Technical Report

O-Plan Knowledge Source Framework

Austin Tate

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

March 6, 1995

ARPA-RL/O-Plan/TR/21 Version 1

1 Introduction to O-Plan Knowledge Source Framework – KSF

O-Plan [6] is an open, modular architecture for building computational agents. The O-Plan Architecture Guide [4] describes a facility within the O-Plan design which allows the Knowledge Sources in an O-Plan agent to be described to that agent. This is done through a descriptive language termed the *Knowledge Source Framework (KSF)*.

In an open and modular system in which components can be plugged in, it is important that there be straightforward and clear mechanisms to define those components to the system. A combination of an interface protocol or application programming interface (API) and a description of the module's range of usage of that protocol or interface is needed.

The O-Plan Knowledge Source Framework and a corresponding Knowledge Source Protocol has been outlined in the design of O-Plan to allow for the description and plugging in of an O-Plan agent's Knowledge Sources. Knowledge Sources in O-Plan are the main functional processing assets of the system. Knowledge Sources are executed on one or more O-Plan *Knowledge Source Platforms* (which can be computer platforms or people involved in the planning process). A *Knowledge Source Protocol* defines the way in which an executing Knowledge Source interfaces to the rest of the O-Plan system. This particularly governs the way in which a Knowledge Source gets its task description (from the agenda controller), the way in which it accesses and alters the plan state through the agent data base manager, constraint associator [5] and support modules, and the way in which the Knowledge Source communicates its results or alternative results to the alternatives controller. This is described in more detail in the O-Plan Architecture Guide [4].

Although the KSF is only a concept within O-Plan, it is important to the overall O-Plan design. It is included to give important capabilities needed within a flexible, open computational architecture. There is a need to provide descriptions of plug-in modules to such an open architecture. Recent work on a Systems Integration Architecture for the UK Enterprise Project [2], itself a generalisation of the general open architecture principles in O-Plan has reinforced the need for a processing asset description capability – such as the O-Plan Knowledge Source Framework.

2 Open Systems Integration Architecture

Figure 1 shows a systems integration architecture whose general structure has been adopted on a number of AIAI projects including Enterprise [2] and O-Plan [6]

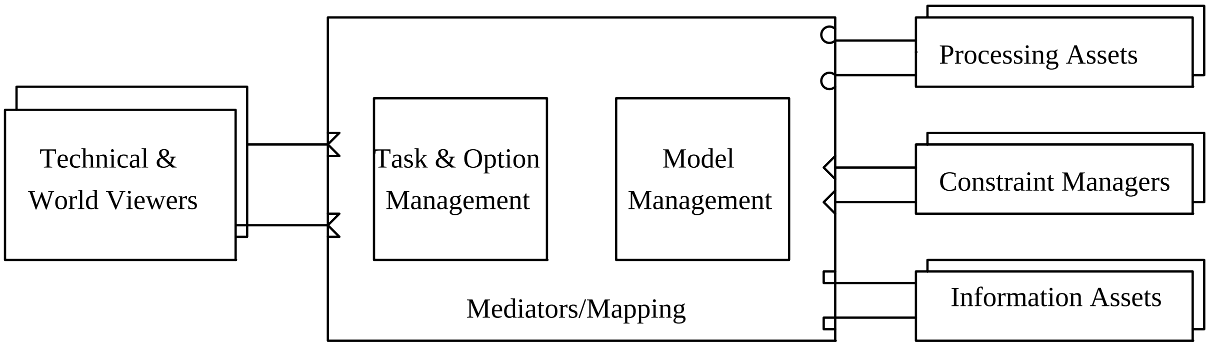

Figure 1: Generic Open Systems Integration Architecture

The various components "plug" into "sockets" within the architectural framework. The sockets are specialised to ease the integration of particular types of "component".

The components are as follows:

Viewers – User interface, visualisation and presentation viewers for the model - sometimes differentiated into *technical* model views (charts, structure diagrams, etc.) and *world* model views (simulations, animations, etc.)

Task and Option Management – The capability to support user tasks via appropriate use of the processing and information assets and to assist the user in managing options being used within the model.

Model Management – coordination of the capabilities/assets to represent, store, retrieve, merge, translate, compare, correct, analyse, synthesise and modify models.

Mediators – Intermediaries or converters between the features of the model and the interfaces of active components of the architecture (such as viewers, processing assets, constraint managers and information assets).

Processing Assets – Functional components (model analysis, synthesis or modification).

Constraint Managers – Components which assist in the maintenance of the consistency of the model.

Information Assets – Information storage and retrieval components.

The generic architecture has been derived from work on a range of projects such as O-Plan. The O-Plan architecture [4] presented in the same general form as described above can be redraw as shown in figure 2.

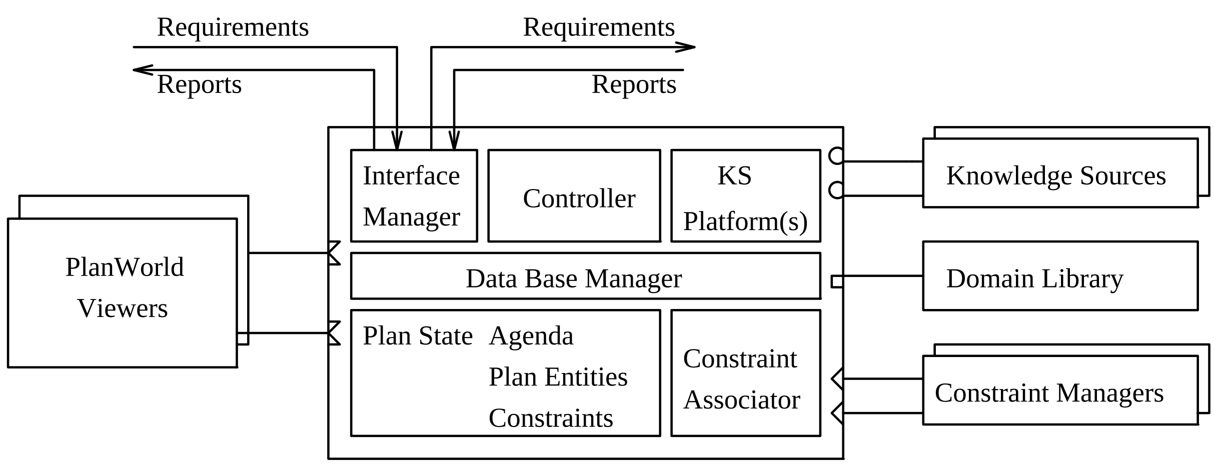

Figure 2: O-Plan Agent Architecture

3 Extracts about KSF from the O-Plan Architecture Guide

As further background, this section includes relevant extracts from the O-Plan Architecture Guide [4] which relate to the Knowledge Source Framework.

In O-Plan, Knowledge Sources run on Knowledge Source Platforms, which are basically processing engines for the knowledge source code. It is hoped that the eventual O-Plan will exploit multiprocessor architectures, where possible, so the current system has a clean separation of its knowledge source platforms from the other system modules, and locking mechanisms need to be put in place to ensure that data in the system is up to date and consistent. Only the final stages of a knowledge source can change any of the plan state; earlier stages merely build up information locally. It is intended to investigate a language for describing Knowledge Sources (Knowledge Source Framework). Amongst other things this may allow for information concerning the selective locking of parts of the database to be gathered.

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 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*).
4. Support Modules - functions and constraint managers which support the processing capabilities of the planner and its components.
5. Controller - the decision maker on the *order* in which processing is done.

3.1 Support Modules

Support Modules may either be Constraint Managers or other types of modules 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 constraints they are managing or to respond to questions being asked of them by he decision making level. The support modules normally act to manage information and constraints in the plan state. Examples of Support Modules in O-Plan include:

- Effect/Condition (TOME/GOST) Manager including Question Answering (QA)

- Resource Utilisation Manager
- Time Point Network Manager
- Object Instantiation (Plan State Variables) Manager
- Alternatives Handler
- Instrumentation
- Monitors for output messages, etc.

A guideline for the provision of a good support module 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 support module should be called by a knowledge source or from a component of the architecture).

3.2 Protocols

In addition, a number of external interface specifications and protocols for inter-module use have been established. Only first versions of these interfaces have been established at present, 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 into the planning framework.

The protocols for regulating the processing conducted by components of O-Plan are:

1. *Knowledge Source Protocol*
   for 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 Database Manager.
2. *KS_USER Protocol*
   for the ways in which the user (in the role of *Planner User*) can assist the planning system via a specially provided knowledge source.
3. *Inter-agent Communications Protocol*
   controls the way in which the KS_EXTRACT_LEFT, KS_EXTRACT_RIGHT and KS_PATCH Knowledge Sources operate and may use the Interface Manager's support routines which control the agent's input and output event channels (LEFTIN, LEFTOUT, RIGHTIN and RIGHTOUT).

3.3 Internal Support Facilities

The internal support provided within the planner to assist a 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 O-Plan 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-Plan provides a limited number of triggers of this kind, but we anticipate this being expanded significantly in future.

3. *Controller Priority Language*
   allows the input of guidance rules for the ordering decisions taken by the O-Plan Controller on which triggered agenda entries to process next. Currently, only simple numerical priorities are used to guide the controller.

The following sections give further details of these facilities.

3.4 Knowledge Source Framework (KSF)

The KSF allows information about a knowledge source to be provided to the O-Plan architecture. A KSF description of a knowledge source is the mechanism by which a new capability is declared to an O-Plan agent.

The KSF gives the following details:

- the name of the knowledge source. This is used by other knowledge sources to indicate that they want to call this capability by posting suitable agenda entries as they run. It is also used for nominating a knowledge source to deal with an event.
- parameters match description. This is used to restrict the legal parameters that may be passed to the knowledge source.
- agenda posting/information field match description. This is used to restrict the legal entries for the posting/information field accepted back by a knowledge source (used for information temporarily kept between stages of a suspended knowledge source). It can also be used to ensure that any modification of this posting information field not done by the knowledge source itself is verified as being acceptable to the knowledge source itself.
- stages information in the form

  - <stage number> <trigger> -> <ks stage function> <locking information>
  - the stage number need only be given if there is more than one stage.
  - The <trigger> description can be composed from the O-Plan Agenda Trigger Language - which itself will evolve over time.
  - the <ks stage function> is the actual procedure that will be called to implement the current knowledge source stage.
  - <locking information> is provided to define whether this stage needs the plan state in READ mode or WRITE mode. If not provided, the default assumption is that the stage is a READ mode stage and that all effects of the stage are created by communication with the controller (normally also saving information in the information field of the agenda record when the knowledge source terminates if asked to do so at the stage end). It is also possible to give information about the specific parts of the plan state that can be READ by or WRITTEN to by this stage to allow for selective locking strategies to be explored in future versions of O-Plan.

- controller priority function. To provide heuristic guidance to the controller based upon the overall information in the agenda record nominating this knowledge source. This will only be applied to triggered agenda entries. It may use Branch 1 and Branch N information [1] in an agenda entry to provide heuristic guidance to the controller.

- plan state poison handler. The function to be called whenever *this* knowledge source terminates with a request to poison the plan state (i.e. when this knowledge source thinks that the plan state is inconsistent and that it cannot recover from the problem itself).

The KSF is used to build a capability library for the agent and to define the event information that may be passed by the guards on the external event channels of the agent. Extensions to the KSF will be needed as further refinement of the agent properties of an O-Plan system are defined.

3.5 Agenda Trigger Language

An agenda entry can be set to await a *trigger*. This trigger can relate to information in the plan state, to external events, etc. The facility can be used by a Knowledge Source writer to allow Knowledge Source processing to be suspended at a *stage* boundary and made to await the trigger condition before resumption. The responsibility for reactivating the computation is taken by the O-Plan system using facilities within the Database Manager.

The trigger can be composed from the O-Plan Trigger Language - which itself will evolve over time. Triggers will be composed to form a boolean function.

Example triggers available within the O-Plan planner are:

- “always triggered”.

- dependencies on the plan state to be selected including:

  wait for a suitable effect matching some specification.

  wait for a fully instantiated binding for a Plan State Variable.

- links to events triggers at a specific time via the O-Plan Diary.
- empty agent agenda.

3.6 Controller Priority Language

Currently, the O-Plan 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 based controller which can make use of priority information provided in the form of rules in an O-Plan 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 Priority 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.

4 Characterising Processing Components in an Open System

This section seeks to abstract the features which we may wish to use to characterise a Knowledge Source or functional processing asset to an Open Systems Integration Architecture. These features could thus form a basis for a language in which to describe plug in processing assets to such a system.

4.1 Task Orientated Characterisation

It may be possible to use a description of the task carried out by the asset in terms of the O-Plan activity ontology (Triangle Model of Activity or <I-N-OVA> model [3]. This would allow it to be described in terms of:

1. the name of the task,
2. a task decomposition to describe the way in which the task is done (including sub-activities),
3. temporal constraints on the task and its sub-activities,
4. world (pre-)conditions and context conditions describing the environment in which this task can be used and is appropriate.
5. resource usage and production,
6. authority requirements and provision,
7. outstanding issues to be addresses in order to render the task enactable or executable.

4.2 System Orientated Characterisation

The O-Plan Knowledge Source Framework (KSF) principally is intended to provide a characterisation of the processing component in terms which allow for it to be used and controlled in a well behaved way within the overall system.

It can be considered to have five features that we wish to capture:

1. Whether the processing component is *monolithic* or whether it can be broken into processing *stages*[1].

[1]It can be advantageous to consider a processing element as having a number of stages rather than being separate processing elements with their own KSF descriptions, since this allows processing state to be saved and reinstated for the stages in a form that would not be allowed by the more strict KSF interaction with the state (plan state). In particular it is common in O-Plan to save inter-stage processing information for a Knowledge Source in an information field in an agenda item for the re-activation of the next stage.

2. The parts of the system state (Plan state in O-Plan) it can *read* from. This can be described per stage for multi-stage processing elements.

3. The parts of the system state (Plan state in O-Plan) it can *write* to or alter. This can be described per stage for multi-stage processing elements.

4. Whether the processing element produces a *single outcome* or whether it can generate multiple *options* that must be taken under management by the overall system. This can be described per stage for multi-stage processing elements.

5. Whether the results provided give complete coverage of all answers possible (*algorithmic*) or whether the results provided are in-complete (and thus represent a *heuristic* selection from the complete set of answers).

4.3 Characteristic Modules

We have found it convenient to group some of the properties within the five characteristics described above in order to define commonly used processing components, as follows:

Knowledge Source can read and write the plan state, and may generate options.

Constraint Manager has limited scope of read and write (to one type of auxiliary constraint under the <I-N-OVA> model) within the plan state and produces complete answers (i.e., is algorithmic).

Information Asset Read only on the plan state[2].

[2]Information Assets are not characterised separately within the current implementation of O-Plan.

5 Summary

This paper has collected together information on the O-Plan Knowledge Source Framework concept. It has sought to place this work in the context of its possible use as the basis for describing functional processing assets to an Open Systems Integration Architecture that allows modules to be plugged in.

References

[1] Currie, K.W. and Tate, A. O-Plan: the Open Planning Architecture, *Artificial Intelligence* Vol 51, No. 1, Autumn 1991, North-Holland. World Wide Web URL `ftp://ftp.aiai.ed.ac.uk/pub/documents/ps/1991/91-aij-oplan.ps.gz`

[2] Fraser, J.L., The Enterprise Project, Artificial Intelligence Applications Institute, University of Edinburgh, 1995. World Wide Web URL `http://www.aiai.ed.ac.uk/ entprise/enterprise/`

[3] 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. World Wide Web URL `ftp://ftp.aiai.ed.ac.uk/pub/documents/ps/1995/95-sigart-inova.ps.gz`

[4] Tate, A., Drabble, B. and Dalton, J., O-Plan Architecture Guide, Version 2.2, Artificial Intelligence Applications Institute, University of Edinburgh, July 1994. World Wide Web URL `ftp://ftp.aiai.ed.ac.uk/pub/documents/ps/ANY/oplan-architecture-guide.ps.gz`

[5] Tate, A., Drabble, B. and Dalton, J., Reasoning with Constraints in O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop at Tucson, (ed. M.Burstein), Morgan-Kaufmann, 1994. World Wide Web URL `ftp://ftp.aiai.ed.ac.uk/pub/documents/ps/1994/94-arpi-constraints.ps.gz`

[6] 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. World Wide Web URL `ftp://ftp.aiai.ed.ac.uk/pub/documents/ps/1994/94-is-oplan2.ps.gz`

Note that papers concerning O-Plan are available on the World Wide Web via the O-Plan Project URL `http://www.aiai.ed.ac.uk/ oplan/`