UK Planning and Scheduling SIG 13

The O-Plan Constraint Associator

Jeff Dalton, Brian Drabble & Austin Tate

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

September 15, 1994

ARPA-RL/O-Plan/TP/14 Version 1

The O-Plan Constraint Associator

Jeff Dalton, Brian Drabble and Austin Tate

Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom
Tel: +(44) 131 650 2732 Fax: +(44) 131 650 6513
Email: A.Tate@ed.ac.uk; B.Drabble@ed.ac.uk; J.Dalton@ed.ac.uk

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.

The aim of this paper is to describe the Constraint Associator which is resident in the database manager (which looks after the plan state) and which forms a new core components of the O-Plan2 architecture. The constraint Associator mediates between the decisions made by the decision making parts of the architecture i.e. knowledge sources and the underlying constraint managers.

This paper describes the motivation for the development of the constraint associator and 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 which were needed to support the constraint associator. These guidelines are 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. The paper concludes with a description of a planning agent whose architecture and components would make it suitable across a number of planning and representation domains.

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 [5] 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 [17] 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.

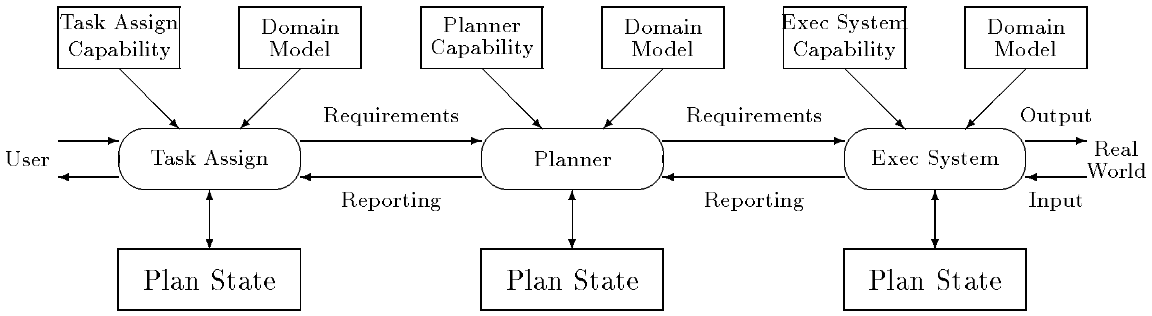

Figure 1: Communication between Strategic, Tactical and Operational Levels

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 [23].

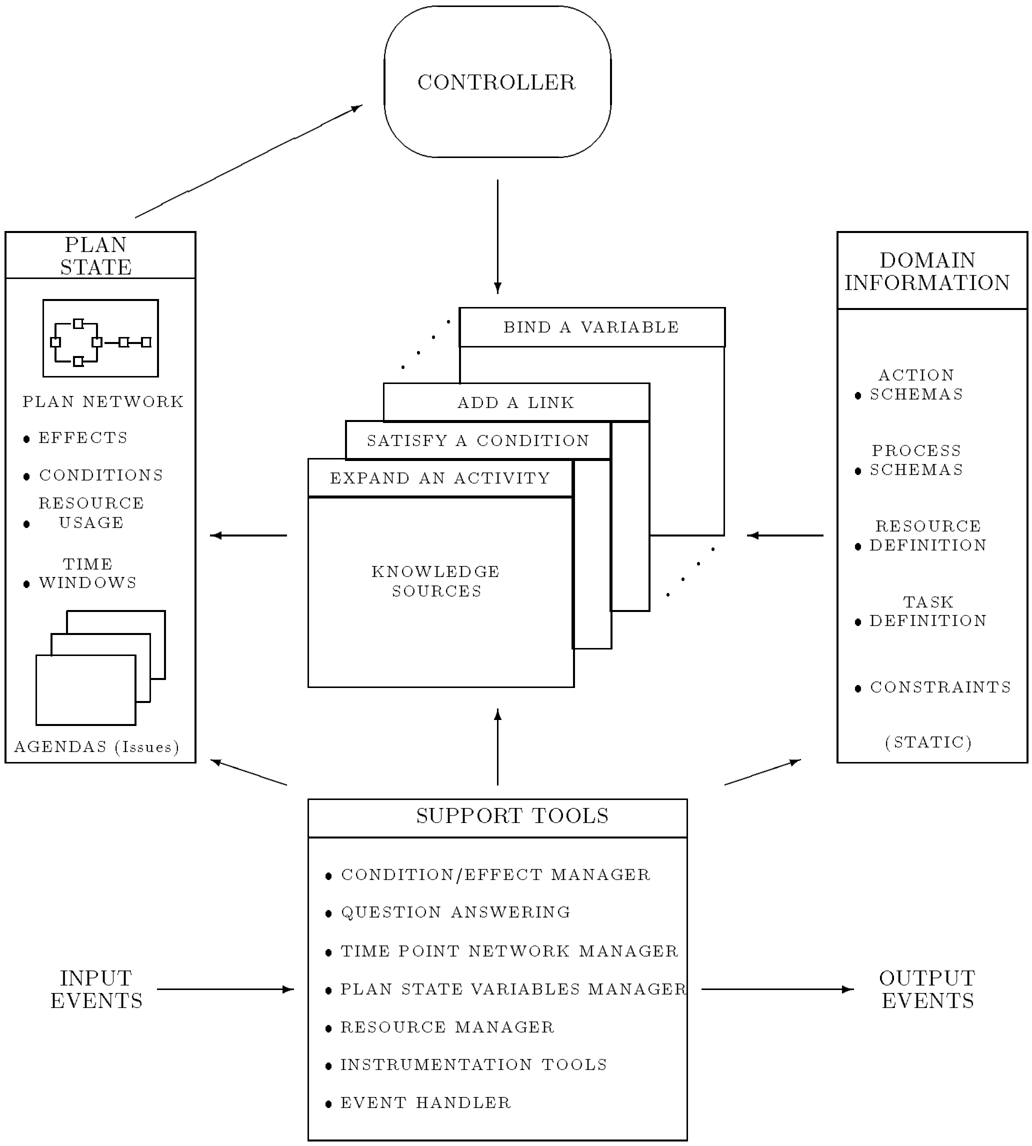

Figure 2: O-Plan2 Architecture

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.

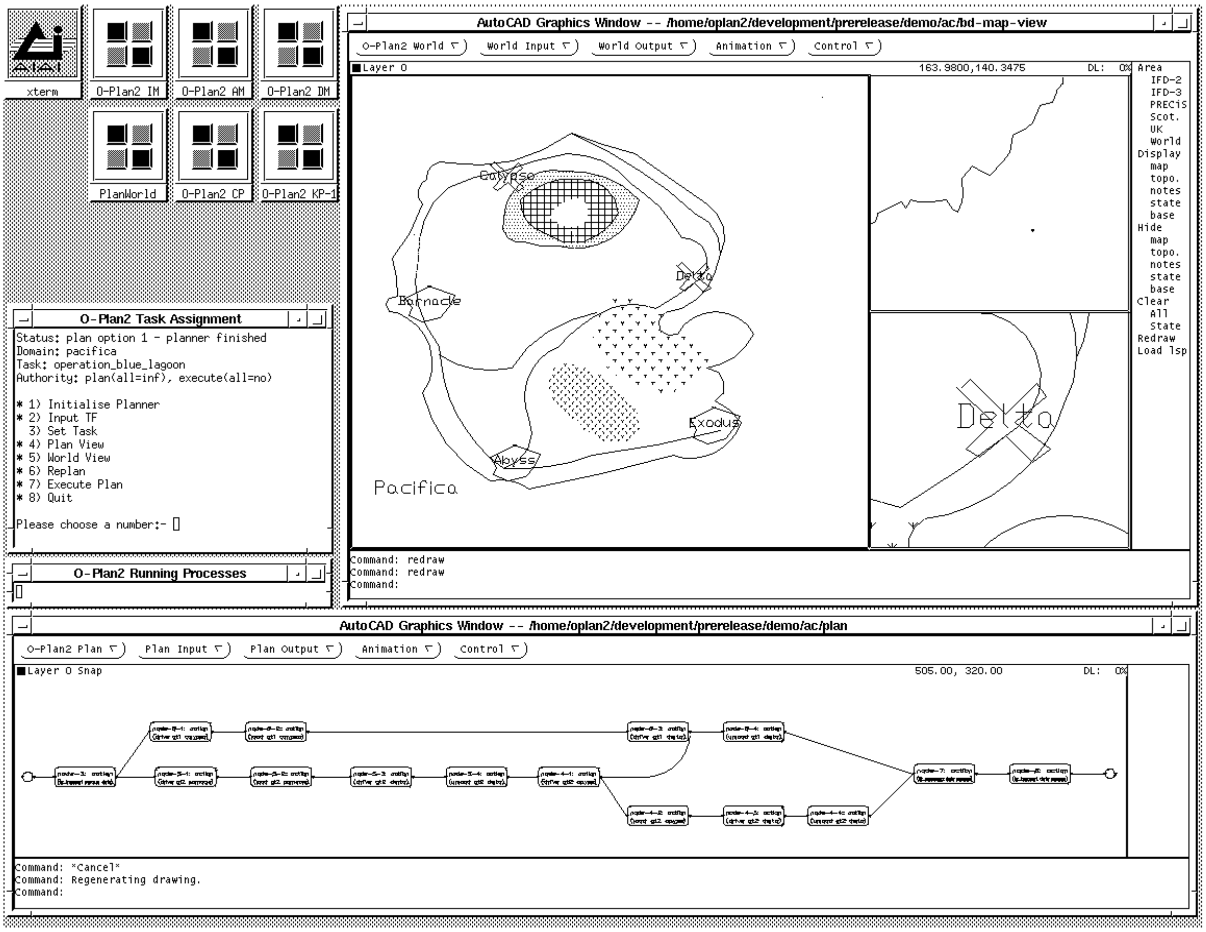

Figure 3: Example Output of the AutoCAD-based User Interface

2 Introduction to the Constraint Associator

The aim of this section is to provide an overview of the aims of the constraint associator and the motivation behind its inclusion in the O-Plan2 system.

Plans in the O-Plan2 systems are represented as a series of constraints which represent the valid plans reachable from a given plan state. The constraints are used to represent different plan entities, e.g. authority, variables, resources, causality, time, etc. The core constraints in the O-Plan2 system are time and variables, the other e.g. resources, authority, causality, etc are defined as *other constraints*. In some applications additional constraint managers may be added to deal with spatial constraints, physical constraints, etc.

Each of the different constraint types is dealt with by a separate constraint manager. As described in the previous section O-Plan2 is an agenda driven architecture in which each outstanding agenda or *issue* type is dealt with by a separate knowledge source. In earlier versions of O-Plan2 it was the responsibility of the knowledge source writer to ensure the relevant constraints were added and updated in the correct order. This meant that the knowledge source writer needed to be sensitive to the cases where a change in one constraint type would require the checking of another constraint type. For example, the restriction of a time window on a point in the time manager may lead to problems in the resource manager. To alleviate the knowledge source writer of this problem it was decided to add a *constraint associator* to the O-Plan2 architecture to act as a mediator between the knowledge sources and the constraint managers. A secondary problem with the earlier architecture was dealing with request for information between different constraint managers. For example, the TOME and GOST manager (which deals with causality constraints) requires temporal information on points when deciding whether and effect of condition protection range can be added to the plan. The functions of the constraint associator are:

1. to take constraints and to route them to the appropriate constraint manager
2. to inform a constraint manager of the need to re-evaluate the constraints under its management due to changes in another constraint manager.
3. to allow the constraint layer to inform the planner that on further investigation a problem has arisen and the plan state should be poisoned[1]

This second requirement lead to an investigation of when constraints need to be re-calculated and the types of constraints involved changes which were involved. From this investigation it was identified that constraints involving *time* and *plan variables* needed to be treated differently as a change in them would necessitate a re-evaluation of other constraints. For example, the restriction of a time point may cause a resource allocations problem, e.g. the resource is no longer returned by the required time. The identification of time constraints and variables as being special have been further strengthened by work at AIAI on the I-N-OVA model [22] and work by [10]. The details of the I-N-OVA model are described in Section 4.

[1] Poisoning a plan state means that current plans state should be abandoned and an alternative chosen.

As a result of the inclusion of the constraint associator a number of changes and requirements were identified for the current O-Plan2 design. These included the development and definition of a minimal planning ontology to allow the different constraint managers to communicate through the same language to the decision making layer of the planner. The constraint managers are queried at to whether they are prepared to manage the constraint through a commit or abort framework. When considering the query the constraint manager is allowed to reply in *one* of *three* ways:

1. **Yes**: the constraint can be added with problem
2. **No**: the constraint cannot be added under any circumstances
3. **Maybe**: the constraint can be added if certain changes are made to the plan state.

The changes which can be added are to either restrict the binding a plan variable can take of to add further temporal restrictions to the plan state.

The main aim was to develop an interface which would allow information (representing impacts on the plan state) from different constraint managers to be *analysed* and *merged*. This would allow all pertinent information to be brought together at one point and thus reduce the size of the search space. An example of such a point would be the expansion of a node where information concerning time, objects, causality, resources and authority is used to ensure that the expansion is valid in that particular plan state. If the planner was forced to explore all options for adding each constraint *separately* the search space could be enormous.

The use of different constraint managers to handle separable parts of the plan state represents O-Plan's view of plans being represented as a series of constraints on *possible* plan elaborations. The different types of constraints and the ways in which they can be manipulated is described in [21].

3 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.

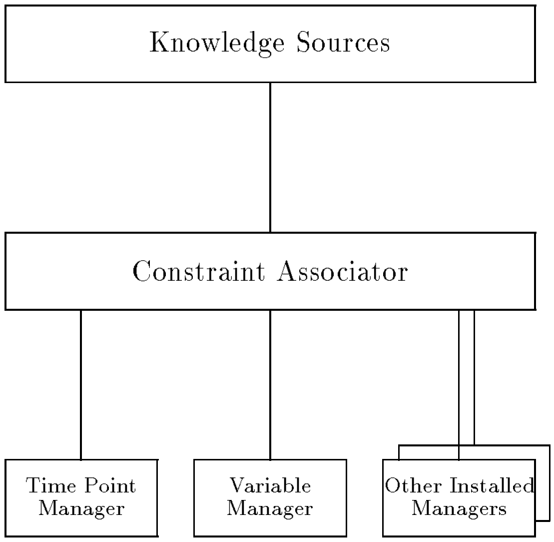

Figure 4: Associator to mediate between Knowledge Sources and Constraint Managers

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 4). 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.

4 I-N-OVA Model of Plans

We have recently been seeking to clarify the model of constraints used within plans. This relates to our approach to separating specific aspects of detailed constraint management in the O-Plan planner. Our model is referred to as <I-N-OVA>. Details of this model are provided in [22].

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 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., [20], [24]).

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.

Recently, formal representations have begun to address issues of realistic plan representations and to model hierarchical planning [2],[12],[13],[26]. 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 [11]:

```
<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[2].

5 Plans Represented as Constraints on Plan Elaborations

It is useful to present a simple abstraction of how a planner or scheduler operates. Figure 5 shows such an abstraction that will be useful in this paper.

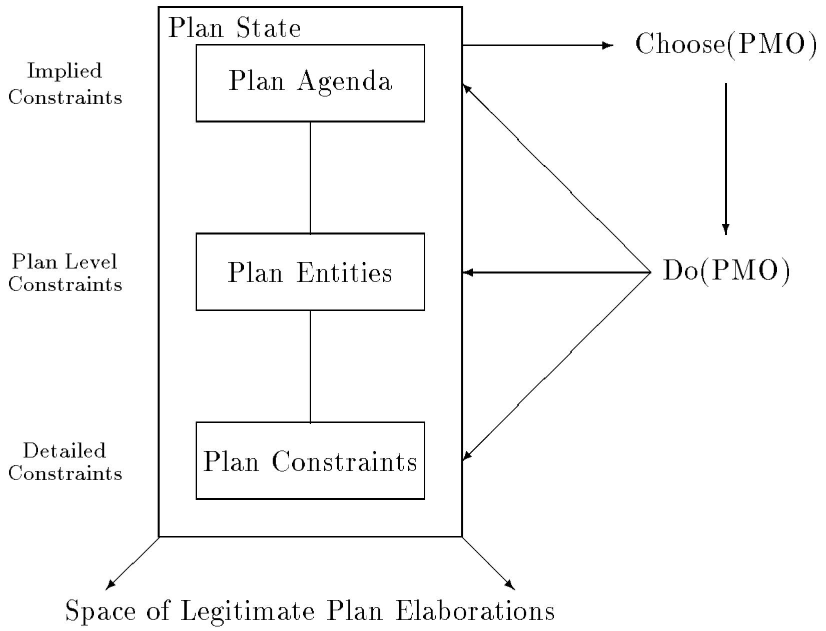

Figure 5: A Framework of Components in a Planning/Scheduling System

Many planners and schedulers work by refining a "current" plan (shown in figure 5 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.[3] The planner or scheduler needs to know what outstanding

[2]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.

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

processing requirements exist in the plan (shown in figure 5 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 has an *Associated Data Structure* (ADS) level of representation [8] 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 [3] 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.

6 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 [6] and used in the FORBIN planner [7], are a good example of the approach. The management of effect and condition (Goal Structure) tables in Nonlin [17] 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 [8]. A commercial Time Map Manager (TMM) is available from Honeywell based on the concepts described in [6]. More powerful temporal relationships are managed by the General Electric TACHYON temporal system [15]. 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 [9] 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 [3] 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.

7 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).

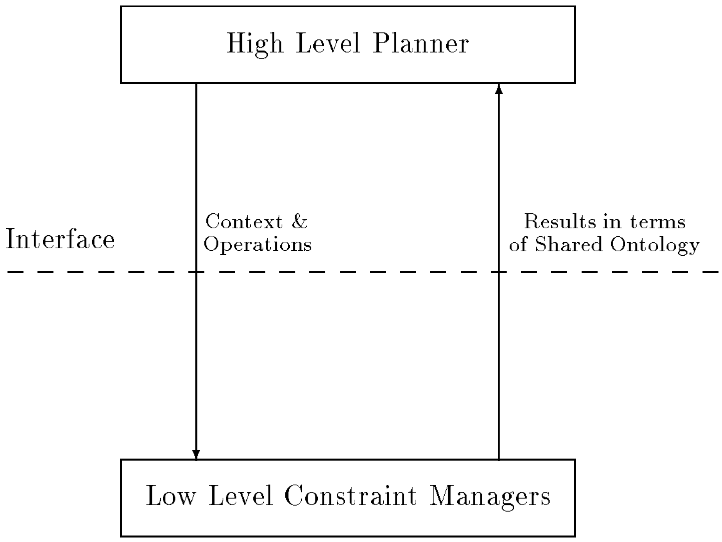

Figure 6: The Interface to Constraint Managers

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 6 shows an overview of the interface.

7.1 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>[4].
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[5].
   - 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 [8]) 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.

7.2 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.

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

[5] 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.

- 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[6].

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.

7.3 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"[7] of *all* (for search space completeness) the legal ways in which the Plan State can be altered

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

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

(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.

8 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.

One advantage of having all constraints pass through a single point and adopting a single standard interface is it becomes possible to engineer planners in a more systematic way. Other constraint managers can be added if they abide by the ontology. A further advantage is the ability to analyse more closely the approach taken by planner such as O-Plan2. The work of [11] aims to define plans as a 5-tuple in which the constraints associated with nodes, time and objects [8] are separated from those constraints involving resources, causality, authority, etc.

[8]Objects in the O-Plan2 system are defined as Plan State Variables

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] 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.

[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. Also available as AIAI Technical Report AIAI-TR-121.

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

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

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

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

[8] 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.

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

[10] 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.

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

[12] Kambhampati, S. and Hendler, J., A Validation-Structure-Based Theory of Plan Modification and Reuse, *Artificial Intelligence*, May, 1992.

[13] 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.

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

[15] Stillman, J., Arthur, R. and Deitsch, A., Tachyon: A Constraint-based Temporal Reasoning Model and its Implementation, *SIGART Bulletin*, 4:3, July 1993.

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

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

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] Yang, Q. Formalizing Planning Knowledge for Hierarchical Planning, *Computational Intelligence*, Vol. 6, No. 1, pp12-24, 1990.