simple versions of the interfaces are supported in the current O-Plan2 system. However, even the early versions of such interfaces are proving useful to isolate functionality that may be generally required in a number of applications and across a number of different planning, scheduling and control systems.

This paper is intended to further discussions on the identification of suitable "standard" re-usable components in planning and scheduling systems.

2 A Successful "Standardisation" - The Truth Criterion in Planners

One early example of a standardised component is the Truth Criterion condition establishment algorithm. This was first described for the Nonlin planner in 1976 where it was called the QA algorithm [16]). Such a Truth Criterion is now found at the heart of most activity based planning systems. The modularisation of the capability to establish the truth of a given statement at a given point in a partially ordered network of time points in a partial plan has led to an explosion of further development, empirical and theoretical study of this component.

Nonlin used the partially ordered plan representation of NOAH [14], but brought this together with a teleological approach to defining the search space and searching through the space of all legal ways to resolve goal interactions. Thus it combined the protection interval maintenance approach in HACKER [15] with proper planning for the first time. Nonlin was the first planner to have an explicit criterion/algorithm to establish the value of a statement at a point in a partially ordered plan.

Nonlin used an algorithm we call Question Answering in a partially ordered network of nodes to establish the truth value of a statement at a particular point in the partial ordered plan. This took the form of a question:

Does P=V at N (using tactics)?

This asked "does the proposition P have a required value V at the indicated point N in a partially ordered network of time points". The answer is in the form of one or more possible

"contributor" time points. A set of allowable "tactics" to use to compute the answer could be given in terms of legitimate changes that could be proposed to the plan (say in terms of variable bindings or temporal orderings) to establish the required value for the proposition. The QA algorithm came back with "yes" the proposition has the required value, "no" it does not and cannot in the current plan, or "maybe" if one of the indicated conjuncts of plan constraints (in terms of variable bindings and time point orderings) is applied.

This QA algorithm became a basis for work by Chapman on the formalisation of the concept in his Modal Truth Criterion (MTC) [6]. This led to boom in formal analysis of planners from Chapman onwards. It has been a valuable *packaging* attempt since it has led to practical and theoretical advances. If the algorithm had been buried in a planner implementation and not drawn out as a separate module, this would have been more difficult. As will be seen in a later section of this paper, the interface adopted for the Condition Question Answerer should be of general utility in the packaging of other modules in planning systems.

3 A Framework for Discussing "Standard" Components

In order to benefit from advances in various technologies and to allow improved implementations of components to be used, we need to be able to recognise separable functions and capabilities within our planners and schedulers. By separating the processing capabilities at the *architecture* level of a planner or scheduler from the *plan or schedule representation* we can begin to address modularity issues of this kind.

3.1 Plan Representation

There have been a number of research and development efforts directed at defining planning and scheduling system *products* in a way which is independent of the planner or scheduler that produces or uses them. This allows results of planning to be passed between various different systems or components. Ontologies of plans and schedules have been created to underpin such representations. An example is the KRSL plan representation language defined for the ARPA/Rome Laboratory Planning Initiative programme in the USA [2].

Query languages have been defined such that one part of a planning system can query other parts or can use information in repositories. An example is the use of KQML [5] with embedded KRSL plan representations on the ARPA/Rome Laboratory Planning Initiative. The definition of a general purpose PQL (Plan Query Language), a SQL for the planning world, has been attempted in a project intended to allow interfacing between natural language front ends and various back end planning systems [7],[8].

3.2 A Planner Architecture Abstraction

It is useful to present a simple abstraction of how a planner or scheduler operates. Figure 1 shows such an abstraction that will be useful in this paper.

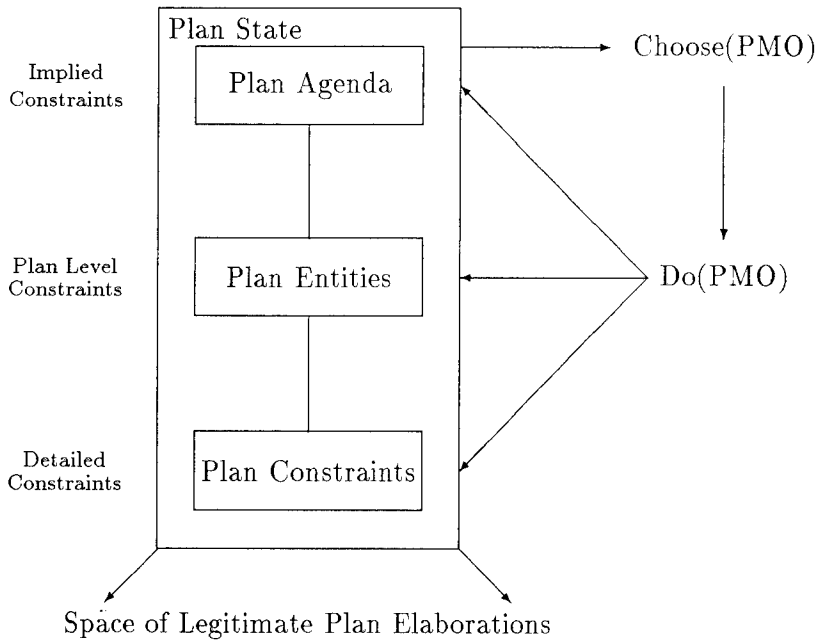


Figure 1: A Framework of Components in a Planning/Scheduling System

Many planners and schedulers work by refining a "current" plan (shown in figure 1 as the *Plan State*). They maintain one or more *partial plans* in this Plan State in which the previous decisions taken during the planning process restrict the space of plan elaborations which can be reached from that point.[1] The planner or scheduler needs to know what outstanding processing requirements exist in the plan (shown in figure 1 as the *Agenda*). These represent the implied constraints on valid plan solutions. One (normally) of these outstanding processing requirements is chosen to be worked upon next. This calls up processing capabilities within the planner which can make decisions and modify the Plan State - these are sometimes called *Plan Modification Operators*. The modifications can be in terms of definite plan structure in the Plan State or by noting further processing requirements (as a result of Plan State critiquing, etc).

We have found it to be useful to separate the plan entities representing the decisions already made during planning into a high level representing the main plan entities shared across all planning system components and known to various parts of the systems, and more detailed specialised plan entities which form a specialised area of the representation of the plan. These lower level more compartmentalised parts can represent specialised constraints within the plan such as time, resource, spatial and other constraints. This separation can assist in the identification of modularity within planning and scheduling systems.

O-Plan2 [19], for example, has an *Associated Data Structure* (ADS) level of representation [12] which holds the main plan entities (such as activities). The lower level constraints then separately handle constraints on ordering and time points in the plan, resource constraints, etc.

[1] Plan constraint relaxation is also possible to increase the space of plan elaborations in some systems.

The lower level constraints are tied to the higher ADS level entities via associations. The TOSCA manufacturing scheduling system [4] which was based on the O-Plan2 architecture makes use of quite a different ADS level based on resource reservations, but shares the same time point constraint management code at the lower level.

4 "Standardising" Constraint Management in Planners

Moves to provide powerful constraint management languages and tools could lead to a rapid expansion of the benefits to be gained by identifying more standard components that can be combined and re-used in planning and scheduling systems. This can allow time network management, management of the persistence of facts across time, resource management, spatial constraint management and other such constraints to be managed by separate components provided by someone other than the original developer or integrator.

As one example, consider the provision of the management of temporal relationships in a planner. All modern planners embed some degree of time management for temporal relationships between time points or across time intervals and may provide support for metric (definite) time "stamps" on time points. Many planners also relate their time management to the management of the persistence of facts or propositions across time. This allows planners to reason about whether some required condition is true at a given time. The Time Map Management concepts, clearly described in [10] and used in the FORBIN planner [11], are a good example of the approach. The management of effect and condition (Goal Structure) tables in Nonlin [16] uses a similar approach.

This type of packaging has led to separate study of the support for time management and fact persistence management in planners at various research centres. O-Plan2 has a Time Point Network Manager [12]. A commercial Time Map Manager (TMM) is available from Honeywell based on the concepts described in [10]. More powerful temporal relationships are managed by the General Electric TACHYON temporal system. In some cases, it has already proved possible to replace some simpler level of time constraint management in a planner with a better packaged and more powerful capability. One example of this has been the combining of the SRI SIPE-2 planner with the GE TACHYON temporal system. Other studies have indicated that the O-Plan2 TPNM can be replaced quite straightforwardly with the Honeywell TMM.

Studies at Edinburgh [13] relating to Resource Management have shown how progressively more capable resource management systems can be incorporated into O-Plan2 to replace the simple consumable resource handler in the system at present. These studies have developed a *Resource Criterion* interface to a Resource Utilisation Manager for the O-Plan2 planner which has many similarities to the interface used for the Truth Criterion/QA algorithm. This mechanism could allow resource handling by mechanisms as powerful as those based on the Habographs [4] constraint management mechanism incorporated in the Edinburgh TOSCA manufacturing scheduler.

Spatial constraint management which is not currently provided inside O-Plan2 has also been explored. We believe that clear modular interfaces can allow even such a "foreign" type of constraint management not understood by the core system at all to be be added reasonably straightforwardly to O-Plan2.

We will return in a later section to a proposal for a "standard" constraint management interface now being considered for O-Plan2. First we will introduce the O-Plan (Open Planning Architecture) work at Edinburgh and examine the ways in which modularity and interfaces are being explored in this research.

5 O-Plan – the Open Planning Architecture

The O-Plan2 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-Plan2 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 planning systems and a chart showing how O-Plan2 relates to other planning systems. See [9] for details of O-Plan (now referred to as O-Plan1), the planning system that was a forerunner to the O-Plan2 agent architecture.

The O-Plan2 system combines a number of techniques:

- A hierarchical planning system which can produce plans as partial orders on actions.
- A control architecture in which each control cycle can post further processing steps on an agenda which are then picked out 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 can alter.
- O-Plan2 is derived from the earlier Nonlin planner [16] from which it takes and extends the ideas of Goal Structure, Question Answering (Truth Criterion) and typed conditions.
- We have extended Nonlin's style of task description language Task Formalism (TF).

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 and satellites such as VOYAGER, ERS-1, etc.

6 A Sample O-Plan2 Scenario

- 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 *execution system* seeks to carry out the detailed activities specified by the planner while working with a more detailed model of the execution environment.

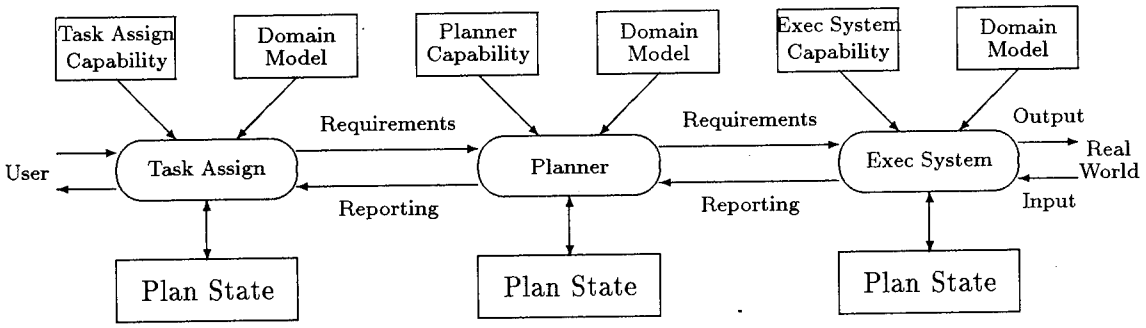


Figure 2: Communication between Strategic, Tactical and Operational Levels

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. 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. However, we believe that the ideas are relevant to the more general case of a *cooperative, hierarchical and distributed command, planning and control* environment.

A common representation is sought to include knowledge about the capabilities of the task assigner, the planner and the execution agent, and the information used to represent the requirements of the plan and the plan itself either with or without flaws (see Figure 2).

The planner components described in outline form in Figure 3 can be mapped to the system and process architecture detailed in Figure 4. Communication between the various processes and support modules in the system is shown in the latter figure. More detail of the internal structure of O-Plan2 can be found in [19].

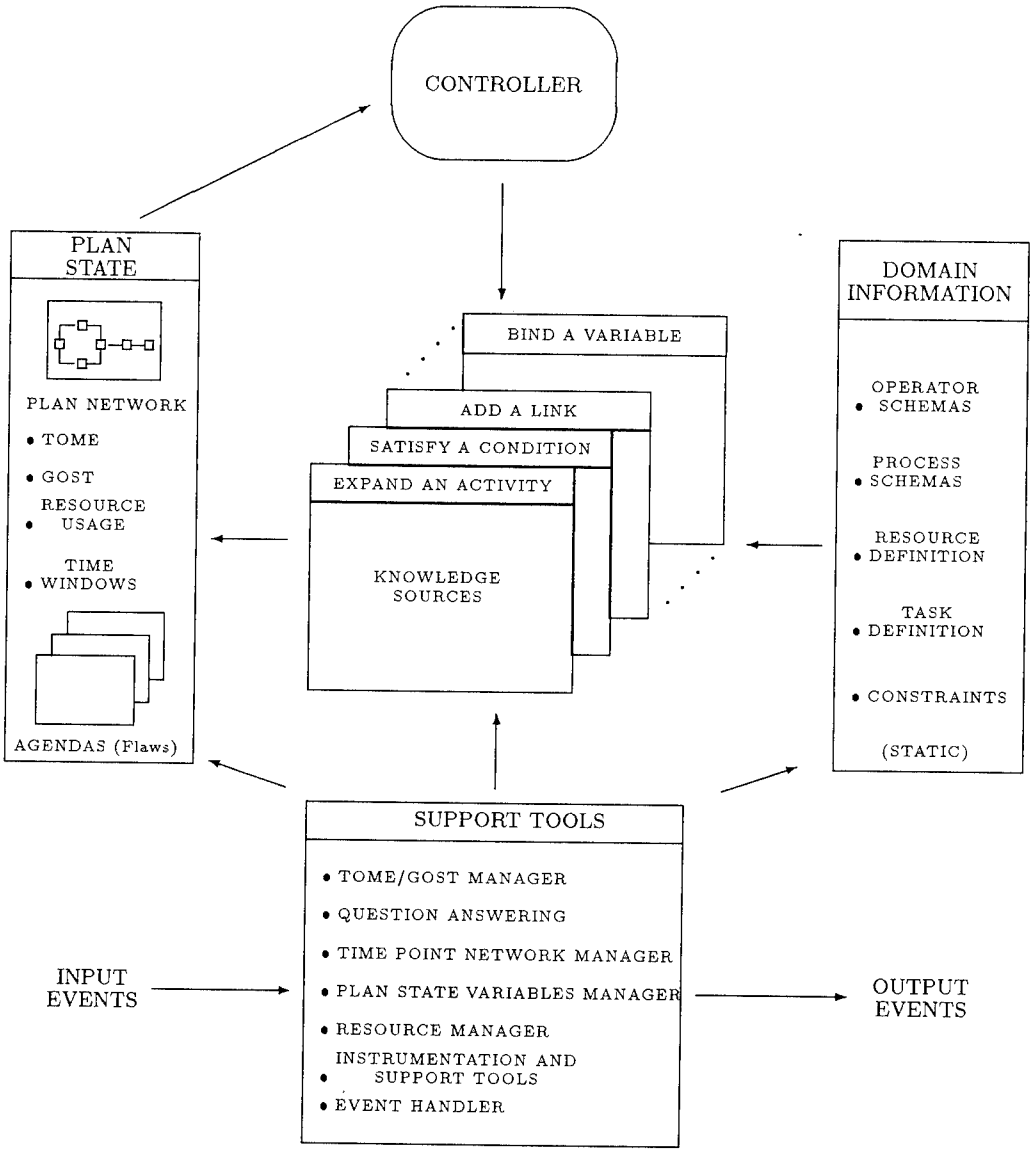


Figure 3: O-Plan2 Architecture

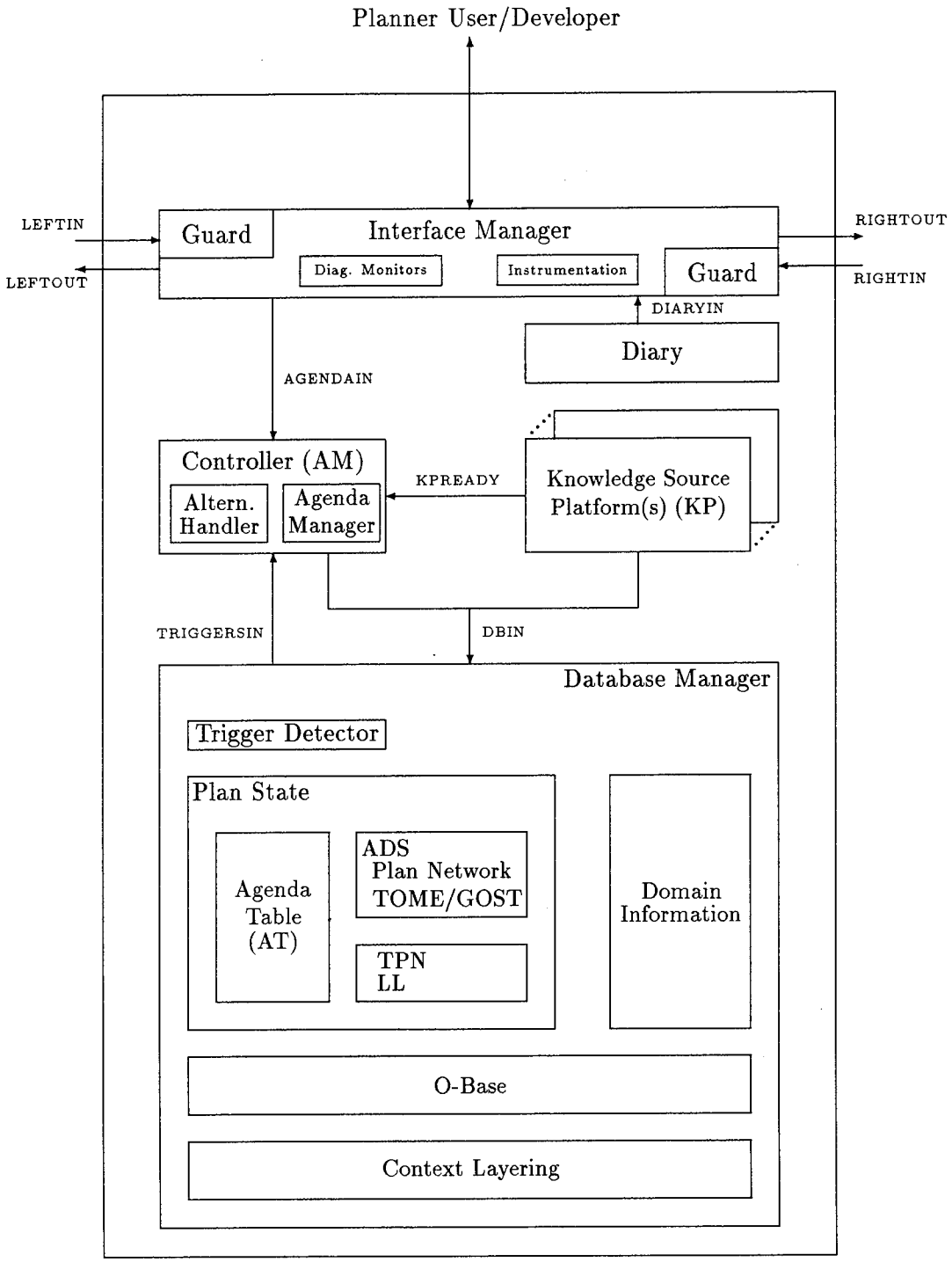


Figure 4: Internal Structure of the Current O-Plan2 Planner

7 Developer Interface

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. It therefore has a clear separation of agent roles (as shown in figure 2) and the various components (as shown in Figure 3). Each of these may be run on a separate processor and multiple *platforms* may be provided to allow for parallelism in knowledge source processing. Lower level constraint management and support routines are intended to allow for the exploitation of massively parallel computational and data base architectures and special purpose hardware.

A sample screen image as seen by the O-Plan2 developer or an interested technical user is shown in Figure 5.

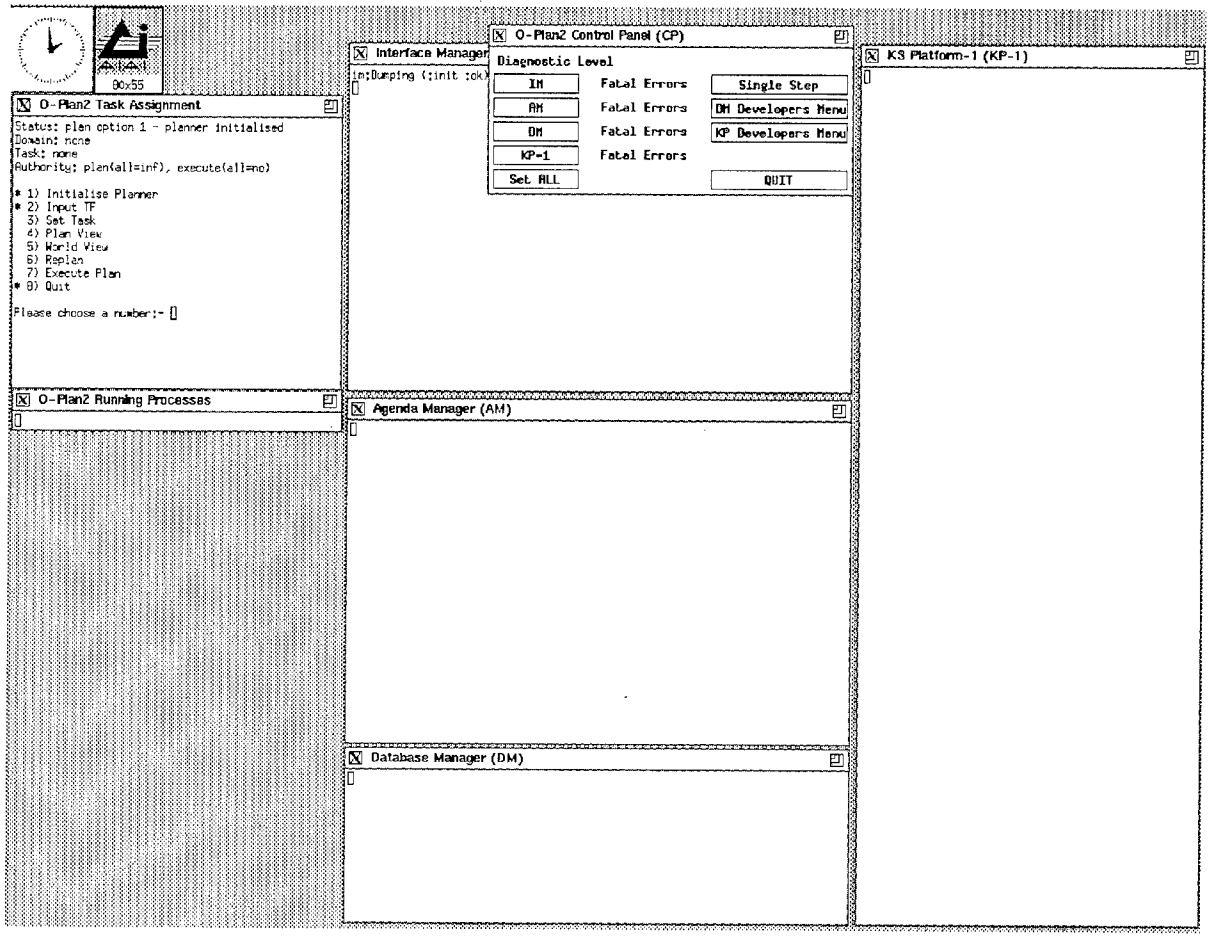


Figure 5: Example Developer Interface for the O-Plan2 Planning Agent

8 User Interface

AI planning systems are now being used in realistic applications by users who need to have a high level of graphical support to the planning operations they are being aided with. An interface to AutoCAD [3] has been built to show the type of User Interface we envisage (see Figure 6). 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 showing the plan 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 and the interfaces between these and the planner uses a defined interface to which various implementations can conform. Most of the developer aspects of the planner interface are not shown to the normal user. In figure 6, the developer windows are shown in iconic form along the top edge of the screen.

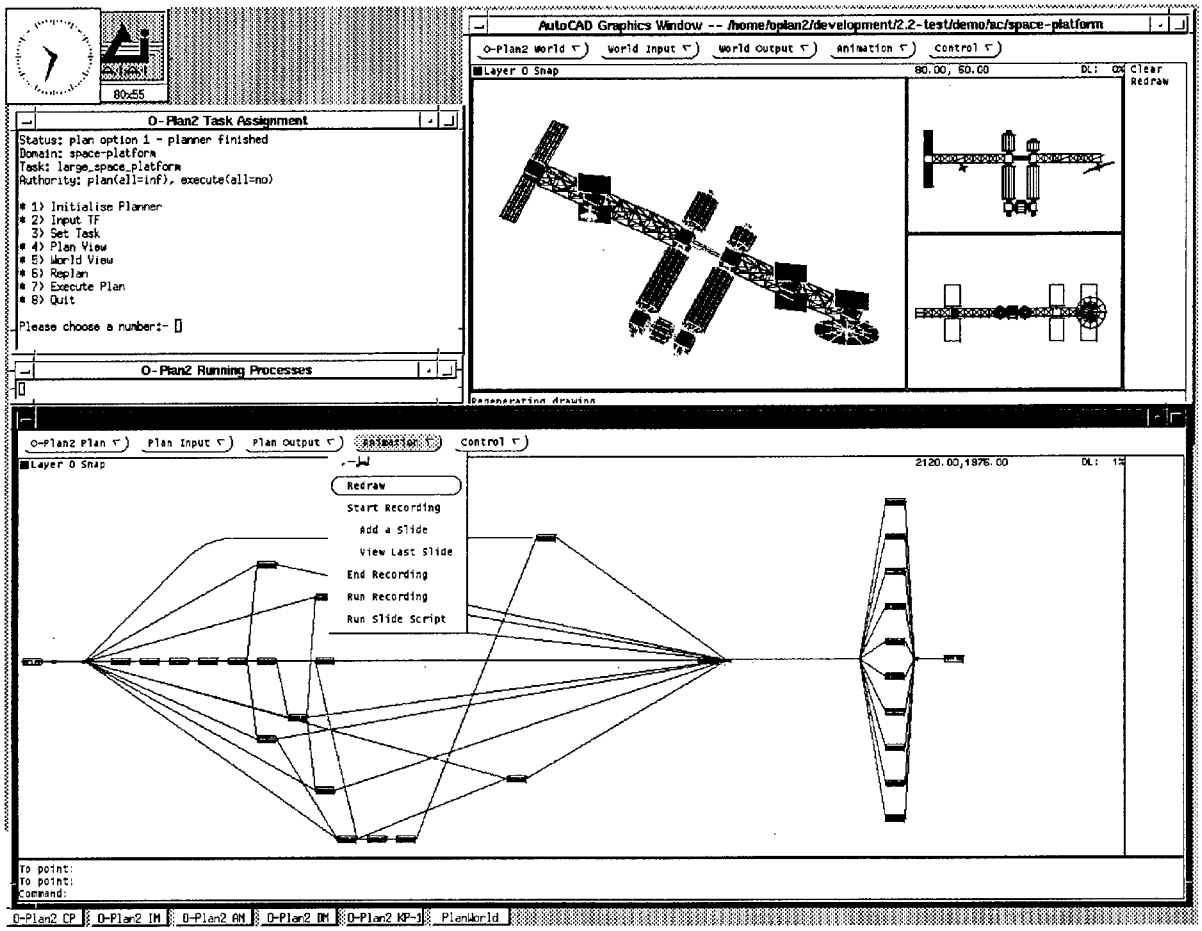


Figure 6: Example Output of the AutoCAD-based User Interface

9 O-Plan2 Modularity, Interfaces and Protocols

9.1 O-Plan2 Components

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.

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 (*Plan Modification Operators* – PMOs).
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.

9.2 Constraint Managers and Support Modules

Constraint Managers and Support Modules are intended to provide efficient support to a higher level where decisions are taken. They should not take any decision themselves. They are intended to provide complete information about the questions asked of them to the decision making level itself. They normally act to manage information and constraints in the Plan State. Examples of Constraint Managers and Support Modules in O-Plan2 include:

- Effect/Condition (TOME/GOST) Manager and Question Answering (QA) [18] – TGM.
- Resource Utilisation Manager – RUM.
- Time Point Network Manager [12] – TPNM.
- Object Instantiation (Plan State Variables) Manager – PSVM.
- Alternatives Manager.
- Interface and Event Manager.
- Instrumentation.
- Monitors for Output Messages, etc.

9.3 Protocols

In addition, a number of external interface specification and protocols for inter-module use have been established. Only first versions of these interfaces have been established at present in O-Plan2, but we believe that further development and enhancement of the planner can take place through concentrating effort on the specification of these interfaces. This should greatly assist the process of integrating new work elsewhere into the planning framework too.

The protocols for regulating the processing conducted by a component of O-Plan2 are:

1. *Knowledge Source Protocol* – describing the ways in which a Knowledge Source is called by the Controller, can run and can return its results to the Controller and for the ways in which a Knowledge Source can access the current Plan State via the Data Base Manager.
2. *KS-USER Protocol* – describing the ways in which the user (in the role of *Planner User*[2]) can assist the planning system via a specially provided Knowledge Source.
3. *Inter-agent Communications Protocol* – controls the way in which the Knowledge Sources operate and may use the Interface Manager's support routines which control the agent's input and output event channels.

9.4 Internal Support Facilities

The internal support provided within the planner to assist a System Developer and Knowledge Source writer includes:

1. *Knowledge Source Framework* (KSF) – is a concept for the means by which information about a Knowledge Source can be provided to an agent. This will ensure that a suitable Knowledge Source Platform is chosen when a Knowledge Source is run inside an agent. It will also allow a model of the capabilities of other agents to be maintained. The KSF will also allow for triggers to be set up for releasing the Knowledge Source for (further) processing. It will allow a description of the parts of a Plan State which can be read or altered by each stage within the Knowledge Source (to allow for effective planning of concurrent computation and data base locking in future).
2. *Agenda Trigger Language* – gives a Knowledge Source writer the means by which a computation can be suspended and made to await some condition. The conditions could relate to information within the plan, for external events or for internally triggered Diary events. O-Plan2 currently provides a limited number of monitorable triggers of this kind, but we anticipate this being expanded significantly in future.
3. *Controller Priority Language* – currently, the O-Plan2 Controller selects agenda entries based on a numerical priority which is simply a statically computed measure of the priority of outstanding agenda entries in a Plan State. Our aim for the future is to provide a rule

[2]The O-Plan2 design identifies a number of distinct *roles* or ways in which a user may interact with the system.

based Controller which can make use of priority information provided in the form of rules in an O-Plan2 Controller Priority Language. This concept will allow us to clarify our ideas on what information should govern Controller ordering decisions. Domain information linking to generic Controller Language statements which can affect the Controller decisions is likely to be considered as part of a link between Task Formalism (TF) and the operation of the Controller.

9.5 External Interfaces

The external interfaces provided by the planner are:

1. *Task Formalism* (TF) – as the language in which an application domain and the tasks in it can be expressed to the planner.
2. *Plan Viewer User Interface* – which allows for domain specific plan drawing and interaction to be provided.
3. *World Viewer User Interface* – which allows for domain specific world state input and simulation facilities to be provided.
4. *External System Interface* – provided by TF **compute conditions** [17] for ways in which external data bases, modelling systems, CAD packages, look-up tables, etc., can be used and for ways in which these external systems can access plan information and provide qualifications on the continued validity of their results if appropriate.

10 Constraint Managers in the O-Plan2 Architecture

O-Plan2 uses a number of *Constraint Managers* to maintain information about a plan while it is being generated. The information can then be used to prune search (where plans are found to be invalid as a result of propagating the constraints managed by these managers) or to order search alternatives according to some heuristic priority.

It is intended that some of these Constraint Managers could be replaced by more efficient or more capable systems in future. This section considers the interfaces between the O-Plan2 architecture components and Constraint Managers to help others consider packaging and integration issues.

Our experience with earlier AI planners such as Nonlin and O-Plan1 was that a large proportion of the processing time of a planner could be spent in performing basic tasks on the plan network (such as deciding which nodes are ordered with respect to others) and in reasoning about how to satisfy or preserve conditions within the plan. Such functions have been modularised and provided in O-Plan2 as Constraint Managers (such as a Time Point Network Manager, an Effect/Condition Manager and a Resource Utilisation Manager), and Support Routines (such as a Graph Operations Processor) to allow for future improvements and replacement by more efficient versions.

Constraint Managers are intended to provide efficient support to a higher level of the planner where decisions are taken. They should not take any decision themselves. They are intended to provide complete information about the constraints they are managing or to respond to questions being asked of them by the decision making level. Examples of Constraint Managers in O-Plan2 include:

- Time Point Network Manager (TPNM).
- Effect/Condition (TOME/GOST) Manager (TGM) and the related Question Answerer (QA).
- Resource Utilisation Manager (RUM).
- Object Instantiation (Plan State Variables) Manager (PSVM).

A guideline for the provision of a good Constraint Manager in O-Plan2 is the ability to specify the calling requirements for the module in a precise way (i.e. the *sensitivity rules* under which the Constraint Manager should be called by a knowledge source or from another component of the architecture).

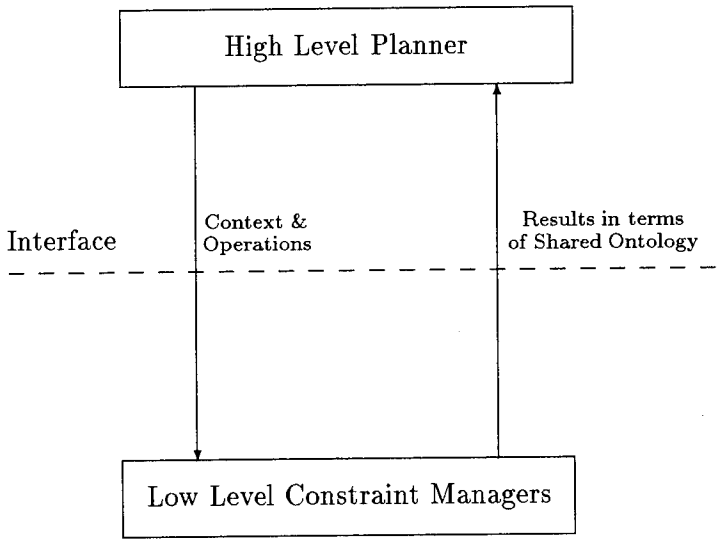


Figure 7: The Interface to Constraint Managers

The following sections explore the definition of an interface between the higher level decision making part of a planning or scheduling system and a lower level constraint manager. Figure 7 shows an overview of the interface.

10.1 Constraint Manager Procedural Interface

A Constraint Manager is a part of the Database Manager (DM) component in O-Plan2 which looks after the Plan State and all of its alternatives (if any). A Constraint Manager may look after a specialised aspect of the Plan State on behalf of the DM.

The O-Plan2 design is being rationalised so that a Constraint Manager has the following generic procedural interface:

1. initialise Constraint Manager and name base context with given <tag>[3].
2. terminate Constraint Manager
3. push context and name new context with given <tag>
4. pop context to parent of current context
5. restore a previously created context which has the <tag> specified
6. open update transaction, and within this allow:
   - allow changes to managed entities[4].
   - queries can be made inside an open transaction. Any query reflects the changes made within the transaction to date.
   - nested open update transactions are not allowed (in O-Plan2 at present).
7. commit changes made within the update transaction
8. abort changes made within the update transaction

Some of the above routines may be inoperative or null for specific managers. In particular, context management as specified above is not needed for any Constraint Manager which chooses to make use of the O-Plan2/O-Base context managed structures – since the Associated Data Structure (ADS [12]) layer in O-Plan2 guarantees that Constraint Managers will only ever be called when the contexts being referred to are preset within the O-Plan2 planner.

10.2 Shared Plan Ontology between O-Plan2 and Constraint Managers

There are specialised update and query routines supported by each onstraint Manager. These share a common plan entity model within the planner and its Associated Data Structure (ADS) layer. The design intention has been to keep this minimal, including only those elements that allow relevant communication between higher level planning decisions and lower level constraint management. This model includes *only*:

[3] Contexts specify alternative views of a Plan State. A tree of such contexts is manipulated by O-Plan2.

[4] An extra standard update routine is needed in our implementation to handle O-Plan2 TF **other_constraints** statements (constraints not directly understood by the planner) relating to this particular constraint manager.

- a directed acyclic graph of time points.
- ability to map a plan activity node end to a unique time point and a time point to all associated node ends.
- time points as plan entities.
- an ordering relation on two time points – before(tp1,tp2).
- context <tag>s to represent alternative Plan States.
- An understanding of the meaning of a Plan State Variable[5].

These entities allow for information to be communicated about constraints and options for correcting constraint violations in terms of the shared model. All other more specific entities may be unique to a specific Constraint Manager or shared only between pairs of caller and manager.

10.3 An Emerging "Standard" General Interface for Constraint Managers

The aim in O-Plan2 is to provide a standardised interface between each Constraint Manager and the rest of the planner. For this we are seeking to employ a very similar interface to that used by the Nonlin or O-Plan style Condition Question Answerer (QA) or Truth Criterion.

A Constraint Manager cannot take any decisions and cannot change parts of the Plan State not under its immediate management. It must return all legitimate answers for the query it is given or must undertake reliably the task it is given. One focus of the O-Plan2 research has been to build a *planning ontology* which describes those concepts which are shared between constraint managers and those parts of the Plan State which are private to the relevant manager.

A Constraint Manager's primary function is to manage the current set of constraints relevant to that manager (time, resource, spatial, objects, etc) which are part of the Plan State. It must signal to the caller when there is an inconsistent set of such constraints.

The interface allows for a constraint entry to be tested against existing managed constraints to see what the impact of making the entry would be, and then a commit or abort can be done to add it or not (either the commit or the abort could be active – the caller not being able to tell).

All Constraint Manager update routines return one of three results:

- **yes** – constraint is now under management (to be confirmed later by a caller using a commit update transaction).
- **no** – constraint cannot be added within the capabilities of the Constraint Manager and its communications capability to the caller (in terms of the shared ontology of entities).

[5]The exact nature of what needs to be understood in the shared ontology needs to be considered further.

- **maybe** – constraint can be added if plan entities are altered as specified in terms of the shared entity model. This normally means returning a standard O-Plan2 "or-tree"[6] of *all* (for search space completeness) the legal ways in which the Plan State can be altered (sets of Plan State Variable restrictions and ordering constraints between time points) to maintain consistency.

  The constraint is *not* added after this maybe response. However, an "actually add constraint" routine may be provided to more cheaply add the constraint immediately following a query which returned "maybe". This would follow action by the caller to ensure at least one of the relevant binding constraints and/or time point orderings options were either dealt with or noted as necessary in the Plan State - thus the caller takes responsibility for resolving inconsistencies (*not* the Constraint Manager).

It is hoped to be able to take the result or-trees generated by the various Constraint Managers in O-Plan2 (TGM, RUM, PSVM and the TPNM) and merge them into a consistent or-tree which would represent an efficiently ordered set of possibilities – thus reducing the size of the search space.

We believe that this style of interface between the higher level decision making level of the planner and the various Constraint Managers could improve modularity in planning systems.

11 Summary

This paper was intended to further discussions on the identification of suitable "standard" re-usable components in planning and scheduling systems.

This paper has presented an overview of the O-Plan2 system under development at the Artificial Intelligence Applications Institute of the University of Edinburgh. Aspects of the system concerned with separation of functionality within the system, internal and external interfaces have been addressed. The O-Plan2 system is starting to address the issue of what support is required to build an evolving and flexible architecture to support command, planning and control tasks.

One particular area highlighted has been the interface between planning systems and Constraint Managers able to look after certain specialised aspects of parts of a plan on behalf of the overall planning system. An interface to such Constraint Managers has been developed to show how improved packaging can be beneficial to re-use of components. The value of the type of interface developed for the Condition Question Answering procedure in planners (the Truth Criterion) to act as a general interface to a number of different Constraint Managers has been explored.

[6] a data structure representing the alternative ways in which the Plan State may be altered in terms of the shared plan ontology.

acknowledgements

O-Plan2 is an on-going project at Edinburgh. I am grateful to Brian Drabble and Jeff Dalton 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.

Parts of section 9 of this paper were previously presented at the European Space Agency's Round Table on Planning and Scheduling Tools, 6-7th April 1992, ESTEC, Noordwijk, The Netherlands.

References

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

[2] ARPA/Rome Laboratory Planning and Scheduling Initiative, Knowledge Representation Specification Language (KRSL) Reference Manual (eds, Allen, J. and Lehrer, N.), Version 2.0, September 1991. ISX internal technical report, ISX, 4353 Park Terrace Drive, Westlake Village CA 91361, USA.

[3] AutoDesk, AutoCAD 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. Also available as AIAI Technical Report AIAI-TR-121.

[5] Chalupsky, H., Finin, T., Fritzson, R., McKay, D. Shapiro, S. and Wiederhold, G., An Overview of KQML: A Knowledge Query and Manipulation Language, Proceedings of Concurrent Engineering and Computer Aided Logistics Conference, Washington, USA, June 1992.

[6] Chapman, D. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333-377, 1991.

[7] Crabtree, B., Crouch, R.S., Moffat, D.C., Pirie, N., Pulman, S.G., Ritchie, G.D. and Tate, A., A Natural Language Interface to an Intelligent Planning System, Proceedings of the UK Information Engineering Directorate IT Conference, Swansea, July 1988. Also available as Department of AI Research Paper No. 407, University of Edinburgh.

[8] Crouch, R.S. and Pulman, S.G., Time and Modality in a Natural Language Interface to a Planning System, *Artificial Intelligence* 63, pages 265-304, 1993.

[9] Currie, K.W. & Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence* 51(1), Autumn 1991, North-Holland.

[10] Dean, T. and McDermott, D., Temporal Database Management, *Artificial Intelligence* 32(1):1-56, 1987.

[11] Dean, T., Firby, J. and McDermott, D., Hierarchical Planning Involving Deadlines, Travel Time and Resources, *Computational Intelligence*, 6(1), 1990.

[12] Drabble, B. and Kirby, R.B., Associating A.I. Planner Entities with an Underlying Time Point Network, European Workshop on Planning (EWSP) 1991, Springer-Verlag Lecture Notes in Artificial Intelligence. Also available as AIAI Technical Report AIAI-TR-94.

[13] Drabble, B. and Tate, A., Resource Representation and Reasoning in O-Plan2, AIAI Technical Report ARPA-RL/O-Plan/TR/6, April 1993.

[14] Sacerdoti, E., *A Structure for Plans and Behaviours*, Artificial Intelligence Series, North Holland, 1977.

[15] Sussman, G.J., A Computational Model of Skill Acquisition, MIT AI Laboratory Technical Report TR-297, 1973.

[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., Goal Structure, Holding Periods and "Clouds", Proceedings of the Reasoning about Actions and Plans Workshop, Timberline Lodge, Oregon, USA, (eds, Georgeff, M.P. and Lansky, A.) Morgan Kaufmann, 1986.

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

[20] Wilkins, D., *Practical Planning*, Morgan Kaufmann, 1988.

Appendix C – O-Plan: A Situated Planning Agent

Brian Drabble and Austin Tate

Appears in the Proceedings of the Third European Workshop on Planning (EWSP'95), Assisi, Italy, September 1995.

O-Plan: A Situated Planning Agent

Brian Drabble and Austin Tate

Abstract

This paper describes the need to view a planner as situated in an environment dealing with the whole "planning" problem. While a planning agent deals with plan generation aspects, other agents are concerned with aspects such as task elicitation, plan analysis, reactive execution, plan repair, etc. Each of these systems has its own perspective on the planning problem and each of the systems must be capable of communicating in a way which allows other systems to assimilate new information into their perspective of the problem. Within such a collection of agents a situated planner takes task assignments from a superior agent and creates a plan or further elaborates it before passing it to the execution support agents for further processing or enactment.

The paper describes the O-Plan system which has been developed as an architecture within which situated agents, such as planning agents, can be created. The paper provides a summary of work to date on the planning and execution agents. The paper then goes on to describe current research involving O-Plan which aims to address the communication between a task assignment agent and a planning agent. It concentrates on three key issues in this area: communication of plans, assessment of the quality of plans and the role of authority in the planning process.

1 Introduction

The aim of this paper is to describe the need for AI planners to be viewed as situated agents, where planning is one of a number of tasks involved in dealing with the whole problem of task assignment, planning, execution and control. While the planner deals with the plan generation aspect of the problem other agents deal with problems such as task elicitation, plan analysis, reactive execution, plan repair, etc. Each of these systems has its own perspective on the planning problem and each of the systems is capable of communicating in a way which allows other systems to assimilate new information into their perspective of the problem. Within such a collection of agents a situated planner takes task assignments from a superior agent and creates a plan or further elaborates it before passing it to the execution support agents for further processing or enactment.

The reason for taking this view is that planners cannot be considered as functioning in isolation. In addition to being able to communicate about the overall task being performed, the planner must be able to interact closely with the environment in which it is placed.

In many domains such as manufacturing, construction assembly, logistics, spacecraft control, etc. the planner needs to deal with a changes occurring in two very different ways:

1. **Change of Task and Requirements:**
   The task set to the planner may change as the plan is being generated requiring the planner to:

- alter its focus, 'plan to move the 82nd airborne now rather than later',
- choose alternative methods, e.g. 'move the 82nd by sea rather than air',
- abandon the task altogether, 'abandon the deployment task and return the 82nd airborne to their home base'.

2. **Change in the Environment:**
   Events may occur in the domain which require the plan to be repaired by the insertion of new constraints or activities. In some cases the failure may be so severe that the entire plan needs to be abandoned and an alternative chosen.

Generating plans and reacting to changes in the environment have been the primary focus to date for the O-Plan project [4, 16] as well as a many other researchers. However, the problem of dealing with task assignment and its link to the generative planner has been neglected by planning researchers. In many domains the problems of command, task setting, planning, plan analysis and enactment have been compartmentalised leading to many systems having an inability to assimilate new information into existing plan options. Current research on the O-Plan project aims to address this area and in particular the problem of allowing different situated agents to maintain their own perspective on the planning problem while at the same time allowing plans to be communicated between them. This will make it possible to communicate and use commands, plans, and tasks with improved precision, timeliness and level of detail between a number of situated agents. Research has already begun on addressing three key issue of the task assignment problem:

- communication of plans between agents,
- assessment of the quality of plans being generated by an agent,
- the role of authority in the task assignment process.

Each of the these key issues is dealt with in a separate section.

The structure of the paper is as follows. Section 2 provides an overview of the O-Plan system and the areas of AI planning research which it has covered to date. Section 3 describes the new focus of the O-Plan project and provides details of the research already undertaken in the area of Task Assignment. Section 4 provides a summary of the paper and outlines future work in the area of Task Assignment.

2 O-Plan - the Open Planning Architecture

O-Plan is a command, planning and control architecture which has an open modular structure intended to allow experimentation on or replacement of various components. The research is seeking to isolate functionality that may be generally required in a number of applications and across a number of different command, planning, scheduling and control systems. O-Plan has been applied to logistics tasks in which flexibility of response to changing situations is required.

O-Plan is intended to be a domain-independent, general planning and control framework with the ability to make use of detailed knowledge of the domain. See [2] for background reading on AI planning systems. See [4] 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 [16].

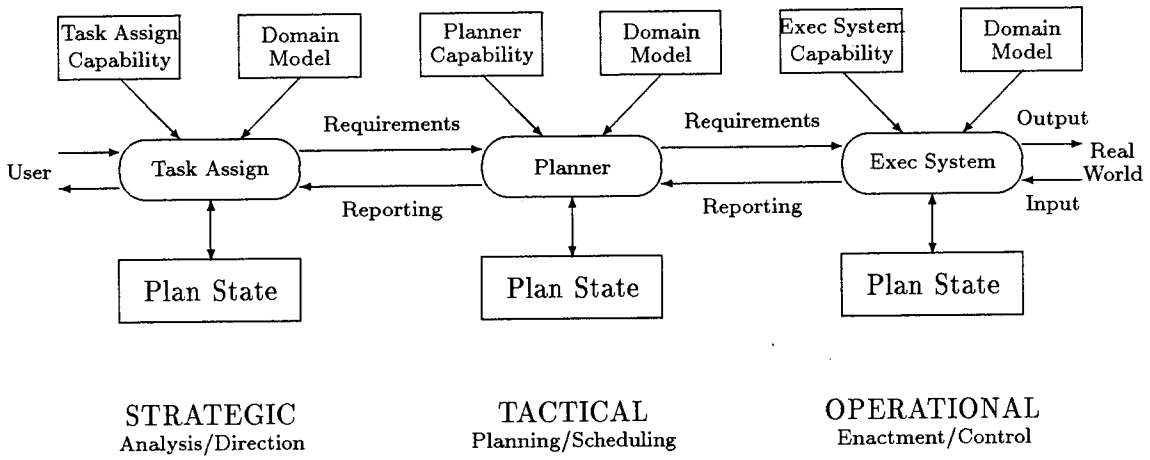


Figure 1: Communication between Strategic, Tactical and Operational Agents

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
- 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 [13]. 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.

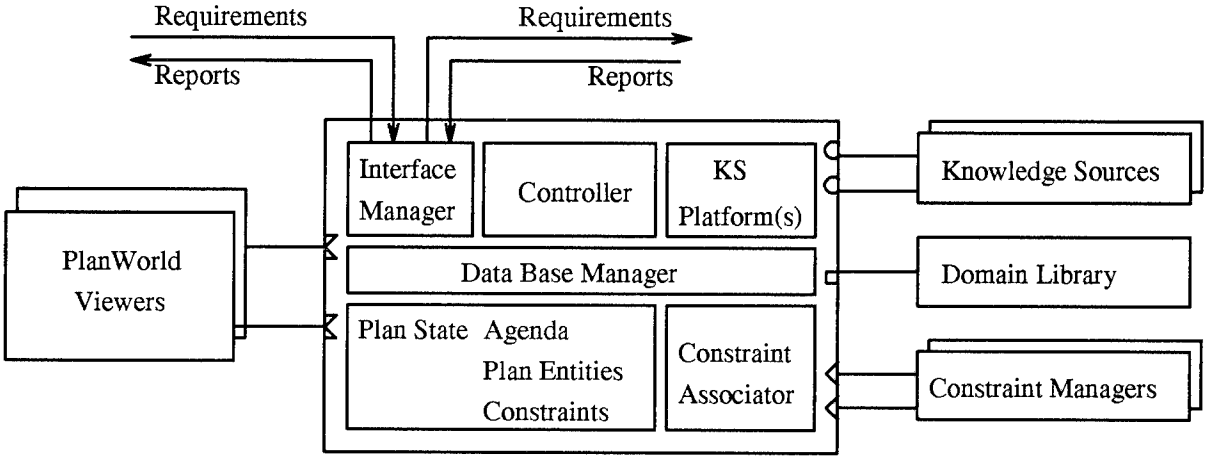


Figure 2: O-Plan Agent Architecture

The various components of the architecture for each O-Plan agent 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.

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 Manager – 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.

3 Task Assignment to a Situated Planning Agent

The aim of this section is to provide an overview of the new focus of the O-Plan project. The main emphasis is on the development of a task assignment agent and the link between it and the planning agent. In developing the designs for a task assignment agent the project has addressed three key issues:

- **Communication:**
  Within a number of situated agents each with their own perspective of the planning problem it is essential that they can communicate plans, tasks, etc, in a form which can be easily generated and integrated into different perspectives. The <I-N-OVA> [1] constraint model of plans [14] has been developed to address this and a number of other issues related to the communication of plans.

- **Plan Quality:**
  The task assigner needs to analyse the quality of the plans being generated and to provide feedback and direction concerning the options and plans which should be explored further. Joint work with USC/ISI to link O-Plan to their EXPECT system [7] has shown that plans can be generated and analysed to provide valuable feedback to human planners.

- **Role of Authority:**
  The activities of the various situated agents need to be coordinated and authority management is viewed as one of the main constraints to be considered. For example, in generative planning authority is important in considering which options are available and what level of planning detail should be considered. In enactment it is important to identify (and possibly name) which phases of the plans can be executed and which parts should be held back.

Each of these items will be dealt with in a separate subsection.

[1] <I-N-OVA> is pronounced as in "Innovate".

3.1 Communication of Plans between Situated Agents

The <I-N-OVA> (*Issues – Nodes – Orderings/Variables/Auxiliary*) Model is a means to represent plans as a set of constraints . By having a clear description of the different components within a plan, the model allows for plans to be manipulated and used separately to the environments in which they are generated.

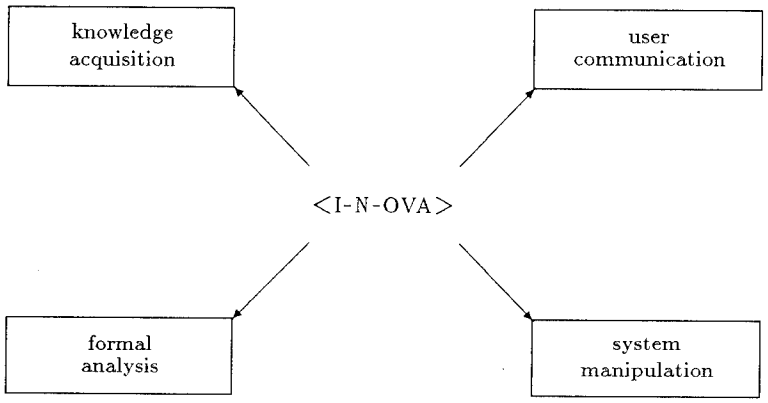


Figure 3: <I-N-OVA> Supports a Number of Requirements

As shown in Figure 3 the <I-N-OVA> constraint model underlying plans is intended to support a number of different uses of plan representations:

- automatic manipulation of plans and to act as an ontology to underpin such use.
- human communication about plans.
- principled and reliable acquisition of plan information.
- formal reasoning about plans.

These cover both formal and practical requirements and encompasses the needs of both human and computer-based planning systems.

Our aim is to characterise the plan representation used within O-Plan [4],[16] and to more closely relate this work to emerging formal analyses of plans and planning. This synergy of practical and formal approaches can stretch the formal methods to cover realistic plan representations, as needed for real problem solving, and can improve the analysis that is possible for production planning systems.

A plan is represented as a set of constraints which together limit the behaviour that is desired when the plan is executed. Work on O-Plan and other practical planners has identified different entities in the plan which are conveniently grouped into three types of constraint. The set of

constraints describes the possible plan elaborations that can be reached or generated as shown in Figure 4.

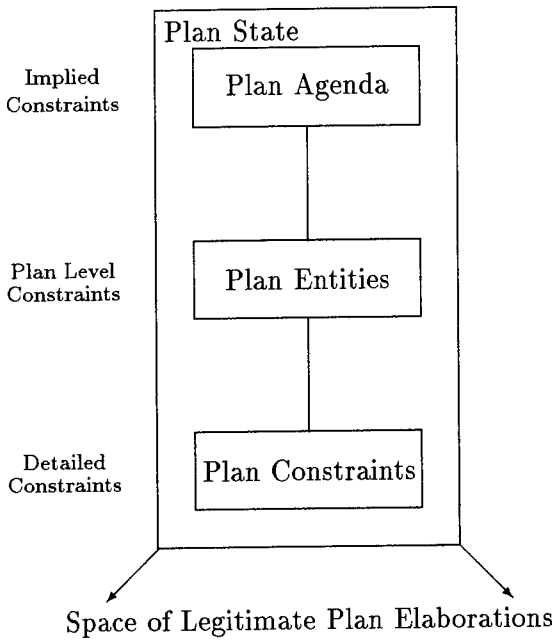


Figure 4: Plan Constraints Define Space of Plan Elaborations

The three types of constraint in a plan are:

1. Implied Constraints or "Issues" – the pending or future constraints that will be added to the plan as a result of handling unsatisfied requirements, dealing with aspects of plan analysis and critiquing, etc. The implied constraints are the issues to be addressed, i.e., the "to-do" list or agenda which can be used to decide what plan modifications should be made to a plan by a planner (user or system).

2. Plan Entities or Plan Node Constraints – the main plan entities related to external communication of a plan. They describe a set of external names associated to time points. In an activity planner, the nodes are usually the actions in the plan associated with their begin and end time points. In a resource-centred scheduler, nodes may be the resource reservations made against the available resources with a begin and end time point for the reservation period.

3. Detailed Constraints – specialised constraints on the plan associated with plan entities. Empirical work on the O-Plan planner has identified the desirability of distinguishing two special types of detailed constraint:

    - Ordering or Temporal Constraints (such as temporal relationships between the nodes or metric time properties).

- Variable Constraints (co-designation and non-co-designation constraints on plan objects in particular).

These two constraints are highlighted since they often form part of other constraints within a temporal reasoning domain such as occurs in planning and scheduling problems. Knowing that these constraints have such cross "associations" has been found to simplify planner system design of constraint handling mechanisms and ease implementation issues [13],[15].

Other Detailed Constraints relate to input (pre-) and output (post-) and protection conditions, resources, authority requirements, spatial constraints, etc. These are referred to as:

- Auxiliary Constraints.

Auxiliary Constraints may be expressed as occurring at a time point (referred to as "point constraints") or across a range of the plan (referred to as "range constraints"). Point constraints can be used to express input and output constraints on nodes or for other constraints which can be expressed at a single time point. Range constraints relate to two or more time points and can be used to express protection intervals, etc.

There is a deliberate and direct mapping of the model of plans and activity used within O-Plan and the <I-N-OVA> Constraint Model of Plans to existing structured analysis and diagraming methods such as IDEF, R-Charts, etc. Other researchers have recognised the value of merging AI representation concepts with structured analysis and diagramming techniques for systems requirements modelling (e.g., [3],[11]).

3.2 Integrating Plan Quality Considerations into Planning

In producing plans, human planners take into account a variety of criteria that guide their decisions. Besides constraints imposed by the domain itself, these criteria often express preferences among alternative plans that meet the given requirements. Human planners can use these criteria for two important purposes:

- when asked to generate one plan, human planners are able to discern between an ordinary solution and a better quality one and propose the latter.
- when asked to generate several alternative plans, human planners are able to discern between similar alternative solutions and qualitatively different ones. They can relax different criteria to explore tradeoffs.

Current AI planners are good at generating a solution that satisfies the requirements that they are given. Some planners provide facilities to control the quality of the solution to be returned, by using evaluation functions or search control rules. However, they do not usually integrate plan quality considerations across several plans. In addition, there is not enough data on the adequacy of these representations to reflect the plan quality criteria that are necessary in practice. Often, the quality criteria that human expert planners consider:

- are highly dependent on the situation and the scenario at hand. Some criteria may be more important if there is a certain deadline, or new criteria may need to be considered if new considerations come up.
- include complex factors and tradeoffs that are often not represented by an automatic planner

Research in the area of plan analysis has concentrated on addressing two keys issues:

- to provide a tool – EXPECT – [7] which allows human planners to define criteria for plan quality and for preferences among alternative plans and options.
- to operationalise these criteria to guide a generative planner – O-Plan – in proposing better quality plans [8, 5, 6].

The approach taken has been to combine the O-Plan planner with a knowledge-based system that reasons about plan evaluation EXPECT Figure 5 describes the architecture of the O-Plan and EXPECT systems and the way in which plans and analysis information flows.

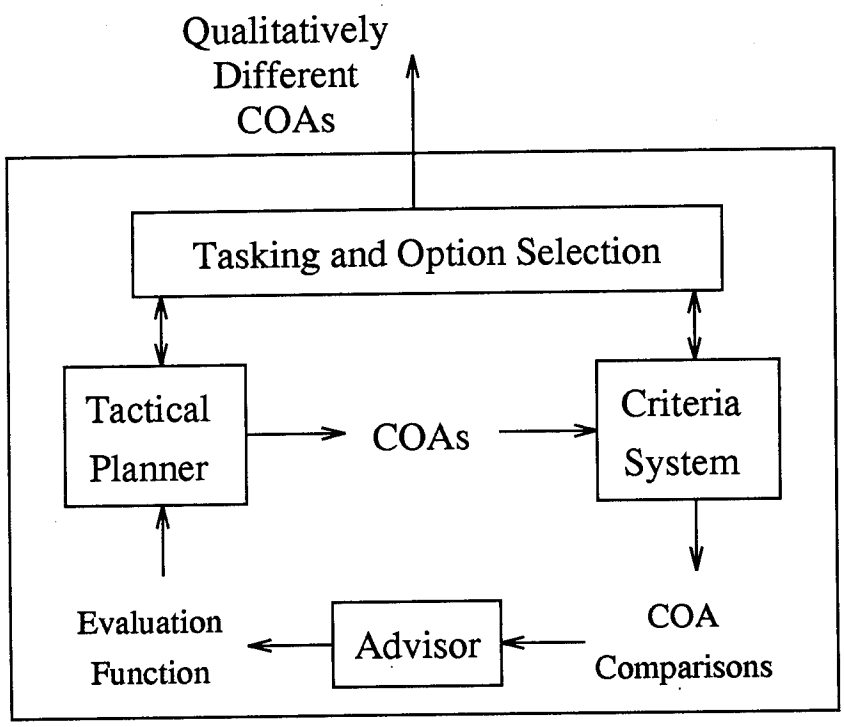


Figure 5: Situated Planner and Plan Analysis Tool

Using these two systems it has been possible to build an interface between the planner and the user that provides the following functionality:

	COA 1	COA 2	COA 3	COA4
A-PORTS:				
- airports	1	1	1	2
- sorties/hr	315	315	315	480
- sq ft ac parking	2M	2M	2M	3M
S-PORTS:				
- seaports	1	1	1	2
- piers	6	6	6	15
- berths	6	6	6	16
- max vessel size in ft	600	600	600	765
- oil facilities	1	1	1	3
CLOSURE DATE	C + 29	C + 22	C + 23	C + 23
LOG PERS	1154	5360	5396	7362
LOCs:				
- number locations	1	5	7	6
- miles max distance	20	99	140	120
- air and sea?	yes	yes	yes	yes

Figure 6: EXPECT's evaluation of several alternative plans generated by O-Plan.

- support to the user in defining criteria for evaluating plan quality through a knowledge acquisition tool,
- evaluation of the quality of plans proposed by the planner,
- provision of justifications for good and bad plan quality.

The work on plan analysis is motivated by the transportation planning domain that is the focus of the ARPA/Rome Laboratory Planning Initiative [10]. This domain involves the movement of materials and forces with a mixture of aircraft, ships, trucks and trains. The task being investigated is to generate multiple Courses of Action (COAs) and an evaluation of the tradeoffs among them using the relevant plan quality evaluation factors from a logistics perspective. This allows the human planners to identify those options which are critical to a plan's success and those parts of the plan which need further exploration and refinement. An evaluation matrix from a series of different COAs is described in Figure 6.

To date the O-Plan system is able to generate plans which can be evaluated by the EXPECT system. Work is continuing to extend EXPECT and O-Plan to strengthen the ability to support a user in specifying, comparing and refining the constraints on qualitatively different plans at the task assignment level of a planning support environment and to allow this information to be used directly by O-Plan in guiding it in its search for the best solution.

3.3 The Role of Authority for a Situated Planning Agent

At the moment, the Task Assignment agent in O-Plan informs 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. It is intended that O-Plan will support authority management in a more comprehensive and principled way in future [12]. O-Plan 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[2] 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[3].
- 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[4].
- the notion of plan *levels*. Greater precision of authority management is possible by specifying more explicit levels in the O-Plan domain description language 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 Task Assignment agent of O-Plan 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 current release version of O-Plan has a simple authority scheme and reports this in the Task Assignment menu shown here.

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

This reports that the system is planning for task `Operation_Blue_Lagoon` in the domain `pacifica` and that it is planning within `plan option 1` currently. It is authorised to plan to any level of detail for all phases (`plan all=inf`) but is not yet authorised to execute any actions (`(execute all=no)`.

A prototype HARDY-based[5] user interface for the Task Assignment agent has been created.

[2]Mutiple conjunctive tasks in one scenario is also possible.

[3]Plan options may be established and explicitly switched between by the Task Assignment agent.

[4]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-Plan planning agent.

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

4 Summary

There is a need to view AI planning systems as being situated within an environment where there are a number of other agents and systems which deal with the whole planning problem. While the planner is responsible for the plan generation aspects of the problem, other agents should be responsible for dealing with other aspects of the whole "planning" problem, e.g. task elicitation, plan analysis, reactive execution monitoring, etc. This view is motivated by the obvious realisation that planning systems cannot operate in isolation and for a task to be solved successfully its needs to be communicated and reasoned with between a number of systems.

The O-Plan architecture has been designed to support the creation of situated agents and work to date has concentrated on building generative planning agents and an execution agent with links between these two agents. The outcome of this research has been used in a number of systems. For example, the reactive execution agent work of Reece [9] and the Optimum-AIV [1] system developed for Assembly, Integration and Verification of spacecraft at the European Space Agency.

Future research on the O-Plan architecture will concentrate on its ability to support a task assignment agent and the link between it and the planning agent. This is an area of planning research which has been neglected by planning researchers. However, it is an important aspect of the planning problem as a planner needs to fully understand the requirements set by the task assigner and needs the guidance which this can provide in identifying an appropriate solution. The planner also needs to provide feedback on plan feasibility to the Task Assigner. As part of this research three key issues have already been investigated. The key issues are the representation and communication of plans as sets of constraints, the use of quality criteria to analyse and direct the generative planner and the role of authority in coordinating the activities of a number of situated agents.

References

[1] Aarup, M., Arentoft, M.M., Parrod, Y., Stokes. I., Vadon, H. and Stader, J. *Optimum-AIV: A Knowledge-Based Planning and Scheduling System for Spacecraft AIV*, in Intelligent Scheduling (eds. Zweben, M. and Fox, M.S.), Published by Morgan Kaufmann Inc, San Francisco, USA, 1994.

[2] Allen, J., Hendler, J. and Tate, A., *Readings in Planning.* Published by Morgan-Kaufmann Inc, San Francisco, USA, 1990.

[3] Borgida, A., Greenspan, S. and Mylopoulos, J., Knowledge Representation as the Basis for Requirements Specifications, IEEE Computer Magazine, Special Issue on Requirements Engineering Environments, April 1985.

[4] Currie, K.W. and Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence* 52(1), pp. 49-86, North-Holland, 1991.

[5] Drabble, B. and Gil, Y. *Acquiring Criteria for Plan Quality Control*, in the proceedings of the AAAI Spring Symposium workshop on Integrated Planning Applications, June 1995. Stanford Univerisity, CA, USA. Published by the American Association for Artificial Intelligence, Menlo Park, California.

[6] Drabble, B. and Gil, Y. *Yes, but why is that plan better?*, submitted to the International Conference on Artificial Intelligence in the Petroleum Industry, 13th-15th September 1995, Lillehammer Hotel, Lillehammer, Norway.

[7] Gil, Y. *Knowledge Refinement in a Reflective Architecture*, in the proceedings of the Twelfth National Conference on Artificial Intelligence, Seattle, WA, USA. August 1994. Published by AAAI Press/ The MIT Press Menlo Park, CA, USA.

[8] Gil, Y., Hoffman, M. and Tate, A. *Domain Specific Criteria to Direct and Evaluate Planning Systems*, in proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, Tucson, February 1994. Published by Morgan Kaufmann, Published Inc, San Francisco, USA, 1994.

[9] Reece, G. A. and Tate A. *Synthesising Protection Monitors from Protection Intervals*, in the proceedings of the Second International Conference on Artificial Intelligence Planning Systems, (eds. K. Hammond), University of Chicago, June 13-15 1994, pp 146-152. AAAI Press, Menlo Park, California.

[10] Reece, G.A., Tate A., Brown, D.I., Hoffman, M. and Bernard, R.E. *The PRECiS Environment*, in the proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop at the National Conference on Artificial Intelligence (AAAI-93), Washington, DC, 1993.

[11] Ramesh, B. and Dhar, V., Representing and Maintaining Process Knowledge for Large-Scale Systems Development, IEEE Expert, pp. 54-59, April 1994.

[12] Tate, A, *Authority Management - Coordination between Task Assignment, Planning and Execution*, in the working papers of the IJCAI-93 workshop on Knowledge-based Production Planning, Scheduling and Control, August 1993.

[13] 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.

[14] 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.

[15] Tate, A., Drabble, B. and Dalton, J. Reasoning with Constraints in O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. Burstein. M.), Tucson, Arizona, USA, Morgan Kaufmann, 1994.

[16] Tate, A., Drabble, B. and Kirby, R.B., O-Plan2: an Open Architecture for Command, Planning and Control, in *Intelligent Scheduling* (eds. Zweben. M. and Fox. M.S.). Published by Morgan Kaufmann Inc, San Francisco, USA, 1994.

Appendix D – The Use of Condition Types to Restrict Search in an AI Planner

Austin Tate, Brian Drabble & Jeff Dalton

Appears in the Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, August 1994.

The Use of Condition Types to Restrict Search in an AI Planner

Austin Tate, Brian Drabble & Jeff Dalton

Abstract

Condition satisfaction in planning has received a great deal of experimental and formal attention. A "Truth Criterion" lies at the heart of many planners and is critical to their capabilities and performance. However, there has been little study of ways in which the search space of a planner incorporating such a Truth Criterion can be guided.

The aim of this document is to give a description of the use of condition "type" information to inform the search of an AI planner and to guide the production of answers by a planner's truth criterion algorithm. The authors aim to promote discussion on the merits or otherwise of using such domain-dependent condition type restrictions as a means to communicate valuable information from the domain writer to a general purpose domain-independent planner [1].

1 Introduction to Condition Typing

Research in AI planning has introduced a range of progressively more powerful techniques to address increasingly more realistic applications (Allen, Hendler & Tate 1990). A lesson learned in the expert systems and knowledge-based systems field is that it is important to make maximum use of domain knowledge where it is available in order to address many real problems. One powerful means of using domain knowledge to restrict and guide search in a planner is to recognise explicit precondition types, as introduced into Interplan (Tate 1975) and Nonlin (Tate 1977) and subsequently used in other systems such as Deviser (Vere 1981), SIPE–2 (Wilkins 1988) O-Plan (Currie & Tate 1991) and O-Plan2 (Tate, Drabble & Kirby 1994).

An explicit account of the *Goal Structure* or *teleology* of a plan can be kept in these systems. This records the causal relationships between actions in the plan and can show the intentions of the domain writer or planner in satisfying conditions on actions. In some circumstances, such domain knowledge can be used to prune the search of a planner. The information is provided to the planner via a planner's domain description language (e.g., Task Formalism – TF – in Nonlin and O-Plan). The domain writer takes the responsibility for a deliberate pruning of the search space or for providing preferences via condition types. This caused us to adopt the term *knowledge based planning* to describe our work.

Nonlin and O-Plan TF extends the notion of a precondition on an action and mates it with a "process" oriented view of action descriptions. A TF schema description specifies a method by which some higher level action can be performed (or higher level goal achieved). Each schema is thought of as provided by its own "manager". The schema introduces lower level actions under the direction of its manager and uses that manager's own resources. The schema may

[1] 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 project is monitored by Dr. Northrup Fowler III at Rome Laboratory.

say that some specific sub-action is included in order to set up for some later sub-action as part of the overall task. In TF, such internally satisfied requirements in actions are specified as **supervised** conditions. The "manager" also relies on other (normally external) agents to perform tasks that are their own responsibilities, but affect the ability of this manager to do the task. These are given as **unsupervised** conditions. There are other conditions which the "manager" may wish to impose on the applicability of particular solutions (e.g. don't try this method for house building if the building is over five stories tall). These are termed **holds** and **usewhen** conditions in different versions of Nonlin and are now called **only_use_if** conditions in O-Plan2.

Condition typing can be used to restrict search in a planner, but there is work to be done on how far this technique can be developed. It is often difficult for a domain writer to choose the correct type for a condition to most effectively restrict the search space while not over-indulging and throwing away plans which should be considered valid in the domain. Tool support to aid in the reliable modelling of large domains will undoubtably be needed. In practice, we have found that condition typing is an essential aspect of encoding realistic problems to an AI planner in order to reduce search spaces to a manageable level.

2 Other Related Work

The concept of providing explicit domain encoder input to guide planning has its roots in early research on the Planner language family. POPLER (Davies 1973) identified the search space implications of providing only a single type of "goal" which can either already be true or which can induce subgoaling to be made true. Interplan (Tate 1975) provided a simple facility to indicate that nominated conditions should not be sub-goaled upon. That is, that no method of *achieving* them should be introduced into the plan. Nonlin (Tate 1977) provided a comprehensive set of condition types as described earlier. These were used to restrict the options considered to satisfy a condition in the Nonlin "QA Algorithm". QA was a precursor to the Truth Criterion used in many planners which use a partial order plan representation and make use of Goal Structure or causal links to direct search. See (Tate 1993) for a historical perspective.

A more general condition satisfaction approach, not using such domain knowledge, is used in TWEAK based on Chapman's formalisation of the Modal Truth Criterion (MTC) (Chapman 1987). This approach does not address search control issues. Chapman's work provides a description of the search space, but not a specification of how to control or prune search in that space.

There has been little study of ways in which the search space of a planner incorporating a Truth Criterion can be guided. Drummond (Drummond 1993) argues that there has been too much concentration on planner aspects that deal with logically or syntactically complete condition achievement, and too little attention has been paid to other capabilities of practical planners such as Nonlin, SIPE-2 and O-Plan. These other capabilities include hierarchical expansion, a simple but effective resource allocation mechanism, and explicit languages to describe and allow for the protection of the a plan's causal structure (effects/conditions).

A number of researchers have pre-analysed operator information to guide search.

Collins and Pryor (Collins & Pryor 1993) provide the first critical analysis on the use of condition types intended to filter out options that would otherwise have to be considered by a planner. They conclude that in the majority of cases such filter conditions are misused and may not have the effect intended. Their arguments assume that changes to the set of operators available might invalidate domain modelling assumptions about the use of filters(true[2]), that most providers of systems employing condition types did not fully appreciate that use of filter conditions would restrict the search space (false), and that restricting the search space using such filter conditions is not useful due to the restrictions under which they correctly apply (false, their argument assumes that hierarchical modelling is not used properly in planning or that filter conditions can be "hierarchically promiscuous"[3] – they must not be). Although the Collins and Pryor critical analysis paper is flawed in making some of these assumptions, it is none-the-less a useful document in raising the issue of the validity or otherwise of utilising condition type information to restrict search in a planner and may start wider study and debate on whether such condition types are valid and useful. Unfortunately, the work takes too simplistic a view of how condition types (filter and otherwise) are already used in planning systems today.

It is hoped that the current paper goes some way towards providing an information base on which comment, study and analysis will be possible.

3 O-Plan2 Domain Description Language Task Formalism

TF is used by a domain encoder 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 them to a more detailed level) in the plan and including the relevant set of more detailed sub-activities described therein. Ordering constraints are then introduced 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 language. Other TF structures hold global information and heuristic information about preferences of choices to be made during planning.

4 O-Plan2 Triangle Model of Activity

O-Plan2 uses a hierarchical model of activity which gives emphasis to an owner's perspective of how an activity is performed and the environment in which it can be sanctioned, resourced and

[2]Tool support may help in avoiding such domain encoding errors.

[3]Hierarchical promiscuity occurs when a domain modeller confuses the levels at which effects are introduced and conditions are required. This is especially problematic for the ways in which AI planners typically handle filter conditions.

used. This is reflected in the "triangle" model of an activity (see Figure 1). The vertical dimension reflects activity decomposition, the horizontal dimension reflects time. Inputs and outputs are split into three principal categories (authority, conditions/effects and resources). Arbitrarily complex modelling is possible in all dimensions. "Types" are used to further differentiate the inputs and outputs and their semantics.

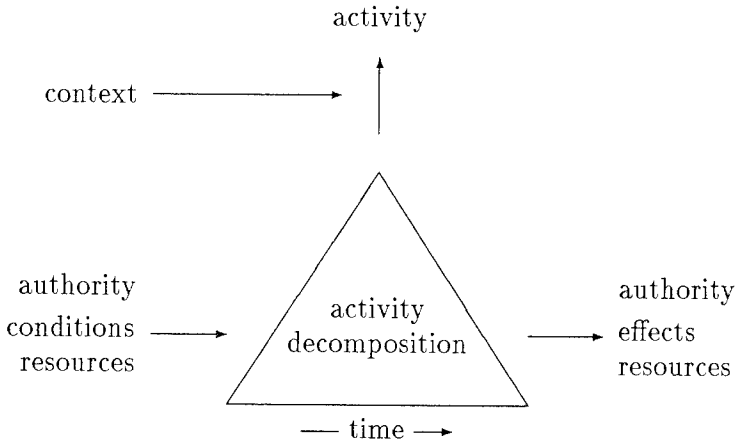


Figure 1: Triangle model of Activity

"Entry" to the model can be from any of three points in the triangle model. From the top vertex it is possible to ask for activity expansions or decompositions. From the right side of the triangle, it is possible to ask for activities satisfying or providing the output requirement (a desired effect or "goal", a required resource, or a needed authority). These two points are used mostly by our planners to date. The third point on the left side can reflect triggering conditions for an activity and will be needed when improved models of independent processes are used as in our Excalibur (Drabble 1993) extension to Nonlin. A "context" requirement permits use of each particular expansion or decomposition of an activity.

The triangle model of activity is a generalisation of process models used in many structured analysis and design techniques (SADT) such as IDEF, R-Charts, etc., and can be directly related to them.

5 O-Plan2 Condition Types

Condition typing allows relevant information to be kept about when, how and why a condition present in the plan has been satisfied and the way it is to be treated if the condition cannot be maintained. All condition statements appear in O-Plan2 Task Formalism action schemas. Conditions play a greater role in O-Plan than in previous planning systems since there is no *special* notion of *goal*. Nonlin (Tate 1977), NOAH (Sacerdoti 1977) and SIPE–2 (Wilkins 1988) style goal nodes in action expansions become simply **achieve** conditions in O-Plan. The **achieve** condition type is the only one on which sub-goaling is permitted.

Conditions are one of the most elaborate of all TF statements due to the variety of condition types identified as being needed for practical planning in O-Plan2. The "process" or "manager" view of hierarchical activity description used in Nonlin contributed the three basic condition types of **supervised**, **unsupervised** and **usewhen**. O-Plan research and applications experience identified the need to separate two different uses being made of the Nonlin **usewhen** condition type. This led to the introduction of **only_use_if** and **only_use_for_query**. A more flexible **achieve** condition definition was also required to remove temporal scope limitations on the ways in which earlier planners such as Nonlin could satisfy goals by adding new activities into a plan.

The O-Plan2 condition types are thus:

- **only_use_if** conditions provide an applicability check on the context in which a schema can be used. These are sometimes referred to as filter conditions.
- **only_use_for_query** conditions are used to make queries at a point in the plan to instantiate or restrict variables in a schema.
- **unsupervised** conditions must be satisfied at the required point, but it is assumed that, in circumstances in which the schema introducing such a condition is used, that the condition will have been satisfied elsewhere. Therefore, they act as a sequencing constraint.
- **supervised** conditions are satisfied directly within the schema containing them by the deliberate introduction of a suitable effect (or alternative effects) at an earlier point or by the direct inclusion of an action known to achieve the necessary effect (at some more detailed level in the action's decomposition). They may be used as a means to explicitly record a protection interval within the causal structure of a plan.
- **achieve** conditions can be satisfied by any means available to the planner including the addition of new actions into the plan.

Other condition types can be identified but the ones above have been found to be useful ways to extract knowledge from a domain writer in a communicable form that can be used to restrict search in an AI planner.

Condition typing helps direct the planning process, but it also requires that the domain encoder structures the hierarchy of the tasks or actions clearly. It forces checks to be made on processes or actions which should communicate with others – ensuring they actually do advertise their results through a common vocabulary.

6 O-Plan2 Plan Levels

Before describing condition types and their definitions, it is useful to describe how O-Plan2 uses hierarchical modelling levels in its operation.

Definition

Each action and effect is introduced at a single domain modelling level and higher level activities introduce activities and effects at the same or a lower level.

A plan level can be introduced for two distinct purposes:

1. For convenience of abstraction and aggregation.
2. To place an order on the commitments and constraints made during planning.

The numerical plan levels are assigned by the O-Plan2 TF compiler in quite an intuitive way. Level numbers increase as lower level, more detailed action and effect descriptions are given. However, there can be "loops" in the structure, such that some actions can expand recursively or may expand back to themselves via other schemas. In such cases, all the actions and effects in the "loops" are mapped to the same plan modelling level. The detailed way in which level numbers are assigned is as follows.

Each schema represents a way to perform the action indicated by its **expands** clause. The first word of the **expands** pattern is referred to as the *action name* of the schema. Each schema S links its action to a number of direct successor actions: the sub-actions listed in the schema's definition. These successor actions are normally at the next lower level (except when loops are involved). A further set of direct successors can be found by taking the action names of all schemas that have an **only_use_for_effects** that matches any **achieve** condition of S.

This will define a graph in which the action names are vertices and there is an edge from each action name to each of its directly reachable successors. The next step is to find the *strongly connected components* (SCCs) in this graph in order to build a new graph in which each SCC is treated as a unit. This new graph is acyclic and the level of an action can be found by taking the longest path to the SCC that contains it through this graph. This will also identify the level at which *effects* are introduced into the plan. The plan level mapper is sensitive to loops in the graph and the SCCs components represent such loops. Whenever you can get from A to B *and* from B to A in a directed graph, A and B are in the same SCC.

7 Condition Types for the Domain Writer

This section gives definitions of O-Plan2 condition types in terms of what information a domain writer providing a library of action or plan components can state, hopefully in an understandable way without knowledge of how the AI planner would go about using this in detail.

For each condition type used within O-Plan2 we provide below the following information:

- **Purpose**: This describes the condition type in domain terms for use by the domain encoder and describes the circumstances under which the condition type should be used.

- **Definition**: This describes the condition type in planner terms and describes in more detail how the planner goes about dealing with the condition type on behalf of the domain encoder.
- **Examples**: These clarify the use of each type.

7.1 Only_use_if

- **Purpose**: This is a filter condition on the applicability of a particular schema.
- **Definition**: It may be given on statements introduced as effects at a higher level or on the same modelling level as the schema introducing it.
- **Examples**: On static facts (those never refuted in the plan and referred to as **always** facts in O-Plan2):

```
only_use_if {type_of soil} = sandy
```

and on dynamic facts (whose value can change over time):

```
only_use_if {apportioned_force ?regiment} = unallocated
```

In the first example the condition would be used to allow a schema to be selected which was suitable for use if the soil type is sandy. In the second example, the condition would only allow the schema to be chosen if a particular force was available at this point in the plan. During the course of the plan the force's status may vary and with it the ability to use the schema.

7.2 Only_use_for_query

- **Purpose**: A query mechanism to establish current values for variables.
- **Definition**: It may be given on statements at a higher or on the same modelling level as the schema including it. There should *always* be an answer for such a query when it is evaluated at an appropriate level.
- **Examples**: On static facts:

```
only_use_for_query {country_of ?city} = ?country
```

and on dynamic facts:

```
only_use_for_query {position_of ?robot} = ?location
```

The first example would allow the country in which a city is located to be looked up. The second example allows the dynamic lookup of the position of the robot.

7.3 Unsupervised

- **Purpose**: Specifies a scheduling constraint on the schema which is (normally) satisfied externally. Exceptionally, it may also specify an internal ordering requirement within the schema making use of actions introduced for other reasons.
- **Definition**: It may be given on statements at the same or a higher modelling level if the condition is satisfied externally to the sub-actions of the schema or at the same or lower modelling level if the condition is satisfied from the sub-actions within the schema.
- **Example**:

```
unsupervised {status ground_buffer} = empty at 2
```

This would make a sub-action number 2 introduced by the schema occur after some other action in the plan which empties the ground_buffer.

7.4 Supervised

- **Purpose**: To protect an intended effect of some earlier sub-activity up to the point required.
- **Definition**: It may be given on statements at the same or on a lower modelling level as the schema including it.
- **Example**:

```
supervised {status ground_buffer} = full at 3 from 2
```

This would protect the ground_buffer as being full between the end of a schema sub-action number 2 (say where some data was captured into the ground_buffer) to the beginning of a later schema sub-action number 3 where the data might be used.

7.5 Achieve

- **Purpose**: To allow a condition to be satisfied by the optional inclusion of sub-activities.
- **Definition**: Specifies a requirement which the schema writer is optionally prepared to meet by adding new structure into the plan at the same or lower modelling level as the schema including the condition.
- **Example**:

```
achieve {in_position ?tool} = workbench at 1
```

This allows the required tool to be moved to the required place if it is not already there.

The following table summarises the ways in which each O-Plan2 condition type may be satisfied.

Condition	Levels Considered			Type
only_use_if	above	same		EXTERNAL
only_use_for_query	above	same		EXTERNAL
unsupervised	above	same		EXTERNAL or
		same	below	INTERNAL
supervised		same	below	INTERNAL
achieve		same	below	OPTIONAL

8 Condition Type Correspondence to Nonlin, SIPE-2 and ACT

Nonlin (Tate 1977) was the first Edinburgh planner to use the Task Formalism (TF) language and made use of **supervised**, **unsupervised** and **usewhen** condition types. It handled achievable conditions by including goal nodes in the action expansion.

The SIPE–2 planner (Wilkins 1988) also includes support for a number of condition types and is converging on similar types to those available in O-Plan2. A development of the SIPE–2 domain description language to link to work on the PRS (Procedural Reasoning System) reactive execution support system (Georgeff 1986) is now underway to create a shared domain description language called ACT. The following sections compare O-Plan2 condition type usage with the those used in Nonlin, SIPE–2 and ACT.

O-Plan2	Nonlin	SIPE-2	ACT
only_use_if	usewhen	precondition	precondition
only_use_for_query	none	none	setting
unsupervised	unsupervised	external	wait-until
supervised	supervised	protect-until	require-until
achieve at N	none	none	none
achieve after	goal	goal	achieve

only_use_if This is the same as Nonlin's **usewhen** (originally called **holds**). It is the same as a **precondition** in either SIPE–2 or ACT. The ***already** condition in later releases of Deviser (Vere 1981) performs the same function.

only_use_for_query In Nonlin and SIPE–2, such conditions were modelled as **usewhen** conditions or **preconditions** respectively and not treated separately. Nonlin or SIPE–2 treated a query condition in the same way as an **only_use_if** filter condition, except that it was assumed that variables would be bound by satisfying it. This incorrectly limits the range

of legitimate solutions. As in O-Plan2, ACT separates query type conditions for clarity – calling them the **setting**.

unsupervised Later releases of SIPE–2 allow an **external** condition to give this capability. In ACT, this is called **wait-until**.

supervised The same as a **protect-until** in SIPE–2 and **require-until** in ACT.

achieve at N This imposes no restriction on the temporal scope of any activity inserted in the plan to satisfy the condition. There is no equivalent in Nonlin, SIPE–2 or ACT all of which make such temporal scope restrictions and thus restrict the domains which can be modelled.

achieve at N after <time point> This is an **achieve** with a temporal restriction on the points within the plan from which a contributor may be chosen to satisfy the condition. This is a more general way to describe Nonlin, NOAH and SIPE–2 goal nodes which appear in the expansion/decomposition part of their operator schemas. For these systems, the <time point> is restricted to be after the start of the time range of the expansion of the schema containing the condition. In ACT this is called **achieve** and has the same fixed restriction on temporal scope as Nonlin, NOAH and SIPE–2.

9 Summary

In large realistic domains, we believe that significant domain knowledge must be made available to a planner in order to reduce search spaces to a manageable level. One important way in which this can be done effectively is to get a domain writer to provide information about a domain from which we can extract instructions to the planning system about how to satisfy and maintain conditions required in the plan.

Condition types can be a valuable aid to providing knowledge about a domain to a domain-independent planner. Condition typing can successfully restrict the search for a plan, but there is work to be done on how far this technique can be developed. It is often difficult for a domain writer to choose the correct type for a condition to most effectively restrict the search space while not over-indulging and throwing away plans which should be considered valid in the domain. Improved planning knowledge capture aids now under development may assist in this process.

One aim of this paper has been to seek to separate the domain writer oriented description of condition types from the mechanisms used by a planner to satisfy, maintain and re-satisfy conditions.

By making this information more widely available, the authors aim to promote discussion on the merits or otherwise of using such domain-dependent condition type restrictions as a means to communicate information from the domain writer to a general purpose domain-independent planner. The control of planner search via condition types is worthy of a serious study in its own right, and could form an ideal Ph.D. topic.

Acknowledgements

We are grateful for discussions on condition types with Drew McDermott, Nancy Lehrer, Glen Reece, Mark Drummond, Subbarao Kambhampati, Dan Weld, Craig Knoblock and Louise Pryor, as well as from the comments of the reviewers and conversations with other members of the AI planning community. Sorry we had to be ruthless to achieve the final page count. The work reported here has benefited from input by talented researchers who have participated in the O-Plan and related projects. Thanks to David Wilkins who helped in checking the comparison to SIPE–2 and ACT usage of condition types. Errors remain ours.

References

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

[2] Chapman, D. 1987. Planning for Conjunctive Goals, *Artificial Intelligence*, Vol. 32, pp 333-377.

[3] Collins, G. & Pryor, L. 1993. On the Misuse of Filter Conditions: A Critical Analysis, in *Current Trends in AI Planning*, (eds. Backström, C. & Sandewall, E.), IOS Press.

[4] Currie, K.W. & Tate, A. 1991. O-Plan: the Open Planning Architecture, *Artificial Intelligence* Vol 51, No. 1, pp 49-86, North-Holland.

[5] Davies, D.J.M. 1973. POPLER 1.5 Reference Manual, Theoretical Psychology Unit Report no.1., Department of Artificial Intelligence, University of Edinburgh.

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

[7] Drummond, M.E. 1993. On Precondition Achievement and the Computational Economics of Automatic Planning, in *Current Trends in AI Planning*, (eds. Backström, C. & Sandewall, E.), IOS Press.

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

[9] Sacerdoti, E. 1977. *A Structure for Plans and Behaviours*, Artificial Intelligence Series, North Holland.

[10] Tate, A. 1975. Using Goal Structure to Direct Search in a Problem Solver. Ph.D. Thesis, University of Edinburgh.

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

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

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

[14] Vere, S. 1981. Planning in Time: Windows and Durations for Activities and Goals, *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 5.

[15] Wilkins, D. 1988. *Practical Planning*, Morgan-Kaufmann.

Appendix E – The Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner

Brian Drabble and Austin Tate

Appears in the Proceedings of Second International Conference on AI Planning Systems (AIPS-94), AAAI Press, Chicago, USA, June 1994.

The Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner

Brian Drabble and Austin Tate

Abstract

Resource reasoning has been at the heart of many of the successful AI based scheduling systems – yet no attempt has been made to integrate the best techniques from scheduling with the best techniques from AI activity based planning. This paper presents a set of incremental algorithms which create two separate profiles to represent the optimistic and pessimistic use of resources within an activity plan. These allow the planner to ensure that there is a feasible assignment of resources available within any plan state being considered. The paper demonstrates how these profiles can be used to track the usage of a variety of different resource types and how they can be used to provide detailed and relevant information when a resource constraint conflict is detected.

1 Introduction

Resource reasoning has been at the heart of many of the successful AI based scheduling systems – yet no attempt has been made to integrate the best techniques from scheduling with the best techniques from AI activity based planning. The reason for wishing to reason about resources in an activity based planner is clear. One of the prime motivations for not considering a particular course of action is that you have insufficient resources with which to carry it out. These resources can vary from people, to money, to space in a car park. Resource reasoning provides a powerful way of pruning the search space and guiding the planner towards a successful plan.

Scheduling problems have tended to be dominated by complex resource contentions and relatively simple process plans whereas activity plans have tended to have complex process options with simple resource uses. However, this view of planning and scheduling as being two separate problems has been enforced by the different approaches of AI researchers and not by the nature of the problem itself. Activity based planning and resource based scheduling can be viewed as opposite ends of a continuum with the middle area being of particular interest in real world applications. This middle area of the continuum contains problems such as:

- 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 spacecraft such as Voyager and ERS-1.

Activity based planning systems have attempted to address some of the problems in reasoning about resources – NOAH [6], Nonlin+ [9],[12] and SIPE–2 [14] are examples, but they have had limited success. At the same time resource based schedulers such as OPIS [8], Micro-Boss [7]

and TOSCA [1] have attempted to use more complex process plans than those used in earlier scheduling systems. Again these attempts have met with limited success. There have also been attempts to handle a richer model of resources primarily within a constraint management framework such as in CAMPS [2, 13], AMPS [3, 5] and EMPRESS [4]. Figure 1 shows how these differing systems can be related in the maturity of their approaches to resource reasoning and handling process plans.

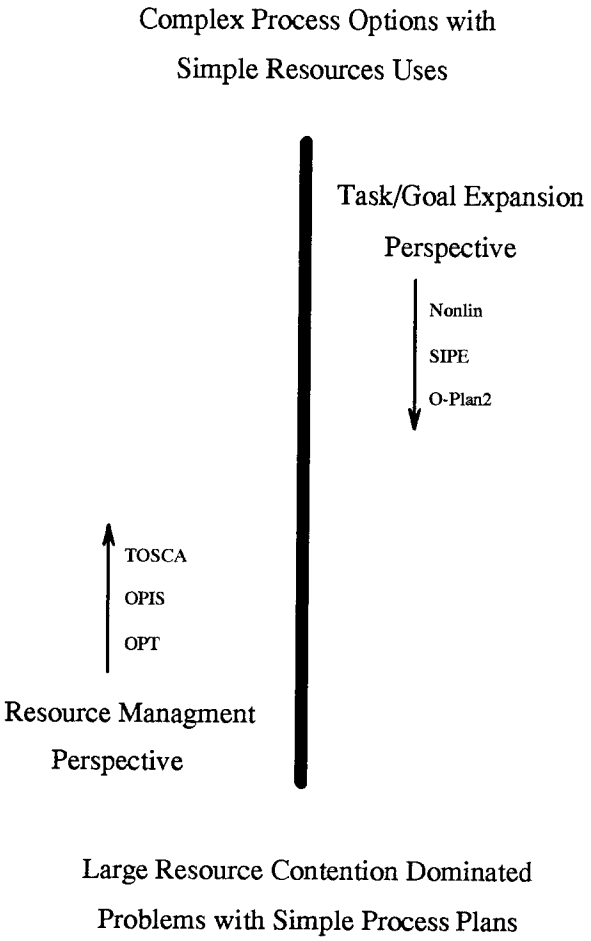


Figure 1: The Continuum of Planning and Scheduling Systems

This paper describes an approach to resource reasoning which takes the idea of a rich resource model as developed in AI based scheduling systems and presents a series of incremental algorithms which allow such a resource model to be used in an activity based planner framework. The techniques described in this paper are currently being integrated into the Resource Utilisation Manager (RUM) of the O-Plan2 planner [11].

O-Plan2 is aimed to be relevant to the types of problems which were outlined above. O-Plan2 uses a number of *Constraint Managers* to maintain information about a plan while it is being generated [10]. The information can then be used to prune search (where plans are found to be invalid as a result of propagating the constraints managed by these managers) or to order

search alternatives according to some heuristic priority. Constraint Managers are intended to provide efficient support to a higher level of the planner where decisions are taken. They do not take any decision themselves. They are intended to provide complete information about the constraints they are managing or to respond to questions being asked of them by the decision making level.

2 Resource Management in an Activity Planner

Resource constraint management within the O-Plan2 system is carried out by a Resource Utilisation Manager (RUM). It is the function of the RUM to check on the levels of resources being used at certain points in the plan. The RUM is informed of resources level changes in a plan by means of Resource Utilisation Entries (RUE's). A RUE can change resource levels in one of five different ways:

1. **Set** a resource level to be a particular value (or within a particular range) For example, top up a fuel tank to its maximum capacity.
2. **Allocate** a certain amount of resource i.e. reduce the amount of resource remaining as available from that point within the plan. Semantically, an allocation must be paired with a subsequent deallocation.
3. **Deallocate** a certain amount of resource back to the resource pool, i.e. increase the amount of resource available from that point in the plan.
4. **Consume** a certain amount of resource.
5. **Produce** a certain amount of "new" resource.

The initial declaration of resource types (e.g. fuel, food, money, plumbers, etc.) is accomplished by using a `resource_type` definition in the O-Plan2 domain description language (Task Formalism – TF), together with the information required to define that resource. For example,

```
types
    fuel_loc = (port1 port2 port3 ship1),
    fuel_storage = (tank1 tank2 tank3 tank4 tank5),
    prov_type = (frozen chilled fresh),
    prov_loc = (port1 port2 port3 port4),
    prov_storage = (warehouse1 warehouse2 warehouse3);

resource_types
    consumable_producible_by_agent
                {resource fuel ?{type fuel_loc}
                               ?{type fuel_storage}} = gallons,
    consumable_producible_by_agent
                {resource provisions ?{type prov_type}
                                     ?{type prov_loc}
                                     ?{type prov_storage}} = kilos;
```

The actual usage and setting of resource levels in the plan is achieved by RUE's which are derived from **resource** statements in TF action schemas. These provide changes to resources levels i.e. increments (**produces, deallocates**), decrements (**consumes, allocates**) and **sets.**

The RUM maintains resource usage profiles that reflect the changes of resource levels indicated by the RUEs. There can be uncertainty in two dimensions: in the actual level of resource changes and in the time at which such a change occurs. The RUM manages resource usage profiles in order to provide the following functionality for the planner:

- **Adding a new resource utilisation into a resource profile**
  As actions are expanded in the plan new resource utilisations will need to be added to the resource profile. The RUM will need to constrain the resources affected and monitor for resource violations.

- **Modifying an existing resource utilisation entry**
  Existing resource entries will be modified during the plan as their time and resource windows are constrained by other activities. The RUM will propagate the effects of these changes through only those resource entries affected.

- **Providing feedback when a constraint violation occurs**
  The RUM is able to provide specific advice relevant to the particular problem which has arisen. By using the type of a resource to restrict the options proposed, the RUM can suggest altering the resource levels in other related resource entries and/or modifying the time constraint of related resource entries.

3 Management of Resource Specification and Aggregate Resource Usage

Resource information in O-Plan2 action schemas is used to to restrict search and to ensure that resource usage in a plan stays within the bounds indicated. There are two types of resource statements. One gives a *specification* of the **overall** limitation on resource usage for a schema (over the total time that the schema's expansion can span). The other type of statement describes actual resource *utilisation* **at** points in the expansion of a schema. It must be possible (within the flexibility admitted by the actual resource utilisation statements) for a point in the range of the aggregate of the resource utilisation statements to be within the overall resource specification given.

1. the **specification** of the limits on resources used within a schema and all its possible expansions. For example, a schema to move a ship from one port to another may specify that it may consume between 100 and 1000 gallons of fuel depending on which ship and which pair of a specified set of ports is chosen.

```
resource consumes (resource fuel) = 100:1000 gallons overall;
```

2. the **utilisation** of resources on a particular action or at a particular time point within a schema. In this example ship1 receives 5000 gallons of fuel into its single fuel tank at the end of action 5 from tank1 at port1.

```
resource produces (resource fuel ship1 tank1) = 5000 gallons
                          at end_of node-5,
          consumes (resource fuel port1 tank1) = 5000 gallons
                          at end_of node-5;
```

All resource specifications and utilisations are maintained as **min/max** pairs, specifying the upper and lower bounds known at the time. Resource declarations which describe resource specifications and utilisation statements (perhaps still only partially specified) are held in the plan being developed by O-Plan2. The current best numerical bounds on resource utilisation statements are also converted to RUE's which are stored in a Resource Utilisation Table (RUT) with (notionally) one table per specific resource available. The entries of the RUT are held in ascending time point order. The following table (Table 1) is a fragment from a RUT for a specific fuel tank.

The entries within the RUT are fully qualified entries and as such represent actual resource utilisation. A schema which states that it produces 500 gallons of fuel from port1 is viewed as a *specification* as the actual change in a resource cannot be specified relative to a port – but only for a specific fuel tank at a location (a port or a ship).

4 Optimistic and Pessimistic Resource Profile Management

The algorithm used to track resource levels uses two distinct measures:

1. **Optimistic Resource Profile** (ORP)
   This describes the maximum resource that could be available with optimistic assumptions and is calculated from:
   (a) the **set** resource statements
   (b) the minimal resource usage at the maximum time value of a time point for an RUE with negative influences, i.e. **allocates**, **consumes**.

No	Type	Resource	Quantity	Time Point	Min	Max
1	*	(resource fuel port1 tank1)	20:20	tp1	0	0
2	-	(resource fuel port1 tank1)	20:30	tp2	4	8
3	+	(resource fuel port1 tank1)	15:15	tp19	6	6
4	*	(resource fuel port1 tank1)	10:15	tp36	7	7

Table 1: Example Resource Utilisation Table

(c) the maximal resource usage at the minimum time value of a time point of an RUE with positive influences, i.e. **deallocates**, **produces**.

For example, if action 1 **allocates** between 20 and 30 resource units between time 4 and time 8 then the ORP normally decreases by 20 at time 8 (unless a **set** is given at the same time point).

2. **Pessimistic Resource Profile** (PRP)
This describes the minimum resource that would be available with pessimistic assumptions and is calculated from:

   (a) the **set** resource statements

   (b) the maximal resource usage at the minimum time value of a time point of an RUE with negative influences, i.e. **allocates**, **consumes**.

   (c) the minimal resource usage at the maximum time value of a time point of an RUE with positive influences, i.e. **deallocates**, **produces**.

   For the above ORP example, the PRP normally decreases by 30 units at time 4 (again unless a **set** is given)

By calculating the changes in anticipated resource levels at specified points along a time line, a profile can be generated for the ORP and the PRP. Using the first three entries of the RUT described in Table 1, the following graph (Figure 2) of resource levels against time can be generated.

To generate the profile the RUM needs to keep track of various pieces of information and to be sensitive to the type of change which is being carried out for each RUE in the RUT. The changes which the RUM must deal with are the addition of a new RUE or the modification of an existing RUE. The information which is maintained is as follows:

1. Optimistic Increment (**OptInc**) which is defined as the incremental change in the level of resource at a particular time point ignoring **set**s. It is calculated from summing:

   (a) If the time point in question is the maximum time point of an **allocates** or **consumes** then add in the minimum change in resource for each record

   (b) If the time point in question is the minimum time point of a **deallocates** or **produces** then add in the maximum change in resource for each record.

2. Pessimistic Increment (**PesInc**) which is defined as the incremental change in the level of resource at a time point ignoring **set**s. It is calculated from summing:

   (a) If the time point in question is the maximum time point of a **deallocates** or **produces** then add in the minimum change in resource for each record

   (b) If the time point in question is the minimum time point of an **allocates** or **consumes** then add in the maximum change in resource for each record.

3. A *base* value of PRP to assist in incrementally computing the actual PRP.

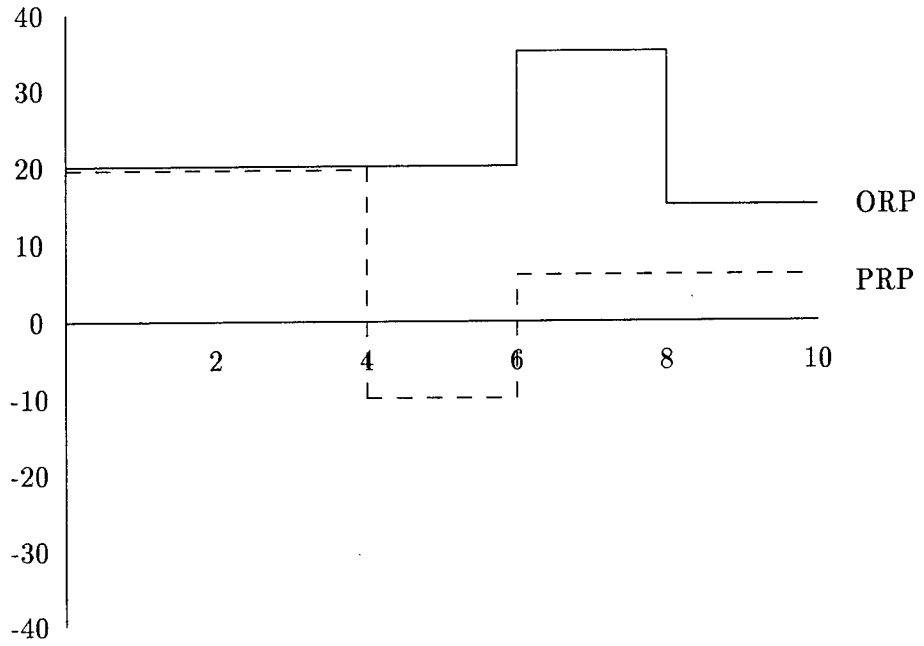


Figure 2: Optimistic and Pessimistic Profiles of Resource Utilisation

4. A *base* value of ORP to assist in incrementally computing the actual ORP.
5. Whether a **set** is involved as one of the RUE's at the time point.
6. Dependency records containing lists of RUE's affected by resource information at time point.

Formulae to Maintain PRP and ORP

The *base* value for PRP and ORP and the ORP and PRP themselves can be calculated for any time point using the following formulae:

```
FORMULA 1 - PESSIMISTIC PROFILE:

      IF one or more set entries are present THEN
         IF there are over lapping sets THEN
            PRP = minimum of all overlapping sets
         ELSE
             IF there are overlapping deallocates or produces then
                base_PRP = base_PRP + the maximum resource value for all
                           overlapping deallocate or produce RUE's
             ELSE
```

```
            base_PRP = minimum of the minimum of the set value
         ENDIF
      ENDIF
      PRP = base_PRP
   ELSE
      base_PRP = PRP at a previous time point in the RUT or
                 0 if none available
      PRP = base_PRP + PesInc
   ENDIF

FORMULA 2 - OPTIMISTIC PROFILE:

   IF one or more set entries are present THEN
      IF there are overlapping sets THEN
         ORP = maximum of all overlapping sets
      ELSE
          IF there are overlapping allocates or consumes then
             base_ORP = base_ORP + the maximum resource value for all
                        overlapping allocate or consume RUE's
          ELSE
             base_ORP = maximum of the maximum set value
          ENDIF
      ENDIF
      ORP = base_ORP
   ELSE
         base_ORP = ORP at a previous time point in the RUT or
                    0 if none available
         ORP = base_ORP + OptInc
   ENDIF
```

Example when a Resource "Set" is Involved

The above formulae will now be demonstrated on an example in which there are positive or negative resource changes which may occur within the time range of a `set` (e.g. see Figure 3).

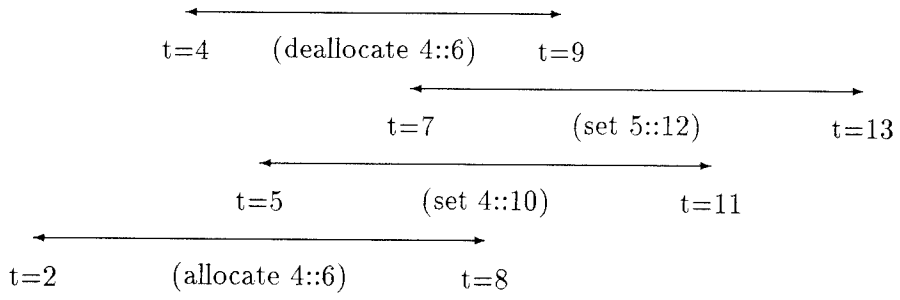


Figure 3: Set Spanning a Resource Change

In this example the `allocation` of the resource will take place some time between `t=2` and `t=8` and will allocate between 4 and 6 units. However, there are two `sets` which may occur in parallel with this `allocation`. This is further complicated for illustrative purposes in this example by including a `deallocate`. By using the algorithms described above it is possible for the RUM to construct the PRP and ORP profiles and to find that there is at least some possible allocation of resources which is valid.

5 Detection of Resource Utilisation Failures

The failure of the addition of a RUE or during propagation of RUE entries represents an attempt by the plan to use more of the resource than there is available. The failures types which have been identified so far are as follows:

1. **ORP less than ZERO** This failure means the even with that most optimistic assumptions there is insufficient resource available.
2. **ORP less than PRP** This failure means the resource utilisation has been declared incorrectly within the domain description (TF) definitions.

The RUM informs the planner of the RUE which has a fault and the possible tactics available to resolve the conflict. These are as follows:

1. increase the earliest start time of a failing action i.e. start it later.
2. alter the lastest finish or earliest start time of possible actions which contribute to the problem. This will depend on whether the actions are taking or giving back a resource. It may make more sense to give some resource back earlier or take a resource later (if time constraints allows) rather than reduce your own resource utilisation.
3. increase the maximum resource level available at a point by adding a `set` of a particular resource. For example, if the authority can be found an extra shift of workers or an extension to the working day may resolve the resource problem.

The actual tactics proposed are sensitive to the resource type for the RUE involved.

6 Summary

This paper has described a mechanism for the incremental management of optimistic and pessimistic resource usage profiles in an activity planning framework. A rich resource model can be handled which can manage uncertainty in the time at which resources are used and the absolute resource levels involved in any resource level change.

The technique allows for an AI planner to check the feasibility of resource availability for plans being considered in the search for a solution. The techniques allows for the maintenance of resource usage profiles within which a specific resource allocation should be possible.

Acknowledgements

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. The project is monitored by Dr. Northrup Fowler III at the USAF Rome Laboratory.

All views expressed are those of the authors only. The information contained in the paper has benefited from discussions with many talented researchers who have worked on the O-Plan and related projects. Thanks are due for productive discussions to our co-workers on the O-Plan2 project: Jeff Dalton and Glen Reece.

References

[1] Beck, H. *TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints*, in Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, Amsterdam, 12-14 May 1993, (eds, C. Kooij and P.A. MacConaill and J. Bastos), pp138-149.

[2] Brown, R., *Knowledge-based Scheduling and Resource Allocation in the CAMPS Architecture*, in Proceedings from the International Conference on expert Systems and the Leading Edge in Production Planning and Control (ed.. M. Oliff.), Benjamin/Cummings, Menlo Park, CA 1987.

[3] Dawson, B., Day, D.S. and Mulvehill, A., *The AMPS Final Report*, Final Report, USAF Rome Laboratory Technical Report RADC-TR-90-131. July 1990.

[4] Hankins, G.B., Jordan, J.W., Katz, J.L., Mulvehill, A.M., Dumoulin, J.N. and Ragusa, J., *EMPRESS: Expert Mission Planning and REplanning Scheduling System*, in Expert Systems in Government Symposium, 1985.

[5] Mulvehill, A., *CAMPS/AMPS FY88 Year End Report*, The MITRE Corporation, Technical Report M89-28, May 1989.

[6] Sacerdoti, E., *A Structure for Plans and Behaviours*, Artificial Intelligence Series, North Holland, 1977.

[7] Sadeh, N., *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling*, Ph.D., CMU-CS-91-102, School of Computer Science, Carnegie Mellon University, 1991,

[8] Smith, S.F., *A Constraint-Based Framework for Reactive Management of Factory Schedules*, in Proceedings International Conference of Expert Systems and the Leading Edge in Production Planning and Control, Charleston, South Carolina, May, 1987

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

[10] Tate, A., *The Emergence of "Standard" Planning and Scheduling System Components*, in Proceedings of the Second European Workshop on Planning (EWSP-93), IOS Press, Sweden.

[11] 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.

[12] Tate, A. & Whiter, A., *Planning with Multiple Resource Constraints and an Application to a Naval Planning Problem*, in Proceedings of the First Conference on Artificial Intelligence Applications, pp. 410-416, AAAI, Denver, Colorado, USA, December 1984.

[13] Zweben, M., *CAMPS: A Dynamic Replanning System*, The MITRE Corporation, Technical Report M87-50, December 1986.

[14] Wilkins, D., *Practical Planning*, Morgan Kaufmann, 1988.

Appendix F – Authority Management - Coordination between Task Assignment, Planning and Execution

Austin Tate

Appears in the Papers of the Workshop on Knowledge-based Production Planning, Scheduling and Control at the International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France, 1993.

Authority Management - Coordination between Task Assignment, Planning and Execution

Austin Tate

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.

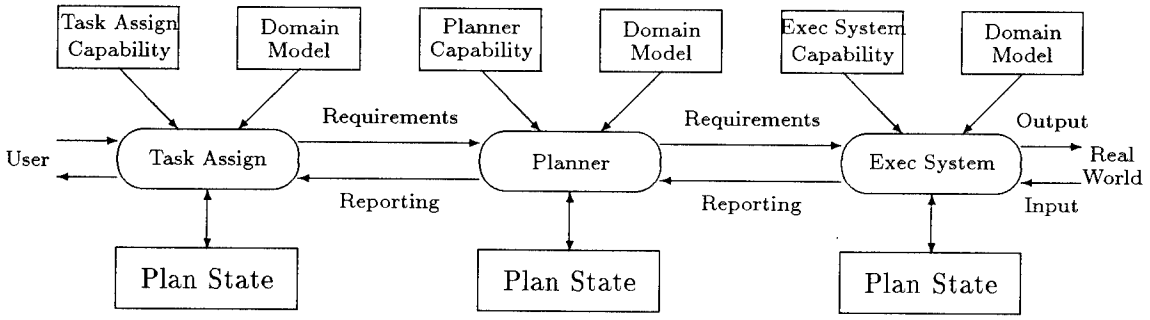


Figure 1: Communication between Task Assignment, Planning and Execution Agents

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.

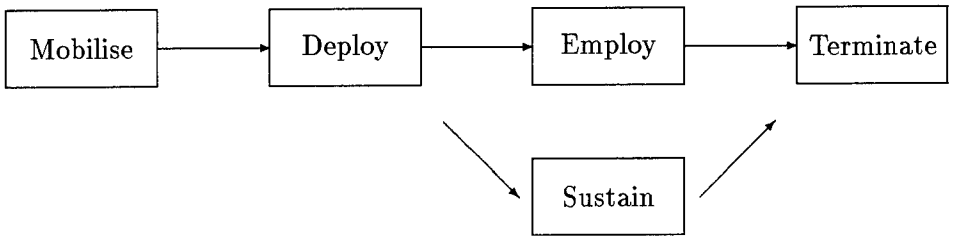
This simple evacuation operation shows the development of a plan for Non-combatant Evacuation Operation (NEO) from a hypothetical island named Pacifica. Though this scenario is completely fictious, the objectives, issues addressed, and underlying data are intended to be sufficiently realistic for the 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.

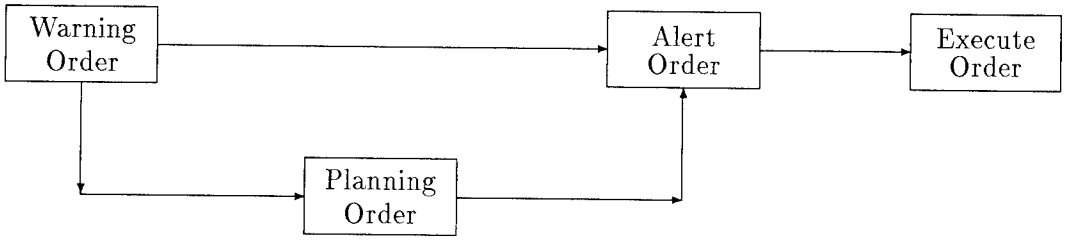
There are two levels of plan which can be stored and retrieved within the military planning community and are commonly referred to:

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

[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.

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.

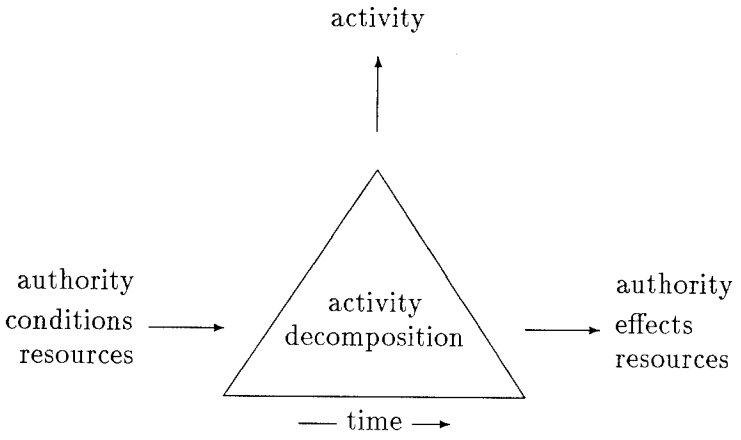
Changes of authority are possible via Task Assignment agent communication to the Planner agent. This is in the context of a current plan option and task provided previously. No TF for changing authorities is envisaged at this stage.

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

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

(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]).

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.*

Appendix G – Domain-Specific Criteria to Direct and Evaluate Planning Systems

Yolanda Gil, Mark Hoffman and Austin Tate

Appears in the Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. Burstein. M.), Tucson Arizona, USA, Morgan Kaufmann, 1994.

Domain-Specific Criteria to Direct and Evaluate Planning Systems

Yolanda Gil, Mark Hoffman and Austin Tate

Abstract

This document is the result of a joint effort to understand what are relevant factors to consider when there are several possible courses of action (COAs) to accomplish a Non-combatant Evacuation Operation (NEO) military mission. These relevant factors are useful for generation and evaluation of COAs and provide the basis for a good decision in selecting a COA. The document compiles the relevant factors from the perspective of logistics that are useful to evaluate whether or not alternative proposed COAs can be supported logistically, and which ones seem to be better alternatives compared to the others. The ultimate goal of this joint effort is to use these factors to automate the evaluation and comparison of COAs and use the comparison to determine what are critical aspects of a COA that may be changed to produce a better option with a generative planner. We discuss how we envision using EXPECT and O-Plan2 for this purpose.

1 Introduction

Generating qualitatively different plans is crucial in decision-making support systems within the Planning Initiative. Current planners are tasked such that all the alternative COAs generated are pretty much produced under some fixed patterns. Typical patterns are to produce one COA that uses many resources but can be deployed very fast, another that uses less resources and the deployment takes longer, another is somewhere in the middle, and another is a bit more extreme. Generating qualitatively different plans would allow more variety and better quality solutions.

What we foresee as the framework is that an outer "strategic/task assignment" layer of the system performs some task analysis and sets direction. This would be used to set up definite targets and constraints for the "tactical" planner to flesh out. The tactics planner would thus establish that a plan was possible within the framework specified (keeping certain elements of evaluation at favorable levels). The planner would be tasked with different such requirements to produce alternate plans which are qualitatively different.

The intent of this document is to add to the PRECiS domain description [5] such that together they provide a rich domain example that is simple enough for enabling technology research, but also that can be realistically evaluated and recognized as addressing real issues.

This document attempts to clarify the following issues:

1. Clear separation of *task assignment* and scoping of a request to a *tactical* planner. Why these differ and how it helps to clearly separate the two.
2. Need for criteria against which plans will be evaluated. Idea that the *same* criteria can be used to *direct* the planner from the task assigner and can also be used to *evaluate* alternatives produced.

Our main goals are the following:

- To understand how domain criteria will be used to evaluate a plan *however it was produced - manually, automatically or with mixed initiative.*
- To relate each of these *domain criteria* to *plan features* in order to ensure that these plan features can be reasoned about by future planners.
- To give feedback to plan representation design efforts, to indicate which parts of the KRSL plan representation should be the primary targets for our work as being most relevant to domain issues of concern.
- To design an evaluation function to rate plan alternatives which will guide alternatives selection, such that the planner is using the *same* knowledge in choice making that will be used to rate COA options by the higher level analysis and direction people.
- To influence planner design and features to ensure that support is available to generate plans with the desirable domain features required.

This document runs as follows. After laying some background on the purpose of COA evaluations, the paper shows the evaluation factors relevant for NEO operations. We then describe in detail how to evaluate relevant factors from a logistics perspective. Finally, we discuss how the O-Plan2 and EXPECT systems can cooperate in the generation and evaluation of alternative courses of action. The paper includes an appendix with a concrete example of how tentative COAs are described, evaluated, and compared.

2 Background

During the concept development phase of a plan, it is crucial to develop careful estimates of the situation and the alternative courses of action. This analysis can help in making certain that:

a) a broad spectrum of possible courses of action is considered;

b) the uncertainties in each COA are analyzed and estimated to reduce unknowns;

c) the analysis can be used as the basis for a commander's estimate and subsequent selection of the appropriate options.

The concept development phase is composed of the following steps [6]:

1. *Mission Analysis.* The CINC analyzes the mission and the assigned task. The result is a *mission statement* that contains the tasks to be accomplished and the purpose they achieve. These tasks are described by who/what/when/where/why/how.

2. *Planning Guidance.* The supported commander produces a *planning directive*, that contains several tentative courses of action and other information that is used as initial guidance for the analyses. Each tentative COA is described a sa series of elements composed of who/when/what/where.

3. *Staff Estimates.* The six staff divisions use the planning directive to analyze the situation, each one from a different perspective. J-1 is concerned with personnel, J-2 with intelligence, J-3 with operations, J-4 with logistics, J-5 with transportation, and J-6 is concerned with C^3. The result of this analysis is a more refined description of each tentative COA, as well as staff estimates of relevant factors.

4. *Commander's Estimate.* A commander's estimate that summarizes the staff estimates is put together that is the basis to select one of the tentative COAs.

5. *Concept of Operations.* Produce an OPLAN (operation plan) that fully develops the CINC's concept of operations and includes time-phased force and deployment data (TPFDD).

The preparation of the staff estimates and the commander's estimate may be the most critical and time consuming task of task-sensitive planning operations. This is currently done by human planners, and our goal is to contribute to the automation (or partial automation) of this process.

Another important problem is that the generation of alternative courses of action cannot be fine-tuned because of time constraints. Courses of action turn out to be one of three types [7]:

1. conservative, using few forces,

2. use massive forces,

3. take little force with the hope that the operation will succeed anyway.

These three types are too gross grain and lie on stereotypical positions of the spectrum of possible alternatives. There are many tradeoffs that should be considered. For example, using a large force is a trivial way to make an operation succeed. However, such COA is considered unacceptable because it is too expensive. The goal is to use the minimum amount of force sufficient to hold the operation and of acceptable cost. If we increase automation during this phase, more satisfactory COAs will be produced.

3 Evaluation Factors for NEO Operations

In the staff estimates process, 23 of the 39 JOPES identified elements of evaluation (EEs) [4] are applicable to most NEO operations and should therefore be considered in the identification and recommendation of a NEO COA. Of these factors, many will remain constant across all COAs and are usually not addressed. Of those that differ, a few are identified as *critical factors* and are thus instrumental in the nomination of the recommended COA.

The 23 EEs are:

1. *Agreements and treaties*

   Do we have overflight rights and freedom of navigation for all lines of communications?

   Do we have basing rights for all staging bases, intermediate locations, and safe havens?

   Do we have all necessary host nation support at each location?

   Would we be violating any treaties with any country involved while conducting the proposed activities?

2. *Airfields and air facilities*

   Are the airfields close to the evacuation areas?

   Are the airfields capable of supporting the proposed evacuation aircraft types?

   Are the airfields capable of supporting the proposed aircraft quantities?

   Are there enough of the right types of staff available (refuelers, air traffic control, maintenance, etc.)?

   Do the airfields have facilities for refueling (only if necessary) or do we need to bring it in?

   Are the airfields capable of providing the equipment necessary to support aircraft operations (radios, radar, etc.)?

   Do the airfields have maintenance facilities (hardstands, hangers, etc.) if maintenance is going to be needed there?

3. *Allied and friendly cooperation*

   Is this a joint operation? If so, have tasks/missions been allocated?

   Do we have the political backing of our friends and allies for this operation?

4. *American firms overseas*

   Are there firms that will require staff and essential records/equipment evacuation?

5. *Ammunition*

   Do we have access to sufficient quantities?

   Do we have access to sufficient types?

   Can we acquire the ammunition in a timely manner to support operations?

   Are we prepared for contingencies with respect to needed ammunition?

6. *Communications*

   Will the Host Nation communications be sufficient (phones)?

   Do we need secure communications? If so, can we provide it?

7. *Concept of operations*

   Is the concept of operations in accordance with all guidance and constraints currently supplied?

Is the concept robust (no/minimal single point failure)?

Is the concept flexible (is this option able to adapt to worsening / improving conditions)?

Are the success, termination, and transition criteria well defined?

8. *Effects of US response*

   Will there be repercussions based on our response (sanctions, diplomatic relations, etc.)?

   Will the American people support the operation?

9. *Environment, weather, and oceanography*

   Can critical portions of the operation be done at night?

   Will weather potentially hamper / delay our operation?

   Can the weather be used to hamper / delay enemy activities / reaction?

   Do the tides negatively affect the operation?

10. *Facilities (US and allied)*

    Are allied and US facilities sufficient to support operations?

    Intermediate locations: food, water, shelter, safety?

    Safe Havens: food, water, shelter, hospital, political, onward transportation?

11. *Facilities (enemy)*

    Are enemy facilities a "center of gravity" for their operations? Can they be disabled?

    Can enemy facilities be captured / utilized for our benefit?

12. *Forces (US and allied)*

    Are the forces trained for this type of operation?

    Are there sufficient forces to offset anticipated and contingency enemy reactions?

    Can the forces be in position in the timeframe identified?

    Do the forces have sufficient equipment?

    Can we accomplish the mission with a "minimum footprint" (minimal troops, destruction, minimum area, etc.)?

13. *Forces (enemy)*

    Can enemy forces be countered during the operation to minimize their impact, especially loss-of-life?

14. *Geography and terrain*

    Are the friendly forces trained to support operations in this type of area and terrain?

    Does the terrain / geography inhibit / facilitate the operation?

    Are beaches accessible as transportation alternative?

15. *Legal authorities*

    Would we be violating any local or international laws or treaties in conducting these operations?

    Will we be coordinating with local peacekeeping authorities?

16. *Maps and chart availability*

    Do we have sufficient information about the local geography and topology?

17. *Medical services*

    Sufficient (in both quantity and type) medical facilities must be provided both en-route and at each safe haven.

    Medical units must be available at each of the evacuation centers in country.

18. *Non-combatant personnel*

    Accommodations (both transportation, food, and lodging) must be made available for all evacuees including both US and other friendly nationals evacuated by US.

19. *Operational comparison (US and adversary)*

    What activities might the enemy undertake to undermine our operation?

    How susceptible is our operation to enemy activities?

20. *Reconnaissance reporting*

    Can we get assessments of enemy activities for this operation?

    Can we get information regarding the agencies, facilities, and resources involved and updates on that status over the course of the operation?

21. *Rules of engagement (ROE)*

    Will the operation be able to be conducted within the specified rules of engagement?

22. *Seaports and port facilities*

    Are the seaports close to the evacuation areas?

    Are the seaports capable of supporting the proposed evacuation ship types?

    Are the seaports capable of supporting the proposed ship quantities?

    Are there enough of the right types of staff available (refuelers, sea traffic control, maintenance, etc.) if necessary?

    Do the docks have facilities for refueling (only if necessary) or do we need to bring it in?

    Are the docks capable of providing the equipment necessary to support ship operations (radios, etc.)?

23. *Transportation (local)*

    Is sufficient local transportation available for transport to assembly areas?

Can transportation be rented or purchased locally as opposed to provided by the evacuation forces?

Are the routes susceptible to enemy intervention?

Can the local lines of communications be protected during use?

The remaining 16 are normally not a consideration during NEO operations but are included here for completeness:

1. *Construction*
2. *Critical Assets*
3. *Emergency Response Elements*
4. *Intelligence Collection Assets*
5. *Intelligence Collection Priorities*
6. *LERTCON Actions*
7. *Manpower*
8. *Mobilization (Forces)*
9. *Mobilization (Industrial Base)*
10. *National/Regional Interests and Objectives*
11. *Nuclear Weapons Accounting*
12. *Political, Economic, and Social Factors*
13. *Petrol and Lubrication (POL)*
14. *Security Assistance/Military Aid Programs*
15. *Sustainment*
16. *World Reaction*

4 Relevant Logistics Factors for COA Evaluation

As we described before, each staff division produces evaluations of COAs that take into account the factors relevant to that division. For example, the logistics directorate (J-4) is concerned with ensuring effective logistic support for all forces, including transportation, supply, and maintenance issues. This section describes relevant factors to evaluate COAs from a logistics perspective in more detail than the previous section. The main factors from a logistics perspective are the following five:

A-PORTS (*Airports*) — For each airport mentioned in COA, two aspects are evaluated: (1) number of sorties/day, and (2) the number of square feet of aircraft parking.

S-PORTS (*Seaports*) — For each seaport mentioned in COA, the aspects considered are: (1) number of piers, (2) number of berths, (3) the max size of vessels allowed in the seaport (in feet), and (4) number of oil facilities or POLs (petrol and lubrication.)

LOG PER (*Logistics Personnel*) — The number of people needed to support the operation. Support personnel includes unloading personnel, stevedores, and military police.

Closure Date (*Earliest deployment closure allowed by COA*) — This is also known as the COA *closure date*, and is given as an offset from D-day (D+X).

LOCs (*Lines of Communication*) — This factor evaluates the operation in terms of how the different force modules involved will be able to communicate when they are physically distributed in different locations. It is usually qualified as good, ok, or bad.

Other factors considered include resupply capability of airports and seaports in terms of storage and refrigeration, pre-positioned war reserve material stock, covered storage areas, logistics command and control, host nation support in terms of resources allocated by host country for the operation, medical services, the logistic over the shore, whether ships are stacked up at the seaports waiting to be unloaded, onward movement coordination, oil facilities gained, who is in charge of C2, whether forces must move to other locations, topography, C3 physical protection, climate and weather, and enemy C3CM.

4.1 Estimating the Value of Relevant Factors for COA Evaluation

The value of most factors is estimated using back of the envelope calculations. The estimates for the five logistics factors being considered are calculated as follows:

A-PORTS: For all the airports mentioned in COA, add

- number of sorties/day allocated to the operation by the host nation.
- aircraft parking space available.

S-PORTS: For all the seaports mentioned in COA, add

- number of piers in the seaport.
- number of cargo berths.
- maximum size of vessels allowed by the seaports of the COA. This is calculated by taking the maximum length of the types of cargo berths available in all the seaports.

Closure Date: Maximum of airlift and sealift closure times. The procedure to calculate the airlift closure is described in detail in [8].

LOG PER: The logistics personnel needed is a function of the size of the personnel involved in the operation. It can be estimated as a percentage of the people who compose the force modules

involved in the COA. First, the total amount of troops to be moved is calculated. These troops come from any non-organic units involved in the COA. We take an 8% of the total personnel as unloading support personnel, 0.42% of the total personnel for each airport as airport support personnel, and 1% of the total personnel for each seaport as seaport support personnel.

LOCs: There are three relevant aspects to evaluate:

- number of locations
- maximum distance between those locations (in miles)
- whether or not there are both air and sea locations.

5 Comparing Alternative COAs

Once the factors relevant for the evaluation have been estimated for each COA, the COAs can be compared against each other to produce a comparison matrix. The matrix is filled out with pluses and/or minuses depending on how the alternative COAs compare.

A-PORTS is better the more throughput they have, which depends mostly on sorties and parking. S-PORTS is better the more berths of bigger size that they have. The closure date is better the closer it is to the D day. LOG PERS is good if it is not a large number.

LOCs are compared as follows. If only one geoloc involved in COA, then they are good. If two geolocs, then they are ok. If three or more geolocs, they are bad. It is better if the locations are close to each other and also if they are far from the enemy border. It is also good if there are both air and sea locations.

In general there are tradeoffs in these factors. For example, the more ports in the COA the better A-PORTS and S-PORTS, but LOG PERS increases and that is not so good. This is key to give feedback to a generative planner from this evaluation: to keep a good value in a factor while improving in another one.

6 Generating Qualitatively Different Plans: EXPECT and O-Plan2

This section describes our ideas to combine the COA generation via O-Plan2 and the COA evaluation capabilities of EXPECT within the Planning Initiative. We first present very briefly the two systems, then we show how they can be combined.

6.1 O-Plan2

The O-Plan2 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-Plan2 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 planning systems. See [2] for details of O-Plan (now referred to as O-Plan1), the planning system that was a forerunner to the O-Plan2 agent architecture. That paper also includes a chart showing how O-Plan relates to other planning systems. Further detail on O-Plan2 is available in [3].

The overall O-Plan2 plan representation and system allows for "tasks" (Missions, constraints, resources, etc) to be explored and compared in a supportive interface for doing plan option analysis. This strategic "Task Assignment" level gives more specific tactical requirements to the computer planner and human planner who work with mixed initiative alongside each other. Neither is "in charge" in our system - they both are "editing" plans constrained by the mission options being explored and the "authority" given to them for planning or execution. Finally, when a COA to be used as a basis for operations is selected, operational planning and execution monitoring support is offered along with some simple forms of plan repair to keep things on track.

The Edinburgh O-Plan2 prototype is currently being demonstrated generating plans in a cut down Tunisian IFD-2 scenario. Work is now underway for mid 1994 to demonstrate the O-Plan2 planner working with an enriched resource model of NEO evacuee transportation in the PRECiS domain. A later demonstration in 1995 is intended to show how plans can be generated and their execution monitored and simple fixes applied in the PRECiS domain.

6.2 EXPECT

The goal of the EXPECT project of the Information Sciences Institute of the University of Southern California is to provide an environment for the development of knowledge-based systems that aids in the acquisition, maintenance, and documentation of the knowledge about a task.

The EXPECT architecture [9, 10, 11] is being applied to producing staff estimates for tentative courses of action to produce briefings for a commander. To date, we have a prototype system that takes an assessment of the situation and evaluates relevant factors for the alternative courses of action from the logistics perspective. The system has a map-based interface that displays force deployment, and allows the user to analyze factor evaluations through interactive dialogues. The user can correct the system's knowledge about how to compute these evaluations if a knowledge deficiency is detected. The user can also correct the system's knowledge base to add new relevant factors or to expand the level of detail at which the evaluations are computed.

6.3 Generating Qualitatively Different Plans

Figure 1 shows how the two systems could cooperate to produce better alternatives. O-Plan2's generated COAs are given to EXPECT. EXPECT evaluates these COAs, and gives feedback to O-Plan's evaluation function in terms of what factors can be improved to produce a better COA.

A higher level Mission Tasking component provides the framework within which options are being explored and compared.

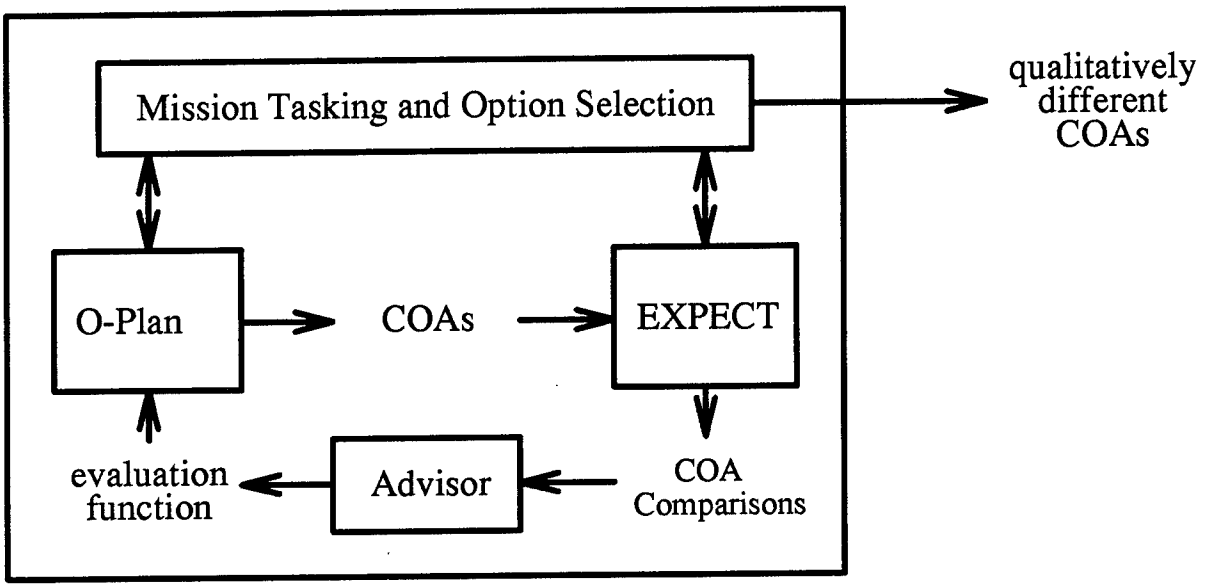


Figure 1: O-Plan2 and EXPECT could cooperate to produce better alternatives for COAs.

The Advisor module would provide the feedback to make a COA of better quality. This feedback can be at different levels of detail. The more details, the easier it is for a generative planner to operationalize the feedback. For example, a high-level piece of feedback could be "The airlift closure date needs to be a day earlier," while a more detailed one would be "use a bigger airport."

Acknowledgements

The EXPECT project is supported by the Advanced Research Projects Agency under contract no. DABT63-91-C-0025. The view and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARPA or the U.S. Government. 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.

Dave Brown from MITRE and Tom Karmiezak from SRA provided very useful expertise that we included in this document. Of course, any inaccuracies and misinterpretations remain ours.

References

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

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

[3] Tate, A., Drabble, B. and 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.

[4] JOPES Planning Policies and procedures Vol. 1, Appendix C, Criteria for the Evaluation of Military Options and Courses of Action.

[5] Reece, G., Tate, A., Brown, D., and Hoffman, M. *The PRECiS Environment* Paper to the ARPA/Rome Laboratory Planning Initiative Workshop at the National Conference on Artificial Intelligence (AAAI-93), Washington D.C., USA. ARPI Report ARPA-RL/CPE/Version 1, August 1993. Also available as Artificial Intelligence and Applications Institute Technical Report AIAI-TR-140, University of Edinburgh.

[6] Armed Forces Staff College, Pub 1., *The Joint Staff Officer's Guide*, 1991.

[7] Rear Admiral Vernon E. Clark, address to the participants of the 1992 DRPI Meeting. San Antonio, TX, 1992.

[8] Air Force Pamphlet 76-2, *Airlift Planning Factors*, 1987.

[9] Swartout, W. R., Paris, C. L., and Moore, J. D. *Design for explainable expert systems.* IEEE Expert 6(3):58–64. 1991.

[10] Paris, C. L. and Gil, Y. *EXPECT: Intelligent support for knowledge-based system development.* In Proceedings of the Seventh European Knowledge Acquisition for Knowledge-Based Systems Workshop. Also published in "Knowledge Acquisition for Knowledge-Based Systems", N. Aussenac, G. Boy, M. Linster, J-G. Ganascia and Y. Kodratoff (eds), New York, NY: Springer Verlag, 1993. ISI Technical Report # RR-93-339.

[11] Gil, Y., and Paris, C. L. *Towards Method-Independent Knowledge Acquisition.* To appear in "Knowledge Acquisition".

Appendix G-A: An Example Scenario

This appendix shows with concrete examples what are the relevant inputs and outputs of the various steps of the development of the concept of operations. The examples used are extracted from the PRECiS scenario.

1 Tentative Courses of Action

Tentative COAs are described as a set of elements composed of who/when/what/where specifications. These correspond to a force module, a time frame (a start date and an end date as offsets from D-day), an action, and a location.

The following are the alternative COAs for this scenario.

COA 1 (Delta) — On D day, the MEU[1] will conduct amphibious operations in Delta and the LIB[2] will airland in Delta. Starting on D+2 and ending no later than D+5, the ACR[3] will begin unloading in Delta. Starting on D+5 and ending no later than D+15, the MID[4] will begin unloading in Delta. The MEU will reimbark no later than D+9. On D day, the CVBG[5] will MODLOC near Barnacle.

COA 2 (Calypso) — On D day, the MEU will conduct amphibious operations in Calypso and the LIB will airland in Calypso. Starting on D+2 and ending no later than D+5, the ACR will begin unloading in Calypso. Starting on D+5 and ending no later than D+15, the MID will begin unloading in Calypso. The MEU will reimbark no later than D+9. On D day, the CVBG will MODLOC near Barnacle.

COA 3 (Delta and Calypso) — On D day, the MEU will conduct amphibious operations in Calypso and the LIB will airland in Delta. Starting on D+2 and ending no later than D+5, the ACR will begin unloading in Delta. Starting on D+2 and ending no later than D+15, 1 Brigade of the MID will begin unloading in Calypso. Starting on D+5 and ending no later than D+15, the rest of the MID will begin unloading in Delta. The MEU will reimbark no later than D+9. On D day, the CVBG will MODLOC near Barnacle.

COA 4 (Delta and Calypso and Abyss) — On D day, the MEU will conduct amphibious operations in Calypso and the LIB will airland in Delta. On D+1, a LI Battalion will airland in Abyss. Starting on D+2 and ending no later than D+5, the ACR will begin unloading in Delta. Starting on D+2 and ending no later than D+15, 1 Brigade of the MID will begin unloading in Calypso. Starting on D+5 and ending no later than D+15, the rest of the MID will begin unloading in Delta. The MEU will reimbark no later than D+9. On D day, the CVBG will MODLOC near Barnacle.

2 Staff Estimates

Staff estimates are presented as matrices of factors and alternative COAs. Section 4 describes how these evaluations are produced based on the description of each COA. The following is example of a logistics staff estimate.

---

[1] Marine Expeditionary Unit
[2] Light Infantry Brigade
[3] Armored Cavalry Regiment
[4] Mechanized Infantry Division
[5] CV Battle Group

	COA 1	COA 2	COA 3	COA 4
A-PORTS:				
- airports	1	1	2	3
- sorties/hr	315	165	480	580
- sq ft ac parking	2M	.9	2.9	4.4
S-PORTS:				
- seaports	1	1	2	3
- piers	6	9	15	18
- berths	6	10	16	21
- max vessel size in ft	600	none	none	none
- oil facilities	1	2	3	4
CLOSURE DATE	D + 15	D + 12	D + 9	D + 9
LOG PERS	3300	3300	3800	4300
LOCs:				
- number locations	1	1	2	3
- miles max distance	20	20	64	208
- air and sea?	yes	yes	yes	yes

3 Comparison Matrices

Based on the estimates, each staff division produces a comparison matrix that compares the alternative COAs. Section 5 shows how these comparisons are constructed. This is an example of a logistics staff comparison matrix:

	COA 1	COA 2	COA 3	COA 4
A-PORTS	++	+	+++	+++
S-PORTS	+	++	+++	+++
CLOSURE	+++	++	+	+
LOG PERS	+	+	+/-	-
LOCs	+	+	++	+/-

These comparisons are represented as pluses and minuses. Based on the data in this figure, COA 3 would probably be selected.

Appendix H – Synthesizing Protection Monitors from Causal Structure

Glen A. Reece and Austin Tate

Appears in the Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), AAAI Press, Chicago, USA, 1994.

Synthesizing Protection Monitors from Causal Structure

Glen A. Reece and Austin Tate

Abstract

Protection monitors synthesized from plan causal structure provide execution systems with information necessary to detect potential failures early during execution. By detecting early, the execution system is able to address these problems and keep the execution on track. When the execution system finds that the necessary repairs are beyond its capabilities, early detection gives the planning system additional time to suggest a repair. This paper discusses how protection monitors are synthesized directly from plan causal structure, and the options which are available to an execution system when protection violations occur.

1 Introduction

Experience in planning for execution in realistic domains tells us that we cannot consistently generate plans that will succeed because of the uncertainty which is inevitably present. A planning system is not able to determine all possible interventions *a priori*, and the model of the world which it uses to base assumptions on is destined to be out of date. The situated agent which must carry out the plans generated by the planning system also has uncertainty to contend with. It is neither in total control of the environment in which it is situated, nor necessarily alone. Thus, we must be able to manage uncertainty during execution.

Many execution systems take a simplistic approach to monitoring. That is, they simply try to test preconditions and postconditions of actions to determine when execution is not going to plan as a way of managing this uncertainty. This is a reasonable mechanism for doing so in some domains. However, as it has been shown [3], such an approach would not be reasonable in domains where actions take long periods of time to complete. For example, if an action N-1 completed successfully at time t1 and its effects were required by an action N-19 at t1+7days, then the fact that some event(s) had taken place which nullified one or more of the required effects between t1 and t1+7days would not be detected until the preconditions of the action were verified at t1+7days.

In order to detect and resolve such problems, an execution system must actively monitor actions during their execution and subsequently up to the point where their conditions are required. Passive monitoring, or only checking the pre- and post-conditions, informs the execution system not to attempt to perform certain actions due to failed preconditions, or that certain actions have not produced all of their expected effects so something else must be done. Active monitoring informs the execution system on how a particular action is progressing to achieve its effects. However, active monitors as they have been defined [10, 4] in the past do not address the whole monitoring picture. In addition to monitoring the progress of an action, we need active monitors to detect protection violations during protection intervals. Such violations typically manifest themselves as later failures thus, possessing the ability to detect them provides an early warning for potential failures.

In order to comprehensively provide execution monitoring in complex and dynamic environments an execution agent must be able to: (1) monitor preconditions and essential postconditions, (2) actively monitor for protection violations, (3) actively monitor situations which are known to cause failures, (4) actively monitor for potential beneficial opportunities, and (5) actively monitor the progress of an action during its execution. Our focus in this paper is to detail how protection violations can be detected early during execution and how protection monitors are synthesized directly from plan causal structure.

In the next section, we describe what is meant by the causal structure of a plan. Section 3 presents a model of plan execution which is based on having casual structure information available. Section 4 discusses the process of how execution monitors are synthesized from causal structure and how they become activated, and in Section 5 we discuss what happens when a violation of a condition is detected during a protection interval.

2 Plan Causal Structure

Causal structure is a high level representation of information about a plan which states the relationship between the purposes of actions with respect to the goals or sub-goals they achieve for some later point in the plan. This information may be used by a planner to detect and correct conflicts between solutions to sub-problems when higher level plans are refined to greater levels of detail.

Various forms of causal structure representations are found in most planning systems for a variety of purposes. During plan generation its main use is for interaction detection and correction. The representations include Goal Structure (or GOST) [12, 2], causal links [6], protection intervals [11], and plan rational [15] to name a few.

During the generative planning process a causal structure table is maintained to record what facts have to be true at any point in the plan network and the possible "contributors" that can make them true. A contributor in this sense is a node in the plan network whose effects are required elsewhere in the network to satisfy a condition of another node. The planning system is able to plan without choosing one of the (possibly multiple) contributors until it is forced to by interaction of constraints. The causal structure is used to detect important interactions (ignoring unimportant side effects) and can be used to find the small number of alternative temporal constraints to be added to the plan to overcome each interaction. This "Question Answering" procedure [12] is the basis for work by Chapman on the Modal Truth Criterion [1]. Multiple interactions arising at the same time further restrict the possible solutions and a minimal set of temporal constraints can be proposed.

We believe that this plan causal structure can be extended to represent information which an execution agent can use to effectively monitor action execution and detect protection violations [13]. Causal structure statements represent precisely the outcome of any operation which should be monitored (i.e., protected). If lower level failures can be detected and corrected while preserving the stated higher level causal structure, the fault need not be reported to a higher level (e.g., a planning system).

3 A Model of Plan Execution Monitoring Based on Causal Structure

An execution agent is given a plan generated by a planner together with information on what the individual plan steps achieve, by what time, and for which subsequent tasks (the causal structure). It must supervise the execution of actions (based on a capabilities data base which might be trivial or quite complex in nature). It should use any available monitoring capabilities to monitor the execution of each action to ensure (as far as possible) that it achieves its purpose(s).

When failures occur, recovery steps may be taken which might be of various types:

- Recovery procedures for the effector chosen (e.g., reset and repeat).
- Recovery procedures for the action type chosen (e.g., generic procedures for ensuring that an action can be successfully accomplished by passing it to a special purpose effector or skilled supervisor).
- Recovery procedures for the particular failed action (e.g., by procedural methods, etc.).

Recovery on failures can be simple or complex depending on the local intelligence of the effectors chosen, the closeness of coupling of actions in the domain, the predictability of error outcomes, etc. When a failure is found which cannot be locally recovered from within the given causal structure constraints (of required outcomes, resource usage or time limits), the execution agent must prepare a statement of the failure and changed plan circumstances to communicate back to the planner (which can then be used to suggest a plan repair).

As shown in Figure 1 (from [13]), an activity can be executed as soon as all the incoming causal structure requirements are satisfied (by any potential "contributor" if there are several alternatives). A decision on the allocation to a particular effector must then be made. The activity and any associated constraint information is then passed to the effector.

The relevant effector executes the action and its controller must report when the activity is completed. Time-out conditions related to the time limits for the follow on actions to the causal structure outcomes are used to prevent the system hanging up on effector controller failure.

The condition monitor is triggered to check all associated causal structure outcomes of the activity. This same model of execution and condition monitoring applies when the activity involves the use of a sensor to capture information needed at some point in the plan. The causal structure outcomes in such a case my contain variables which will be bound to definite values when the condition is checked.

If failures occur, local recovery is possible (by either the effector or by using procedural methods accessible to the execution agent) within the given resource or time limits set for the follow on activities resultant on each monitored outcome. The parallel causal structure (i.e., post-conditions of actions before the failed activity which are required later in the plan) provides a guide to the local recovery system on what should be preserved if the local recovery is to

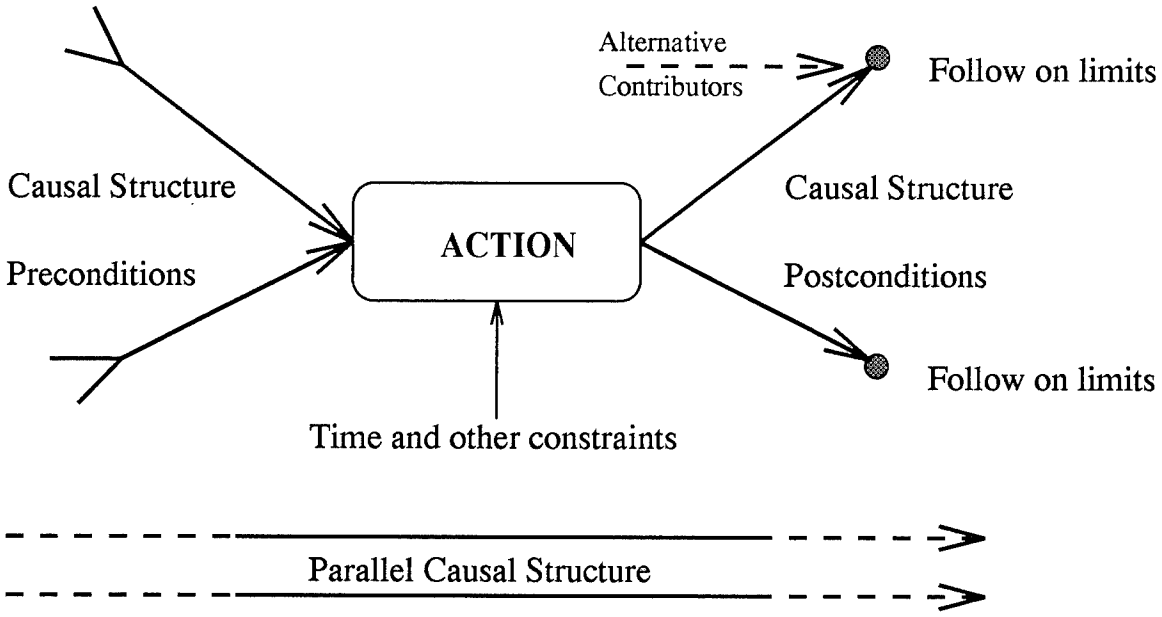


Figure 1: Execution from the viewpoint of a single action.

avoid interference with other important parts of the existing plan. Any interference with such parallel causal structure should be reported to the execution agent as it must be re-considered by the planner to work out the actual effect on the plan.

4 Monitor Synthesis and Activation

The model described in Section 3, is the basis of the execution monitoring functionality of the Reactive Execution Agent (REA) design proposed in [8]. The REA is designed to handle multiple, simultaneously executing plans and to possess the ability to monitor conditions between plan executions. This design utilizes a communication protocol called Inter-Agent Communication Language (IACL) to transmit information between the execution agent and the planner. Tasks are specified by a planning system (in the form of synthesize messages) to the REA which are then carried out using a more detailed model of the execution environment than is available to the planner. The REA executes the plans by choosing the appropriate activities to achieve the various sub-tasks within the plans, using its knowledge about the particular resources under its control. It communicates with the environment in which it is situated by executing the activities within the plans and responding to failures fed back from the environment. 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.

When the planner has generated a plan it intends to execute, it sends a synthesize message that contains the actions of the plan, commitment information, ordering constraints, and plan causal structure. This information is then used by the REA to synthesize a Task-Directive object which it can execute. The causal structure information contained in a synthesize message (see

```
(causal-structure
 ((CSTR-7  (AT GT1) DELTA (N-6-1) (N-3))
  (CSTR-24 (AT GT2) DELTA (N-5-1) (N-3))
  (CSTR-6  (STATUS GT1) AVAIL (N-6-1) (N-3))
  (CSTR-23 (STATUS GT2) AVAIL (N-5-1) (N-3))
  (CSTR-14 (AT C5) DELTA (N-8) (N-3))
  (CSTR-9  (STATUS GT2) ABYSS (N-4-2) (N-4-1))
  (CSTR-8  (AT GT2) ABYSS (N-4-2) (N-4-1))
  (CSTR-18 (STATUS GT2) ABYSS (N-4-4) (N-4-1))
  (CSTR-17 (AT GT2) DELTA (N-4-4) (N-4-3))
  (CSTR-13 (AT GT2) DELTA (N-8) (N-4-3))
  (CSTR-12 (AT GT2) DELTA (N-8) (N-4-4))
  (CSTR-16 (STATUS GT2) BARNACLE (N-5-2) (N-5-1))
  (CSTR-15 (AT GT2) BARNACLE (N-5-2) (N-5-1))
  (CSTR-26 (STATUS GT2) BARNACLE (N-5-4) (N-5-1))
  (CSTR-22 (AT GT2) DELTA (N-4-1) (N-5-3))
  (CSTR-25 (AT GT2) DELTA (N-5-4) (N-5-3))
  (CSTR-21 (STATUS GT2) AVAIL (N-4-1) (N-5-4))
  (CSTR-5  (STATUS GT1) CALYPSO (N-6-2) (N-6-1))
  (CSTR-4  (AT GT1) CALYPSO (N-6-2) (N-6-1))
  (CSTR-20 (STATUS GT1) CALYPSO (N-6-4) (N-6-1))
  (CSTR-19 (AT GT1) DELTA (N-6-4) (N-6-3))
  (CSTR-11 (AT GT1) DELTA (N-8) (N-6-3))
  (CSTR-10 (AT GT1) DELTA (N-8) (N-6-4)))))
```

Figure 2: Causal structure information from a synthesize message

an example in Figure 2[1]) is used by the REA to synthesize monitor objects which actively monitor for protection violations during the execution of the Task-Directive.

Each CSTR, or causal structure record, provides the execution agent with important monitoring information as follows:

(<Tag> <Pattern> <Value> <R-Node> <C-Node(s)>)

The *tag* provides a reference to the planning system for use when a failure has occurred which cannot be addressed locally by the REA. The *pattern* specifies the exact property which is to be protected for the range *C-Node(s)* to *R-Node*. The *R-Node* is the node in the plan network which *requires* the pattern to have the *Value*, and the *C-Node(s)* field specifies one or more *contributors* of the value.

The causal structure record contains all the information necessary to synthesize a monitor object. The mapping of information contained in the CSTR to the monitor object is shown in Figure 3.

The complexity of protection monitors comes from deciding when the monitor should be active.

[1]This example is from a logistics transportation domain in which people are moved by ground transports (GTs) between towns on the fictional island of Pacifica [9, 7]

Monitor-Slot	Value	CSTR-info
NAME	MONITOR-25	
TAG	CSTR-25	TAG
TASK-DIRECTIVE	#<TD-1>	
SCHEMA	#<NODE-9>	
EXPECTED-VALUE	DELTA	VALUE
KNOWN-CONTRIBUTORS	(8)	C-NODE(s)
BEING-MONITORED	(AT GT2)	PATTERN
RANGE-START	8	
RANGE-END	9	R-NODE

Figure 3: Monitor object created from CSTR-25

Basically, a protection monitor is active only while the REA *intends* to execute the associated Task-Directive to which it belongs. The monitors become active immediately upon synthesis of the Task-Directive and are removed when either they expire or all actions of the Task-Directive have been executed. During the "life" of a protection monitor it could find itself in one of three states—activated, inactivated, and expired.

When a synthesize message is received by the REA and a Task-Directive object is being synthesized, any causal structure information is used to synthesize protection monitors and associated with that Task-Directive. All protection monitors are initially in the inactivated state when they are synthesized. For example, the causal structure information shown in Figure 2 is used to synthesize protection monitors for a 15 node plan giving the coverage shown in Figure 4. A single monitor (e.g., M25) is synthesized for each causal structure record (e.g., CSTR-25).

When the REA begins to execute any Task-Directive from its agenda the state of the monitors can change. What a protection monitor object is concerned about is when the REA's world model is updated with new information. When updates occur a set of activation-rules are applied to each protection monitor to determine if it should change its state. These rules determine whether a monitor is to be activated or has expired.

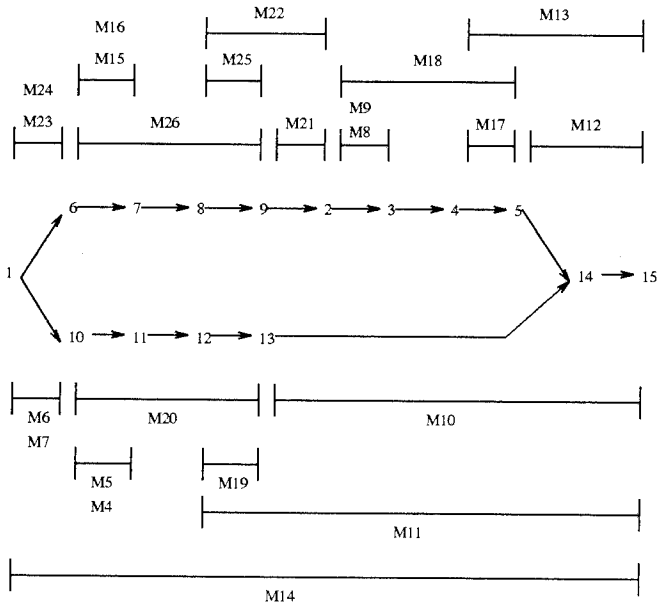
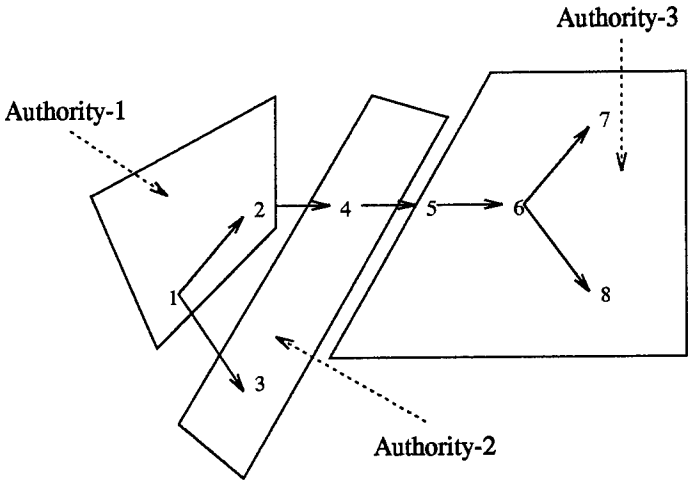


Figure 4: Causal Structure Coverage of a Plan by Protection Monitors

Protection monitors become activated when execution has progressed to the point where the monitor's range is valid. Once a monitor is in the activated state it remains in that state until either what is being monitored by the object does not have the value it expected (in which case it is a violation), or execution has progressed past the range-end of the monitor (in which case it has expired). Once a monitor has expired it is removed from contention and is no longer considered when the activation-rules are applied. Protection violation will be further discussed in Section 5.



(CSTR-1 (condition-1) (N-2 N-3) (N-1))
(CSTR-2 (condition-2) (N-4) (N-2))

Figure 5: Planner Authority Levels

An advantage of this approach to activation is that violations can be detected across Task-Directives so the planner can improve the probability that the assumptions it makes about the future will be valid by protecting them. That is, if the planner "knew" that it was, for the time being, only going to execute a portion of the plan which it was working on, it could submit that plan to the REA with causal structure that would essentially protect the effects of that plan until the remaining portion of the plan could be executed. This requirement stems from the need to plan and/or execute particular "phases" of plans only to specified levels when *authority* is given to do so [14]. For example, we see in Figure 5, that the planner has a plan that it wishes to execute. However, in the first instance it is only given authority to execute actions N-1 and N-2. So, it sends a synthesize message to the REA with causal structure telling it that condition-1 would be introduced by action N-1 which is required by action N-2 (and some action N-3 in the future), and condition-2 would be introduced by action N-2 which would be required by some action N-4 in the future. The REA would then use the causal structure information to monitor condition-1 and condition-2 until actions N-3 and N-4 were executed.

5 Protection Violation

When updates are made to the REA's world model, the activated protection monitors are examined to determine whether a violation has occurred. When the world model is updated with new information a process within the REA is notified and informed which information changes. This process then initiates the determination of whether violations have occurred.

Each activated protection monitor that is monitoring the type of condition which changed in the world model is examined to see if the value it expects the condition to have is the same as its new value. This examination only takes place if all of the contributors of the condition have been executed (i.e., the condition should exist). If one or more contributors remain to be executed then it is likely that a premature violation has occurred, so this potential violation is ignored and the monitor remains activated. If it is the same then the monitor remains activated otherwise, if it is not the same then a violation has occurred and must be examined further.

The planner uses the causal structure to prevent plan state interactions, but the execution agent does not have the ability to prevent things from happening. Not everything in the environment is under the agent's control and some other agent might have interfered.

When a violation has occurred several considerations must be made. First, did the contributor (or last known contributor in the case of multiple alternative contributors) fail? If so, then the violation can be ignored since it was the failure to produce the condition and not any interaction in the environment. This type of failure is handled by another component of the REA. If the violation was not due to a failure then there must have been something acting in the environment which caused the violation. In this situation one of three things could be done—reintroduction, repair, or failure.

Reintroduction is the process of executing another action which will yield the same condition that was violated. However, there are several issues which must be addressed here. This can only occur if the execution agent's representation is rich enough to allow for it to perform the reasoning required to find such a candidate. Then there is the issue of what interactions the introduction of this new action could cause. It could have effects which would interact with other actions waiting to execute and cause additional violations to occur. Reintroduction allows the system to detect and correct a causal structure violation before it manifests itself as a failure in the executing plan.

The second way to address the violation is through repair initiated by the planner. In this case, the REA would communicate with the planner to inform it that the preconditions of a particular node (i.e., the range-end node) were not going to be satisfied when it is eligible to execute. This would make it the planner's responsibility to generate a repair and communicate that back to the REA so the violation was removed. The REA would then ignore any future violation detections by that particular monitor. Repair also allows the system to detect and correct a causal violation, but does so with the assistance of the planning system. It provides an early warning system so the planning system can help the REA to avert possible future execution failures.

The third measure that could be taken to address the violation would be to report total failure of the Task-Directive. Though drastic, it could save resources which could be used by other

Task-Directives the REA intends to execute. Total failure is only an option when the time remaining before the need to execute the action requiring the condition is so small as for it not to be reasonable to expect the planner to generate a repair in time.

6 Discussion

To actively monitor for protection violations during execution a planning system must provide the causal structure of the plan to the execution agent. We have shown how valuable execution monitoring information can be synthesized from this causal structure. This information allows an execution agent to detect (and possibly correct) potential failures before they manifest themselves as actual failures in a executing plan. It also provides a means for developing an early warning system so a planning system can assist the execution agent to avert execution failures by suggesting repairs.

But, just how much casual structure is enough? The causal structure sent to the REA is dependent upon the domain description given to the planner when the plan is generated. Therefore, the more causal structure information available, the more likely it is that the REA will be able to monitor the plan's execution using this approach. The limitation is that this approach is only as good as the sensing capabilities of the REA. As defined, the basic approach comes at no cost since these monitors are triggered from updates to the REA's world model and are not actively sensing the environment. The reality however, is that this approach is most effective when sensors are used often to keep the REA's world model up-to-date. To that end, research is continuing to provide the REA with the capability to actively sense the environment (keeping the world model up-to-date) to improve the utility of the approach presented here. The issue then will be is the ability to monitor *during* protection intervals to detect problems early worth the cost.

The utility of the approach presented here cannot be fully realized until many open issues of how a planning systems can use such information have been addressed. It is one thing to know where a plan has failed, and another to be able to repair the plan from that point. However, some interesting research on the use of causal structure information in plan reuse and modification is being done to address such issues [5]. Kambhampati uses a *validation structure* to represent the internal dependencies of a plan and then uses that structure to help in modifying plans to suit new situations. Though not exactly what its needed to allow a planner to repair a plan based upon the information from the approach presented in this paper, it is a step in the right direction. Hopefully, this approach will be seen to complement his work and the work of others addressing the issues of plan repair.

7 Conclusion

The value of using causal structure information in planning systems has been widely recognized. However, its utility in execution systems has not received much attention. The benefits of providing such information to execution systems are realized during deliberation and while reacting to change. The planning system is able to reduce the uncertainty of the information

in its model of the world by tasking an execution system to monitor conditions it expects to be valid in the future. The execution system is able to avert potential failures by identifying them sooner thus, giving it more time to make repairs.

8 Acknowledgments

This research has benefited from discussions with members of the O-Plan Project Team at Edinburgh. The authors would also like to thank the reviewers for their comments and suggestions regarding this paper.

References

[1] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333–377, 1987.

[2] K.W. Currie and A. Tate. O-Plan: the Open Planning Architecture. *Artificial Intelligence*, 51(1), 1991.

[3] R. Doyle, D. Atkinson, and R. Doshi. Generating perception requests and expectations to verify the execution of plans. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, PA, 1986. Morgan Kaufmann.

[4] D. Hart, S. Anderson, and P. Cohen. Envelopes as a Vehicle for Improving the Efficiency of Plan Execution. In *Proceedings of DARPA Workshop on Innovative Approaches to Planning Scheduling and Control*, pages 71–76, San Diego, California, 1990. Morgan Kaufmann.

[5] S. Kambhampati. A Theory of Plan Modification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-90)*, pages 176–182, Boston, MA., July 29 – August 3 1990. Morgan Kaufmann.

[6] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-91)*, 1991.

[7] G. Reece, A. Tate, D. Brown, and M. Hoffman. The PRECiS Environment. In *Proceedings of National Conference on Artificial Intelligence (AAAI-93) DARPA-RL Planning Inititive Workshop*, Washington, D.C., 1993. Available as ARPA-RL/O-Plan2/TR/11 from the Artifical Intelligence Applications Institute.

[8] G. A. Reece. Reactive Execution in a Command, Planning, and Control Environment. Technical Report 121, Department of Artificial Intelligence, University of Edinburgh, Scotland, 1992.

[9] G. A. Reece and A. Tate. The Pacifica NEO Scenario. Technical Report ARPA-RL/O-Plan2/TR/3, Artificial Intelligence Applications Institute, March 1993.

[10] J.C. Sanborn and J.A. Hendler. A Model of Reaction for Planning in Dynamic Environments. *Artificial Intelligence in Engineering*, 3(2):95–102, 1988.

[11] G. Sussman. *A Model of Skill Acquisition.* Elsevier Scientific, 1975.

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

[13] A. Tate. Planning and Condition Monitoring in a FMS. In *Proceedings of International Conference on Flexible Automation Systems*, London, England, 1984. Available as AIAI-TR-2 from the Artifical Intelligence Applications Institute.

[14] A. Tate. Authority Management Coordination between Task Assignment Planning and Execution. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-93) Workshop on Knowlege-based Production Planning, Scheduling and Control*, Chambery, France, 1993. Available as AIAI-TR-133 from the Artifical Intelligence Applications Institute.

[15] D.E. Wilkins. Domain-Independent Planning: Representation and Plan Generation. *Artificial Intelligence*, 22(3):269–301, 1984. Available as SRI-TR (May 1983) from SRI International.

Appendix I – Putting Knowledge Rich Process Representations to Use

Austin Tate

Appears in *IOPener - The Journal of the IOPT Club for the Introduction of Process Technology*, Vol. 2 No. 3 pp 12-14, March 1994, UK Introduction of Process Technology (IOPT) Club, c/o Praxis Ltd, UK.

Putting Knowledge Rich Process Representations to Use

Austin Tate

1 Introduction

Research into the generation of plans using knowledge based techniques is now maturing and finding practical application in the commercial, industrial and defence sectors. This has led to a rapid expansion in the last couple of years of investment in this area. Ways to represent plans which have emerged from the research have been found to be of benefit in a number of areas – even when the *generation* of a plan is not the primary concern. One way in which the research is being exploited is via the use of knowledge rich plan *representations* to allow systems to improve their monitoring, analysis and advisory capabilities.

These knowledge rich plan representations are now finding use is in the area of Process Modelling – in particular for business process modelling. Enriched process models can lead to enhancements of the analysis and critiquing of business processes and can open up a variety of ways to support the synthesis and re-engineering of processes, the creation of plans and intelligent workflow management systems.

Knowledge based approaches are acknowledged as a key component in facilitating the integration of "islands of automation" in today's enterprises. Means to connect executive strategic decision making, analysis and direction with tactical planning and scheduling capabilities and on to effective operations management within an organisation may be facilitated by using AI based plan representation approaches.

The First Conference on Enterprise Integration Modelling [6] identified support for the management of change as an important area for the success of enterprise integration efforts. The working groups of the conference also believed that a combination of Artificial Intelligence (AI) and Operations Research (OR) methods as explored by the knowledge based planning community could be a good basis for this work.

2 Knowledge Rich Plan Representations

Plan representations have been developed over several decades of AI planning research [2]. They can support a rich model of processes, tasks, plans, resources and agents. These representations can be used for purposes other than plan generation. Some key concepts include hierarchical plan representations, rich activity and resource models, the capturing of the intentions behind plan steps, and languages or ontologies in which to express the activity and process models.

Knowledge rich plan representations such as those in the Edinburgh O-Plan [4] and O-Plan2 [8] planners have been used successfully in a number of projects.

The PLANIT work [5] is a prototype system produced during the UK Alvey Programme in which rich plan representations were *used* without plans being actually generated. In PLANIT, flexible

plan representations provided integration across an enterprise involving project management (interfaced to the 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.

OPTIMUM-AIV [1] is a more recent example of the use of flexible plan representations in a project management domain alongside ARTEMIS project support tools. OPTIMUM-AIV is a flexible planning and re-planning system for spacecraft assembly, integration and verification at the European Space Agency.

These two systems explicitly represent the causal structure of a plan, to hold the dependencies between the preconditions and effects of activities involved in the plan – therefore showing the rationale or intentions behind the plan. Dependencies of the same kind are useful in all aspects of plan generation, execution monitoring and plan repair.

3 The O-Plan Triangle Model of Activity

The O-Plan2 team at Edinburgh are working to simplify some of these 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).

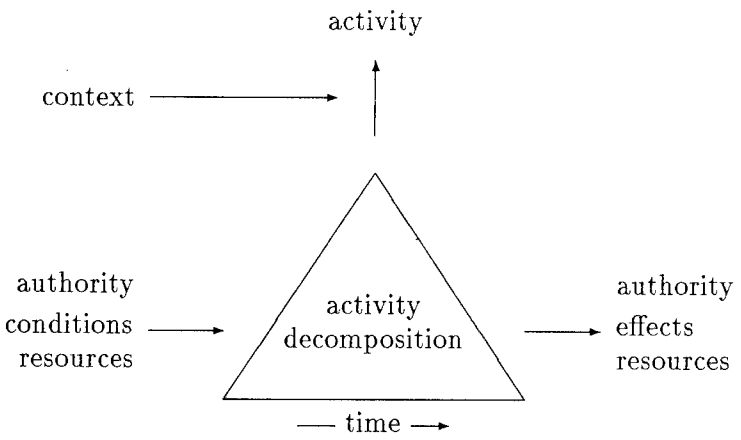


Figure 1: O-Plan2 Triangle model of Activity

This work is reflected in our "triangle" model of an activity (see figure 1). 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 provision may also be stated. This makes it possible to use a uniform approach to the modelling of authority, product flow and resource requirements.

Note that there is a deliberate and direct mapping between the O-Plan2 triangle model of activity and the existing IDEF methodology (see figure 2). IDEF-0 is compared here since it has been used for modelling processes[1].

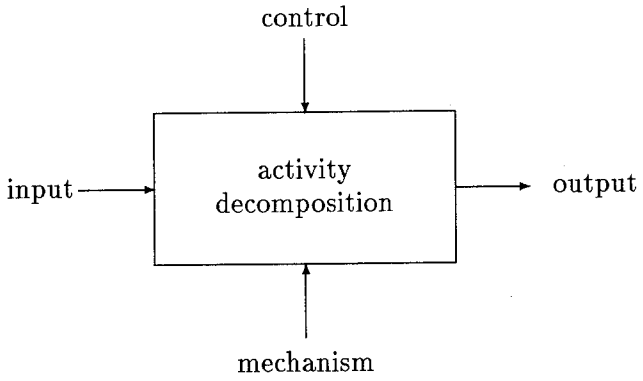


Figure 2: IDEF-0 model

IDEF modellers usually use "control" for authority related triggers and "mechanism" to reflect resource availability. A criticism of IDEF is the lack of direct support for modelling the different types of output and their intended destination. Experienced IDEF modellers use the arc labels, naming conventions and the "notes" system in an IDEF support "kit" to encode this information.

[1] IDEF-3 is a later more comprehensive IDEF method specifically targeted at the modelling of processes.

The O-Plan2 triangle model more directly supports this and will allow for improved support tools.

4 Using Knowledge Rich Plan Representations for Process Management

O-Plan2 has a three level modelling approach[2] that relates a high level of executive strategic decision making, analysis and direction with middle level tactical planning and scheduling capabilities and on to effective lower level operations management within an organisation.

Our approach is to offer an overall vision of an *Executive Communication and Control* environment in which an enriched corporate model (knowledge and data bases) is utilised to offer an "add-on" to existing tools used in companies such as those for option analysis, risk analysis, business case analysis, project management, workflow management, etc. In this framework, we are seeking to improve the types of process models that can be captured and to enrich these models in ways that may be done informally in today's requirements capture and modelling tools. The enriched representations allow for improved analyses and open the way to a new generation of tools for business process management that will provide enhanced aids to:

- reliably capture and maintain process knowledge and models
- make decisions using knowledge based simulation and analysis
- synthesise processes
- re-engineer parts of a process
- reliably execute processes
- simulate, animate, explain and justify processes

Our approach uses *open* and *inspectable* knowledge based representations of processes, tasks, plans and schedules in which dependencies, constraints and preferences are maintained. Our work in process management draws on experience with Nonlin [7], O-Plan, O-Plan2, PLANIT and OPTIMUM-AIV as follows:

- the use of a 3-level view (at strategic, tactical and operational levels) with task assignment, planning and control roles (O-Plan2).
- knowledge rich plan representations summarised in the triangle model of activity (Nonlin, O-Plan and O-Plan2).
- process and plan impact assessment/critiquing (Nonlin, PLANIT).

[2]What is being done is to *augment* earlier methods and approaches, rather than starting afresh and replacing or reinventing what has already been achieved.

- constrained plan editing and single step plan modification option review (O-Plan and PLANIT).
- plan generation technology (O-Plan).
- plan question answering, state generation, simulation and animation (Nonlin and O-Plan2).

With enriched process representations and using knowledge based approaches, it is possible to go beyond *what if?* option analysis and to generate *how to?* option proposals. Higher level component processes can be selected, combined and tailored to specific requirements. Intentions and dependencies captured in process models allow for the enhancement of the capabilities which can be provided to the executive and operational staff.

5 AIAI's Vision for Process Management

Our visualisation of knowledge base supported process management in an organisation is:

Strategic Level Consider that there is support to establish the following:

- organisational processes and constraints reliably captured along with their underlying intentions and dependencies
- lead applications identified, cost benefit analysed, risk analysed and rated using business case support tools
- key objectives and tasks stated, constraints identified
- top level options generated and evaluated
- programme road map created

Tactical Level On the basis of the information generated above, we use constrained plan editing, partially or fully automatic plan elaboration, scheduling, etc. We fit this level to project management tools in the market today and show how these can inter-work. We add plan creation, querying, simulation and animation capabilities to significantly enhance the capabilities available to planners and schedulers today.

Operational Level Then we use the information and its embedded rationale, enriched resource and authority model to fit to (intelligent) work flow managers or process support environments (e.g. such as ProcessWise [3]) and use these as the point of delivery and control of reliable enactment of the plans and schedules in an enriched environment where plan tracking, question answering, explanation, options reworking, reaction and recovery are all possibilities future.

Acknowledgements

Prof. Austin Tate is Technical Director at the Artificial Intelligence Applications Institute of The University of Edinburgh. Established in 1984, AIAI is a non-profit technology transfer organisation working with system providers and user companies throughout the world. AIAI uses knowledge based methods to support process management through process modelling, synthesis, analysis and modification. O-Plan work is funded as part of the $40 million US ARPA and USAF Planning Initiative which is developing next generation planning, command and control support infrastructure relevant to US military logistics. AIAI is pursuing its approach to the use of knowledge rich process representations in commercial applications though its collaboration in the Enterprise project. Enterprise is a consortium including AIAI, IBM, Unilever, Logica and Lloyds' Register and is the largest project supported by the UK government's Intelligent Systems Integration Programme.

References

[1] 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.

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

[3] Bruynooghe, R.F., Parker, J.M. & Rowles, J.S., *PSS: A System for Process Enactment*, published paper from ICL, Kidsgrove, Staffs ST7 1TL, UK, 1992.

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

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

[6] Petrie, C.J. (ed.), *Enterprise Integration Modelling*, Proceedings of the First International Conference, MIT Press, 1992.

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

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

Appendix J – Characterising Plans as a Set of Constraints – the <I-N-OVA> Model – a Framework for Comparative Analysis

Austin Tate

Appear in the Special Issue on "Evaluation of Plans, Planners, and Planning Agents", *ACM SIGART* Bulletin Vol. 6 No. 1, January 1995.

Characterising Plans as a Set of Constraints – the <I-N-OVA> Model – a Framework for Comparative Analysis

Austin Tate

1 Motivation

The <I-N-OVA> (*Issues – Nodes – Orderings/Variables/Auxiliary*) Model is a means to represent plans as a set of constraints. By having a clear description of the different components within a plan, the model allows for plans to be manipulated and used separately to the environments in which they are generated.

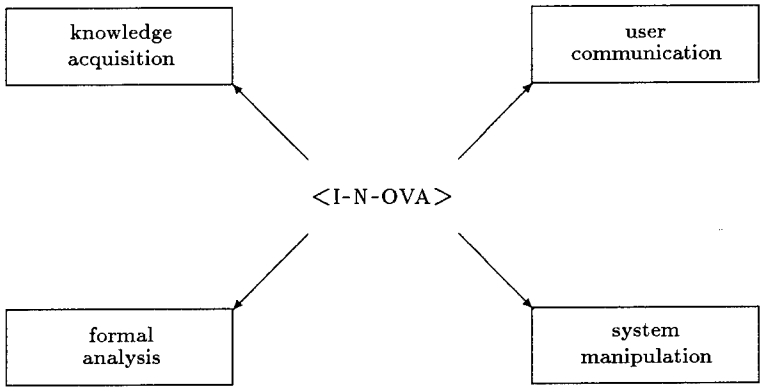


Figure 1: Roles for <I-N-OVA>

As shown in figure 1 the <I-N-OVA> constraint model underlying plans is intended to support a number of different uses of plan representations:

- suitability for automatic manipulation of plans and to act as an ontology to underpin such use.
- suitability for human communication about plans.
- suitability for principled and reliable acquisition of plan information.
- suitability for formal reasoning about plans.

These cover both formal and practical requirements and encompassing the needs of both human and computer based planning systems.

Our aim is to characterise the plan representation used within O-Plan [8],[34] and to more closely relate this work to emerging formal analyses of plans and planning. This synergy of practical and formal approaches can stretch the formal methods to cover realistic plan representations as needed for real problem solving, and can improve the analysis that is possible for production planning systems.

2 Representing Plans as a Set of Constraints

A plan is represented as a set of constraints which together limit the behaviour that is desired when the plan is executed. Work on O-Plan [8],[34] and other practical planners has identified different entities in the plan which are conveniently grouped into three types of constraint. The set of constraints describe the possible plan elaborations that can be reached or generated as shown in figure 2.

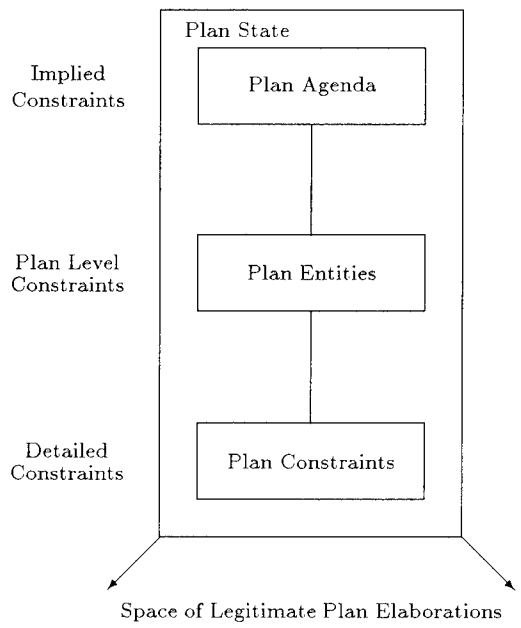


Figure 2: Plan Constraints Define Plan Space

The three types of constraint in a plan are:

1. Implied Constraints or "Issues"[1] – representing the pending or future constraints that will be added to the plan as a result of handling unsatisfied requirements, dealing with

[1]We have previously used a variety of different names for these constraints: *Agenda Entries* reflecting the chosen method of representation in O-Plan; *Flaws* as suggested by Sam Steel in the mid 1980s and reflecting the original concentration of representing the outcome of plan critics which found interactions in the teleological structure which had to be corrected; *To-do list entries* reflecting common usage in business; *Pending Processing*

aspects of plan analysis and critiquing, etc. The implied constraints are the issues to be addressed, i.e., the "to-do" list or agenda which can be used to decide on what plan modifications should be made to a plan by a planner (user or system).

2. Plan Entities or Plan Node constraints - the main plan entities related to external communication of a plan. They describe a set of external names associated to time points. In an activity planner, the nodes are usually the actions in the plan associated with their begin and end time points. In a resource centred scheduler, nodes may be the resource reservations made against the available resources with a begin and end time point for the reservation period.

3. Detailed Constraints - associated with plan entities and representing specialised constraints on the plan. Empirical work on the O-Plan planner has identified the desirability of distinguishing two special types of detailed constraint:

   - Ordering or Temporal Constraints (such as temporal relationships between the nodes or metric time properties).
   - Variable Constraints (co-designation and non-co-designation constraints on plan objects in particular).

   These two constraints are highlighted since they may form part of other constraints within a temporal reasoning domain such as occurs in planning and scheduling problems. Knowing that these constraints have such cross "associations" has been found to simplify planner system design of constraint handling mechanisms and ease implementation issues [29],[30].

   Other Detailed Constraints relate to input (pre-) and output (post-) and protection conditions, resources, authority requirements, spatial constraints, etc. These are referred to as:

   - Auxiliary Constraints.

   Auxiliary Constraints may be expressed as occurring at a time point (referred to as "point constraints") or across a range of the plan (referred to as "range constraints"). Point constraints can be used to express input and output constraints on nodes or for other constraints which can be expressed at a single time point. Range constraints relate to two or more time points and can be used to express protection intervals, etc.

3 The <I-N-OVA> Model

A plan is represented as a set of constraints of three principal types. To reflect the three main types of constraint identified and their differentiation in the model, the constraint set for a plan is written as <I-N-OVA> (*Issues – Nodes – Orderings/Variables/Auxiliary*). I stands for the the

Requirements reflecting the notion that they implied future plan manipulation or constraints; and others. We have settled on *Issues* suggested by Craig Wier in 1994 as being an easily understood term that reflects both the need to handle problems and the positive opportunities that present themselves.

issues agenda or implied constraints, N for the node or plan entity constraints, and OVA for the detailed constraints held as three types (O for ordering constraints, V for variable constraints, and A for the other auxiliary constraints).

The auxiliary constraints are given 4 types: Authority, Conditions, Resources and Other and all may be stated as point (related to a single time point) or range (related to two or more time points) constraints. Sub-types are possible for any of the Auxiliary Constraints and the nature of these reflects on-going work on knowledge modelling for planning and scheduling domains (e.g., [28], [33]).

The <I-N-OVA> constraint model for plans thus contains a hierarchy of constraint types and sub-types as follows:

```
Plan Constraints
  I   - Implied Constraints
  N   - Node Constraints relating to
                a set of time points
  OVA - Detailed Constraints
        O - Ordering Constraints
        V - Variable Constraints
        A - Auxiliary Constraints
            - Authority Constraints
              - subtypes
            - Condition Constraints
              - subtypes
            - Resource Constraints
              - subtypes
            - Other Constraints
              - subtypes
```

The node constraints in the <I-N-OVA> model set the space within which a plan may be further constrained. The issues and OVA constraints restrict the plans within that space which are valid.

The <I-N-OVA> model currently assumes that it is sufficiently general for each node (referred to as N constraints) to be associated with just two time points. One representing the begin of the node and the other representing the end of the node. Further research may indicate that a more general multiple time point association of nodes to time points may be necessary.

Hierarchical or abstraction level modelling is possible for all constraint types within the <I-N-OVA> model. To reflect this possibility, an <I-N-OVA> model which is described hierarchically or with levels of abstraction will be referred to a Hierarchical <I-N-OVA> model. This will be written as Δ-<I-N-OVA>.

The Δ is a triangle pictogram symbol used to represent hierarchical expansion. It can be written in an alternate all character version as H-<I-N-OVA>.

4 The Triangle Model of Activity

The <I-N-OVA> auxiliary constraints incorporate details from the Triangle Model of Activity used to underpin the Task Formalism (TF) domain description language [32] used for O-Plan [8],[34]. The Triangle Model seeks to give a clear description of activities, tasks and plans in a common framework that allows for hierarchical decomposition and time relationships along with authority, pre- and post-conditions, resources and other constraints. The Triangle Model of Activity can be used as a basis for planning domain modelling and for supportive task description interfaces.

The aim in the Triangle Model is to simplify some of the notions from expressive plan and activity representations from AI planning and to relate them better to existing systems engineering requirements capture and modelling languages and methods (like SADT [24], IDEF [20], CORE [9], HOOD [13], etc.).

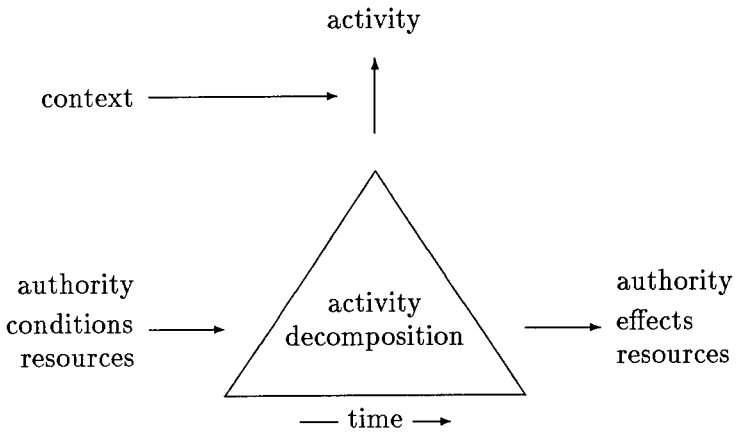


Figure 3: O-Plan Triangle model of Activity

Figure 3 shows the Triangle Model of Activity. The vertical dimension reflects action decomposition, the horizontal dimension reflects time. Inputs and outputs are split into three principal categories (authority, conditions/effects and resources). Arbitrarily complex modelling is possible in all dimensions. Types and sub-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: 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 as in the EXCALIBUR [10] extension to Nonlin [26].

The activity decompositions shows the expansion of the activity to a greater level of detail if that is modelled. It can include details of protection conditions that span points within a decomposition.

Variables may be referred to in an activity description. Differentiation between those variables used in the external specification (outside the triangle) and those only used within the activity decomposition (internal to the triangle) is possible.

The O-Plan time model defines a set of time points which can be related to an absolute start of time (for metric time statements) or which can be related to one another (for relative time relationships). Temporal relationships between an activity (referred to as *self*) and the sub-activities within a decomposition may be stated with reference to the two "ends" of any activity. Arbitrarily complex temporal relationships (e.g., [2]) are possible in the general Triangle Model

The "intentions" or "rationale" behind the use of a particular activity can be related to the features of this triangle model. Causality or teleology modelled via activity pre-conditions/post-conditions has been used in AI planners for many years to record the plan rationale (e.g., in Nonlin [26]). In the richer model now in use in O-Plan, rationale in terms of resource usage and supply or authority requirements or delegation may also be stated. This makes it possible to use a uniform approach to the modelling of authority, product flow and resource requirements.

5 Relationship of Triangle Model to O-Plan TF Schemas

The Triangle Model of activity maps directly to an O-Plan Task Formalism (TF) schema. TF is the domain description language for O-Plan. The following shows the components of a simplified O-Plan TF schema. "..." indicates the detailed part of each component. Further detail is available in [32].

```
schema <schema_name>;
   ;;; public information
   vars                    ... ;
   expands                 ... ;
   only_use_for_authority  ... ;
   only_use_for_effects    ... ;
   only_use_for_resources  ... ;

   ;;; private information
   local_vars              ... ;
   vars_relations          ... ;
   nodes                   ... ;
   orderings               ... ;
   time_windows            ... ;
   authority               ... ;
   conditions              ... ;
   effects                 ... ;
   resources               ... ;
```

```
   other_constraints        ... ;
end_schema ;
```

6 Domain Operators, Tasks and Plans

Figure 4 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.

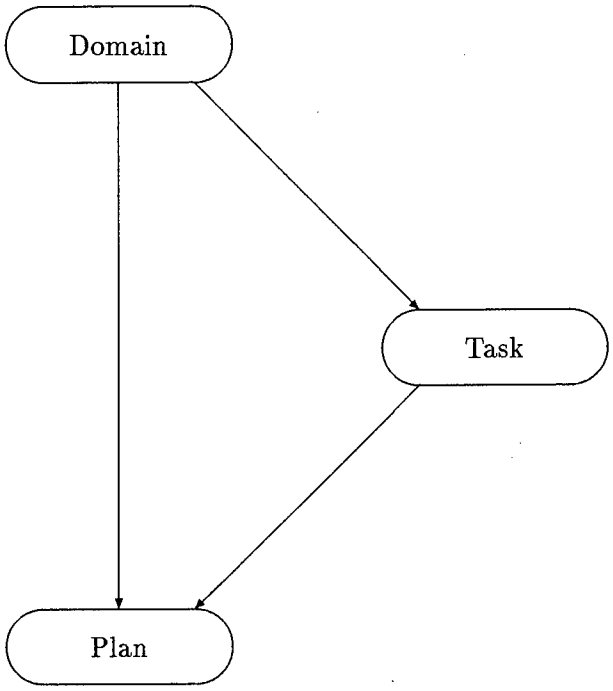


Figure 4: Dependencies between Domain, Task and Plan Knowledge Partitions

- *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.

<I-N-OVA> is intended to underpin domain, task and plan modelling needs in a planning system whether human, computer or mixed agents are involved. Communication between planning agents in O-Plan takes place via Plan Patches [27] which are also based on the Triangle Model of Activity and the <I-N-OVA> constraint components.

7 Relationship of Triangle Model to Structured Analysis and Design Techniques

There is a deliberate and direct mapping between the O-Plan Triangle Model of Activity and the <I-N-OVA> Constraint Model of Plans to existing structured analysis and diagraming methods such as IDEF, R-Charts, etc. Other researchers have recognised the value of merging AI representation concepts with structured analysis and diagramming techniques for systems requirements modelling [6].

IDEF0 [19] is a functional modelling method and diagraming notation that has been used for modelling processes[2]. Figure 5 shows the basic component.

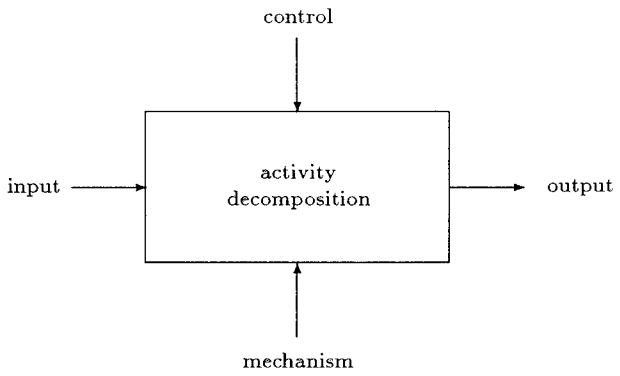


Figure 5: IDEF0 model

IDEF modellers usually use "control" for authority related triggers and "mechanism" to reflect resource availability. A criticism of IDEF is the lack of direct support for modelling the different types of output and their intended destination. Experienced IDEF modellers use the arc labels, naming conventions and the "notes" system in an IDEF support "kit" to encode this information.

R-Charts [35] are one of the ISO approved diagraming conventions for program constructs (ISO/IEC 8631 [14]). Figure 6 shows the basic component which explicitly acknowledges the importance of control (or authority) related outputs.

The O-Plan triangle model represents all three types of input and output more uniformly and

[2]IDEF3 [20] is a later more comprehensive IDEF method specifically targeted at the modelling of processes.

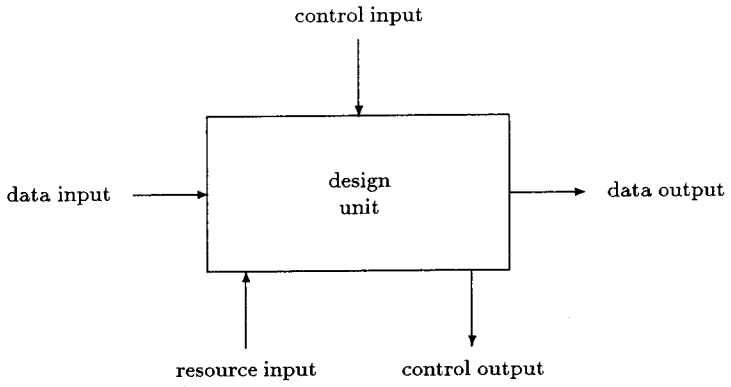


Figure 6: R-Chart Model

directly and will allow for improved support tools.

8 Relationship to Other Work

A general approach to designing AI-based planning and scheduling systems based upon partial plan or partial schedule representations is to have an architecture in which a plan or schedule is critiqued to produce a list of issues or agenda entries which is then used to drive a processing cycle of choosing a "plan modification operator" and then executing it to modify the plan state. Figure 7 shows this graphically.

This approach is taken in systems like O-Plan [8],[34], RT-1 [3], OPIS [25], DIPART [23], TOSCA [5], etc. The approach fits well with the concept of treating plans as a set of constraints which can be refined as planning progresses. Some such systems can act in a non-monotonic fashion by relaxing constraints in certain ways.

Having the implied constraints or "agenda" as a formal part of the plan provides an ability to separate the plan that is being generated or manipulated from the planning system itself. The benefits were first noted by McDermott [21] and are used as a core part of the O-Plan design.

A recently described approach to Mixed Initiative Planning in O-Plan [31] proposes to improve the coordination of planning with user interaction by employing a clearer shared model of the plan as a set of constraints at various levels that can be jointly and explicitly discussed between and manipulated by user or system in a cooperative fashion.

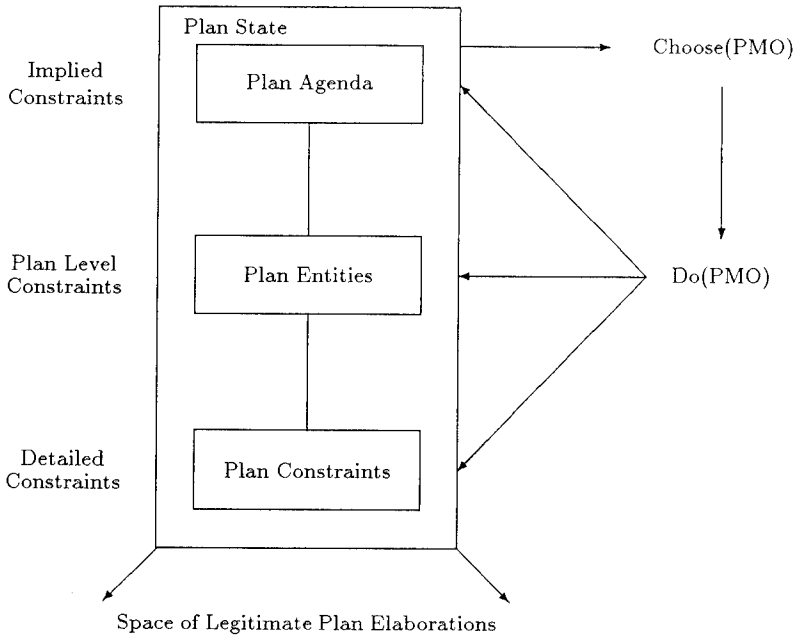


Figure 7: A Framework of Components in a Planning/Scheduling System

9 Relationship to Formal Studies of Plans and Planners

The Nonlin QA Algorithm [26] establishes the modifications that are needed in terms of plan step ordering and variable binding to ensure that a given statement has a required value at a given point in a partially ordered network of nodes. This has been a basis for the formal work by Chapman [7] on the Modal Truth Criterion. However, the MTC uses a simplification of the plans being represented in practical planners such as Nonlin [26], O-Plan [8],[34] and SIPE [37]. It took a non-hierarchical view and ignored specialised domain knowledge of activity condition types and constraints. Many of these were those very features that allowed planners like Nonlin and SIPE to solve problems at a scale that was beyond the more theoretically based planners. Drummond [12] explains that formal approaches have concentrated on goal achievement aspects of planners in a simplified environment that is not representative of the approaches actually taken in practical planners.

Recently however, formal representations have begun to address issues of realistic plan representations and to model hierarchical planning [4],[18],[22],[38]. In particular, Kambhampati has described a formal truth criterion for plans which are represented with greater levels of realism. He describes plans as a 5 tuple [16]:

```
<S, O, B, ST, L>

S  a set of plan steps or nodes
```

```
ST a symbol table mapping each plan step
   or node to a domain operator
O  a partial ordering over S
B  a set of variable binding
   co-designation and
   non-co-designation constraints
L  a set of auxiliary constraints
   (mainly intended for pre- and post-
   conditions)
```

This representation can be related directly to the N (S and ST) and OVA (O, B and L) parts of the <I-N-OVA> model[3].

Hendler and Kambhampati are also studying hierarchical approaches to formal methods in planning [17],[18]. Work is underway by Kambhampati and by Young [39] to understand aspects of the use of "condition types" [33] used to provide domain semantic information to Nonlin, O-Plan and other practical planners.

10 A Framework for Further Study

To provide a framework for further study, the following classification of models related to <I-N-OVA> is provided.

	partial plan	partial plan with issues
single level model	<N-OVA>	<I-N-OVA>
hierarchical model	Δ-<N-OVA>	Δ-<I-N-OVA>

A base model <N-OVA> is used to represent a basic plan without hierarchy or abstraction modelling and not including implied constraints (the issues agenda). The other models extend this basic model along these two dimensions[4]. They are all supersets of <N-OVA>, and are collectively termed *Super*-<N-OVA> models.

[3]The use of the term "Auxiliary Constraints" in <I-N-OVA> was adopted as a means to relate to this formal work. In fact the <S, O, B, ST, L> constraint set acts as a refinement filter on all possible plans, whereas <I-N-OVA> also defines the candidate set from which the solutions may come. This needs further study to relate the two approaches.

[4]Non-determinism is a property of the system (human or computer based) which manipulates the plans and is not necessarily represented in the constraint model. However, it is usual to include explicit dependency information in a plan via constraints to support non-monotonic planners. This may indicate that it would be useful to define a third dimension to this framework for further study.

The <N-OVA> element most closely relates to the model being studied by Kambhampati today [16]. The Δ-<I-N-OVA> element is the closest to the plan representation used within O-Plan today.

11 Summary

The <I-N-OVA> Constraint Model of Plans and its relationship to the O-Plan Triangle Model of Activity has been described to assist in more closely relating new work in formal descriptions of plans and planners to practical work on realistic planning systems. <I-N-OVA> is intended to act as a bridge to improve dialogue between the communities working in these two areas and potentially to support work on automatic manipulation of plans, human communication about plans, principled and reliable acquisition of plan information, and formal reasoning about plans.

Acknowledgements

Prof. Austin Tate is Technical Director at the Artificial Intelligence Applications Institute of The University of Edinburgh. Established in 1984, AIAI is a non-profit technology transfer organisation working with system providers and user companies throughout the world. AIAI uses knowledge based methods to support process management through process modelling, synthesis, analysis and modification. O-Plan work is funded as part of the $40 million US ARPA and USAF Planning Initiative which is developing next generation planning, command and control support infrastructure relevant to US military logistics. AIAI is pursuing its approach to the use of knowledge rich plan and process representations in commercial applications though its collaboration in the Enterprise project. Enterprise is a consortium including AIAI, IBM, Unilever, Logica and Lloyds' Register and is the largest project supported by the UK government's Intelligent Systems Integration Programme.

The O-Plan project is supported by ARPA/Rome Laboratory Knowledge-based Planning and Scheduling Initiative through the Air Force Office of Scientific Research (AFOSR) and their European Office of Aerospace Research and Development by contract number F49620-92-C-0042 (EOARD/92-0001) monitored by Dr. Northrup Fowler III at the USAF Rome Laboratory. The United States Government is authorised to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon.

References

[1] Allen, J.F., Hendler, J. and Tate, A., Readings in Planning, Morgan Kaufmann Publishers, Palo Alto, CA., 1990.

[2] Allen, J.F. and Koomen, J.A., Planning Using a Temporal World Model, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-83), Karlsruhe, Germany, 1993.

[3] D'Ambrosio, B., Raulefs, P., Fehling, M.R., and Forrest, S., Real-time Process Management for Materials Composition in Chemical Manufacturing, Technical Report, Technowledge Inc, 1850 Embarcadero Road, Palo Alto, CA 94303 and FMC Corporation, AI Center, Central Engineering Laboratories, Box 580, 1205 Coleman Avenue, Santa Clara, CA 95052, USA, 1987.

[4] Barrett, A. and Weld, D.S., Task-Decomposition via Plan Parsing, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, 1994.

[5] 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. Also available as AIAI Technical Report AIAI-TR-121.

[6] Borgida, A., Greenspan, S. and Mylopoulos, J., Knowledge Representation as the Basis for Requirements Specifications, IEEE Computer Magazine, Special Issue on Requirements Engineering Environments, April 1985.

[7] Chapman, D., Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333-377, 1991.

[8] Currie, K.W. and Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence* 52(1), Autumn 1991, North-Holland.

[9] Curwen, P., System Development Using the CORE Method, British Aerospace Technical Report BAe/WIT/ML/GEN/SWE/1227, 1990.

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

[11] Drabble, B. and Kirby, R.B., Associating AI Planner Entities with an Underlying Time Point Network, European Workshop on Planning (EWSP) 1991, Springer-Verlag Lecture Notes in Artificial Intelligence. Also available as AIAI Technical Report AIAI-TR-94.

[12] Drummond, M.E., On Precondition Achievements and the Computational Economics of Automatic Planning, in Current Trends in AI Planning (eds. C.Backstrom and E.Sandewall) IOS Press, Sweden, 1993.

[13] HOOD Working Group, HOOD Reference Manual, Issue 3.0 European Space Agency, Noordwijk, Netherlands, 1989.

[14] ISO/IEC 8631-1989 Information Technology - program Constructs and Conventions for their Representation, second edition, ISO/IEC, 1989.

[15] Gil, Y. and Linster, M., On Analyzing Planning Applications, in preparation, 1994.

[16] Kambhampati, S., Design Tradeoffs in Partial Order Planning, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), Chicago, IL., USA, 1994.

[17] Kambhampati, S., Comparing Partial Order Planning and Task Reduction Planning: a Preliminary Report, Proceedings of the Workshop on Comparative Analysis of AI Planning Systems, AAAI-94, Seattle, USA, 1994.

[18] Kambhampati, S. and Hendler, J., A Validation-Structure-Based Theory of Plan Modification and Reuse, *Artificial Intelligence*, May, 1992.

[19] Mayer, R.J. (editor), IDEF0 Functional Modeling: A Reconstruction of the Original Air Force Wright Aeronautical Laboratory Technical Report AFWAL-TR-81-4023 (The IDEF0 Yellow Book), Knowledge Based Systems Inc., College Station, TX, 1992.

[20] Mayer, R.J. and Painter, M., IDEF Family of Methods, Technical Report, Knowledge Based Systems Inc., College Station, TX, 1991.

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

[22] Penberthy, J.S. and Weld, D.S., UCPOP: A Sound, Complete, Partial Order Planner for ADL, Proceedings of the Third International Conference on Knowledge Representation and Reasoning, 1990.

[23] Pollack, M., DIPART Architecture, Technical Report, Department of Computer Science, University of Pittsburgh, PA 15213, USA, 1994.

[24] Ross, D.T., Applications and Extensions of SADT, IEEE Computer Magazine, Special Issue on Requirements Engineering Environments, April 1985.

[25] Smith, S., OPIS: A Methodology and Architecture for Reactive Scheduling, in Intelligent Scheduling, (eds, M.Zweben and M.S.Fox), Morgan Kaufmann Publishers, Palo Alto, CA., USA, 1994.

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

[27] Tate, A., Coordinating the Activities of a Planner and an Execution Agent, 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.

[28] 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, 1993.

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

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

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

[32] Tate, A., Drabble, B. and Dalton, J., O-Plan Version 2.2 Task Formalism Manual, O-Plan Project Documentation, AIAI, University of Edinburgh, 1994.

[33] Tate, A., Drabble, B. and Dalton, J., The Use of Condition Types to Restrict Search in an AI Planner, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, 1994.

[34] 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.

[35] Ushakov, I. and Velbitskiy, I., Visual Programming in R-technology: Concepts, Systems and Perspectives, Proceedings of the Third East-West International Conference on Human Computer Interaction, Moscow, Russia, 1993.

[36] Valente, A., Knowledge-Level Analysis of Planning Systems, Proceedings of the Workshop on Comparative Analysis of AI Planning Systems, AAAI-94, Seattle, USA, 1994.

[37] Wilkins, D., *Practical Planning*, Morgan Kaufmann, 1988.

[38] Yang, Q. Formalizing Planning Knowledge for Hierarchical Planning, *Computational Intelligence*, Vol. 6, No. 1, pp12-24, 1990.

[39] Young, R.M., Pollack, M.E. and Moore, J.D., Decomposition and Causality in Partial-Order Planning, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), Chicago, IL, USA, 1994.

Appendix K – Reasoning with Constraints within O-Plan2

Austin Tate, Brian Drabble and Jeff Dalton

Appears in the Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. Burstein. M.), Tucson, Arizona, USA, Morgan Kaufmann, 1994.

Reasoning with Constraints within O-Plan2

Austin Tate, Brian Drabble and Jeff Dalton

Abstract

O-Plan2 is a command, planning and control architecture which has an open modular structure intended to allow experimentation on or replacement of various components. The research is seeking to isolate functionality that may be generally required in a number of applications and across a number of different planning, scheduling and control systems.

This paper describes the way in which plan constraints are represented and handled in the O-Plan2 architecture. It gives details of a rational reconstruction of the constraint management interfaces now being used as a design principle within the latest version of O-Plan2.

The cooperative manipulation of constraints on plans by a user and by the capabilities provided in computer systems provides a useful and natural paradigm for effective planning and scheduling support systems. The provision of powerful computer based constraint management languages and tools could lead to a rapid expansion of the benefits to be gained by identifying more standard ways in which constraints can be handled in future planning and scheduling systems.

1 O-Plan – the Open Planning Architecture

The O-Plan2 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-Plan2 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 planning systems. See [4] for details of O-Plan (now referred to as O-Plan1), the planning system that was a forerunner to the O-Plan2 agent architecture. That paper also includes a chart showing how O-Plan relates to other planning systems.

The O-Plan2 system combines a number of techniques:

- A multi-agent approach to strategic task assignment, tactical planning elaboration, and operational plan execution support.
- A control architecture within each agent in which each control cycle can post further processing steps on an agenda which are then picked out and processed by appropriate handlers (Knowledge Sources).
- The uniform treatment of the user in the role of planner and computer based planning capabilities as 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.
- A hierarchical planning system which can produce plans as partial orders on actions.

- Constraint posting and least commitment on object variables.
- Temporal and resource constraint handling using incremental algorithms which are sensitively applied only when constraints can alter.
- O-Plan2 is derived from the earlier Nonlin planner [12] from which it takes and extends the ideas of Goal Structure, Question Answering (Truth Criterion) and typed conditions.
- We have extended Nonlin's style of task description language Task Formalism (TF).

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 and satellites such as VOYAGER, ERS-1, etc.

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 *execution system* seeks to carry out the detailed activities specified by the planner while working with a more detailed model of the execution environment.

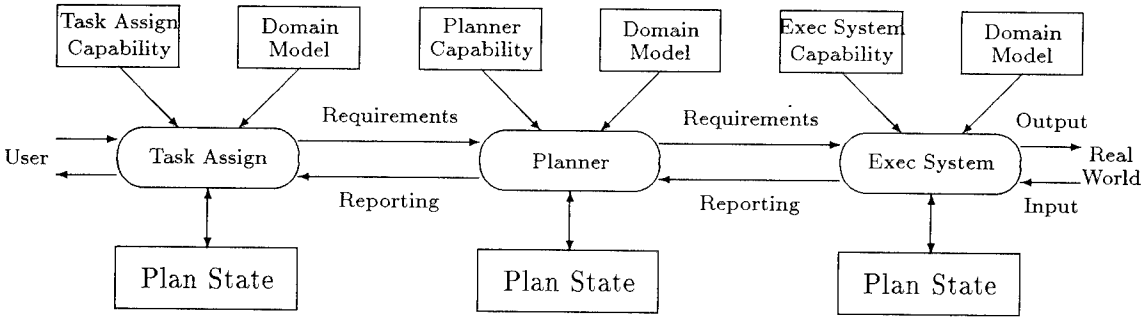


Figure 1: Communication between Strategic, Tactical and Operational Levels

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. See Figure 1.

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.

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 (*Plan Modification Operators* – PMOs).
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.

The planner components are described in outline form in Figure 2. More detail of the internal structure of O-Plan2 can be found in [15].

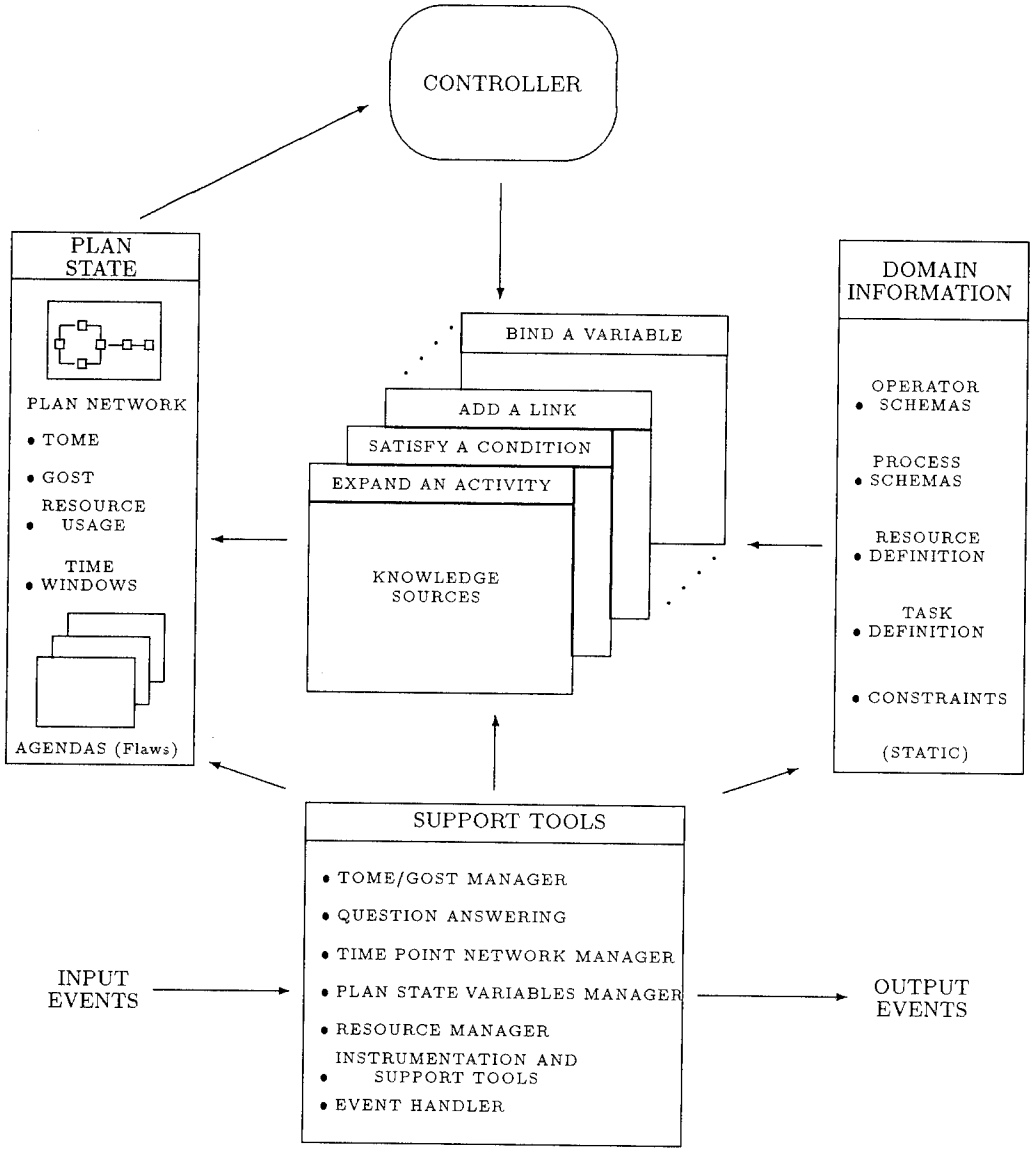


Figure 2: O-Plan2 Architecture

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.

An interface to AutoCAD has been built to show the type of User Interface we envisage (see Figure 3). 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 showing the plan 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 and the interfaces between these and the planner uses a defined interface to which various implementations can conform. Most of the developer aspects of the planner interface are not shown to the normal user. In figure 3, the developer windows are shown in iconic form along the top edge of the screen.

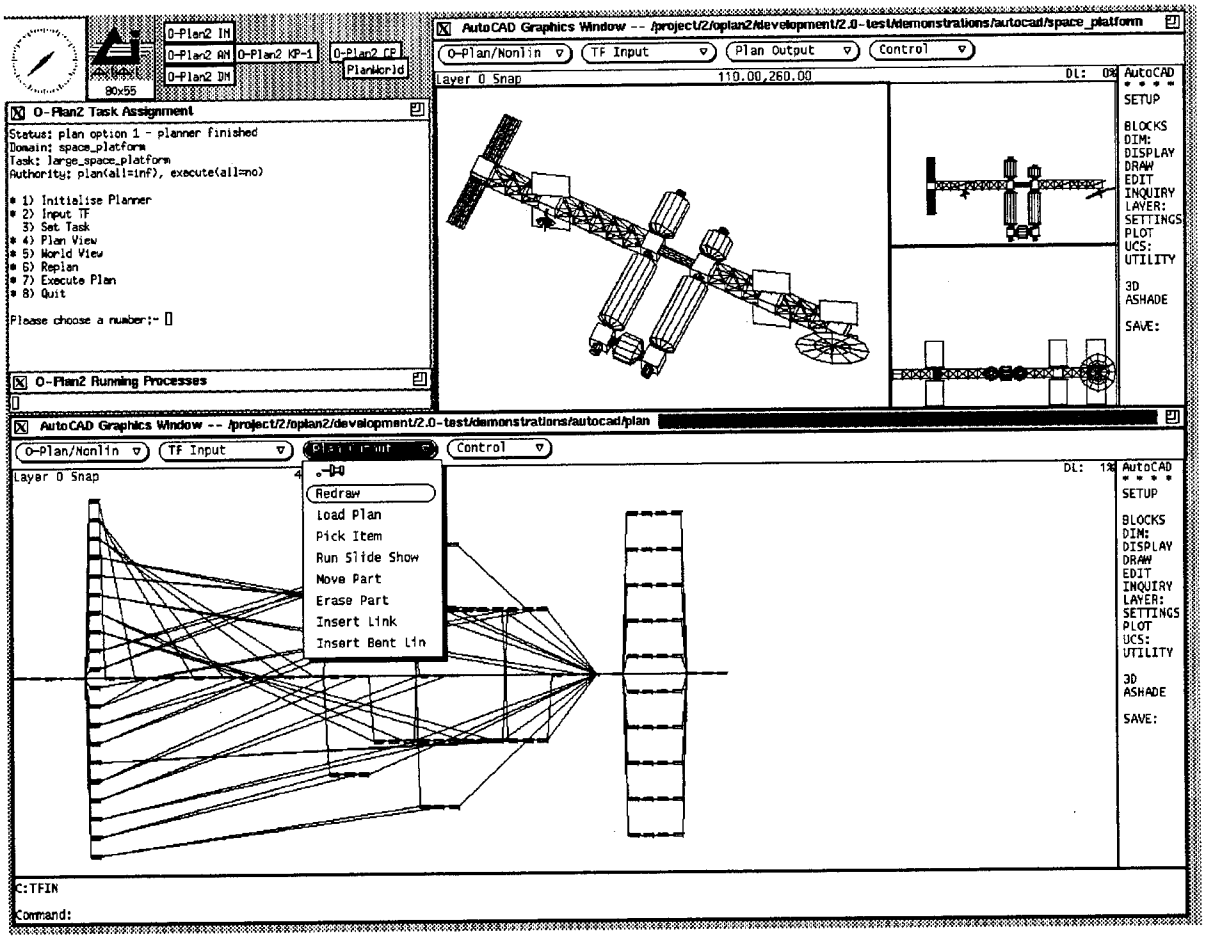


Figure 3: Example Output of the AutoCAD-based User Interface

2 Plans Represented as Constraints on Plan Elaborations

It is useful to present a simple abstraction of how a planner or scheduler operates. Figure 4 shows such an abstraction that will be useful in this paper.

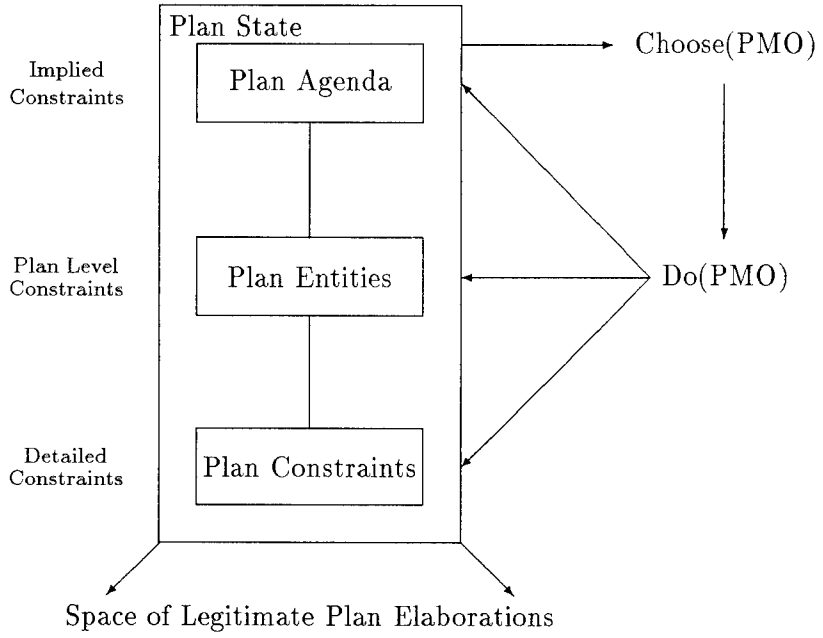


Figure 4: A Framework of Components in a Planning/Scheduling System

Many planners and schedulers work by refining a "current" plan (shown in figure 4 as the *Plan State*). They maintain one or more *partial plans* in this Plan State in which the previous decisions taken during the planning process restrict the space of plan elaborations which can be reached from that point.[1] The planner or scheduler needs to know what outstanding processing requirements exist in the plan (shown in figure 4 as the *Agenda*). These represent the implied constraints on valid plan solutions. One (normally) of these outstanding processing requirements is chosen to be worked upon next. This calls up processing capabilities within the planner which can make decisions and modify the Plan State - these are sometimes called *Plan Modification Operators*. The modifications can be in terms of definite plan structure in the Plan State or by noting further processing requirements (as a result of Plan State critiquing, etc).

We have found it to be useful to separate the plan entities representing the decisions already made during planning into a high level representing the main plan entities shared across all planning system components and known to various parts of the systems, and more detailed specialised plan entities which form a specialised area of the representation of the plan. These lower level more compartmentalised parts can represent specialised constraints within the plan such as time, resource, spatial and other constraints. This separation can assist in the identifi-

[1] Plan constraint relaxation is also possible to increase the space of plan elaborations in some systems.

cation of modularity within planning and scheduling systems.

O-Plan2 has an *Associated Data Structure* (ADS) level of representation [7] which holds the main plan entities (such as activities). The lower level constraints then separately handle constraints on ordering and time points in the plan, resource constraints, etc. The lower level constraints are tied to the higher ADS level entities via associations. The TOSCA manufacturing scheduling system [2] which was based on the O-Plan2 architecture makes use of quite a different ADS level based on resource reservations, but shares the same time point constraint management code at the lower level.

3 Benefits of "Standardising" Constraint Management in Planners

Moves to provide powerful constraint management languages and tools could lead to a rapid expansion of the benefits to be gained by identifying more standard components that can be combined and re-used in planning and scheduling systems. This can allow time network management, management of the persistence of facts across time, resource management, spatial constraint management and other such constraints to be managed by separate components provided by someone other than the original developer or integrator.

As one example, consider the provision of the management of temporal relationships in a planner. All modern planners embed some degree of time management for temporal relationships between time points or across time intervals and may provide support for metric (definite) time "stamps" on time points. Many planners also relate their time management to the management of the persistence of facts or propositions across time. This allows planners to reason about whether some required condition is true at a given time. The Time Map Management concepts, clearly described in [5] and used in the FORBIN planner [6], are a good example of the approach. The management of effect and condition (Goal Structure) tables in Nonlin [12] uses a similar approach.

This type of packaging has led to separate study of the support for time management and fact persistence management in planners at various research centres. O-Plan2 has a Time Point Network Manager [7]. A commercial Time Map Manager (TMM) is available from Honeywell based on the concepts described in [5]. More powerful temporal relationships are managed by the General Electric TACHYON temporal system [10]. In some cases, it has already proved possible to replace some simpler level of time constraint management in a planner with a better packaged and more powerful capability. One example of this has been the combining of the SRI SIPE-2 planner with the GE TACHYON temporal system. Other studies have indicated that the O-Plan2 TPNM can be replaced quite straightforwardly with the Honeywell TMM.

Studies at Edinburgh [8] relating to Resource Management have shown how progressively more capable resource management systems can be incorporated into O-Plan2 to replace the simple consumable resource handler in the system at present. These studies have developed a *Resource Criterion* interface to a Resource Utilisation Manager for the O-Plan2 planner which has many similarities to the interface used for the Truth Criterion/QA algorithm. This mechanism could allow resource handling by mechanisms as powerful as those based on the Habographs [2] con-

straint management mechanism incorporated in the Edinburgh TOSCA manufacturing scheduler.

Spatial constraint management which is not currently provided inside O-Plan2 has also been explored. We believe that clear modular interfaces can allow even such a "foreign" type of constraint management not understood by the core system at all to be be added reasonably straightforwardly to O-Plan2.

4 Constraint Managers in the O-Plan2 Architecture

O-Plan2 uses a number of *Constraint Managers* to maintain information about a plan while it is being generated. The information can then be used to prune search (where plans are found to be invalid as a result of propagating the constraints managed by these managers) or to order search alternatives according to some heuristic priority.

It is intended that some of these Constraint Managers could be replaced by more efficient or more capable systems in future. This section considers the interfaces between the O-Plan2 architecture components and Constraint Managers to help others consider packaging and integration issues.

Our experience with earlier AI planners such as Nonlin and O-Plan1 was that a large proportion of the processing time of a planner could be spent in performing basic tasks on the plan network (such as deciding which nodes are ordered with respect to others) and in reasoning about how to satisfy or preserve conditions within the plan. Such functions have been modularised and provided in O-Plan2 as Constraint Managers (such as a Time Point Network Manager, an Effect/Condition Manager and a Resource Utilisation Manager), and Support Routines (such as a Graph Operations Processor) to allow for future improvements and replacement by more efficient versions.

Constraint Managers are intended to provide efficient support to a higher level of the planner where decisions are taken. They should not take any decision themselves. They are intended to provide complete information about the constraints they are managing or to respond to questions being asked of them by the decision making level. Examples of Constraint Managers in O-Plan2 include:

- Time Point Network Manager (TPNM).
- Effect/Condition (TOME/GOST) Manager (TGM) and the related Question Answerer (QA).
- Resource Utilisation Manager (RUM).
- Object Instantiation (Plan State Variables) Manager (PSVM).

A guideline for the provision of a good Constraint Manager in O-Plan2 is the ability to specify the calling requirements for the module in a precise way (i.e. the *sensitivity rules* under which the Constraint Manager should be called by a knowledge source or from another component of the architecture).

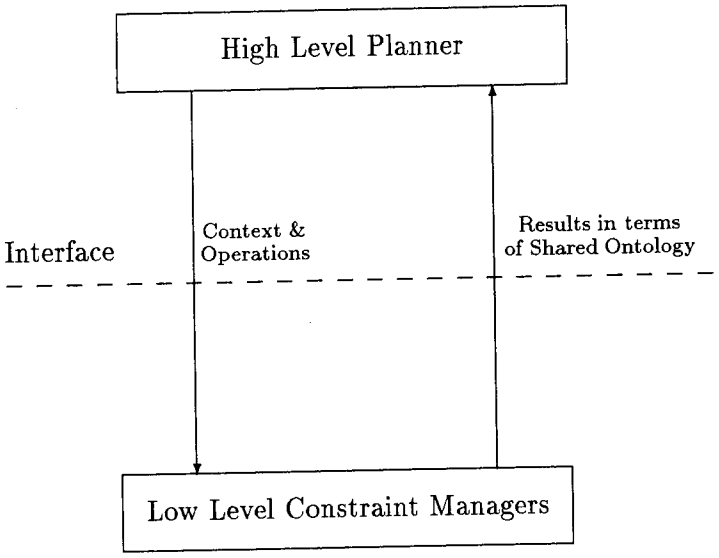


Figure 5: The Interface to Constraint Managers

The following sections explore the definition of an interface between the higher level decision making part of a planning or scheduling system and a lower level constraint manager. Figure 5 shows an overview of the interface.

4.1 Constraint Manager Procedural Interface

A Constraint Manager is a part of the Database Manager (DM) component in O-Plan2 which looks after the Plan State and all of its alternatives (if any). A Constraint Manager may look after a specialised aspect of the Plan State on behalf of the DM.

The O-Plan2 design is being rationalised so that a Constraint Manager has the following generic procedural interface:

1. initialise Constraint Manager and name base context with given <tag>[2].
2. terminate Constraint Manager
3. push context and name new context with given <tag>
4. pop context to parent of current context

[2]Contexts specify alternative views of a Plan State. A tree of such contexts is manipulated by O-Plan2.

5. restore a previously created context which has the <tag> specified

6. open update transaction, and within this allow:

   - allow changes to managed entities[3].
   - queries can be made inside an open transaction. Any query reflects the changes made within the transaction to date.
   - nested open update transactions are not allowed (in O-Plan2 at present).

7. commit changes made within the update transaction

8. abort changes made within the update transaction

Some of the above routines may be inoperative or null for specific managers. In particular, context management as specified above is not needed for any Constraint Manager which chooses to make use of the O-Plan2/O-Base context managed structures – since the Associated Data Structure (ADS [7]) layer in O-Plan2 guarantees that Constraint Managers will only ever be called when the contexts being referred to are preset within the O-Plan2 planner.

4.2 Shared Plan Ontology between O-Plan2 and Constraint Managers

There are specialised update and query routines supported by each constraint Manager. These share a common plan entity model within the planner and its Associated Data Structure (ADS) layer. The design intention has been to keep this minimal, including only those elements that allow relevant communication between higher level planning decisions and lower level constraint management. This model includes *only*:

- a directed acyclic graph of time points.
- ability to map a plan activity node end to a unique time point and a time point to all associated node ends.
- time points as plan entities.
- an ordering relation on two time points – before(tp1,tp2).
- context <tag>s to represent alternative Plan States.
- An understanding of the meaning of a Plan State Variable[4].

These entities allow for information to be communicated about constraints and options for correcting constraint violations in terms of the shared model. All other more specific entities may be unique to a specific Constraint Manager or shared only between pairs of caller and manager.

[3] An extra standard update routine is needed in our implementation to handle O-Plan2 TF **other_constraints** statements (constraints not directly understood by the planner) relating to this particular constraint manager.

[4] The exact nature of what needs to be understood in the shared ontology needs to be considered further.

4.3 The New O-Plan2 "Standard" Interface for Constraint Managers

The aim in O-Plan2 is to provide a standardised interface between each Constraint Manager and the rest of the planner. For this we are seeking to employ a very similar interface to that used by the Nonlin or O-Plan style Condition Question Answerer (QA) or Truth Criterion.

A Constraint Manager cannot take any decisions and cannot change parts of the Plan State not under its immediate management. It must return all legitimate answers for the query it is given or must undertake reliably the task it is given. One focus of the O-Plan2 research has been to build a *planning ontology* which describes those concepts which are shared between constraint managers and those parts of the Plan State which are private to the relevant manager.

A Constraint Manager's primary function is to manage the current set of constraints relevant to that manager (time, resource, spatial, objects, etc) which are part of the Plan State. It must signal to the caller when there is an inconsistent set of such constraints.

The interface allows for a constraint entry to be tested against existing managed constraints to see what the impact of making the entry would be, and then a commit or abort can be done to add it or not (either the commit or the abort could be active – the caller not being able to tell).

All Constraint Manager update routines return one of three results:

- **yes** – constraint is now under management (to be confirmed later by a caller using a commit update transaction).
- **no** – constraint cannot be added within the capabilities of the Constraint Manager and its communications capability to the caller (in terms of the shared ontology of entities).
- **maybe** – constraint can be added if plan entities are altered as specified in terms of the shared entity model. This normally means returning a standard O-Plan2 "or-tree"[5] of *all* (for search space completeness) the legal ways in which the Plan State can be altered (sets of Plan State Variable restrictions and ordering constraints between time points) to maintain consistency.

  The constraint is *not* added after this maybe response. However, an "actually add constraint" routine may be provided to more cheaply add the constraint immediately following a query which returned "maybe". This would follow action by the caller to ensure at least one of the relevant binding constraints and/or time point orderings options were either dealt with or noted as necessary in the Plan State - thus the caller takes responsibility for resolving inconsistencies (*not* the Constraint Manager).

It is hoped to be able to take the result or-trees generated by the various Constraint Managers in O-Plan2 (TGM, RUM, PSVM and the TPNM) and merge them into a consistent or-tree which would represent an efficiently ordered set of possibilities – thus reducing the size of the search space.

[5] a data structure representing the alternative ways in which the Plan State may be altered in terms of the shared plan ontology.

5 The Constraint "Associator"

To improve the separation of functionality with respect to constraint management in O-Plan2, we wish to localise the interactions between changes in one type of constraint that can lead to changes in other types of constraint. This has been problematic in O-Plan2 to date. In particular, changes in constraints on time points and changes to constraints on plan state variables can have implications for most other constraints being managed (such as effect/condition, resources, etc. constraints). Previously Knowledge Sources had to be written such that any change in one constraint type that could influence another was programmed in.

The clarification of constraint manager interface for O-Plan2 as described in this paper has made us realise the special requirements for the handling of time point constraints and variable constraints in the architecture. These form the core elements in the shared ontology in which communication occurs between the plan entity (ADS) layer and the constraint managers in O-Plan2. By recognising that there is a normal constraint management function for time points and variable, but also an *additional* function of association and mutual constraints with other constraint types, we can design better and more modular support for constraints handling in O-Plan2 and simplify the writing of Knowledge Sources.

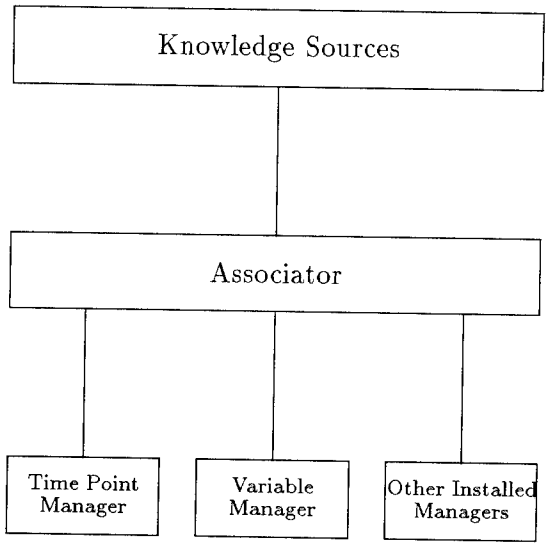


Figure 6: Associator to mediate between Knowledge Sources and Constraint Managers

Accordingly, the O-Plan2 agent architecture design in future will allow for an "Associator" component as part of the data base manager which looks after plan states. The Associator mediates between the decisions made by Knowledge Sources and the underlying constraint managers (see

figure 6). A number of constraint managers can be "installed" into an O-Plan2 agent. As a minimum, each agent will have a time point manager and a variables manager installed into the Associator. Any number of other constraint managers may then be added depending on the requirements. In the current planner this will include the effect/condition manager, the resource utilisation manager, and an "other constraints" manager to keep annotations of other requirements on a plan state. In other applications it may be necessary to include spatial constraint managers, etc.

We believe that this style of interface between the higher level decision making level of the planner and the various Constraint Managers could improve modularity in planning systems.

6 Summary

This paper was intended to further discussions on the identification of suitable "standard" re-usable components in planning and scheduling systems.

This paper has presented an overview of the O-Plan2 system under development at the Artificial Intelligence Applications Institute of the University of Edinburgh. Aspects of the system concerned with separation of functionality within the system, internal and external interfaces have been addressed. The O-Plan2 system is starting to address the issue of what support is required to build an evolving and flexible architecture to support command, planning and control tasks.

One particular area highlighted has been the interface between planning systems and Constraint Managers able to look after certain specialised aspects of parts of a plan on behalf of the overall planning system. An interface to such Constraint Managers has been developed to show how improved packaging can be beneficial to re-use of components. The value of the type of interface developed for the Condition Question Answering procedure in planners (the Truth Criterion) to act as a general interface to a number of different Constraint Managers has been explored.

Acknowledgements

O-Plan2 is an on-going project at Edinburgh. 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.

Parts of this paper were previously presented at the European Workshop on Planning Systems 1993 (EWSP-93), December 1993, Linkoping, Sweden.

References

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

[2] 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. Also available as AIAI Technical Report AIAI-TR-121.

[3] Chapman, D. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333-377, 1991.

[4] Currie, K.W. & Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence* 51(1), Autumn 1991, North-Holland.

[5] Dean, T. and McDermott, D., Temporal Database Management, *Artificial Intelligence* 32(1):1-56, 1987.

[6] Dean, T., Firby, J. and McDermott, D., Hierarchical Planning Involving Deadlines, Travel Time and Resources, *Computational Intelligence*, 6(1), 1990.

[7] Drabble, B. and Kirby, R.B., Associating A.I. Planner Entities with an Underlying Time Point Network, European Workshop on Planning (EWSP) 1991, Springer-Verlag Lecture Notes in Artificial Intelligence. Also available as AIAI Technical Report AIAI-TR-94.

[8] Drabble, B. and Tate, A., Resource Representation and Reasoning in O-Plan2, AIAI Technical Report ARPA-RL/O-Plan/TR/6, April 1993.

[9] Sacerdoti, E., *A Structure for Plans and Behaviours*, Artificial Intelligence Series, North Holland, 1977.

[10] Stillman, J., Arthur, R. and Deitsch, A., Tachyon: A Constraint-based Temporal Reasoning Model and its Implementation, *SIGART Bulletin*, 4:3, July 1993.

[11] Sussman, G.J., A Computational Model of Skill Acquisition, MIT AI Laboratory Technical Report TR-297, 1973.

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

[13] 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.

[14] Tate, A., Goal Structure, Holding Periods and "Clouds", Proceedings of the Reasoning about Actions and Plans Workshop, Timberline Lodge, Oregon, USA, (eds, Georgeff, M.P. and Lansky, A.) Morgan Kaufmann, 1986.

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

[16] Wilkins, D., *Practical Planning*, Morgan Kaufmann, 1988.

Appendix L – PlanWorld Viewers

Austin Tate and Brian Drabble

Appears in a shorter form in Proceedings of the 14th Workshop of the UK Planning and Scheduling Special Interest Group University of Essex, Colchester, Wednesday 22 November - Thursday 23 November 1995.

Sections 4 and 5 are based on material published in at 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. 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.

PlanWorld Viewers[1]

Austin Tate and Brian Drabble

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
- using opportunistic selection of the focus of attention on each problem solving cycle

[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.

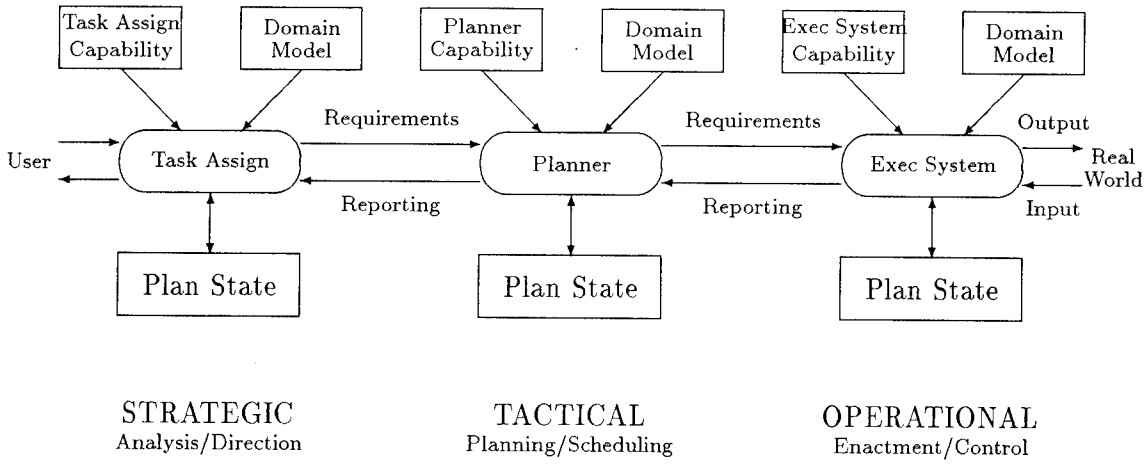


Figure 1: Communication between Strategic, Tactical and Operational Agents

- 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.

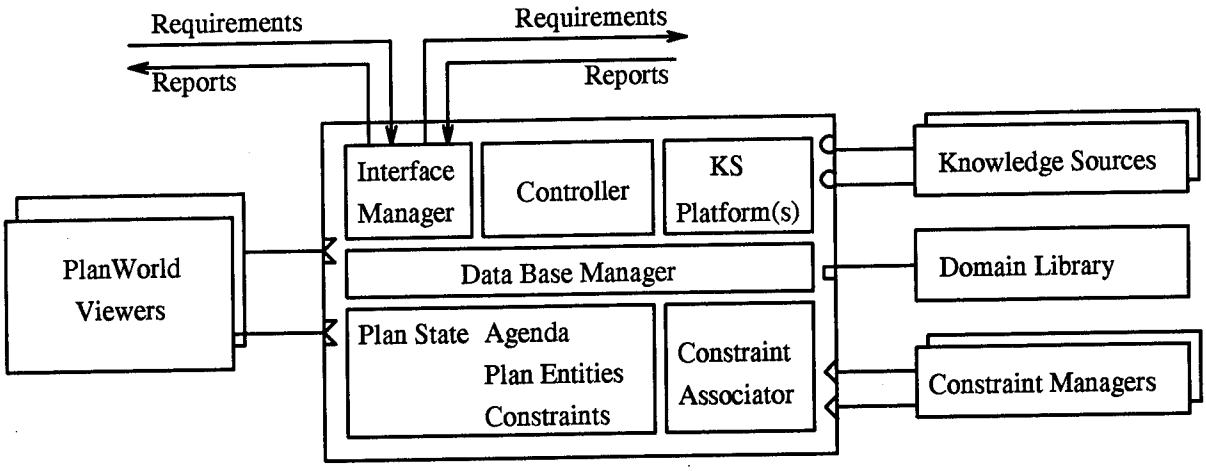


Figure 2: O-Plan Agent Architecture

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 Manager – 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.

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.

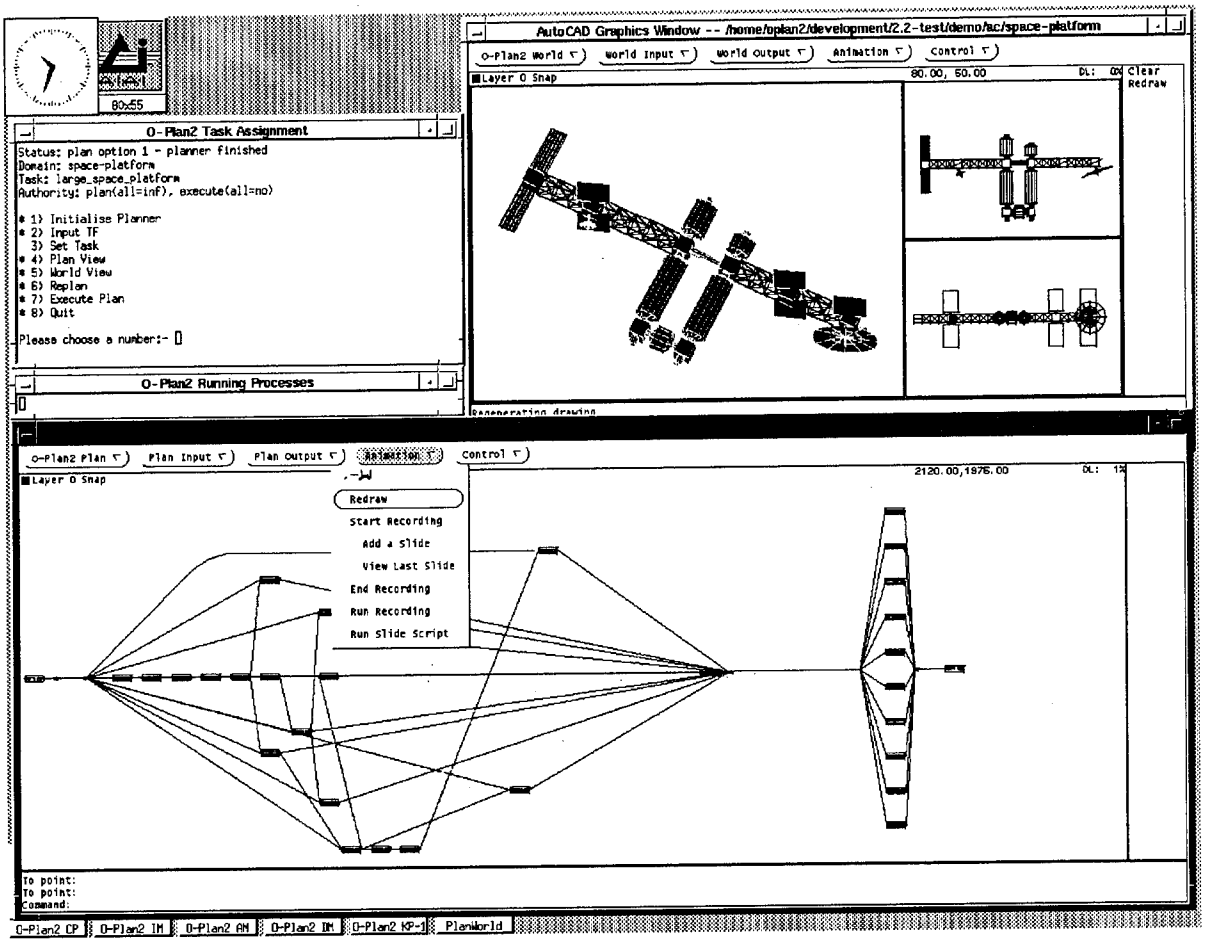


Figure 3: PlanWorld User Interface for Space Platform Construction

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.

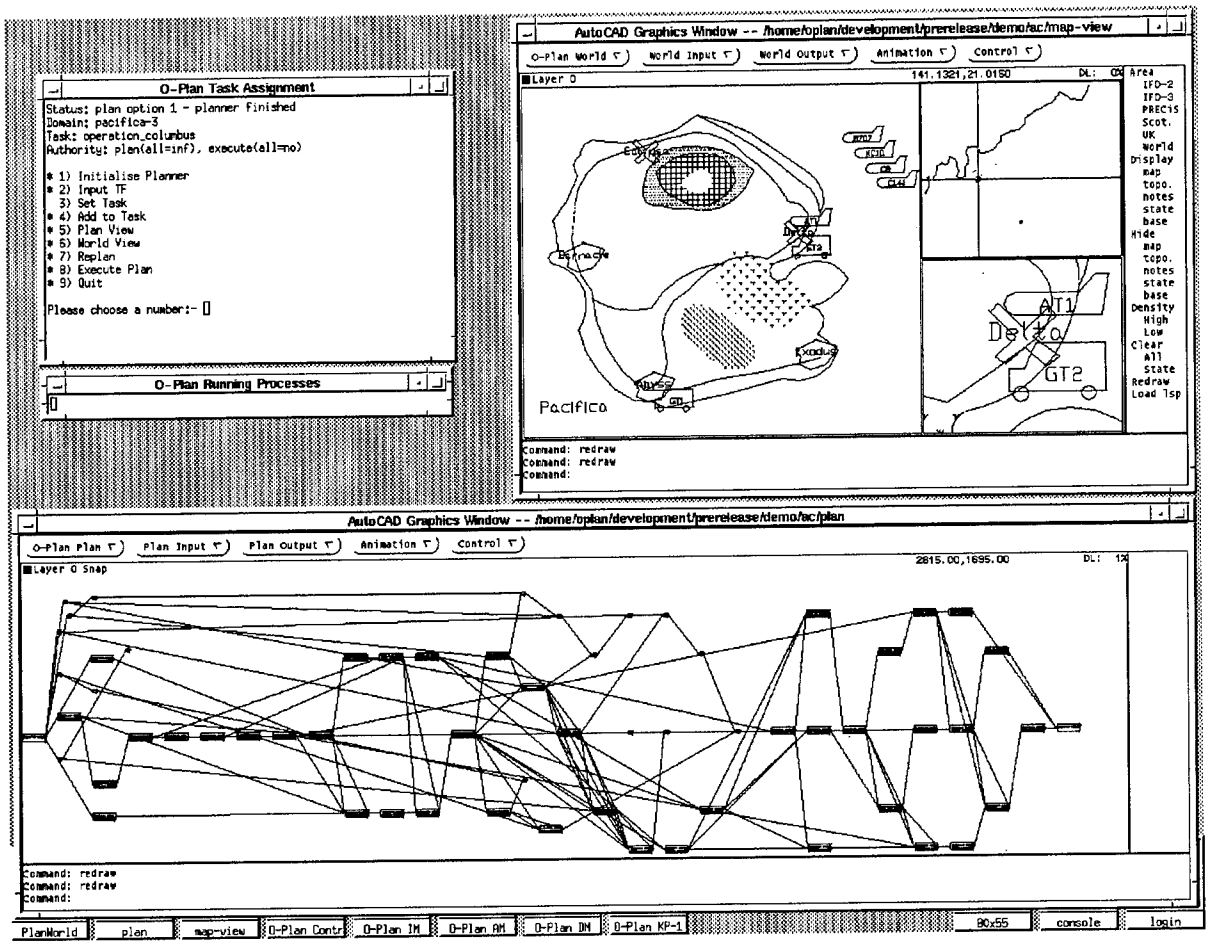


Figure 4: PlanWorld User Interface for Non-Combatant Evacuation Operations

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, re-organisation 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 demon-

strated 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.

Appendix M – Mixed Initiative Planning in O-Plan2

Austin Tate

Appears in the Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop at Tucson, Arizona, USA, (ed. Burstein. M.), Morgan-Kaufmann, 1994.

Mixed Initiative Planning in O-Plan2

Austin Tate

Abstract

The model of Mixed Initiative Planning that can be supported by the O-Plan2 architecture is the mutual constraining of a set of alternative partial plans for some task set.

This paper describes the opportunities for mixed initiative planning within the O-Plan2 architecture. Both the user and the system can work in harmony and neither is seen as at a higher level or "in charge" as far as the architecture is concerned. Ordering and priorities can be applied to impose specific styles of authority to plan within the system. One extreme of user driven plan expansion followed by system "filling-in" of details, or the opposite extreme of fully automatic system driven planning (with perhaps occasional appeals to an user to take predefined decisions) are possible. In more practical use, we envisage a mixed initiative form of interaction in which the user and system proceed by mutually constraining the plan using their own areas of strength.

Appendices describe in more detail the use of the KS-USER knowledge source to "wrap" around the interfaces provided to the user and to ensure integrity of the system, the various user roles identified within the O-Plan2 design, details of the search space explored by O-Plan2 and other relevant information concerning mixed initiative planning within O-Plan2.

1 Partial Plans as a Set of Constraints

An O-Plan2 plan is viewed as containing a set of constraints on the possible plan elaborations that can be entertained. Users and system in a mixed initiative way jointly add (or relax) constraints in the plan as planning proceeds.

A plan conceptually has three levels

1. implied constraints (called the plan agenda)
2. plan level entities (decided upon plan components at various levels of abstraction)
3. detailed constraints (for time, resources, authorities, conditions/effects, object selections, spatial use, etc. These are associated with the plan level entities)

Explicit options (Courses of Action) may exist for plans and may share a lot of common structure.

At any time the system or an user can work on this set of constraints – normally being directed by the agenda.

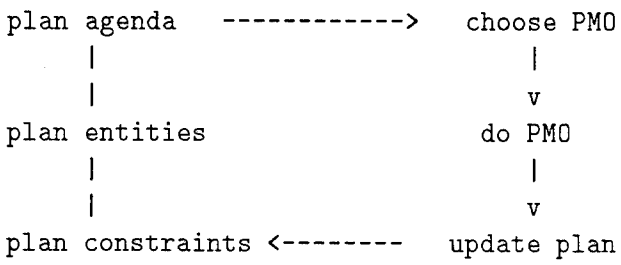
2 Plan Agenda and Control

The agenda keeps everyone straight about what remains to be considered. Inconsistent plans and partially elaborated plans are possible using the agenda to represent such outstanding issues.

Focus of initiative is determined by decisions on the order in which agenda entries are processed. This ordering decision process is separate to the involvement of the user in constructing a plan.

Mixed initiative is possible. The user can delegate to the system by adding suitable agenda entries (implied constraints) for parts of the work that the system can handle best. The system can seek help from the user via the same mechanism. Or both can take a look at what they can do with the agenda. Critiquing can lead to new agenda entries to work on.

3 Plan Modification Operators – Knowledge Sources



Users or the system can (when sanctioned or authorised) work on anything that is outstanding on the agenda. They do this through the "wrapper" of a Plan Modification Operator (PMO). This is like a knowledge source in blackboard systems. An user also interacts via a PMO wrapper to ensure that plan integrity is maintained.

All decision making processing which can alter a plan is done via plan modification operators. The recording of dependencies and who is responsible for changes to the plan is possible in such PMOs to support later plan changes and constraint relaxation. This has been done in some versions of our planners at Edinburgh.

PMOs (called knowledge sources in the O-Plan2 architecture) can run on one or more knowledge source "platforms". Concurrency is possible with multiple platforms. Real time capabilities can be assured by having dedicated platforms for a nominated knowledge source. One or more platforms can run knowledge sources to provide the planner user interface(s).

4 Plan Entities and Detailed Constraints

The user or the system made alterations to a plan are done at the plan entities layer – which expresses most of what can be thought of as the "interesting" contents of the plan, and certainly contains what the users are likely to want to work with. Whether user or system decisions lead to a change of the plan entities, they are subject to lower level constraint management at a detailed level where critiquing of the changes and inconsistency issues are raised directly with the caller or via the agenda. The constraint manager level NEVER takes any decisions, so the user and the decision making level of the system can maintain a simple view of what is going on and who is changing what.

5 User Roles

O-Plan2 identifies quite separate user ROLES with respect to planning. The discussion above relates mostly to the role we call PLANNER USER. We identify other user roles which are quite distinct. For example, an user in a system developer sense is not confused with the PLANNER USER role.

One important distinct role with respect to mixed initiative planning is the role of "user support to agenda controller" which is the place where an user can assist in deciding which agenda entries to process next (i.e., choose a PMO to process on an available knowledge source platform) and thus where the focus of initiative between the planner user and the system plan modification operators lies. This separation of roles allows a better understanding of what an user is doing and what the user's intentions are.

6 User Interfaces

We have characterised features of the User Interface for the O-Plan2 planning system and provide appropriate support for various user roles. We have developed flexible interface specifications between a plan state and users who want quite different views of the plan. Technically orientated PERT diagrams and Gantt charts, resource profiles, etc., can be the means of interaction for some. For others good domain orientated displays, maps, animations or simulations convey much more. O-Plan2 does this via a PlanWorld Viewer interface specification which allows quite disparate external viewers to be connected to the planning support tools. Map displays fit naturally into this, as do AutoCAD style systems, etc. – all with minimal work to write the adaptor code. We have already had discussions with BBN about the ARPI CPE and ways to integrate the TARGET interface and approach with that used in O-Plan2 – and this seems possible.

7 3 Level Model – Strategic, Tactical and Operational Support

The O-Plan2 architecture identifies three levels at which different types of task are performed in a command, planning and control environment.

Strategic: task characterisation: analysis and direction.

Tactical: task characterisation: synthesis.

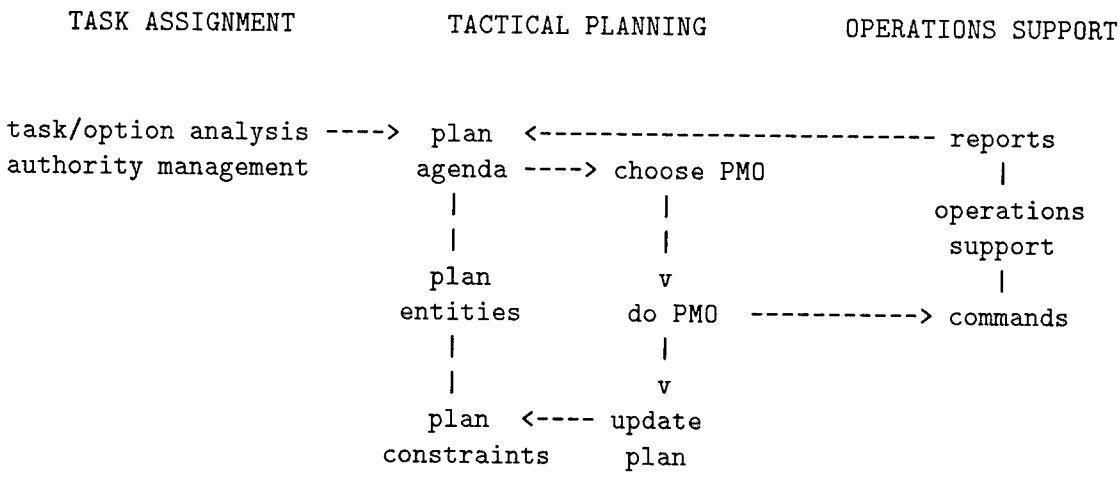
Operational: task characterisation: modification and control.

For more "strategic" plan decisions, the agenda level and manipulation of WHAT to do rather than HOW to do it is more important. We believe that strategic "planning" does not employ the same techniques as "tactical" planning. The strategy level involves much more analysis,

option comparison and direction. The tactical level is closer to what is termed generative AI planning and the resource scheduling we see in manufacturing and logistics support systems today. Similarly, the operational level has different priorities and requirements to the higher strategic and tactical levels. All three levels are needed, and the O-Plan2 architecture is designed to accommodate them all. A clear direction in the form of a 'Task Assignment" is needed from the strategic decision making level to drive the tactical level.

A clear characterisation and analysis of different "task types" has emerged from knowledge engineering. There are some ten basic task types often identified (diagnosis, interpretation, classification, planning, monitoring, learning, etc.). The KADS methodology separate these into three classes: analysis, synthesis and modification tasks. This is consistent with the separation of the roles of the three levels in the O-Plan2 architecture. It is also common practice to relate to these three levels in many organisations – whether military or otherwise.

The core O-Plan2 planner model described earlier fits into this three level (Task Assignment, Tactical Planning, Operations Support) framework as shown in the diagram below.



8 Focus of Initiative

In O-Plan2, there is separation of decisions on focus of initiative at 4 main points. These relate to different types of task.

- mission tasking, option analysis, authority management and direction.

  Task characterisation: analysis and direction.

  Initiative: normally manual.

  O-Plan2 Agent/Component: Task Assignment agent.

- decisions on what to work on next for the human and system components given available human planner and system computational resources.

  Task characterisation: interpretation and classification.

Initiative: normally automatic using pre-defined priorities with the possibility of manual override.

O-Plan2 Agent/Component: Planner agent/controller dispatching to planner agent/knowledge source platforms.

- decisions to add (or relax) constraints on important plan entities.

  Task characterisation: synthesis.

  Initiative: opportunistic with mixed automatic and manual possibilities. System support if constraints are relaxed is essential due to the potential ramifications of such change.

  O-Plan2 Agent/Component: Planner agent/knowledge sources.

- detailed constraint propagation and projection.

  Task characterisation: algorithmic.

  Initiative: normally automatic, with human assistance for speed-up.

  O-Plan2 Agent/Component: Planner agent/constraint managers.

9 Authority Management

It is important to clarify the description of authority for the planner user and system. Authority to plan to given levels of detail for certain parts (phases) of certain plan options, and permission to execute parts of plans should be given explicitly. For example, "give me a CONPLAN for the DEPLOYMENT phase of a specifically nominated COA we are discussing", or "execute the MOBILISATION phase of a specific COA we are discussing".

The O-Plan2 plan representation recognises:

- named plan options
- named plan phases
- named plan levels of abstraction

O-Plan2 research has begun to explore issues of clearer authority management and representation between agents involved in command, planning and control. The O-Plan2 Architecture and plan representation allows a simple form of authority management at present. The current O-Plan2 user interface makes allowance for later more sophisticated authority management.

10 Summary

The O-Plan2 architecture has been designed to support advanced research and prototype development for flexible next generation support systems for command, planning and control environments. This paper has shown how the current architecture already goes some way towards addressing key research and development issues to support flexible mixed initiative planning.

References

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

[2] Beck, H. *TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints*, in Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, Amsterdam, 12-14 May 1993, (eds, C. Kooij and P.A. MacConaill and J. Bastos), pp138-149.

[3] Choueiry, B.Y. and Faltings, B. Resource Allocation by Problem Decomposition and Temporal Abstractions, in Proceedings of the Second European Workshop on Planning (EWSP-93), Vadstena, Sweden, IOS Press.

[4] Currie, K.W. and Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence* 51(1), Autumn 1991, North-Holland.

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

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

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

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

[9] Tate, A, INTERPLAN: a plan generation system which can deal with interactions between goals, Research Memorandum MIP-R-109, Edinburgh: Machine Intelligence Research Unit, December 1974.

[10] Tate, A, Using Goal Structure to direct search in a problem solver, Ph.D. Thesis, University of Edinburgh, September 1975.

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

[12] Tate, A., The Emergence of "Standard" Planning and Scheduling System Components, in Proceedings of the Second European Workshop on Planning (EWSP-93), Vadstena, Sweden, IOS Press.

APPENDIX M-A – The KS-USER Knowledge Source in O-Plan2

The O-Plan2 architecture allows for a KS-USER knowledge source. Knowledge sources are the only places in which decisions relating to the plan entities are taken – other parts of the system being concerned with the ordering in which decisions are taken, and the management of plan states and the constraints included in them.

The KS-USER knowledge source allows a planner user to take decisions within the framework of the architecture. The user can take the initiative by asking for the KS-USER knowledge source to be activated to allow the plan to be viewed and decisions made, constraints applied, etc. Alternatively, the system can seek user input and decisions by asking the KS-USER knowledge source to seek certain kinds of input from the user. Hence both planner user and system are working in harmony and neither is seen as at a higher level or "in charge" as far as the architecture is concerned. Ordering and priorities can then be applied to impose specific styles of authority to plan within the system. One extreme of user driven plan expansion followed by system "filling-in" of details, or the opposite extreme of fully automatic system driven planning (with perhaps occasional appeals to an user to take predefined decisions) are possible. In more practical use, we envisage a mixed initiative form of interaction in which the user and system proceed by mutually constraining the plan using their own areas of strength.

O-Plan2 Design Rationale for KS-USER Knowledge Source

The KS-USER knowledge source is intended to be the single point of interaction with the O-Plan2 planner agent for the user in the role of planner user. The planner user is intended to act at the same level as other decision making components of an O-Plan2 agent (i.e., has the same properties as a knowledge source).

For integrity of the manipulation of an O-Plan2 agent's plan state, the KS-USER knowledge source must respect the O-Plan2 knowledge Source Protocol in its dealings with the Controller (for spawning alternative plan states where necessary, or for adding agenda entries into a plan state). Its O-Plan2 Knowledge Source Framework description must be accurate in describing its read/write interaction requirements (of each knowledge source stage) on the plan state through the O-Plan2 Data Base Manager. Greater levels of concurrency are possible by specifying the interaction details in as constrained a way as possible where this is known.

There are two principal ways in which the planner user will interact with the system:

mode a) user wishes to intervene

mode b) system wishes user to intervene

In addition, there is a requirement for visualisation of some aspects of the plan (via the Plan-World Viewers). This may be at the request directly of the planner user (i.e., as in (a) above but where no changes are to be made to a plan state) or maybe to serve a request from outside the agent (for example, to provide a visualisation of the plan at the request of the Task Assignment

agent). So, we have a third mode of planner user support requirement for this latter service case:

mode c) planner user interface services to other agents

Earlier O-Plan1 systems (1984-1988) utilised a single KS-USER knowledge source for modes (a) and (b). The KS-USER knowledge source implemented in O-Plan2 up to version 2.1 is used for mode (c). In O-Plan2 up to version 2.1, some other user interface aspects related to mode (b) are incorporated in individual knowledge sources (such as KS-BIND). However, these were intended to be centralised in KS-USER in due course. Also, some aspects of support for mode (a) have been available via the system developer interface (the Data Base Manager Developer's menu and especially its break-in option). We now wish to demonstrate in an integrated way the proper support for mixed initiative planning within O-Plan2.

The aim will be to demonstrate the range of ways in which a planner user can interact with the system. These will show the mixed initiative properties of the O-Plan2 architecture in a realistic setting.

KS-USER Specification

KS-USER may be called in any one of three modes (indicated by an entry in the information field of the agenda entry passed to KS-USER).

mode a) User Request Mode

A button on each O-Plan2 agent control panel will allow the principal user of that agent to request interaction in their role as agent user (e.g., planner user role for the planner agent). An agent level agenda entry will be posted with USER REQUEST MODE indicated. This will lead to the activation of the KS-USER knowledge source installed in the agent.

At this level a menu of possible interaction options will be presented. The aim is to eventually provide very flexible editing of the current plan state and the ability to select from open alternatives (leaving those remaining to be handled by the controller), to re-order options available for schema choice, variable binding choice, ordering choice, etc. The immediate target is to provide support for the following:

1. Plan View
2. World View
3. Bind Variables
4. Break-in (with warning not to alter plan state improperly)
5. Quit

Bind Variables would be the only "sophisticated" part of the interface not currently available in KS-USER. This would find all open Plan State Variables (PSVs), and present these in a simple way with their current possible values and their restriction set. Perhaps some list of where the variables occurred in plan entities could also be given.

The interface would allow an user to:

1. select any open variable and to order the possible values
2. restrict any open variable (to one value or to some sub-set of values) (an alternative would be posted via the controller for the excluded choices to guarantee search space integrity).
3. commit valid changes made and quit from KS-USER
4. abort changes made and start again
5. quit from KS-USER

The choices would be made within a new "what-if" context layer such that the user could easily abort any sequence of decisions that was not useful.

It should be noted that sophisticated forms of user interface and compatible binding decision support could be possible in such an interface. We will only provide relatively simple forms in our implementation. One possible variant that would fit directly into the framework adopted would be the use of the VAD (Value-Assignment Delay) Heuristic and a supportive graphics interface for this as described in:

"Interactive Resource Allocation by Problem Decomposition and Temporal Abstractions", Berthe Y. Choueiry and Boi Faltings, AI Laboratory, Swiss Federal Institute of Technology, EPFL-Ecublens, CH-1015 Lausanne, Switzerland, Second European Workshop on Planning (EWSP-93), Vadstena, Sweden, IOS Press.

After any choice, the Plan State Variables (PSV) Manager would be allowed to propagate the consequences of the action taken, to check the immediately implied implications of the user action and to further constrain the remaining open variables.

mode b) System Request Mode

In O-Plan2, a KS-BIND agenda entry is posted to handle any outstanding PSV bindings. If the O-Plan2 control panel indicates that the user should be asked to make bindings for open variables, then when KS-BIND is activated it should delegate its job to a KS-USER agenda entry with a SYSTEM REQUEST MODE indicator for BINDING A VARIABLE and indicate the variable or variables involved. When activated, KS-USER will use the same interface as for Bind Variables under the USER REQUEST MODE described above. It may only allow the indicated variable(s) to be bound or may allow any variable that is still open to be bound (to be determined). If the planner user elects not to bind the variable(s) for which the system request was made, then a KS-BIND request with an automatic bind indicator should be posted to allow the proper termination of the knowledge source with responsibilities fulfilled.

mode c) Agent Services Mode

KS-USER may be called to service requests from outside (or possibly also inside) an O-Plan2 agent for user interface related access to the plan state via the PlanWorld Viewers. In this case the caller posts an agenda entry for KS-USER with the AGENT SERVICES MODE indicator and the specific service required. Currently we will support PLAN VIEW or WORLD VIEW from the Task Assignment agent.

APPENDIX M-B – User Roles in O-Plan2

User interaction with O-Plan2 can occur for a variety of purposes. Various *roles* of an user interacting with O-Plan2 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 provision.

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-Plan2 Task Formalism to define the domain information which the planner will use.

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-Plan2 Task Formalism to provide the detailed domain information which the planner will use.

Task Assignment User Role

The command user interacts only with the Task Assignment Agent to provide user requirements or commands. This is currently the top level menu for the O-Plan2 system. This user is responsible for the selection of the task which the system will try to carry out. The menu currently allows for a domain to be selected and for a selection from the task schemas within the Task Formalism for that domain to be selected. 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.

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.

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.

World Interventionist

If a world simulation is being used to demonstrate the O-Plan2 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.

User Support to Controller Role

The user may assist an O-Plan2 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.

User Support to Alternatives Handler

The user may assist an O-Plan2 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.

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-Plan2 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.

System Builder

The O-Plan2 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-Plan2 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-Plan2 Planner. This scheduler, called TOSCA (The Open SCheduling Architecture), also has different knowledge sources to those used in the O-Plan2 Planner.

APPENDIX M-C – O-Plan2 Planner Search Space Description

Characterisation of the Search Space

The O-Plan2 planner searches a space of teleologically novel solutions to the given task and only progressively expands the search space to include alternative means to satisfy the chosen teleological approach if a solution is not discovered earlier. In short we refer to this as a "teleologically novel, progressively extended search space".

The teleological approach defines the way in which conditions or resource requirements at activities within a plan are satisfied. O-Plan2 guarantees to produce at least one valid solution to a given problem if this is feasible within the constraints specified on the task and within the modelling capabilities provided by the constraint managers installed. It does this by systematically searching a lazy-generated space of solutions for the task given.

O-Plan2 does not guarantee to produce more than one such valid solution for any given teleological approach since it does not generate alternatives unless these prove necessary. It is therefore not suitable for problems in which all (syntactically different) solutions are required or in which an optimal solution is needed.

This approach to defining the search space of a planner was first introduced in INTERPLAN (Tate, 1974, 1975) and subsequently used in Nonlin (Tate, 1977) and O-Plan1 (Currie and Tate, 1991).

Search Space Node – A Plan State – A Conjunction of Constraints

A node of the search space is a partial plan (called a "plan state") which represents the conjunction of all constraints on the partial and fully elaborated plans which can be reached from that search space node without relaxing any constraint.

The systematicity of the search space is not compromised by selections of the order in which any component of the conjunct of constraints is refined. So simple or more complex opportunistic heuristics to select which constraint to refine are possible. Parallel constraint satisfaction techniques can also be utilised because of this property of a search node. The plan state structures themselves are designed to allow a large number of alternative solutions which do not affect other constraints to be built up (by "posting" a complete disjunction of alternative constraints into suitable structures within the plan state).

Search Space Arc – Plan Modification Operators

An arc of the search space represents the application of a Plan Modification Operator (PMO). An O-Plan2 Knowledge Source can branch the search and apply a single PMO in the new branch.

Relaxing the Constraints at a Node – Progressive Expansion (opening up) of the Search Space

On the search space leaves (fringe), if a PMO establishes that no further solutions are possible within the constraint set given using the current means of satisfying the teleological approach, then the PMO may "poison" that plan state (search node).

It is possible, at that time to consider whether alternative means to get to a valid solution within the defined teleological approach is possible. This is done by a special "poison handler" PMO. The poison handler contains all knowledge which the system has about alternative means to handle a given "poison cause". Only the poison handler has the authority to "extend" a poisoned plan state (search space leaf) and may do so progressively (only guaranteeing to use one of any alternate means it has at its disposal). Any relaxation of the search space whether done by the planner role user or the system should be delegated to the poison handler to allow it to maintain the integrity of the search space of the planner.

Mechanisms to Describe Disjunction in the Search Space

The designer of the O-Plan2 planner may choose from a number of mechanisms for holding disjunctions in the search space when seeking to implement the above search space definition. This can cut down on the search space that needs to be manipulated.

- Alternative Plan States.
- "Posting" Choices into separate Agenda Records (in their Information Fields).
- "Posting" Choices of the O-Plan2 Constraint Management Shared Ontological Elements in an "Or-tree" held in each separate Agenda Record (in their Information Fields).
- Progressive Expansion of Search via Poison Handler Capabilities.

APPENDIX M-D – Previous Work of Relevance to Mixed Initiative Planning – PLANIT

The paper "PLANIT Design Rationale and Future Directions" by Mark Drummond and Austin Tate (available as AIAI-TR-108 from AIAI, University of Edinburgh) reports on work conducted for a consortium of 27 organisations within the UK Alvey Programme in 1986-7. The PLANIT project produced an Interactive Planner's Assistant (IPA) that helped an user make use of an integrated set of knowledge rich plans, schedules and process plans for work in an enterprise.

Quotes from this paper:

> The PLANIT IPA (Interactive Planner's Assistant) can be considered as a "spreadsheet" which provides a constraint network linking the various entities involved in representing a "knowledge rich" plan. This model is an useful one and with more powerful representations and operational planning capabilities the systems of the future will be based on a similar notion.
>
> Changes to any part [of a plan state] will be reflected in other constrained parts by the use of suitable constraint propagation systems.

The Plan representation and techniques used within the PLANIT IPA were based on O-Plan research and have much in common with the design principles of O-Plan2. The IPA provided a supportive interface through which a planner role user could make legitimate constrained changes to a plan and could seek the support of the system via the automatic application of Plan Modification Operators in a single step fashion. The system could then propagate the consequences of the user's chosen higher level plan modifications to more detailed constraints within the plan. This allowed the plan to be critiqued following the user driven changes.

Appendix N – The PRECiS Environment

Glen Reece, Austin Tate, David Brown, Mark Hoffman and Rebecca Burnard

Appears in the Papers of the ARPA-RL Planning Initiative Workshop at Eleventh National Conference on Artificial Intelligence (AAAI-93), Washington D.C., USA, July 1993.

The PRECiS Environment

Glen Reece, Austin Tate, David Brown, Mark Hoffman and Rebecca Burnard

1 Introduction

Military crisis management is a complex problem which requires the active participation of countless planners at a myriad of locations around the world. From the joint Chiefs of Staff (JCS) and the Unified Commander's headquarters comes guidance and direction about what needs to be done. Geographically separated Supported Commander's component commands are responsible for detailed planning. These requirements are passed to the supporting commands who identify the specific units who will deploy to support the operation. When all of the thousands of details are completed, the plan is passed to the United States Transportation Command (USTRANSCOM) for analysis and possible implementation.

Planning is a time-consuming and cumbersome process where each participant plays his part in a carefully orchestrated sequence of events.

The system and the procedures to support this process were designed in an era when the transfer of data took hours; rapid communications was restricted to the telephone and conference calls were difficult to arrange; and the facts and information needed to make decisions was usually found in a printed book, document, or map.

This is the problem which the Advanced Research Projects Agency (ARPA) and Rome Laboratory (RL) undertook to examine. Their research projects have focused on innovative approaches and techniques leading to revolutionary advances in state-of-the art for planning and scheduling. Specifically, ARPA and RL have embarked on a joint Planning Initiative (PI) to develop and demonstrate the next generation of generic Artificial Intelligence (AI) planning, resource allocation, and scheduling technology focused on achieving significant performance enhancements over current Department of Defense (DOD) operational planning systems. The vision of the PI is to demonstrate how planners can utilize new technology which will revolutionize the planning process.

This new planning process can best be described as Distributed Collaborative Planning (DCP). It is "distributed" in that planners at multiple locations share data, software, and information on a real-time basis: and it is "collaborative" because planners communicate with each other via video-teleconferences passing written and verbal information instantly to each other. The thrust of this research is to eliminate the sequential nature of planning by providing tools which support the way planners would conduct their business if they were in the same room instead of hundreds of locations around the world. ARPA and RL conduct annual Integrated Feasibility Demonstrations (IFD1, IFD2, ...) which incrementally demonstrate the integrated utility of various maturing advanced technologies to satisfy a portion of the vision. The demonstrations build d upon each other and, with participation of selected joint operational Commanders-in-Chief (CINCs), are demonstrated in a context to show functional feasibility for future integration into a joint CINC's command and control infrastructure,

IFD3, which is currently being produced, is intended to illustrate how planners at United States Pacific Command (USPACOM), USTRANSCOM, United States Army Pacific and Pacific Fleet (both simulated by participants at Defense Information Systems Agency (DISA)), and an analysis agency such as Institute for Defense Analysis (IDA) can collaborate over a "global" network to develop a military plan. The specific operational focus is a Noncombatant Evacuation operation (NEO).

The data in use for Integrated Feasibility Demonstrations (IFDs) uses real locations, peoples and military data some of which is confidential or sensitive. However, some of the Planning Initiative work involves so called "tier 1" or enabling research in which ideas are being generated and tested. It was felt that a "cut-down" realistic scenario would be beneficial to such researchers. The aim was to provide non- confidential data that could be used to show the relevance of the enabling research for military planning problems. The data would be such that publication and public demonstration of results was possible in the scenarios provided.

The PRECIS (Planning, Reactive Execution, and Constraint-Satisfaction) [1] Environment defines the data and hypothetical background for studying logistics and transportation planning/scheduling problems and Non-combatant Evacuation Operations (NEO) scenarios.

The definition of the PRECIS environment has drawn on work by: Brown to describe a realistic NEO scenario for the Planning Initiative IFD2; Reece and Tate to define a fictional environment suitable for planning and reactive execution of plans based on the island of Pacifica ([?]); and work by Hoffman to produce a cut- down demonstration scenario suitable for transportation scheduling research experiments.

Three primary needs of the ARPA/Rome Laboratory Knowledge-based Planning and Scheduling Initiative are to be met by the PRECIS Environment. First, that realistic scenarios can be explored from the data provided in the environment,, for Course-of-Action (COA) generative planning, case-based reasoning, transportation scheduling, and reactive execution of plans. Second, requirements of tier 1 researchers are sufficiently met by the data in order for them to pursue their individual research objectives. Third, entities in the environment are hypothetical and do not reflect actual peoples and locations vet, are realistic in the types of data that would normally be available.

Against the general environment described in this paper, a series of supplemental scenario documents provide individual scenarios suited to a range of research issues. The intention is that this series of supplemental scenario documents can be augmented as desired by individual research teams. The general information in the core of the paper may be extended in the future to support these additional scenarios or scenario detail. As this is a communal document, both created by and of benefit to the research community, the contents of the document can be changed or redirected by the researchers within the community. Any additions or modifications to be made should be sent via electronic mail to arpi@isx.com.

Publicly available documents were used as guides to determine some of the factors used in the paper. Sources include USTRANSCOM [1], [2], Air Mobility Command (AMC), Military Sealift Command (MSC), and Military Traffic Management Command (MTMC) documents.

[1] PRECIS is a short piece of writing which contains the main points of a book or report, but not the details

A glossary of terms and acronyms used is given in Section 5.

2 Theater Geography

The PRECIS environment relates to events which are to take place in a hypothetical theater of operation. This theater is located in the Pacific Ocean and consists of the island of Pacifica and four countries on the Pacific Rim. These include a politically divided Country-W, an unfriendly country Yia (due to its territorial disputes with Pacifica), a friendly Country-X with an airport and seaport in City-K, and a friendly Country-Z with an airport and seaport in City-L, airports in cities City-N and City-O., and a seaport in City-M. Other assets which are available for use in the theater are located in the United States (see Figure 1).

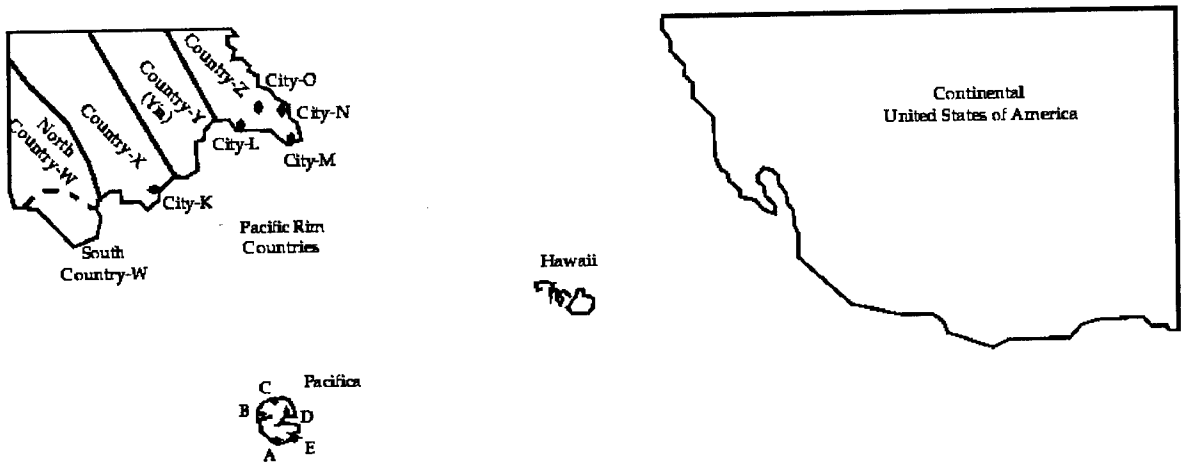


Figure 1: Map of the Pacific Rim

Pacifica (see Figure 2) is an island state located in the Pacific Ocean. It has a very interesting coastline, but remains shrouded in mystery due to its inaccessibility over the centuries with some areas of the island largely unexplored and unmapped- The island was formed by volcanic activity and still has one active volcano. There are active geothermal areas on the Western part of the island with volcanic mud occasionally closing the coastal island road for days at a time. A large fresh water lake has formed in a dormant volcano in the North, and prevailing winds come over the cliffs from the Northeast. The Southern portion of the island consists of the lush, tropical, Abysian Forest, and cotton is grown in the South-Central region. The small fishing village of Exodus is located on the Southeastern tip of the island, and its access is by what can only be described as a trail which limits the types of vehicles that can enter the village. The remainder of the island terrain consists mainly of a mixture of low rowing shrub and vegetation. Typically monsoons occur during the periods of January-February and July-August.

Pacifica has two seaports and airports. A seaport and airport are located in both the capital Delta and the city of Calypso.

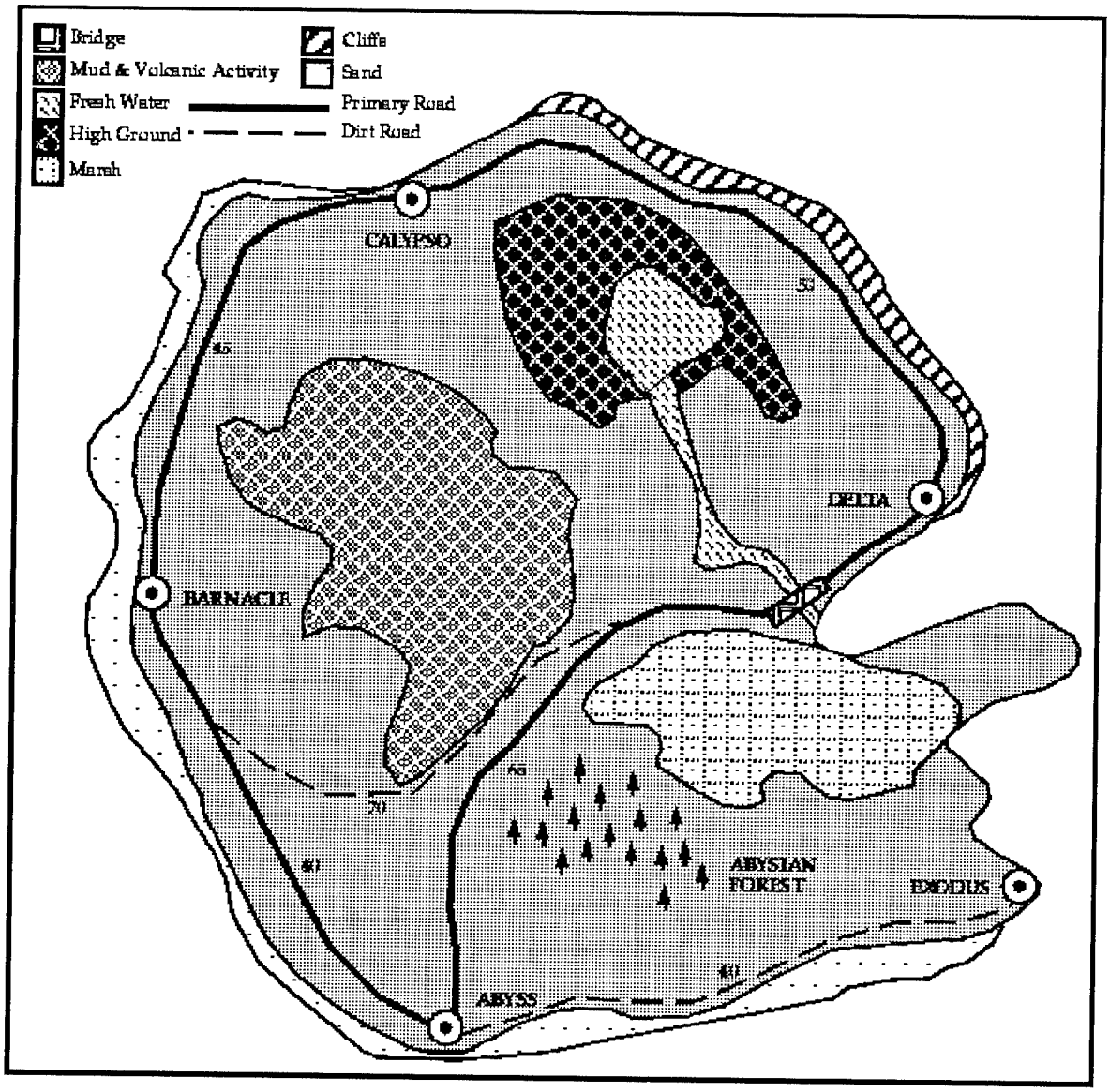


Figure 2: Island Map of Pacific

Also shown on the map are the small fishing village of Exodus (located on the Southeastern tip of the island), a dirt road from the city of Abyss to Exodus, and a dirt road which connects just South of Barnacle on the Coastal Road to just North of the sand flats (marsh) before the bridge.

3 Regional Political Situation

The origins of the Pacificians are shrouded in myth. Most historians believe that they came from North Yia during the 3rd century AD. Buddhism arrived from the Pacific Rim 100 years later. The Portuguese, in search of cinnamon and spices, seized coastal areas beginning in 1505. The Portuguese were ejected by the British in 1796. On 5th April 1950, Pacifica gained independence from the United Kingdom. A US embassy was established the same year.

As with many countries on the Pacific Rim, class, caste, and religion play a part in all Pacifica politics. The most serious difficulties are between the Pacifica Ethnic majority and the Adanan minority, and the Pacifica government and Adanan militants. The two main insurgent groups in Pacifica are the Adanan Liberators (AL) and the Malte Panef (MP). The AL has strong ties to the fishing communities on the Pacific Rim, is most active in the southern provinces and areas of the western province, and has vowed that it will not give up its goal of a separate Tondi state under any circumstances. The MP is a leftist, Salamiese militant group established in the late 1960's with strongholds in the northern part of the country. MP goals are to seize power, advocate the establishment of a socialist society, and include socialist dogma in the government.

Pacifica is a multi-ethnic, multi-religious, and multi-linguistic country. Salamiese comprise 74% of the population and are concentrated in the densely populated northeast. Pacific Tondis, citizens whose South Asian ancestors have lived on the island for centuries, total 12%. Although most live in the south and west, Pacific Tondis are found in Delta, the Capital, and throughout the country. A distinct ethnic group, the Yian Tondis represent 6population. The British brought their forbears to Pacifica in the 19th century as cotton plantation workers. Yian Tondis remain concentrated in the "cotton country" of south-central Pacifica. However, not all Yian Tondis are Pacifica citizens. In November, 1988, in accordance with an agreement with Yia, Pacifica passed legislation extending citizenship to some 23,000 "stateless" Yian Tondis. Under this pact, Yia agreed to grant citizenship to the remainder, approximately 20,000, who now live in Yia. Another 9,000 Yian Tondis who themselves are or whose parents once applied for Yian Citizenship now wish to remain in Pacifica. The Government of Pacifica has stated that this group must eventually return to Yia.

Other minorities include Muslims, which represent about 7% of the population; Burghers, who are descendants of the original European colonists; and aboriginal Veddahs.

Most Pacificians are Buddhist and most Tondis are Hindu. Sizable minorities of both Pacificians and Tondis are Christians, most of whom are Roman Catholic. The 1978 new Constitution, while assuring religious freedom, grants primacy to Buddhism.

Post-1950 Pacifica politics have been strongly democratic. The government is a republic with an elected president as Head of State, Head of Government, Chief Executive, and Commander-in-Chief of the armed forces. The Parliament shares power with the President. The Constitution

explicitly states that the national objective is the establishment of a "Socialist Democracy". The government is to provide full employment and an equitable distribution of wealth.

Pacifica has a competitive party system with two major parties each of which is capable of forming a stable government. The two major parties are the United National Party and the Pacifica Freedom Party. The United National Party (UNP) lead Pacifica to independence. It is currently the ruling party. The UNP's main support comes from professionals, industrialists, and urban entrepreneurs. The Pacifica Freedom Party (PFP) is the largest of the legal opposition parties. It is a non- Marxist party whose followers include Buddhist groups, land-owning rural gentry, Pacifician intellectuals, professionals, and the lower middle class.

From its independence, the Tondi minority has been uneasy with the country's government, fearing that the Pacifician majority would abuse Tondi rights. These fears were heightened when, in 1956, the Government declared Pacif the country's official language. The Tondis view Pacif to be a denigration of their own tongue. This was the first of many Government actions that the Tondis considered to be discriminatory towards their culture and heritage.

The decades following 1956 saw the intermittent outbreak of communal violence and growing radicalization among Tondi groups. The 1974 constitution changed the country's,y's name to the Democratic Republic of Pacifica, made protection of Buddhism a constitutional principle, and created a weak President appointed by the Prime Minister.

By 1978, Tondi politicians were moving from support for federalism to a demand for a separate Tondi state - Tondi Elite - in southern and western Pacifica. Many Tondi politicians sought to gain independence by peaceful, democratic means. The major Tondi political party, the Tondi United Liberation Front (TULF) won all of the parliamentary seats in the Pacifica Tondi areas. Unlike the TULF, the AL sought an independent state by force.

In 1992, the death of 13 Pacifica soldiers at the hands of Tondi militants unleashed the largest outburst of communal violence in the country's history. Hundreds of Tondis were killed in Delta and elsewhere, tens of thousands were left homeless, and more than 10,000 fled to South Yia. Members of the TULF lost their seats in Parliament when they refused to swear an oath of loyalty. The south and west became scenes of bloodshed as security forces attempted to suppress the AL. Terrorist incidents occurred in all major cities. The Pacifica Government accused the Yian Government of supporting the Tondi insurgents.

4 Logistics Domain Information

This section describes the factors which must be addressed in transportation logistics type problems and data which is used in the various scenarios. These factors are sufficient to demonstrate various concepts required to address such issues.

4.1 Unit Sizes

Unit size is determined by the number of persons (PAX), number and category tons BULK OVER OUT and MTONS and ULNS that need to be transported.

FM-Unit	FM-Name	Service	ULNS	PAX	BULK	OVER	OUT	MTONS
SAG	Surface Action Group	NAVY	16	3748	68	0	0	154
ACS	Conv. Carrier Bat. Grp. (F-14 Emb.)	NAVY	27	9435	591	226	0	2687
891	24th PAA F-16 Active Squadron	Airforce	8	725	250	316	0	2687
89B	24th PAA F-16 Active Squadron	Airforce	8	785	244	145	0	2016
LIB	Light Infantry Brigade Army	19	3005	591	1862	93	0	16087
IMF	Mechanized Inf. Brigade	Army	18	4672	1036	13344	7433	65005
IMB	Mechanized Inf. Brigade (Separate)	Army	20	5056	1146	12882	11303	77245
ACR	Armored Cavalry Reg	Army	16	5492	1362	13,348	12905	83250
701	Marine Exp. Brig. (Assault Echelon)	Marine	87	11689	4578	9185	4152	106219
5RG	Ranger Battalion	Army	2	606	120	10	0	377
5SB	Special Forces Battalion	Army	28	896	216	645	19	3771
710	Marine Exp. Unit (MEU)	Marine	53	2579	893	1924	909	23394
AFL	Aerial Port Element (2100 S/T /day)	Airforce	9	174	15	126	24	515
8EV	PAA C-130E Active Wing	Airforce	7	1102	142	76	0	1284
8E2	16 PAA C-130E Active Sq.	Airforce	4	508	57	62	0	697
8T6	05 KC-10A Tanker Task	Airforce	19	600	102	31	6	394
81M	24 PAA A-IOA Active Sq. Depend.	Airforce	7	5501	182	188	0	1912

Some of the data has not yet been checked for consistency, but is indicative of what will be provided. There are also some apparent inconsistencies which are actually simplifications. For example, runway length is not the only factor for aircraft landing. There are many airports

that support the larger C-5 that do not support the smaller C-141. This goes to the fact that the C-5 has more tires to distribute its weight better. The C-141 will "sink" through some runways that the C-5 won't.

4.2 GEOLOC Codes

Table 4.2.1, cross references the GEOLOC codes used in this scenario with the cities in which they are located. This table also indicates the type of US military base, where applicable.

GEOLOC Code	City	Base Name	Service	Location Type
ETZB	Oceanside, CA	Camp Pendleton	Marines	origin
UTBS	San Diego, CA	-	Navy	origin
UTAC	San Diego, CA	-	Navy	seaport
QKJA	San Diego, CA	Miramar NAS	Navy	airport
UTLR	San Francisco, CA	-	Navy	origin
UTLS	San Francisco, CA	-	Navy	seaport
UTKY	San Francisco, CA	-	Navy	airport
SYZP	Honolulu, HI	Pearl Harbor	Navy	origin
SYZZ	Honolulu, HI	Pearl Harbor	Navy	seaport
YVEW	Honolulu, HI	Wheeler AFB	Air Force	airport
JKFQ	Tacoma, WA	Ft. Lewis	Army	origin
PSBD	Tacoma, WA	McCord AFB	Air Force	airport
WPVT	Tacoma, WA	-	Navy	seaport
DCOA	City 0, Country Z	-	-	airport
NCLA	City L, Country Z	-	-	airport
NCLS	City L, Country Z	-	-	seaport
SCMS	City M, Country Z	-	-	seaport
UCKA	City K, Country X	-	-	airport
UCKS	City K, Country X	-	-	seaport
9CNA	City N, Country Z	-	-	airport
CPSA	Calypso, Pacifica	-	-	airport
CLPS	Calypso, Pacifica	-	-	seaport
DLTA	Delta, Pacifica	-	-	airport
DLTS	Delta, Pacifica	-	-	seaport

4.3 Airlift

Aircraft data used in the environment is shown in the tables of this section. Table 2 describes passenger/cargo capacity, range, and landing requirements. "Range" data calculations include assumptions regarding the weight of reserve fuel, aircraft operating weight, the weight of fuel

to an alternative destination, and others. The Flt-hours/day column indicates the maximum authorized aircraft type utilization. That is, for the entire fleet of aircraft type X, the Flt-hours/day value indicates the maximum number of hours that an aircraft of that type may spend IN FLIGHT (onloading and offloading times do not affect these times). The maximums are applied at the "fleet" level so that for example, a maximum Flt-hours/day of 8 hours is satisfied by a situation where we have 3 aircraft, one of which is in the air for 24 straight hours and the other two on the ground during that same time period. These tables also reflect assumptions such as weather conditions, sea level, operating weight, and others.

Type	OUT	OVER	BULK	PAX with cargo	PAX	Speed	Range	Flt-hours/day
C-5	101.0	74.5	82.8	73	73	436	5500	10
C-141	0	29.9	26.0	26	153	425	4000	10
C-130	0	11.4	13.8	8	91	280	2700	5
B747	0	108	107.6	408	408	450	3500	15

Table 3 describes turnaround time data. Turnaround time consists of three separate times: onload time (the time to load the aircraft), enroute time (the time to refuel), and offload time (the time to unload the aircraft). It is assumed that the onload/offload times given are for fully loaded aircraft.

Type	Onload	Enroot	Offload
C-5	1:45	1:15	1:30
C-141	1:30	1:40	1:20
C-130	1:20	1:25	1:00
B747	5:00	1:30	3:00

4.4 Sealift

Sea vessel data is shown in Table 4. It describes MTONS (which can be filled by Outsized, Oversized, or Bulk tons at 1:1 ratio), speed (in knots — nautical miles per hour), berth size required by ships, as well as load and offload times (in days). Note that a sulphite does not normally carry PAX, speeds can be maintained for 24 hours a day, and range is assumed not to be a limiting factor. All units are given in knots.

Type	MTONs	Speed	Berth	Load-time	Offload-time
Breakbulk	20874	20.5	C	5.0	5.0
Container	24520	16.1	B	1.5	1.5
RORO	38755	23.5	A	0.3	0.3
LASH	42042	20.0	A	0.7	0.7
Sea Barge	42400	20.0	A	0.4	0.4

4.5 Ground Transportation

Borrowing from the terminology from the aircraft types used in PRECIS , the ground transport data for the PRECIS environment is shown in Table 5.

Type	Onload	Enroute	Offload	PAX Capacity	Range
Ground Transport	0:20	0:15	0:20	50	348

4.6 Airport Characteristics

The ability of aircraft to land at different airports is determined by a number of factors, two of which are the length of the runway and the weight of the payload. The characteristics given here show which types of aircraft are capable to takeoff/land at which airports, the maximum on ground (MOG), and number of takeoff /landing pairs that can be supported (Sorties) -

Code	Name	State/Country	C-5	C-141	C-130	B747	MOG	Sorties
CPSA	Calypso	Pacifica	T	T	T	F	15	165
DCOA	City-O	Country-Z	T	T	T	T	25	240
DLTA	Delta	Pacifica	T	T	T	T	30	315
NCLA	Citv-L	Country-z	T	T	T	T	25	240
PSBD	McCord	Washington	T	T	T	T	70	500
QKJA	Miramar	California	T	T	T	T	70	500
UCKA	City-K	Country-x	T	T	T	T	25	240
UTKY	San Francisco	California	T	T	T	T	70	500
YVEW	Wheeler	Hawaii	T	T	T	T	70	500
9CNA	Citv-N	Country-Z	T	T	T	T	25	240

4.7 Seaport Characteristics

Similar to airport characteristics, those for seaports define which types and number of vessels can dock at a particular seaport. This data is given in terms of berth sizes for both ships and tankers. The berth sizes for ships (-i.e., A, B, C, D, E, and F) are decreasing size berth types, as are those for tankers (i.e., TA, TB, TC, and TD).

			Berth Types Available									
Code	Name	State/Country	A	B	C	D	E	F	TA	TB	TC	TD
CLPS	Calypso	Pacifica	1	9	6	3	15	31	0	2	2	0
DLTS	Delta	Pacifica	0	6	11	3	25	10	0	0	1	0
NCLS	City-L	Country-Z	8	3	5	5	21	30	2	1	0	0
SCMS	City-M	Country-z	6	7	7	12	0	30	1	1	0	0
SYZZ	Pearl Hbr	Hawaii	-	-	-	-	-	-	-	-	-	-
UCKS	City-K	Country-X	14	11	9	9	0	0	3	2	2	0
UTAC	San Diego	California	-	-	-	-	-	-	-	-	-	-
UTLS	San Francisco	California	-	-	-	-	-	-	-	-	-	-
WPVT	Tacoma	Washington	-	-	-	-	-	-	-	-	-	-

4.8 Travel Distances

Travel distances are given for land, air, and sea in Tables 8, 9, and 10 respectively. No travel time is given as it is dependent upon other factors (such as weather). Note, all land distance data is given only for in-theater locations and not POEs o PODs unless they are to be in-theater. Land distances shown are on primary road where possible. Segment distances have also been shown previously on Figure 2.

4.8.1 Land

Where there are multiple distances shown for a city pair, these numbers indicate the distance following alternate routes between the cities, e.g. when travelling from Abyss to Delta, the two numbers indicate the distance following opposite directions around the island.

	Abyss	Barnacle	Calypso	Delta	Exodus
Abyss,	X	40	85	65/135	40
Barnacle		X	45	95/109	80
Calypso			X	50/150	125/155
Delta				X	105
Exodus					X

4.8.2 Air

	US POEs				Theater Staging & POEs				PODs	
	PSBD	QKJA	UTKY	YVEW	DCOA	NCLA	UCKA	9CNA	CPSA	DLTA
PSBD	x	1050	675	2700	6000	6100	6500	5900	6250	6300
QKJA		x	450	2630	6950	7050	7400	6900	7100	7150
UTKY			x	24	6500	6400	6700	6450	6600	6550
YVEW				x	3500	3600	3700	3450	2800	2830
DCOA					x	100	400	80	700	710
NCLA						x	320	160	610	620
UCKA							x	460	500	520
9CNA								x	670	680
CPSA									x	45
DLTA										x

4.8.3 Sea

	US POEs				Theater Staging & POEs			PODs	
	SYZZ	UTAC	UTLS	WPVT	UCKS	NCLS	SCMS	CLPS	DLTS
SYZZ	x	2630	2450	2700	3700	3600	3550	2800	2830
UTAC		x	460	1100	7400	7050	7000	7100	7150
UTLS			x	700	7150	7100	6950	7300	7350
WPVT				x	7300	7330	7150	8700	8750
UCKS					x	250	300	500	550
NCLS						x	75	600	650
SCMS							x	550	590
CLPS								x	60
DLTS									x

5 Glossary

BULK Bulk cargo; Materiel generally shipped in volume where the transportation conveyance is the only external container, such as liquids, ore, or grain.

MTON Measurement Ton; The unit for volumetric measurement of equipment associated with surface-delivered cargo. Measurement tons equal total cubic feet divided by 40. (I MTON = 40 cubic feet).

OUT Outsized cargo; Cargo that exceeds 1,090" x 117" x 105", that is too large for C-130/C-141 aircraft.

OVER Oversized cargo; Cargo that exceeds the usable dimension of a 436L pallet, 104" x 84" x 96", or a height set by the particular model of aircraft.

PAX Passengers.

POD Port of Debarkation; The geographic point (port or airport) in the routing scheme where a movement requirement will complete its strategic deployment.

POE Port of Embarkation; The geographic point (port or airport) in an objective area that is the terminal point for strategic deployment for non-unit-related supplies and replacement personnel.

ULNS Unit Line Number; A seven-character alphanumeric code that uniquely identifies each force requirement.

6 References

References

[1] Day, D. S. and McAlpin, S. R. (December 1989) Supplementary Material Describing US-TRANSCOM Transportation Planning. Department of the Air Force.

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

[3] Reece, G. A. and Tate, A. (March 1993) The Pacifica NEO Scenario. Technical Report ARPA-RL/0-Plan2/TR/3. Artificial Intelligence Applications Institute, University of Edinburgh, Scotland.

Appendix O – Applying O-Plan to the NEO Scenarios

Brian Drabble, Austin Tate & Jeff Dalton

O-Plan Technical Report ARPA-RL/O-Plan/TR/23 version 1, July 1995.

Applying O-Plan to the NEO Scenarios

Brian Drabble, Austin Tate & Jeff Dalton

Abstract

This paper describes the evaluation experiments conducted as part of the O-Plan project. Each of the experiments conducted is categorised according to the ARPI Evaluation Handbook. The O-Plan system has addressed three types of ARPI experiment: Programmatic, Demonstration and Scientific. A number of experiments are described from each of these categories detailing the aims of the experiments, the method used and the conclusions and results which were found. The principal milestones for the project comprised three annual demonstrations.

1 Introduction

The aim of this paper is to describe the evaluation experiments conducted as part of the O-Plan project. The paper will show how each experiment can be related to the categorisation of experiment types defined in the ARPI Evaluation Handbook [4]. The O-Plan system aims to address three types of ARPI experiment: Programmatic, Demonstration and Scientific. This approach has been taken since the O-Plan project took a domain problem driven perspective to validate the approach of a specified planning architecture while allowing for the integration of new scientific ideas. The principal milestones comprised three annual demonstrations.

The O-Plan project has been targeted at a specific class of problems which include:

- 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 this type represent an important class of industrial relevance. From work on other projects in sectors such as aerospace, logistics, manufacturing, petroleum and business, we believe that this is a richly populated class of problems.

The structure of the paper is as follows. Section 2 describes the evaluation methods used and the experiment types carried out during the project. Section 3 describes a number of major experiments which included:

- Year 1 demonstration which showed a cut-down version of an ARPI Integrated Feasibility Demonstration IFD the IFD2 scenario running in the O-Plan system,

- Year 2 demonstration which showed how a rich model of resources could be used to improve the solutions provided by a planner
- Year 3 demonstration which showed how O-Plan could be employed in a command, planning and control scenario to deal with changes occurring on the environment and in the overall task
- linking of O-Plan with the EXPECT plan analysis tool from USC/ISI.

Section 4 describes a number of additional experiments which were used to evaluate the new functionality and capabilities being added to the system. The Appendix describes the algorithms used to implement the plan repair mechanism demonstrated in the Year 3 major demonstration.

2 Methods of Evaluation

The aim of this section is to describe the evaluation experiments conducted as part of the O-Plan project.

The O-Plan project has identified a number of demonstrations closely related to ARPI programmatic goals in a sufficiently simple domain which allows for the investigation of scientific goals. This domain is called PRECIS (Planning, Reactive Execution and Constraint Satisfaction) [24]. The PRECIS environment defines the data and hypothetical background for a demonstration domain related to logistics and transportation planning/scheduling problems and Non-combatant Evacuation Operations (NEOs). The definition of the PRECIS environment has drawn on work by:

- Brown (at MITRE) to describe a realistic NEO scenario for the Planning Initiative's IFD3
- Reece and Tate (at Edinburgh) to define an openly accessible fictional environment based on the island of Pacifica [23], suitable for enabling technology researchers interested in planning and reactive execution of plans,
- Hoffman and Burnard (at ISX) to produce a cut down demonstration scenario suitable for transportation scheduling research experiments.

Four primary needs of the ARPA/Rome Laboratory Planning and Scheduling Initiative are met by the PRECIS environment:

1. realistic scenarios can be explored from the data provided in the environment, for Course of Action (COA) generative planning, case based reasoning, transportation scheduling and the reactive execution of plans;
2. requirements of "tier-1" enabling researchers are sufficiently met by the data in order for them to pursue their individual research programmes;

3. entities in the environment are hypothetical and do not reflect actual peoples and locations. However, they are realistic in the types of data that would normally be available;

4. the scenario and domain descriptions are not confidential or military critical. They can be openly demonstrated and publications can be based upon them. This is important for enabling researchers.

Using the PRECIS domain as a base, the O-Plan project validated its vision of the component parts of the architecture for a responsive command, planning and control environment which is provided in a modular fashion. This allowed checks to be carried out on the components and pathways to ensure they were suitable and appropriate.

At the same time as validating the architecture, the O-Plan project made scientific progress in this integrated framework. This is addressed in the project's Year 2 and 3 demonstrations which addressed resource reasoning and integrated command, planning and control respectively.

The experiments carried out are covered by three main categories of the ARPI Evaluation Handbook and are as follows:

- **Programmatic:**
  The programmatic element aims to show the relevance of the O-Plan project to the goals of the ARPI and in particular its impact on the US military planning community. The impact was measured as follows:
  - Improved connectivity and consistency between command, planning, scheduling and control.
  - Open, inspectable, explainable and changeable plans.
  - Greater Scope for COA analysis and greater plan reliability.

- **Demonstration:**
  The demonstrations in Year 2 and Year 3 show O-Plan solving a series of problems from an ARPI relevant problem class. In order to help this process the project developed the PRECIS domain description with other members of the Initiative. This allows for the presentation of ideas to researchers outside the ARPI while maintaining the confidentiality of the target domain. In addition to the Year 2 and 3 demonstrations, other demonstration experiments were conducted. These including the linking of the O-Plan system with USC/ISI's EXPECT plan analysis tool [16] to allow plans generated by O-Plan to be evaluated against user provided domain dependent plan evaluation criteria. The evaluation matrix developed by EXPECT for a number of plans generated by O-Plan could allow the user to examine the quality of the solutions being generated by O-Plan on a series of problems from the PRECIS domain.

- **Scientific:**
  The scientific experiments showed how specific technical features of the O-Plan system have improved the quality of the solution presented to the user. The features examined in detail were as follows:

- its ability to use resource reasoning in an activity planner framework,
- its ability to explicitly represent authority in a command, planning and control environment.

The following sections describes the experiments carried out during the project and describes in outline terms the method of the experiment, the results obtained and the measures taken (where appropriate) to overcome any problems the experiment highlighted. Section 3 describes the major programmatic experiments conducted, and Section 4 describes a number of experiments conducted to evaluate the functionality and capabilities of the developing O-Plan system.

3 Experiments

The aim of this section is to describe some of the major experiments which have been carried out during the project. Each experiment will be described in outline and will be classified according to the taxonomy developed in the ARPI Evaluation Handbook [4].

3.1 Demonstration Experiment: Year 1 – 1993: Generation of Plans from the IFD-2 Scenario

One of the first year aims of the O-Plan project was to repeat the ARPI Integrated Feasibility Demonstrator Number 2 IFD-2 with O-Plan taking the place of SIPE-2. From the start of the experiment, it was recognised that SIPE-2 [32] was a more mature system than O-Plan and as such this could only be an approximation to IFD-2. However, using the Task Formalism (TF) (O-Plan's domain input language) then supported within O-Plan Version 2.1 it was possible to encode the SOCAP domain and to identify a number of shortcomings in O-Plan TF [7] [8]. The schema library for this domain contained 63 schemas which defined alternative missions, deployment and employment plans, sea and airlift resources, etc. The Courses of Action (COAs) generated contained an average of 150 actions and were developed in approximately 50 seconds. O-Plan was able to generate plans in the SOCAP domain for two tasks:

- **Task 1**: "Deter three threats"
  The task requires a plan to deter one army, one air force and one navy threat by specified dates. The threats are forces which have crossed the protected border.
- **Task 2**: "Deter three threats and counter a further nine"
  The task requires a plan to deter the same three threats as well as countering a further nine threats: three army, navy and air force respectively. These nine forces are threatening to cross the border but have not yet done so.

The experiment highlighted a number of problems in the way the system handled the satisfaction of `only_use_if` and `only_use_for_query` conditions and in the handling of plan objects (referred to as plan state variables). The experiment allowed these problems to be highlighted and fixed [31].

The experiment also allowed the testing of user defined functions for selecting the next agenda entry to run (instead of the default O-Plan mechanism). The agenda entry selection function was needed to force the planner to decide which military unit was to be used before developing the appropriate deployment plan for it. Off-line analysis showed that the problem could be solved with little or no search being involved. For example, many of the military units which could be chosen for a particular mission were similar and consequently the planner should have left the decision over which military unit to use until it was forced upon it, i.e., developing the military unit's employment plan. The agenda selection function allowed the user to provide this knowledge in a form which can be used by the system. The experiment showed that the user defined function reduced the search time from 10 hours to 50 seconds by removing many redundant paths from the search space. This analysis has proved useful in guiding work in Mixed Initiative Planning mechanisms for O-Plan [29]

3.2 Programmatic & Scientific Experiment: Year 2 – 1994: Use of a Rich Resource Model in an Activity Planner Framework

The aim of the Year 2 demonstration was to show O-Plan in a military-relevant resource-based scenario. The main aim of the demonstration was to show the benefits of using a richer resource model, e.g., the modelling of trucks, helicopters, cargo planes, passengers planes, air tankers, diesel and aviation fuel, storage tanks, runways, etc., within a generative activity planner such as O-Plan. In order to accomplish this a new Resource Utilisation Manager (RUM) [13] was designed which could deal with a number of different resource types and research was conducted into the ways in which the planner could make use of the domain information about resources to restrict its search. The resource type hierarchy defined for the RUM was consistent with that defined for KRSL [19] and extended the KRSL definitions in a number of ways. The demonstration also provided a check on the development of the functionality of the emerging O-Plan system and in particular the system's ability to reason with numbers and numerical ranges. The use of a rich model of resource management in an activity planner was one of the principal research themes of the project.

As part of the preparation for the demonstration a study was carried out into the different types of resources present in planning domains and into previous planning approaches to resource reasoning [9]. The results of this study were twofold.

1. It became possible to identify the type of resource reasoning support which should be possible with an activity planning framework.
2. It resulted in the design of a flexible Resource Utilisation Manager (RUM) for use in an activity planner such as O-Plan and SIPE-2.

The support provided by the new RUM design would allow a range of resources types to be represented and manipulated and went beyond those types supported by KRSL.

We set out to use a simpler Resource Utilisation Manager in O-Plan and existing planner features to deal with resources. The demonstration successfully showed that plans could be generated for a number of different resource constrained tasks specified in the PRECIS domain.

A number of techniques were explored and validated which showed how resources could be defined and manipulated using a range of methods. These methods made explicit use of O-Plan's simple Resource Utilisation Manager to track consumable resources and O-Plan's World Condition and Effect (TOME and GOST) Manager to track reusable/sharable resources. Whilst these techniques allowed some of the coverage as was expected with the new RUM they do not have the same level of flexibility and support. In tasks where the resources were limited, e.g., small amounts of diesel fuel, the system was able to use knowledge of resources to rule out certain options as being impossible. In tasks where the choices were more extensive, e.g., use any transport type with no temporal restrictions, the system was still able to find a solution in an acceptable period of time.

The tasks described in the demonstration represent an interesting class of problems faced by the US military. They demonstrate how a series of transport assets (planes, helicopters, ground transports) and cargo (fuel) can be moved and coordinated between one location and another. The plans produced took into consideration many real world constraints on time and resources. However, in order to facilitate the demonstration a number of simplifications were introduced, e.g., fuel for the C5 and C141, crew schedules and maintenance periods were ignored. It would have been possible however, to introduce these resource and time constraints without drastically altering the performance of the system. The overall performance of the system showed that plan domains can be encoded with resource information and that plans can be generated which use this knowledge to restrict the options and choices to be considered.

While the new RUM has not been implemented in the current O-Plan prototype (Version 2.3), the research has investigated the underlying mechanisms necessary to support the various resource types and to integrate resource reasoning about each type into an activity planner. The approach adopted in the research – as stated in the original proposal – was to take an activity centred reasoning approach and to relate resource reasoning to this. The project does not claim this to be the best way to handle domains in which resource contentions dominate. Approaches such as in KIDS, [15] OPIS [26] or TOSCA [2] may be more appropriate. TOSCA is itself based on the O-Plan architecture but uses a resource centred representation and knowledge sources.

3.3 Programmatic & Scientific Experiment: Year 3: Coordinated Command, Planning and Control

This was a demonstration experiment which showed O-Plan solving a number of tasks from an integrated command, planning and control scenario. The aims of the demonstration were to show:

- O-Plan reacting to changes in the environment and identifying those parts of the plan which were now threatened by these changes.
- O-Plan reacting to changes in the overall task by integrating new plan requirements into the plan.

In both these cases the changes were to be made to an ongoing and executing plan.

The types of changes explored in this demonstration include failures of trucks due to blown engines and tyres and the inclusion of new objectives, e.g., pick up an extra group of evacuees. The PRECIS/Pacifica based example used for the demonstration has been deliberately simplified to allow a number of different aspects to be explored while keeping the plan to a manageable size. This is for viewing purposes only so that the user could follow what was happening in the demonstration. However, while being a simplification, the types of problem encountered and the solutions proposed by the planner are of relevance to military crisis action planning. Larger and more complex plans are available in other Pacifica domains.

The schema library for this domain contained 12 schemas which defined alternative evacuation methods, e.g., trucks or helicopters, fuel supplies, transport aircraft, etc. The COAs generated contained an average of 20 actions and were developed in approximately 40-60 seconds. Four different repair plans were used in the demonstration as follows:

- To repair a blown engine on a ground transport
  - The engine can only be fixed by a repair crew which is dispatched from the Pacifica airport at Delta with a tow truck. The ground transport is then towed to Delta for repairs. The evacuees remain with ground transport while it is being towed.
  - The failure of the transport occurs in a time critical situation and there is insufficient time to tow the broken transport to Delta. The evacuees are moved from the broken ground transport by helicopter to Delta and the transport is abandoned.
  - This is similar to the previous repair in that the failure occurs in a time critical situation except in this plan the evacuees are moved by another ground transport instead of by helicopter.
- To repair a blown tyre on a ground transport
  - The driver of the ground transport can fix the tyre by the side of the road. The effect of the repair action is to delay the ground transport by a fixed amount of time.

Details of the algorithms and methods used to implement the plan repair features are given in the Appendix.

Closely allied to the third year O-Plan demonstration showing the link between a proactive planner and a more comprehensive reactive execution agent, an associated Ph.D student project by Glen Reece showed a reactive execution agent [22] based on the O-Plan architecture. This has been used to reactively modify plans in response to operational demands in a simulation of the Pacifica island in the context of a NEO.

3.4 Programmatic & Demonstration Experiment: Linking of O-Plan and the EXPECT Plan Analysis Tool

This was a demonstration experiment conducted with USC/ISI in which the O-Plan system was linked with their EXPECT plan analysis tool [11],[12]. The ARPI Integrated Feasibility Demonstration Number 2 (IFD2) was used for IFD-2 was chosen for the evaluation domain.

The schema library for this domain contained 63 schemas which defined alternative missions, deployment and employment plans, sea and airlift resources, etc. The Courses of Action (COAs) generated contained an average of 150 actions and were developed in approximately 40 seconds. The different COAs were generated using alternative mission profiles and force packages. EXPECT could allow military planners to analyse these alternative COAs generated by O-Plan against a number of user defined domain evaluation criteria creates an evaluation matrix for a number of chosen COAs. From the analysis, military planners would be able to identify aspects of the COAs which were acceptable (e.g., low numbers of support personnel) and those which were not (e.g., a closure date greater than 29 days). An EXPECT evaluation matrix from a series of different COAs generated by O-Plan for a logistics scenario is shown in Table 1. This information could then be used to impose addition requirements on the planning system to seek to provide a better quality solution.

	COA 1	COA 2	COA 3	COA4
AIRPORTS				
- number of airports	1	1	1	2
- sorties per hour	315	315	315	480
- sq. ft. aircraft parking	2M	2M	2M	3M
SEAPORTS				
- number of seaports	1	1	1	2
- number of piers	6	6	6	15
- number of berths	6	6	6	16
- max. vessel size in ft.	600	600	600	765
- number of oil facilities	1	1	1	3
CLOSURE DATE	C + 29	C + 22	C + 23	C + 23
LOGISTICS PERSONNEL	1154	5360	5396	7362
LINES OF COMMUNICATION				
- number of locations	1	5	7	6
- max. distance in miles	20	99	140	120
- air and sea?	yes	yes	yes	yes

Table 1: EXPECT's evaluation of several alternative plans generated by O-Plan

4 Additional Experiments

In addition to the three main themes explored by the evaluation experiments, a wide range of other experiments were carried out during the research. These were intended to demonstrate the capabilities being added to the O-Plan prototype, to explore the characteristics of problems that we wished O-Plan to address, to explain to others how to encode domains in O-Plan, etc. These problems have been explored in two ways:

- by creating O-Plan Task Formalism domain descriptions to give to O-Plan or extending some of the domains already used with O-Plan such as House Building or Pacifica. Many of the TF domain descriptions created for these experiments are provided with the O-Plan release in the `demo/tf` directory.

- by creating a number of experiments which aim to show the implications on the search spaces of the use of different types of knowledge and search techniques.

The experiments are described in the following subsections.

4.1 Scientific Experiment: Evaluation of O-Plan Condition Types

The main aim of the experiment was to take a fresh look at condition types and in particular the need for each type and the ways in which they should be handled in future implementations of O-Plan. The reason for this reappraisal was to address our concerns that there was confusion and criticism in the technical literature [5] about the use of the various condition types, and that a great deal of effort was required in encoding some domains for what seemed to be a small gain in search efficiency or in the quality of plans being presented.

The use of domain knowledge to restrict the plan search space is vital in any large scale problem, as the use of syntactic information concerning a condition is inadequate. The O-Plan team believe that one effective way to provide this knowledge to a planner is via condition types. Some condition type information can be gathered by lexical analysis of a problem definition. However, at present AI planning researchers know of no way to automatically deduce some of the information we can gather from user-defined types and conditions.

The experiment was centred around the need for each particular condition types. The first point which was addressed was to provide three statements for each O-Plan condition type:

- **Purpose**: This describes the condition in domain terms for use by the domain encoder and describes the circumstances under which the condition should be used.
- **Definition**: This describes the condition in planner terms and describes in more detail how the planner goes about dealing with the condition type on behalf of the domain encoder.
- **Examples**: This clarifies of the use of a condition type.

In providing a description of each condition type in terms of its purpose and definition it became necessary to define further the meaning of a plan level and the plan circumstances in which a condition could be evaluated (i.e., when to trigger the agenda entry to have the condition satisfied). The original definition of a plan level was too loose and vague to be used with the emerging definition of condition types and as a result a cleaner and more precise definition of a level was produced. The time at which an agenda entry is released (or triggered) for processing is very important in the search for a plan. The function of the triggers is to ensure the agenda entry is released for processing when it is possible to process the agenda entry in the planning process. By developing a clearer understanding of the ways in which conditions can be satisfied and maintained it was possible to define a cleaner and more precise definition of triggers. Full details of this evaluation can be found in [14] and the results were published in [31].

At present, one of the current release demonstration domains cannot be encoded with the tighter definitions. This is a block stacking domain (`blocks-2.tf`) used to show examples in

earlier planners such as Nonlin [27] and Noah [25]. On study it became clear that the domain is encoded in a way which limits the solution space artificially. This was necessary for earlier planners, but is not necessary for O-Plan. The encoding of (`blocks-1.tf`) is a general purpose and improved description of this problem, but it would not have been possible to use this with Nonlin and Noah. From O-Plan version 3.1, (`blocks-2.tf`) will be removed from the demonstration suite so that a tighter definition of condition types can be imposed in the future.

The result of the experiment has been a better understanding of the use of condition types and the re-engineering and testing of *all* O-Plan test domains to comply with the new condition definitions.

4.2 Programmatic & Scientific Experiment: Economy of Force

There had been discussion within the ARPI community during 1993-4 which indicated a belief that so-called "generative planners" were inherently incapable of finding to solutions to problems in which the choice of a single action (operator schema) was necessary to address two or more separate problem requirements. This is sometimes called "economy of force". It can be one of the domain elements of evaluation to guide choice of better plans. Ginsberg at the University of Oregon had provided a simple island evacuation domain description in which such a single action choice was necessary to find the shortest solution to a problem[1].

O-Plan does contain all economy of force solutions in its search space, as it is designed to be systematic in preserving search space completeness (modulo restrictions on the search space deliberately encoded by a domain writer through features provided for this purpose - such as condition typing). In order to prove this, the example from Ginsberg was coded in O-Plan TF and provided to O-Plan. As expected, this demonstration showed that O-Plan is easily able to find solutions to such problems.

The O-Plan condition satisfaction procedure (Question Answering) and the Operator Schema choice routines in O-Plan do in fact currently choose between open choices using a single criteria, but they preserve all choices systematically. These choices are available to the search space controller in O-Plan. The O-Plan design allows for the incorporation of heuristic prioritisation of choices made by the operator schema choice function and the pre-ordering of choices available via Question Answering to satisfy conditions. Such heuristic prioritisation is anticipated to support a range of domain dependent elements of evaluation (as described in [17]). This can include economy of force (or as we term it "kill-two-birds-with-one-stone") choice prioritisation. Note that it is not possible to build this heuristic in to a general purpose planner as a hard wired prioritisation routine, since in some domains economy of force is to be avoided. It can also run counter to other preferences such as robustness in plans.

4.3 Scientific Experiment: Missionary and Cannibals

Scientists at Rome Laboratory used the Missionary and Cannibals Problem during 1993-4 to compare O-Plan and SIPE-2 [20]. While the Missionary and Cannibals Problem is not in the

[1] Personal Communication.

problem class for which O-Plan is designed (since it is essentially a mathematical *puzzle*), we used a range of Missionary and Cannibals Problem descriptions in O-Plan TF to demonstrate features of O-Plan prior to support for numeric handling and compute conditions (for external function support [28]) being added. These experiments were done with version 2.1 of O-Plan – the first release to the ARPI CPE. This showed that the Missionary and Cannibals Problem could be encoded using successor arithmetic.

Following the addition of numeric and compute condition support to O-Plan in version 2.2 (the second release to the ARPI CPE in July 1994), the Missionary and Cannibals Problem was recoded to act as a test domain for these features and to show that improved handing of the domain was possible. The Missionary and Cannibals TF encodings for the early and later experiments are available in the O-Plan release within the `demo/tf` directory.

4.4 Scientific Experiment: Spanner

O-Plan includes handling for the satisfaction of conditions within operator schemas, where the introduction of actions to satisfy the conditions turns out to require the insertion of new actions into the plan before the temporal scope of the schema which contains the condition [31]. The O-Plan team have for some time explained that other planners designs are not allowing this possibility in their search spaces. This means that they are *designed* to be incomplete. Some plans that a domain encoder might expect to be possible will not be admitted. The benefits for these planners is that their search spaces can be significantly smaller.

O-Plan provides support which will allow all solutions to be found including those needing the introduction of actions into a plan which "span" the area from the start of the plan to the point at which the condition is needed within the operator schema expansion (rather than just being in the gap between the beginning of the operator schema expansion and the point where the condition is needed).

A simple domain description called `spanner.tf` has been written to describe a very simple domain in which the "spanning" capability is required in a planner. This domain will not be amenable to solution by other planners designed with the more restricted definition of the legal temporal scope for the insertion of new actions to satisfy an achievable condition. This domain when run on O-Plan shows that O-Plan correctly identifies solutions requiring this capability.

Adding this capability to O-Plan has some serious consequences for comparative trials of O-Plan versus other planning systems. As far as we are aware, O-Plan is the only planner to have identified and remedied this problem. The search spaces introduced by handling it make for larger numbers of choices for a range of domains, some of the worst being the "puzzle" orientated domains used by many researchers to test their systems. Even simple problems in large plans can lead to very many more open choices if this capability is added.

Up to and including version 2.3 of O-Plan, the final release to the ARPI CPE from the O-Plan project's work in July 1995, the default handling for achieve conditions assumed that the system should allow for "spanning" solutions to be found. It is anticipated that a future release of O-Plan will alter the default handling to limit solutions to the temporal scope of the operator expansion in which the condition is introduced. However, O-Plan will continue to provide the

more comprehensive "spanning" solution as necessary, and this will be able to be switched on by simply altering a default to the TF Compiler – using `achieve_after_point`.

4.5 Scientific & Demonstration Experiment: Proof of Concept for Mixed Initiative Planning

This experiment provided an early proof of concept demonstration for Mixed Initiative Planning (MIP) within the O-Plan framework [29]. The demonstration was based on the PRECIS/Pacifica domain and showed how the user could provide manual control over some of the choices being taken by the planner. Interaction with the system was via the KS-USER knowledge source and one of the aims of the experiment was to demonstrate that the improved functionality within this knowledge source could provide basic MIP support.

The choices which the user could control were:

- **Choice of schema:**
  The evacuation plans available for a given trip (i.e., city to city) were via helicopter or via ground transports.
- **Choice of variable binding:**
  Once the evacuation plan had been defined a particular transport asset had to be allocated, e.g., a ground transport plan could use any of the ground transports available.
- **Choice of back track point:**
  If the developing plan became invalid, e.g., due to unsatisfiable time or resource constraints then the user could choose which alternative partial plan was to be used as an alternative candidate. Planning would then continue from this partially generated plan.

The schema library for this domain contained 25 schema which defined alternative missions, transport plans, fuel (diesel and aviation), transport assets (helicopters and trucks). The COAs generated contained between 15 and 25 actions and were developed in approximately 40 seconds (discounting time taken for decision making by the user).

The demonstration showed that the O-Plan system could provide a framework within which the user could experiment with MIP related issues and that the support provided by the KS-USER knowledge source was sufficient for basic experimentation with MIP. The demonstration was shown to the ARPI Programme Managers during the O-Plan project review meeting in May 1994.

4.6 Scientific Experiment: Dealing with Plan State Variables

The aim of this experiment was to investigate the use of maintaining tighter information on the possible values for different plan state variables. A Plan State Variables (PSV) can be restricted to disallow a certain value or to require that its value be different from that of another PSV. The latter case is called a "not-same" constraint in O-Plan. The problem of dealing with the

co-designation/non-codesignation of constraints has been a topic of research for many years. Others researchers [3] has developed schemes which attempt to solve parts of this problem. The work of the O-Plan system aims to develop research in this area further.

Each PSV has a "possibles-cache" that lists the values the PSV might take. Restrictions can remove values from the possibles-cache. If the PSV is left with only one possible value, it must be bound to that value; if it's left with zero, the plan is invalid.

The PSV Manager was changed to extract more information from *combinations* of not-same constraints. This was done only when a restriction is added to a PSV, leaving two or more values in PSV's possibles-cache. Then the PSV Manager looks at the PSVs listed in variables' not-sames constraints to check how many of the variables possible values they might take in combination.

The basic idea can be explained by an example. Suppose P-1, P-2, and P-3 all have A and B as their only possible values. Suppose P-2 and P-3 must have different values and that we then restrict P-1 to be different from both P-2 and P-3. Clearly, with three variables and only two possible values, there aren't enough values to go around. The aim is to detect cases of this sort and force the planner to abandon invalid plans earlier.

This change to the PSV Manager did not result in a noticeable increase in overall run-time and significantly reduced the number of O-Plan problem solving cycles required for certain tasks. For instance, the Pacifica task Blue Lagoon went down from 259 O-Plan cycles to 124. A more significant effect was that some tasks became practical (in terms of acceptable run time) for the first time.

4.7 Scientific Experiment: Agenda Choice

The aim of this series of experiment was to investigate the impact on the search space of different schemes for choosing ready to run entries from the agenda. The agenda contains "issues" that must be resolved in order to construct a complete plan: actions to be expanded, conditions to be satisfied, variables to be bound, etc. The order in which the issues (agenda entries) were processed can affect how quickly a plan is found and which plan is found. The different schemes tried were aimed at significantly improving the planner's performance by paying attention to plan levels and by exploiting heuristics that had been developed in earlier Edinburgh planning work. These heuristics include Branch1/BranchN factors which are used to select the most constraining choice next [6]. These have also been studied recently as the "least cost flaw repair" heuristic by Joslin/Pollack [18] who showed that it does lead to search space reductions. Recent work reported, for example at IJCAI-95 in Montreal, has begun the refinement of these methods.

The outcome and results of the experiment were mixed. As anticipated, no simple fixed prioritisation scheme improves performance on all problems. An adaptive and opportunistic scheme is what we are seeking which makes use of constraints in the plan and domain knowledge. The experiments also highlighted problems with individual techniques and these were as follows:

Plan Levels

One problem with using levels is that many of the current O-Plan TF domain definitions were written without a clear hierarchical model and a certain amount of rethinking concerning their encoding is required. In addition we are aware that the O-Plan agenda choice priority mechanism is being used to ensure the planner follows a certain planning "algorithm" (e.g., expands are done before certain types of condition are satisfied). It would be better if these mechanisms were separated from other types of choice to allow improved search control. This is the subject of future O-Plan research.

Branch-1 and Branch-N Estimators

O-Plan maintains two estimators of the branching factors for agenda entries – Branch-1 and Branch-N. Branch-1 is the number of "top level" possibilities that will be considered when an issue is processed. Branch-N is an estimate of the possible final number of alternatives for this issue. When branch-1 is 1, there is only one possibility and the planner has a single committed choice. There is an intuitive and informal argument to the effect that the Planner should prefer committed choices and process them first. All issues on the agenda will have to be processed. Some will create branches in the search space, and all remaining issues will have to be processed in all branches. If committed choices are done first, they will be processed only once. Moreover, they may constrain the plan, thus making things easier (in a sense) when processing other issues. There is also some empirical evidence that it helps to prefer forced moves [18].

However, the experiment showed that preferring committed choices can make things worse. Here, it is important to distinguish between two questions:

1. Is it better overall to process forced moves first?
2. Does it always make things better, or are there some cases where it makes things worse?

The aim of the experiment was to address the second question, not the first. Joslin's work [18] addressed the first problem.

To see that things *can* get worse locally, consider the following example. Suppose the agenda includes items A and B, that A has branch-1 = 1 while B has branch-1 > 1, and that A and B are independent in the sense that processing one will not affect how we process the other. Now suppose that when we process B we will always reach a dead end, so that the Planner must backtrack, and that processing A will not reach a dead end. If we process A before B, the time spent processing A will be wasted. This lead to the consideration of the cost involved in the processing of an agenda entry and that processing some agenda entries was more "costly" (in terms of the amount of effort expanded) than others. Earlier versions of O-Plan did try to maintain knowledge source activation estimates for this reason. Further experimentation is required to resolve this question within O-Plan.

4.8 Scientific Experiment: Alternative Choice

The aim of this experiment was to validate a number of different schemes for chosing alternative plan states to backtrack.

When the Planner cannot continue in the plan state it is currently considering, or chooses not to continue, the Agenda Manager (AM) is asked to pick one of the available alternatives so that the planner can continue from there. That is, the planner returns to some earlier decision and makes a different choice. Alternatives represent such decisions as which schema to use when expanding an action or which value to give to a variable.

Since alternatives are data structures, rather than being expressed procedurally in the Planner's code, the AM can employ a number of different search strategies when deciding which alternative to try. This is done, in part, by assigning each alternative a cost. Both the cost function and the AM's choice method can be redefined.

Initially, a very simple cost function for an alternative was used by counting the number of actions in its associated the plan state, and the choice method was to choose the lowest-cost alternative. If more than one alternative had the lowest cost, the most recent one created was taken. (This was done implicitly by the way the list of alternatives was sorted.)

A number of different cost functions and choice methods were tried and evaluated by planning for a range of O-Plan demonstration tasks. The different schemes investigated were as follows:

- Split the cost function into two parts that would be added, one to measure the work done so far, and one to estimate the work still to be done. This is similar to algorithms such as A* [21] and had some benefits in a number of domains.
- Add a depth-first element by choosing the most recently created alternative, rather than the one with lowest cost, in certain cases. This would provide part of the "local best than global best" strategy. We are seeking to use this in O-Plan. (The rest would be provided by knowledge sources that did some local search on their own.) Of the different schemes used this was by far the most effective change of all the ones described.
- Limit the amount of work done before seeing if an alternative might be better. After a given number of O-Plan problem solving cycles, the AM would switch to a better-rated alternative even if it was still possible to continue from the current plan state. This had no significant improvement in performance in the set of test domains and in some cases made things worse.

In addition to these schemes a number of different cost functions were tried, stopping once an improvement was found.

5 Summary

This paper describes the evaluation experiments conducted as part of the O-Plan project. Each of the experiments conducted has been categorised according to the ARPI Evaluation

Handbook. The O-Plan system has addressed three types of ARPI experiment: Programmatic, Demonstration and Scientific. A number of experiments have been described from each of these categories detailing the aims of the experiments, the method used and the conclusions and results which were found. This approach has been taken since the O-Plan project took a domain problem driven perspective to validate the approach of a specified planning architecture while allowing for the integration of new scientific ideas. The principal milestones for the project comprised three annual demonstrations and one TIE as follows:

- Year 1 demonstration which showed a cut-down version of an ARPI Integrated Feasibility Demonstration IFD the IFD2 scenario running in the O-Plan system,
- Year 2 demonstration which showed how a rich model of resources could be used to improve the solutions provided by a planner
- Year 3 demonstration which showed how O-Plan could be employed in a command, planning and control scenario to deal with changes occurring on the environment and in the overall task
- linking of O-Plan with the EXPECT plan analysis tool from USC/ISI.

A number of additional experiments are also described which were used to evaluate the new functionality and capabilities being added to the system. The Appendix describes the algorithms used to implement the plan repair mechanism demonstrated in the Year 3 major demonstration.

References

[1] Arentoft, M.M., Parrod, Y., Stader, J., Stokes, I. and Vadon, H., Optimum-AIV: a Planning and Scheduling System For Spacecraft AIV, in *Telematics and Informatics* Vol.8, No.4, pp.239-252, 1991.

[2] Beck, H., TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints in Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, (eds. C. Kooij., P.A. MacConaill and J. Bastos), pp138–149, Amsterdam, 12-14 May, 1993.

[3] Choueiry, B.Y. and Faltings B., Interactive Resource Allocation by Problem Decomposition and Temporal Abstractions, in *Current Trends in AI Planning*, (eds. Backstrom C. and Sandewall, E.), (the proceedings of the Second European Workshop on Planning), Vadstana, Sweden, 1993.

[4] Cohen, P., Dean, T., Gil, Y., Ginsberg, M. and Hoebel, L. *Handbook of Evaluation for the ARPA/Rome Laboratory Planning Initiative*, February, 1994.

[5] Collins, G. and Pryor, L. On the Misuse of Filter Conditions: A Critical Analysis, in *Current Trends in AI Planning*, (eds. Backström, C. & Sandewall, E.), IOS Press, 1993.

[6] Currie, K.W. and Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence Journal*, Volume 51, No 1, 1991. Also available as AIAI-TR-67.

[7] Drabble, B., Conversion of SIPE-2 Domain Descriptions to O-Plan Task Formalism, O-Plan2 Technical Report ARPA-RL/O-Plan2/TR/1 Version 1, March 1993.

[8] Drabble, B., Applying O-Plan2 to Military Logistics Planning, O-Plan Technical Report ARPA-RL/O-Plan2/TR/9 Version 1, September 1993.

[9] Drabble, B., Comparison of the CAMPS system with O-Plan2: Architecture and Reasoning Capabilities, O-Plan Technical Report ARPA-RL/O-Plan2/TR/12 Version 1, January 1994.

[10] Drabble, B., NEO Scenarios for O-Plan2 Demonstrations, O-Plan Technical Report ARPA-RL/O-Plan2/TR/4 Version 2, March 1994.

[11] Drabble, B., Gil, Y. and Tate, A., *Acquiring Criteria for Plan Quality Control*, Proceedings of the AAAI Spring Symposium Workshop on Integrated Planning Applications, June 1995. Stanford Univerisity, CA, USA. Publishers the American Association for Artificial Intelligence, Menlo Park, California.

[12] Drabble, B. and Gil, Y., Yes, but why is that plan better?, Proceedings of the International Conference on Artificial Intelligence in the Petroleum Industry, 13th-15th September 1995, Lillehammer, Norway.

[13] Drabble, B. and Tate, A., The Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), AAAI Press, Chicago, USA, June 1994.

[14] Drabble, B., Tate, A., Dalton, G. and Reece, G., Condition Types in O-Plan2: A Discussion, O-Plan2 Discussion Note ARPA-RL/O-Plan2/DN/1 Version 1, January 1994.

[15] Fowler, N., Cross, S.E. and Owens, C., The ARPA-Rome Knowledge-Based Planning and Scheduling Initiative, *IEEE Expert: Intelligent Systems and their Applications*, Vol. 10, No. 1, pp. 4-9, February 1995, IEEE Computer Society.

[16] Gil, Y., Refinement in a Reflective Architecture, Proceedings of the Twelfth International Conference on Artificial Intelligence, Seattle, WA, USA. August 1994. Published by AAAI Press/The MIT Press Menlo Park, CA, USA.

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

[18] Joslin, D. and Pollack, M.E., Least Cost Flaw Repair: A Plan Refinement Strategy for Partial Order Planning, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, August 1994.

[19] Lehrer, N., (ed.), ARPI KRSL Reference Manual 2.0.2, February, 1993. ISX Corporation.

[20] Ludlow, C., Looking at O-Plan2 and Sipe-2 Through Missionaries and Cannibals, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. M.Burstein), Morgan-Kaufmann, 1994.

[21] Nilsson, N.J. *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill Book Company, New York, USA, 1971.

[22] Reece, G.A. and Tate, A., Synthesizing Protection Monitors from Causal Structure, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), pp. 146-152, AAAI Press, Chicago, USA, 1994.

[23] Reece, G.A.. and Tate, A., The Pacifica NEO Scenario, O-Plan2 Technical Report ARPA-RL/O-Plan2/TR/3 Version 1, March 1993.

[24] Reece, G.A., Tate, A., Brown. D. and Hoffman, M., The PRECiS Environment, Presented at the ARPI Workshop as part of AAAI-93 in Washington, DC. Also available as O-Plan2 Technical Paper ARPA-RL/O-Plan2/TP/10.

[25] Sacerdoti, E.D., *The Structure of Plans and Behaviour*, American Elsevier, New York, 1977.

[26] Smith, S.F., Ow, P.S., Potvin, J-Y., Muscettola, N. and Matthys, D., An Integrated Framework for Generating and Revising Factory Schedules, in *Journal of Operational Research Society*, pp539-552, Vol 41, No 6, 1990.

[27] Tate, A., Generating Project Networks, Proceedings of the Fifth International Joint Conference on Artificial Intelligence, William Kaufmann Inc, 1977.

[28] Tate, A., Planning and Condition Monitoring in a FMS, Proceedings of the International Conference on Flexible Automation Systems, Institute of Electrical Enginers, London, July, 1984.

[29] 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.

[30] Tate, A. and Whiter. A., Multiple Resource Constraints and an Application to a Naval Planning Problem, Proceedings of the First Conference on Artificial Intelligence Applications, pp410-416, AAAI, Denver, Colorado, USA, December, 1984.

[31] Tate, A., Drabble, B. and Dalton, J., The Use of Condition Types to Restrict Search in an AI Planner, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, August 1994.

[32] Wilkins, D. *Practical Planning*, Morgan-Kaufmann, 1988.

Appendix O-A – Plan Repair Algorithm

The aim of this Appendix is to describe the approach taken in the O-Plan third year demonstration to the problem of repairing an ongoing plan in the face of changes from the environment and in the task itself.

The aim of the demonstration was to show O-Plan in a integrated command, planning and control environment in which a number of changes were occurring in the domain and in the overall task requirement. The demonstration showed how O-Plan could integrate a number of pre-assembled repair plans, e.g., repairing a blown engine, repair a flat tyre, etc; into an ongoing and executing plan. For the purposes of the demonstration the integration of the repair plans into the ongoing plan was accomplished via the planning agent. However, the techniques and methods used could easily have been integrated into the capabilities of a separate execution agent.

The plan repair algorithms deal with two types of plan state entity as follows:

- **Table of Multiple Effects (TOME) Entry**
  A TOME entry is created for each effect asserted in the plan and is of the form `pattern = value at node-end`, i.e., `p = v @ n`. For example, `(colour_of ball) = green at end_of node-1`
- **Goal Structure Table (GOST) Entry**
  A GOST entry is created for each protected range in the plan and is of the form `condition_type pattern = value at condition-node-end from contributor-node-end`, i.e. `p = v at c from [e]`. This specifies that the pattern is asserted at the contributor-node-end and is required at the condition-node-end. For example, `unsupervised (colour_of ball) = green at (begin_of node-1-2) from (end_of node-1)`. Multiple disjunctive constraints are possible.

Each of these entities is maintained by the O-Plan TOME and GOST Manager (TGM). A plan failure occurs when one or more of the GOST entries are broken, i.e. a contributor of a GOST entry is not asserted as expected or an external world event occurs which asserts extra effects into the plan which breaks the protected GOST range.

Plan repairs are dealt with by a number of knowledge sources. The knowledge sources are responsible for deciding what action to take when a plan failure has been detected by the TGM and for making a repair to the effected plan. A number of knowledge sources have been defined (in addition to those required for plan generation and agent capabilities) to allow the system to repair broken GOST entries as follows:

- KS-EXECUTE starts off the execution of the plan and initialises the datastructures needed by the other execution support knowledge sources.
- KS-EXECUTION-SUCCESS updates the list of executed node-ends to indicate the node-end has executed successfully.

- KS-EXECUTION-FAILURE decides which GOST entries have been effected and posts a number of KS-FIXs to deal with each broken GOST entry.
- KS-UNEXPECTED-WORLD-EVENT deals with the consequences of the occurrence of an unexpected world event and posts a number of KS-FIXs to deal with each broken GOST entry.
- KS-FIX deals with the resatisfaction of a GOST entry broken by a plan failure.
- KS-CONTINUE-EXECUTION decides which node-ends can begin execution after the plan has been repaired.

The main problems dealt with by the repair mechanisms are as follows:

- **Execution Failure**:
  An execution failure occurs when one or more of the expected effects of a node end fail to be asserted. For example, the node end corresponding to the end of the action `Check_out_ground_transport` should assert that the status of the engine and tyres was fine, e.g., `(engine_status gt1) = working` and `(tyre_status gt1) = working`. These may not in fact be satisfied after the plan failure. This type of failure may cause problems if the expected effects of the action are needed to satisfy the preconditions of a later action. For example, the evacuation of people from an outlying city can only precede if the tyres and engine of the ground transport continue to function correctly.
- **Unexpected World Event**:
  Unexpected events cause effects in the world which make planned actions fail. For example, a landslide event may have the effect `(road_status Abyss_to_Barnacle) = closed` and this would interfere with any action requiring the road to be open.

The description of the algorithms of the execution and plan repair system is divided into three main sections. Section 2 describe how the system maintains an execution fringe of those node ends which have been executed and those which are awaiting execution. Section 3 describes how the system deals with plan failures and Section 4 describes how unexpected world events are dealt with.

2 Maintaining the Execution Fringe

An activity, dummy node or event is represented in an O-Plan plan as a node with two ends, a `begin_end` and an `end_end`, each of which is represented via a time point. Conditions and effects can be attached to either end of a node and monitored by the execution system. The execution system reasons purely in terms of node-ends and not in terms of activities or events. The system is driven by the success and failure messages from the model of the world. The system calculates an "execution fringe" of the activities in the partially ordered plan which are currently being executed. The reason for maintaining the execution fringe is to provide a context within which replanning can take place and to provide a focus point when considering

where to insert repair actions, i.e., after all node-ends which have executed and before any node-ends waiting to execute. This point is known as the plan's *neck point* and a single dummy node can be added to the plan by the repair algorithm to neck the plan at this point on need.

The execution fringe instantiated as a list of the node-ends which are currently ready for execution, i.e., node-end E is ready for execution when all node-ends that are linked before it have successfully executed and the required condition contributor are available. This ensures that all explicit ordering constraints (i.e., links from the orderings clauses of O-Plan TF schemas) are satisfied and that all node-ends that provide effects needed by conditions at E have provided those effects. The actual "ready to execute" check considers only whether all the node-ends linked before E have been executed, regardless of whether the execution was successful. It assumes that any problems due to execution failures or world events have been fixed. (It is the responsibility of other parts of the system to ensure that this is so.)

The "ready to execute" check also ignores the temporal constraints in the plan. Temporal constraints are handled by putting node-ends that are ready to execute in a "departure queue" and not dispatching them, (i.e., sending them to the world simulator) until all node-ends with earlier due-times have been executed. This means that node-ends can be dispatched to the world simulator in advance of their actual execution time. The system must obey temporal constraints to this extent because it sometimes need to link a node-end after all the node-ends that have been executed and before all node-ends that have not. If the system looked only at links when deciding when to execute a node-end, it might execute E1 before E2 when the temporal constraints indicate they had to be the other way around. For example, if E1 has executed, E2 had not and the system trys to link something after E1 and before E2 the temporal constraints will not allow it.

During execution, node-ends take on execution status values in the following order:

- **:not-ready**
  Not ready for execution, either because the system has not examined it yet or because some precondition has not yet been met.
- **:ready**
  Assigned when the system determines that all preconditions have been met and the node-end has been queued for execution.
- **:sent** Assigned when the system sends an the appropriate execution message to the execution support system.
- **:finished** Assigned when the system receives a success or failure message from the execution support system.

A node-end with status `:ready` can have its status set back to `:not-ready` when KS-CONTINUE-EXECUTION rebuilds the departure queue. This happens when the system has finished changing the plan after an execution failure or an unexpected world event occur.

3 Dealing with Execution Failures

When an execution failures occur at a particular node-end some of the effects due to be asserted may not occur. They're returned from the execution monitoring system to the planning agent as a list of failed-effects. The task of the planning system is to fix the plan so that any condition that needed one of the failed effects as a contributor is satisfied in some other way. The fix can be relatively simple, e.g., there is already another contributor in the GOST entry or there is a suitable alternative contributor already present in the plan. If these simple fixes cannot be applied then the system will attempt to add a new action to the plan through which a repair plan can be introduced. However, if there are no conditions requiring the failed effects then the execution "failure" can be ignored.

The main algorithm used by the system to track execution and initiate repairs is as follows:

- Mark the node-end as having been executed.
- If there are no failed effects, then a repair is not needed.
- If there are failed effects then remove the TOME entries that correspond to them
- Determine which GOST entries are affected by the failed (removed) effects. If there are none, then a repair is not needed.
- By reaching this point there is a definite failure and the system needs to instigate a repair activity. The search for a repair plan is as follows:
  - Search through the affected GOST entries in turn.
  - If a GOST entry has more than one contributor, check if any are still valid (It is also possible that more than one is from the same node-end, so that a failure might take out more than just one).
  - If there are still some valid contributors, reduce the contributor list; otherwise record the GOST entry as truly broken.
  - If no GOST entries are truly broken, then the repair is complete.
- By reaching this point some GOST entries are truly broken, and the planner will need to post agenda entries for each of the broken GOST entries. For each broken GOST entry, the planner posts a KS-FIX agenda entry which has very similar functionality to the knowledge source KS-ACHIEVE. Its function is to satisfy an **achieve** type condition [31] at a specified node-end in the plan either by:
  - finding an existing alternative contributor in the plan.
  - bringing in additional actions (a repair plan) which asserts the appropriate effect.
- The **achieve** condition for the broken GOST entry will be *after* the *neck* point that is linked after all node-ends that have been executed so far (the execution fringe).

- If there is only one node-end in the execution fringe, use it as the neck point. Otherwise, add a new neck node, link it after all members of the execution fringe.
- Once the neck point has been identified the system can carry out the lower level detail of the repair. The algorithm is as follows: (Where A is the end of the neck node)
  - Step through each of the truly broken GOST enties
  - If the condition is not an `unsupervised` condition type indicate that the contributors for it are not yet defined. Post a KS-FIX to re-establish the condition at the required point, i.e. to "achieve p = v at e after A".
  - If g is a supervised condition the `p = v` must be established over a range, rather than just at a point.
  - Create a new dummy node `d` to act as the "delivery point" and Link `d` after the neck point, before the effect and before all node-ends that are spanned by the condition and have not yet been executed.
  - Change the conditions value to have `d` as the contributor and give `d p=v` as an effect in the TOME.
  - Post a KS-FIX to re-establish `p=v at d`, i.e., to "achieve p = v at d after A"

  The system must be consistent in its use of a single end of d for both conditions and effects to avoid "gaps" in the goal structure which would effect the meaning of the plan. `d` is linked into the plan like this:

  end_of after-node -> begin_of d -> end_of d -> spanned node-ends

  To ensure no gaps are left the system uses the default value for condition_contributor_node_end (obtained from the TF compiler). This is referred to as node-end "ccne_of d" The systems needs to re-establish p=v at the same end of d, so the `achieve` condition should "achieve p = v at ccne_of d after A".
- The current list of alternatives is altered so that the system does not backtrack over alternatives before the fix.
- Post a KS-CONTINUE-EXECUTION to continue execution after the fixes have been made.

4 Dealing with Unexpected World Events

If an unexpected world event occurs in the world that is not anticipated in the current plan. The event is reported as a time, an event pattern, and a list of effects (`pattern` and `value` pairs). For instance, the occurrence of a landslide event would be reported as:

```
event {landslide} with effects
   {status road-a} = blocked,
   {status road-b} = blocked;
```

Events are treated the same way as plan activities except they are not placed in the plan until they have occurred. The effects may break GOST ranges in the plan and if so, the planner must try to satisfy those conditions some other way. However, even if no GOST entries are broken, the planner needs to add a node to represent the world event. This is because, even if the event's effects don't make any difference now, they may matter later on.

The new node represents something that has definitely and already happened. So it must be linked after all node-ends that have already been executed and before all node-ends that have not yet been executed. The new node's effects can't be added until we've removed any broken GOST entries. Otherwise the TOME and GOST Manager might try to preserve those entries when we put in the effects.

The algorithm for dealing with unexpected world events is as follows:

- Add an event node, E, to represent the world event. Link it after the execution fringe. Mark E as having already been executed.
- Edit the GOST to remove any contributors that can no longer contribute, and get a list of the truly broken GOST entries.
- For each truly broken GOST entry g:

  Set up for and post a KS-FIX agenda entry as in the case of an execution failure using end_of E as the neck point.
- Add the world event's effects at end_of E.
- If there were no truly broken GOST entries, then the repair is complete.
- Otherwise, the current list of alternatives is altered so the system does not backtrack over alternatives before the fix. Post a KS-CONTINUE-EXECUTION to continue execution after the fixes have been made.

*U.S. GOVERNMENT PRINTING OFFICE: 1996-710-126-20070

DISTRIBUTION LIST

addresses	number of copies
NORTHRUP FOWLER III ROME LABORATORY/C3C 525 BROOKS ROAD ROME NY 13441-4505	10
AI APPLICATIONS INSTITUTE THE UNIVERSITY OF EDINBURGH 80 SOUTH BRIDGE EDINBURGH EH1 1HN UK	5
ROME LABORATORY/SUL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ROME LABORATORY/C3AB 525 BROOKS RD ROME NY 13441-4505	1
NAVAL WARFARE ASSESSMENT CENTER GIDEP (QA50) ATTN: RAYMOND TADROS PO BOX 8000 CORONA CA 91718-8000	1
AFIT ACADEMIC LIBRARY/LDEE 2950 P STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1

WRIGHT LABORATORY/MLPO
ATTN: R. L. DENISON
BLDG 651
3005 P STREET, STE 6
WRIGHT-PATTERSON AFB OH 45433-7707 — 1

AL/CFHI, BLDG 248
ATTN: GILBERT G. KUPERMAN
2255 H STREET
WRIGHT-PATTERSON AFB OH 45433-7022 — 1

AIR FORCE HUMAN RESOURCES LAB
TECHNICAL DOCUMENTS CENTER
OL AL HSC/HRG, BLDG 190
WRIGHT-PATTERSON AFB OH 45433-7604 — 1

AUL/LSE
BLDG 1405
600 CHENNAULT CIRCLE
MAXWELL AFB AL 361126424 — 1

US ARMY SPACE & STRATEGIC
DEFENSE COMMAND
CSSD-IM-PA
PO BOX 1500
HUNTSVILLE AL 35807-3801 — 1

NAVAL AIR WARFARE CENTER
6000 E. 21ST STREET
INDIANAPOLIS IN 46219-2189 — 1

COMMANDING OFFICER
ATTN: W. BRATT
CODE 772, NCCOSC RDTE DIVISION
53560 HULL STREET
SAN DIEGO CA 92152-5001 — 1

COMMANDER, TECHNICAL LIBRARY
4747000/C0223
NAVAIRWARCENWPNDIV
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6001 — 1

SPACE & NAVAL WARFARE SYSTEMS
COMMAND (PMW 178-1)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200 — 2

COMMANDER, SPACE & NAVAL WARFARE 1
SYSTEMS COMMAND (CODE 32)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200

US ARMY MISSILE COMMAND 2
AMSMI-RD-CS-R/DOCUMENTS
RSIC BLDG 4484
REDSTONE ARSENAL AL 35898-5241

LOS ALAMOS NATIONAL LABORATORY 1
PO BOX 1663
REPORT LIBRARY, P364
LOS ALAMOS NM 87545

AEDC LIBRARY 1
TECHNICAL REPORTS FILE
100 KINDEL DRIVE, SUITE C211
ARNOLD AFB TN 37389-3211

COMMANDER 1
USAISC
ASHC-IMD-L, BLDG 61801
FT HUACHUCA AZ 85613-5000

US DEPT OF TRANSPORTATION LIBRARY 1
FB10A, M-457, RM 930
800 INDEPENDENCE AVE, SW
WASH DC 22591

AIR WEATHER SERVICE TECHNICAL 1
LIBRARY (FL 4414)
859 BUCHANAN STREET
SCOTT AFB IL 62225-5118

AFIWC/MSO 1
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016

SOFTWARE ENGINEERING INSTITUTE 1
CARNEGIE MELLON UNIVERSITY
4500 FIFTH AVENUE
PITTSBURGH PA 15213

DIRNSA 1
R509
9800 SAVAGE ROAD
FT MEADE MD 20755-6000

NSA/CSS 1
K1
FT MEADE MD 20755-6000

DCMAD/WICHITA/GKEP 1
SUITE B-34
401 N MARKET STREET
WICHITA KS 67202-2095

PHILLIPS LABORATORY 1
PL/TL (LIBRARY)
5 WRIGHT STREET
HANSCOM AFB MA 01731-3004

THE MITRE CORPORATION 1
ATTN: E. LADURE
D460
202 BURLINGTON RD
BEDFORD MA 01732

OUSD(P)/DTSA/DUTD 2
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

DR JAMES ALLEN 1
COMPUTER SCIENCE DEPT/BLDG RM 732
UNIV OF ROCHESTER
WILSON BLVD
ROCHESTER NY 14627

DR YIGAL ARENS 1
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

DR RAY BAREISS 1
THE INST. FOR LEARNING SCIENCES
NORTHWESTERN UNIV
1890 MAPLE AVE
EVANSTON IL 60201

MR. JEFF BERLINER 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

DR MARIE A. BIENKOWSKI 1
SRI INTERNATIONAL
333 RAVENSWOOD AVE/EK 337
MENLO PRK CA 94025

DR MARK S. BODDY 1
HONEYWELL SYSTEMS & RSCH CENTER
3660 TECHNOLOGY DRIVE
MINNEAPOLIS MN 55418

DR PIERO P. BONISSONE 1
GE CORPORATE RESEARCH & DEVELOPMENT
BLDG K1-RM 5C-32A
P. O. BOX 8
SCHENECTADY NY 12301

MR. DAVID BROWN 1
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269

DR MARK BURSTEIN 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

DR GREGG COLLINS 1
INST FOR LEARNING SCIENCES
1890 MAPLE AVE
EVANSTON IL 60201

MR. RANDALL J. CALISTRI-YEH 1
ORA CORPORATION
301 DATES DRIVE
ITHACA NY 14850-1313

DR STEPHEN E. CROSS 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213

DR THOMAS CHEATHAM 1
HARVARD UNIVERSITY
DIV OF APPLIED SCIENCE
AIKEN, RM 104
CAMBRIDGE MA 02138

MS. LAURA DAVIS 1
CODE 5510
NAVY CTR FOR APPLIED RES IN AI
NAVAL RESEARCH LABORATORY
WASH DC 20375-5337

MS. GLADYS CHOW 1
COMPUTER SCIENCE DEPT.
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

DR THOMAS L. DEAN 1
BROWN UNIVERSITY
DEPT OF COMPUTER SCIENCE
P.O. BOX 1910
PROVIDENCE RI 02912

DR WESLEY CHU 1
COMPUTER SCIENCE DEPT
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

MR. ROBERTO DESIMONE 1
SRI INTERNATIONAL (EK335)
333 RAVENSWOOD AVE
MENLO PRK CA 94025

DR PAUL R. COHEN 1
UNIV OF MASSACHUSETTS
COINS DEPT
LEDERLE GRC
AMHERST MA 01003

DR JON DOYLE 1
LABORATORY FOR COMPUTER SCIENCE
MASS INSTITUTE OF TECHNOLOGY
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139

DR. BRIAN DRABBLE 1
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH/80 S. BRIDGE
EDINBURGH EH1 LHN
UNITED KINGDOM

MR. SCOTT FOUSE 1
ISX CORPORATION
4353 PARK TERRACE DRIVE
WESTLAKE VILLAGE CA 91361

MR. STU DRAPER 1
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269

DR MARK FOX 1
DEPT O INDUSTRIAL ENGRG
UNIV OF TORONTO
4 TADDLE CREAK ROAD
TORONTO, ONTARIO, CANADA

MR. GARY EDWARDS 1
4353 PARK TERRACE DRIVE
WESTLAKE VILLACA 91361

MR. RUSS FREW 1
GENERAL ELECTRIC
MOORESTOWN CORPORATE CENTER
BLDG ATK 145-2
MOORESTOWN NJ 08057

DR MICHAEL FEHLING 1
STANFORD UNIVERSITY
ENGINEERING ECO SYSTEMS
STANFORD CA 94305

MR. RICH FRITZSON 1
CENTER OR ADVANCED INFO TECHNOLOGY
UNISYS
P.O. BOX 517
PAOLI PA 19301

DR KRISTIAN J. HAMMOND 1
UNIV OF CHICAGO
COMPUTER SCIENCE DEPT/RY155
1100 E. 58TH STREET
CHICAGO IL 60637

RICK HAYES-ROTH 1
CIMFLEX-TEKNOWLEDGE
1810 EMBARCADERO RD
PALO ALTO CA 94303

DR JIM HENDLER
UNIV OF MARYLAND
DEPT OF COMPUTER SCIENCE
COLLEGE PARK MD 20742
1

MS. YOLANDA GIL
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292
1

DR MAX HERION
ROCKWELL INTERNATIONAL SCIENCE CTR
444 HIGH STREET
PALO ALTO CA 94301
1

MR. MORTON A. HIRSCHBERG, DIRECTOR
US ARMY RESEARCH LABORATORY
ATTN: AMSRL-CI-CB
ABERDEEN PROVING GROUND MD
21005-5066
1

MR. MARK A. HOFFMAN
ISX CORPORATION
1165 NORTHCHASE PARKWAY
MARIETTA GA 30067
1

DR RON LARSEN
NAVAL CMD, CONTROL & OCEAN SUR CTR
RESEARCH, DEVELOP, TEST & EVAL DIV
CODE 444
SAN DIEGO CA 92152-5000
1

DR CRAIG KNOBLOCK
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292
1

MR. RICHARD LOWE (AP-10)
SRA CORPORATION
2000 15TH STREET NORTH
ARLINGTON VA 22201
1

MR. TED C. KRAL
BBN SYSTEMS & TECHNOLOGIES
4015 HANCOCK STREET, SUITE 101
SAN DIEGO CA 92110
1

DR JOHN LOWRENCE
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PARK CA 94025 — 1

DR. ALAN MEYROWITZ
NAVAL RESEARCH LABORATORY/CODE 5510
4555 OVERLOOK AVE
WASH DC 20375 — 1

ALICE MULVEHILL
MITRE CORPORATION
BURLINGTON RD
M/S K-302
BEDFORD MA 01730 — 1

DR ROBERT MACGREGOR
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL REY CA 90292 — 1

DR DREW MCDERMOTT
YALE COMPUTER SCIENCE DEPT
P.O. BOX 2158, YALE STATION
51 PROPSPECT STREET
MEW HAVEN CT 06520 — 1

MS. CECILE PARIS
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 — 1

DR DOUGLAS SMITH
KESTREL INSTITUTE
3260 HILLVIEW AVE
PALO ALTO CA 94304 — 1

DR. AUSTIN TATE
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH
80 SOUTH BRIDGE
EDINBURGH EH1 1HN - SCOTLAND — 1

DIRECTOR
ARPA/ITO
3701 N. FAIRFAX DR., 7TH FL
ARLINGTON VA 22209-1714 — 1

DR STEPHEN F. SMITH
ROBOTICS INSTITUTE/CMU
SCHENLEY PRK
PITTSBURGH PA 15213 — 1

DR. ABRAHAM WAKSMAN
AFOSR/NM
110 DUNCAN AVE., SUITE B115
BOLLING AFB DC 20331-0001 — 1

DR JONATHAN P. STILLMAN
GENERAL ELECTRIC CRD
1 RIVER RD, RM K1-5C31A
P. O. BOX 8
SCHENECTADY NY 12345 — 1

DR EDWARD C.T. WALKER
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138 — 1

DR BILL SWARTOUT
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 — 1

GIO WIEDERHOLD
STANFORD UNIVERSITY
DEPT OF COMPUTER SCIENCE
438 MARGARET JACKS HALL
STANFORD CA 94305-2140 — 1

DR KATIA SYCARA/THE ROBOTICS INST
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIV
DOHERTY HALL RM 3325
PITTSBURGH PA 15213 — 1

DR DAVID E. WILKINS
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PARK CA 94025 — 1

DR. PATRICK WINSTON
MASS INSTITUTE OF TECHNOLOGY
RM NE43-817
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139 — 1

HUA YANG 1
COMPUTER SCIENCE DEPT
UNIV OF CALIORNIA
LOS ANGELES CA 90024

MR. RICK SCHANTZ 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

MR JOHN P. SCHILL 1
ARPA/ISO
3701 N FAIRFAX DRIVE
ARLINGTON VA 22203-1714

DR STEVE ROTH 1
CENTER FOR INTEGRATED MANUFACTURING
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGH PA 15213-3890

DR YOAV SHOHAM 1
STANFORD UNIVERSITY
COMPUTER SCIENCE DEPT
STANFORD CA 94305

MR. MIKE ROUSE 1
AFSC
7800 HAMPTON RD
NORFOLK VA 23511-6097

MR. DAVID E. SMITH 1
ROCKWELL INTERNATIONAL
444 HIGH STREET
PALO ALTO CA 94301

JEFF ROTHENBERG 1
SENIOR COMPUTER SCIENTIST
THE RAND CORPORATION
1700 MIN STREET
SANTA MONICA CA 90407-2138

DR LARRY BIRNBAUM 1
NORTHWESTERN UNIVERSITY
ILS
1890 MAPLE AVE
EVANSTON IL 60201

MR. LEE ERMAN
CIMFLEX TECKNOWLEDGE
1810 EMBARCARDERO RD
PALO ALTO CA 94303 — 1

MR DICK ESTRADA
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138 — 1

MR HARRY FORSDICK
BBN SYSTEMS AND TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138 — 1

DR MATTHEW L. GINSBERG
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403 — 5

MR IRA GOLDSTEIN
OPEN SW FOUNDATION RESEARCH INST
ONE CAMBRIDGE CENTER
CAMBRIDGE MA 02142 — 1

DR MOISES GOLDSZMIDT
INFORMATION AND DECISION SCIENCES
ROCKWELL INTL SCIENCE CENTER
444 HIGH ST, SUITE 400
PALO ALTO CA 94301 — 1

MR JEFF GROSSMAN, CO
NCCOSC RDTE DIV 44
5370 SILVERGATE AVE, ROOM 1405
SAN DIEGO CA 92152-5146 — 1

JAN GUNTHER
ASCENT TECHNOLOGY, INC.
64 SIDNEY ST, SUITE 380
CAMBRIDGE MA 02139 — 1

DR LYNETTE HIRSCHMAN
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730 — 1

DR ADELE E. HOWE 1
COMPUTER SCIENCE DEPT
COLORADO STATE UNIVERSITY
FORT COLLINS CO 80523

DR LESLIE PACK KAELBLING 1
COMPUTER SCIENCE DEPT
BROWN UNIVERSITY
PROVIDENCE RI 02912

DR SUBBARAO KAMBHAMPATI 1
DEPT OF COMPUTER SCIENCE
ARIZONA STATE UNIVERSITY
TEMPE AZ 85287-5406

MR THOMAS E. KAZMIERCZAK 1
SRA CORPORATION
331 SALEM PLACE, SUITE 200
FAIRVIEW HEIGHTS IL 62208

DR PRADEEP K. KHOSLA 1
ARPA/ITO
3701 N. FAIRFAX DR
ARLINGTON VA 22203

DR CRAIG KNOBLOCK 1
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

DR CARLA GOMES 1
ROME LABORATORY/C3CA
525 BROOKS RD
ROME NY 13441-4505

DR MARK T. MAYBURY 1
ASSOCIATE DIRECTOR OF AI CENTER
ADVANCED INFO SYSTEMS TECH G041
MITRE CORP, BURLINGTON RD, MS K-329
BEDFORD MA 01730

MR DONALD P. MCKAY 1
PARAMAX/UNISYS
P O BOX 517
PAOLI PA 19301

DR KAREN MYERS 1
AI CENTER
SRI INTERNTIONAL
333 RAVENSWOOD
MENLO PARK CA 94025

DR MARTHA E POLLACK 1
DEPT OF COMPUTER SCIENCE
UNIVERSITY OF PITTSBURGH
PITTSBURGH PA 15260

DR RAJ REDDY 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213

DR EDWINA RISSLAND 1
DEPT OF COMPUTER & INFO SCIENCE
UNIVERSITY OF MASSACHUSETTS
AMHERST MA 01003

DR MANUELA VELOSO 1
CARNEGIE MELLON UNIVERSITY
SCHOOL OF COMPUTER SCIENCE
PITTSBURGH PA 15213-3891

DR DAN WELD 1
DEPT OF COMPUTER SCIENCE & ENG
MAIL STOP FR-35
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

MR JOE ROBERTS 1
ISX CORPORATION
4301 N FAIRFAX DRIVE, SUITE 301
ARLINGTON VA 22203

COL JOHN A. WARDEN III 1
ASC/CC
225 CHENNAULT CIRCLE
MAXWELL AFB AL 36112-6425

DR TOM GARVEY 1
ARPA/ISO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

MR JOHN N. ENTZMINGER, JR. 1
ARPA/DIRO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

LT COL ANTHONY WAISANEN, PHD 1
COMMAND ANALYSIS GROUP
HQ AIR MOBILITY COMMAND
402 SCOTT DRIVE, UNIT 3L3
SCOTT AFB IL 62225-5307

DIRECTOR 1
ARPA/ISO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

OFFICE OF THE CHIEF OF NAVAL RSCH 1
ATTN: MR PAUL QUINN
CODE 311
800 N. QUINCY STREET
ARLINGTON VA 22217

BBN SYSTEMS AND TECHNOLOGY 1
ATTN: MR MAURICE MCNEIL
9655 GRANITE RIDGE DRIVE, SUITE 245
SAN DIEGO CA 92123

DR OREN ETZIONI 1
DEPT OF COMPUTER SCIENCE
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

DR GEORGE ERGUSON 1
UNIVERSITY OF ROCHESTER
COMPUTER STUDIES BLDG, RM 732
WILSON BLVD
ROCHESTER NY 14627

DR STEVE HANKS 1
DEPT OF COMPUTER SCIENCE & ENG'G
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

DR WILLIAM S. MARK 1
LOCKHEED PALO ALTO RSCH LAB
DEPT 9620, BLDG 254F
3251 HANOVER ST
PALO ALTO CA 94304-1187

MR DON MORROW
BBN SYSTEMS & TECHOLOGIES
101 MOONGLOW DR
BELLEVILLE IL 62221 — 1

DR KAREN MYERS
AI CENTER
SRI INTERNATIONAL
333 RAVENSWOOD
MENLO PARK CA 94025 — 1

DR CHRISTOPHER OWENS
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138 — 1

DR ADNAN DARWICHE
INFORMATION & DECISION SCIENCES
ROCKWELL INT'L SCIENCE CENTER
1049 CAMINO DOS RIOS
THOUSAND OAKS CA 91360 — 1

DR JAIME CARBONNEL
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIVERSITY
DOHERTY HALL, ROOM 3325
PITTSBURGH PA 15213 — 1

NR NORMAN SADEH
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIVERSITY
DOHERTY HALL, ROOM 3315
PITTSBURGH PA 15213 — 1

DR JAMES CRAWFORD
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403 — 1

DR TAIEB ZNATI
UNIVERSITY OF PITTSBURGH
DEPT OF COMPUTER SCIENCE
PITTSBURGH PA 15260 — 1

DR MARIE DEJARDINS
SRI INTERNATIONAL
333 RAVENSWOOD AVENUE
MENLO PARK CA 94025 — 1

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.