Integrating Constraint Management into an AI Planner

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

Abstract

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 planning, scheduling and control systems.

This paper describes the way in which plan constraints are represented and handled in the O-Plan 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-Plan.

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-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 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 [21].

The O-Plan 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 "issues" remaining on its agenda, 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 alter.
- O-Plan is derived from the earlier Nonlin planner [15] 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 domain description language – Task Formalism (TF).

O-Plan 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 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.

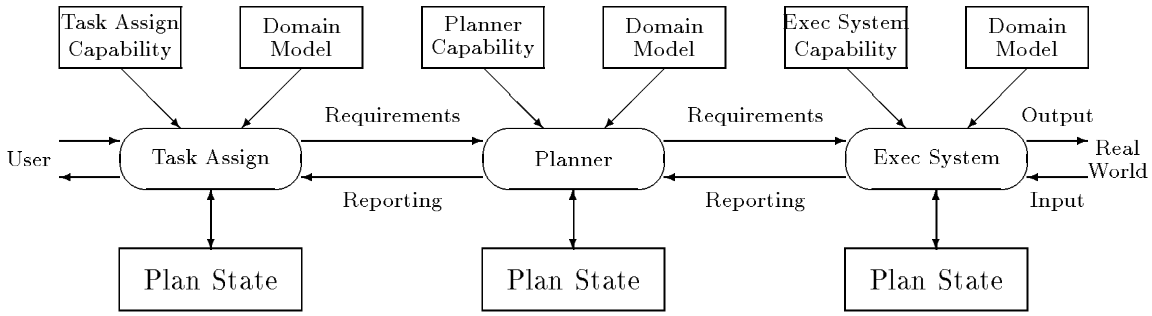

Figure 1: Communication between Strategic, Tactical and Operational Agents

Figure 1 shows the communications between the 3 agents in the O-Plan architecture. The current O-Plan system has a comprehensive planner agent and a simple execution agent [21]. A comprehensive reactive execution agent has also been built in the O-Plan architecture [11]. The task assignment function is provided by a separate process which has a simple menu interface and is not currently in the form of an O-Plan agent.

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 main components within a single O-Plan agent 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 (also referred to as *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 agent components as they appear within the O-Plan planner agent are shown in Figure 2.

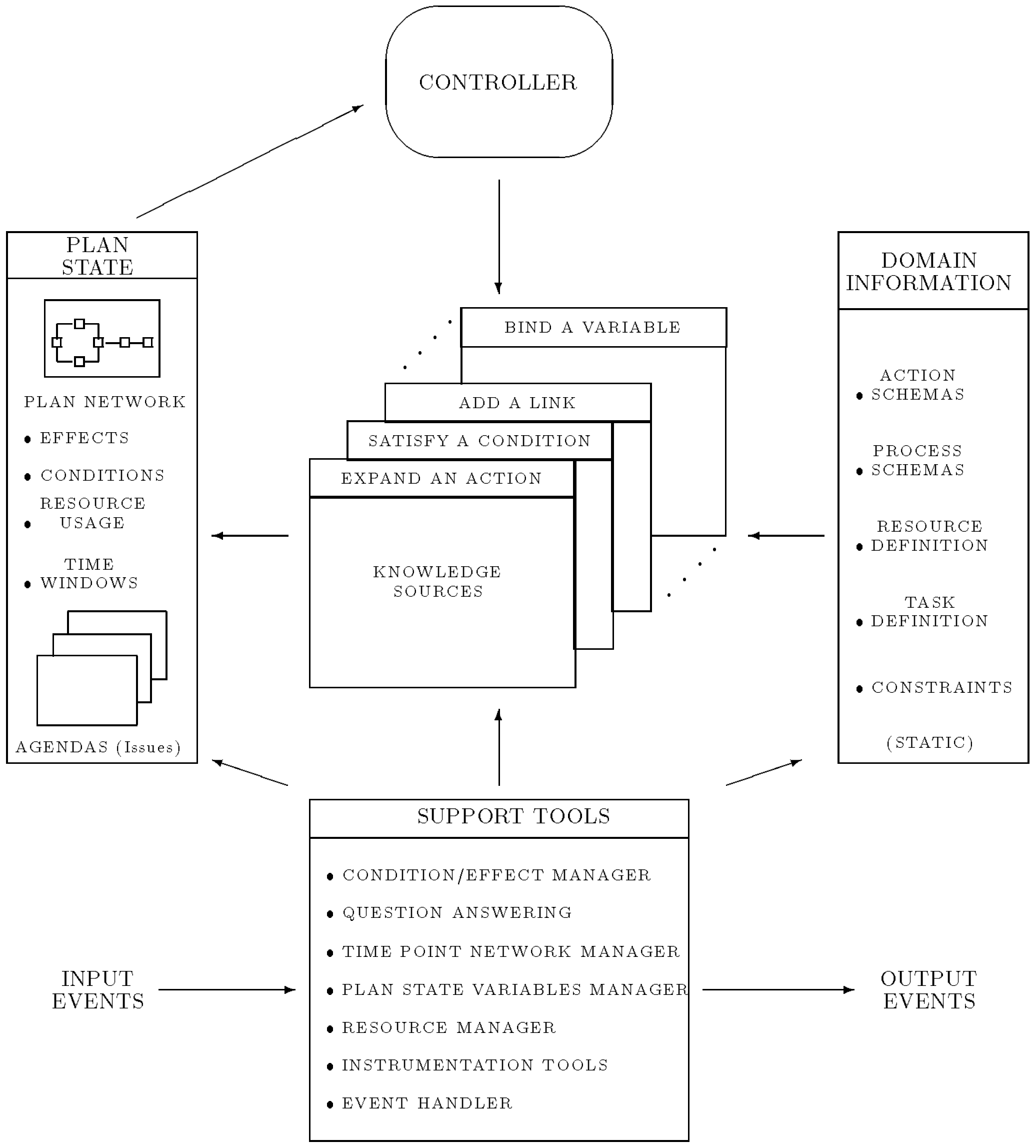

Figure 2: O-Plan Planner Agent Components

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 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 [18] to plan and execute plans for a given task. The lower window shows a *Plan View* (such as 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 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. In figure 3, developer window icons appear along the bottom edge of the screen.

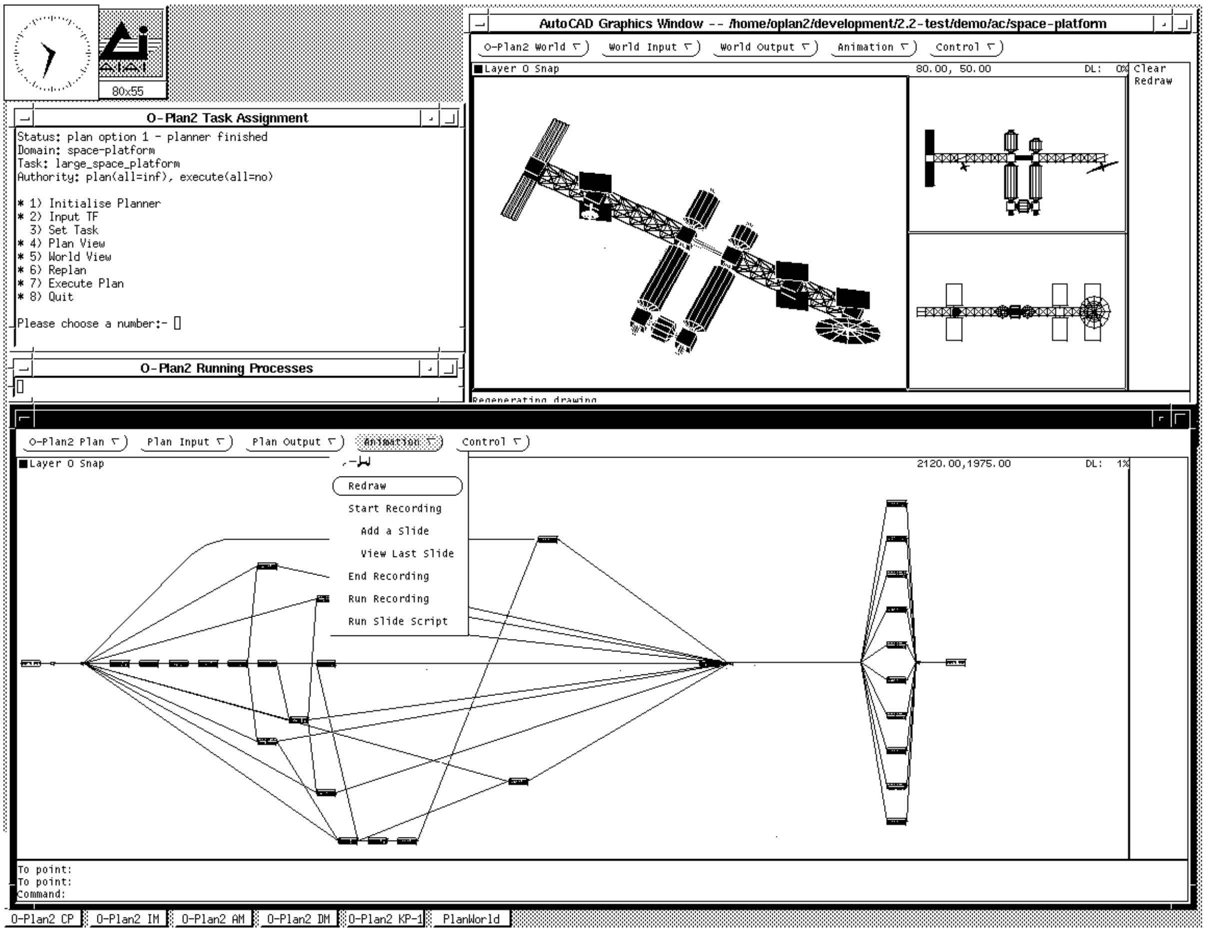

Figure 3: Example Output of the AutoCAD-based User Interface

Recent work on O-Plan has focussed on the representation and management of constraints in planning, particularly in order to simplify some aspects of the architecture (the subject of this paper) and to act as a mechanism for user/system mixed initiative planning [19].

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.

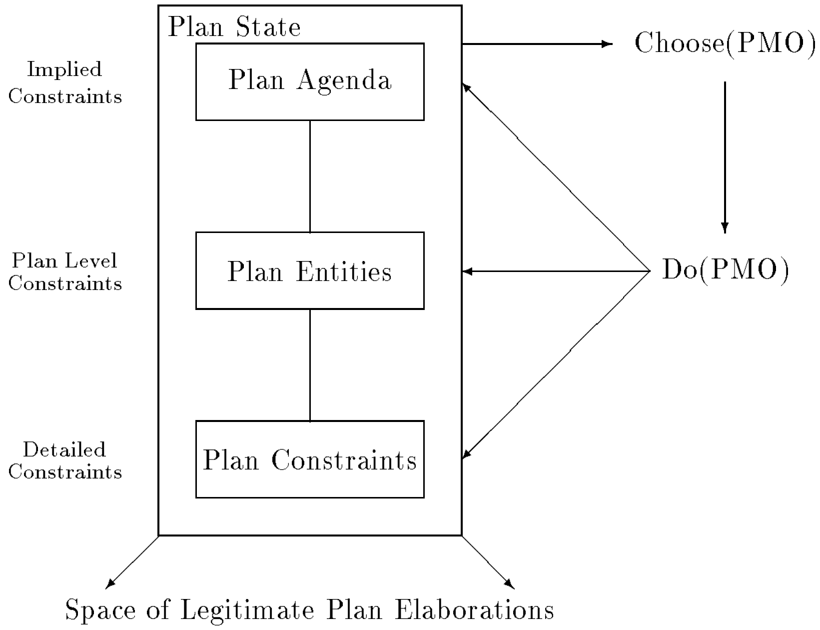

Figure 4: A Framework of Components in a Planning/Scheduling System

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 plan entities which form a particular 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

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

modularity within planning and scheduling systems.

O-Plan has an *Associated Data Structure* (ADS) level of representation [7] which holds the main plan entities (such as actions). The lower level constraints, such as those on on time points and resources in the plan, are managed separately. These 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-Plan 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 and possibly using more efficient algorithms.

As one example, consider support for 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 satisfied 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 [15] 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-Plan 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 [13]. 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-Plan Time Point Network Manager 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-Plan 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-Plan planner which has many similarities to the interface used for the Truth Criterion/QA algorithm used in our systems [15]. This framework could incorporate resource handling by mechanisms as powerful as those based on the Habographs [2] constraint management mechanism incorporated in the Edinburgh TOSCA manufacturing scheduler.

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

4 Constraint Managers in the O-Plan Architecture

O-Plan 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-Plan architecture components and Constraint Managers to help others consider packaging and integration issues.

Our experience with earlier AI planners such as Nonlin and the early versions of O-Plan 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 later versions of O-Plan 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 do not take any decision themselves. They are intended to provide maintain all the information about the constraints they are managing and to respond to questions being asked of them by the decision making level. Examples of Constraint Managers in O-Plan include:

- Time Point Network Manager.
- Effect/Condition Manager and the related Question Answerer.
- Resource Utilisation Manager.
- Object Instantiation (Plan State Variables) Manager.

A guideline for the provision of a good Constraint Manager in O-Plan 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).

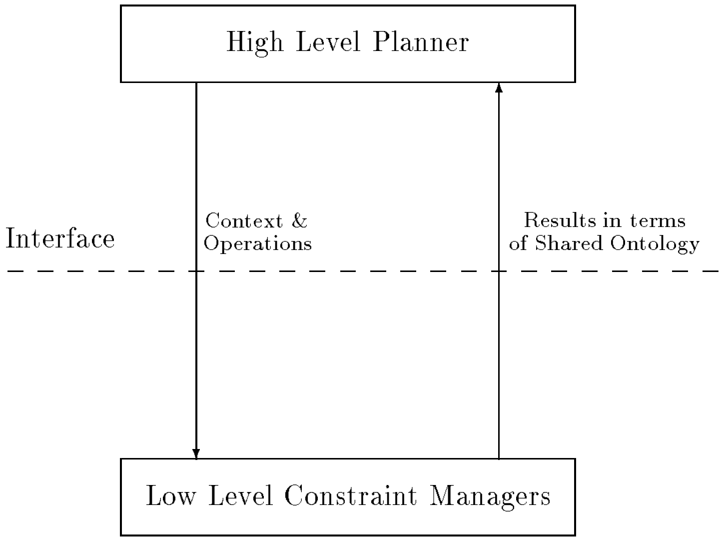

Figure 5: The Interface to Constraint Managers

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 component in an O-Plan agent 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 O-Plan Database Manager.

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

1. initialise Constraint Manager and name base context with a given tag[2].
2. terminate Constraint Manager.
3. push context and name new context with a 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-Plan.

5. restore a previously created context which has the tag specified.

6. open update transaction, and within this:

   - allow changes to managed entities.
   - 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-Plan 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-Plan/O-Base context managed structures – since the implementation of the Associated Data Structure layer in O-Plan guarantees that Constraint Managers will only ever be called when the contexts being referred to are preset within the O-Plan planner.

4.2 Shared Plan Ontology between O-Plan 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 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 an 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[3].

These entities allow for information about constraints and options for correcting constraint violations to be communicated 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]Currently we represent equality (variable codesignation), inequality (non-codesignation) and other restriction (range or property) constraints on the variable.

4.3 The New O-Plan "Standard" Interface for Constraint Managers

The aim in O-Plan 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 [15].

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 and must undertake reliably the task it is given. One focus of the O-Plan 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-Plan "or-tree"[4] 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, from an implementation perspective, 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-Plan (Condition/Effect manager, Resource Utilisation Manager, Plan State Variables Manager and the Time Point Network Manager) 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.

[4] 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-Plan, we wish to localise the interactions between changes in one type of constraint that can lead to changes in other types of constraint. 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 effects/conditions, resources, etc.). The detection and cross-relating of such mutual constraints has been problematic in O-Plan to date. Previously, Knowledge Sources had to be written such that any change in one constraint type that could influence another was programmed in. This was a source of complexity and dependency in teh design that we wish to avoid.

The clarification of the constraint manager interface for O-Plan 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[5]. 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-Plan. 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-Plan and simplify the writing of Knowledge Sources.

Accordingly, the O-Plan 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). The function of detecting mutual constraints in which changes to time and/or variable constraints may affect other constraints which themselves refer to the affected time points or variables is localised in the Constraint Associator.

A number of constraint managers can be "installed" into an O-Plan 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. To give the functionality of the current O-Plan 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 (beyond those managed actively by the currently installed managers). 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].

[5]Other evidence from formal studies is also highlighting the value of separating the constraints on time and the variable codesignation/non-codesignation constraints from other aspects of plan representation (e.g., in [9]). We are developing a description of plans as a set of constraints differentiated into *Issues – Nodes – Orderings/Variables/Auxiliary constraint types* that we refer to as the <I-N-OVA> model [20] to act as a framework for further study and comparison.

[6]Recent work by others (e.g., [10]) is also recognising the practical benefits of being able to isolate the work done for parts of a planning problem into well defined managers which can use specialised algorithms. By not relying on a general search mechanism for all aspects of planning, more realistic tasks can be handled without combinatorial search problems becoming a problem too quickly.

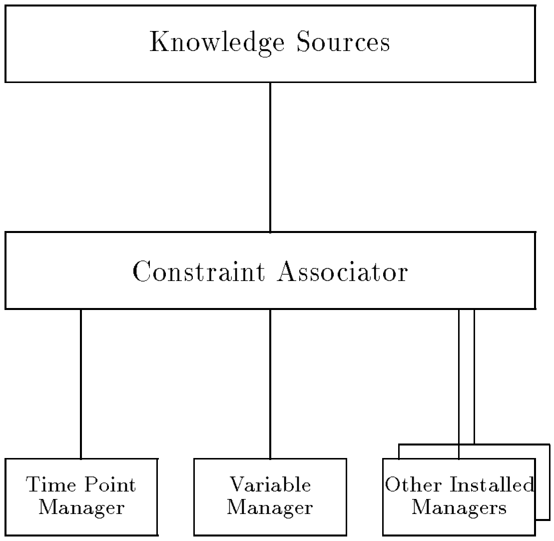

Figure 6: Associator to mediate between Knowledge Sources and Constraint Managers

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-Plan 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-Plan 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-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 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. Thanks to my colleagues on the O-Plan project, Brian Drabble and Jeff Dalton for the discussions we have held to adopt the architecture discussed here.

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.

[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* 52(1), pp. 49-86, 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.

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

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

[10] Penberthy, J.S. and Weld, D.S., Temporal Planning with Continuous Change, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), pp. 1010-1015, AAAI Press, 1994.

[11] 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.

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

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

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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