Knowledge Engineering: Issues for the Planning Community

T. L. McCluskey
School of Computing and Mathematics,
University of Huddersfield, UK
email: lee@zeus.hud.ac.uk

Abstract

Knowledge engineering for AI planning is the process that deals with the acquisition, validation and maintenance of planning domain models, and the selection and optimization of appropriate planning machinery to work on them. Evidence from the growing body of experience in applying planning technology suggests that knowledge engineering issues are crucial to an application's success. The Knowledge Engineering Technical Co-ordination Unit of PLANET[1] has been active for several years now in carrying out workshops and sponsoring cross-site visits on the subject. Here I briefly summarise some of the material in our roadmap document(McCluskey *et al.* 2000), selecting some of the important research questions from it, and introduce the papers that are to be presented in this workshop.

Introduction

Knowledge Engineering (KE) for AI Planning is the process that deals with the acquisition, validation and maintenance of planning domain models, and the selection and optimization of appropriate planning machinery to work on them. Hence, knowledge engineering processes support the planning process - they comprise all of the off-line, knowledge-based aspects of planning that are to do with the application being built.

KE issues have to be engaged by our community if we are to get non-AI Planning people to use our technology. These issues are recognised as a major problem in the application of planning systems. Experience with planners adapted for aerospace and military applications (Wilkins 1999; Tate, Drabble, & Dalton 1996; Muscettola *et al.* 1998) has pointed to KE aspects as being those most in need of attention.

In the field of Knowledge-based Systems, from whence the term 'Knowledge Engineering' originates, the need for modelling knowledge at the conceptual level has long been accepted in the development of KBS methodologies. Specifying components and their interfaces at the knowledge rather than implementational level leads to the kind of abstractions that facilitates interoperability and re-use. The *knowledge-level principle* of Alan Newell (Newell 1982) influenced and directed much KBS work into this direction. Hence the pursuit of KE within planning may be seen as a special case of KE within the general knowledge-based system field. It may prove useful to derive methods and adapt tools from KBS, as the work of Tate et al (Tate, Polyak, & Jarvis 1998) has attempted. There are peculiarities of planning that clearly distinguish engineering planning knowledge from general expert knowledge:

- the ultimate use of the planning domain model is to be part of a system involved in the 'synthetic' task of plan construction. This makes it very specific in the world of KBS, where many successful systems are, in contrast, aimed at solving diagnostic or classification problems.
- the knowledge elicited in planning is largely knowledge about actions and how objects are effected by actions. This knowledge has to be adequate in content (and ultimately in form) to allow efficient automated reasoning and plan construction.

In branches of requirements capture in software engineering knowledge is elicited about processes or actions in the domain of interest, similar to that in Planning. Very expressive and formal languages and development environments have been introduced for this purpose. In software engineering however, the purpose of this capture in very different - it is to help in the analysis and understanding of a system, and to be used in the creation and validation of a new system model.

Despite the peculiarities mentioned above, it seems fruitful to pursue research and developments in KE for planning in the context of related developments in KBS and software requirements engineering. In particular, it is appears inevitable that research in AI Planning must adopt multi-disciplinary approaches to knowledge-based planning.

The Growth of Tool Support

Not too many years ago tools for planning domain acquisition and validation amounted to little more than syntax checkers. 'Debugging' a planning application would naturally be linked to bug finding through dynamic testing. The two pioneering KBS planners O-

[1]The EU-funded Network of Excellence in Planning

Plan and SIPE have of course, by necessity, developed methods and tool support. The O-Plan system has for example its 'Common Process Editor' (Tate, Polyak, & Jarvis 1998) and SIPE has its Act Editor (Myers & Wilkins 1997). These visualisation environments arose because of the obvious need in knowledge intensive applications of planning to assist the engineering process. They are quite specific, however, having been designed to help overcome problems encountered with domain construction in previous applications of these planning systems.

One of the earliest types of tool support for AI Planning grew from research into Machine Learning (ML). From an ML point of view, siting a learning mechanism with a clean, classical planner led to an attractive way to evaluate the automated learning algorithms. Hence ML tools for planning have received considerable attention over the last 20 years. For example, tools have been built to induce operator descriptions from traces of actions, and to acquire or tune heuristics in the form of macro-operators, state evaluation functions and goal orders.

More recently interest has grown in the area of *domain analysers*. These tools process a domain model with the goal of making explicit useful information, and they may be embedded in an online planner or be standalone. In the latter case, they can function as part of a modelling environment, helping a user to perform *static validation* on an acquired model. For example, tools may: check that a planning operator is consistent (e.g. it never inputs a valid state and outputs an invalid state); reason with operators and output state invariants to be visually checked by a user; or output necessary goal orderings, to check for impossible goal combinations. Also, domain analysis tools can help in the acquisition of heuristics that customise a general planning engine to an application, or more importantly to identify the kind of planner appropriate for solve problems within the application domain.

A further step is to produce *tools environments* for acquiring, modelling and prototyping planning applications in such a way that the tools are integrated and the environment is *open*. An open environment means that users can attach their own tools which integrate with the other tools. Research into and the development of such an environment for classical AI planning was one of the goals of the PLANFORM project (Planform 1999). GIPO, an outcome of this project, is a GUI designed to integrate tools in support of knowledge engineering for AI planning. GIPO is purely a 'laboratory' system aimed as a testbed for knowledge acquisition techniques and planning tool integration. Users can attach their planners to GIPO via a PDDL (AIPS-98 Planning Competition Committee 1998) interface, but other more knowledge-rich interfaces are yet to be developed. Truly open environments are essential for the development of planning technology, but this requires a level of standardisation not yet present in the community.

Representation Languages and Standardisation

Both to help the Planning field mature, and to help engineers apply and integrate the technology, representation language conventions should be sought. This has been achieved in a very limited way with PDDL, a community accepted standard for communicating minimal dynamical models of a domain. PDDL has been a lowest common denominator for planning systems running under many of the classical STRIPS-assumptions, allowing sets of domain models to be distributed and certain classes of planning engines to be compared in competitions. There is a need to exchange and to some degree standardise much more than bare domain dynamics - for example, domain structures and heuristics. Specifically, it would be useful to share and exchange generic object structures that could be re-used over a range of applications. More generally, creating a planning knowledge base *without* re-use seems at best inefficient. There is some work emerging on planning ontologies (for example see (Gill & Blythe 2000)) but there is still a long way to go.

When considering standard representations one must consider the function and content of the representation itself. For example, in contrast to domain specification languages, there have been attempts at creating standard languages for *plan specifications* - notably the work surrounding the creation of SPAR (Tate 1998). Another class of representation language, which concerns knowledge engineers in particular, is one in which languages are specifically designed with pragmatic features that help the process of domain acquisition and modelling. Our roadmap (McCluskey *et al.* 2000) postulates criteria for such languages, asserting that they should be well structured, tool supported, expressive, customizable, well founded, and finally, embedded within a modelling method. Languages that have been developed from the point of view of knowledge acquisition, and fulfill some of these criteria, include DDL.1 (Cesta & Oddi 1996), Act (Myers & Wilkins 1997), TF (Tate, Polyak, & Jarvis 1998) and OCL_h (McCluskey & Kitchin 1998).

A Note about Terminology

Research papers that concern KE in planning often appears to use inconsistent, confusing or imprecise terminology. I will choose a simple but pervasive example to illustrate the point. In this volume of papers phrases such as 'domain description', 'domain specification', 'domain definition', 'domain model', 'domain theory' and simply 'domain' are used. Firstly, a distinction: let the *domain* denote the reality being modelled within the corresponding planning system. Let any form of symbols representing parts of the domain be called a *domain description* – for example a document containing natural language describing the domain. Domain description is the most vague term of the set.

The term *domain specification* is less vague than a

description. It implies something that is finished, and something that can be reasoned with. In other words, we expect a domain specification to be complete and precise, and often formal in the sense that valid inferences can be made using it, about the domain itself. Of course most specifications fail in these aspects!

The term *domain model* implies that we have a representation that can be used to perform operations in the same manner that occur in the domain; and that there is a well-known operational semantics for constructs in the model. Further, the term 'model' implies that named objects within it correspond directly to named objects in the domain, and there is an obligation on the developer to check that the model accurately predicts changes in the domain. I would argue that the traditional operator-based domain descriptions fall exactly into this last case - they are domain models. In software engineering there is a divide between implicit or property-based formal specifications on the one hand, and executable formal specifications on the other. While the former might state *properties* of the domain, it may or may not contain operational details. Hence, a domain *model* is rather like the idea of an executable specification in software engineering.

In summary, we call the outside reality the Domain; the Domain Description is any set of documents about the Domain, possibly in natural language; the Domain Specification is an abstract, formalised account of the Domain; and the Domain Model is a Domain Specification which is in an operational form, containing explicit details of domain dynamics, and suitable for processing by a planning engine.

Issues for the Planning Community

Planet's KE Roadmap (McCluskey *et al.* 2000) lists a number of concerns and research challenges for the future in the area. Here I list several but expect that many more will arise as a result of the workshop.

evaluation of knowledge engineering methods: How do we evaluate knowledge engineering methods, tools and techniques? Case studies and controlled experiments are very expensive compared to evaluation of a planner against a set of benchmarks. Is the introduction of *challenges* or *competitions* feasible or desirable to promote this area?

improved representation languages: Pragmatic aspects of programming languages (objects, types, modules) are very well developed to help one to program. On the other hand, it can be argued that PDDL is at the level of a 'machine code' for domain description. What kind of standard, pragmatic structures are needed in domain modelling languages?

further standardisation: There are many reasons why standardisation can help advance a field - one in particular is to help us develop components of a planning system flexibly. Should we be developing languages for standardising the exchange of heuristics, and other planning - related knowledge? Given the potential for applying planning technology through the internet, should we not be developing web-friendly 'semantic' mark-up languages for this purpose?

ontologies: Given the emphasis on Ontologies in Knowledge-based Systems, should we be developing Planning Ontologies, and if so, in what form? The availability of libraries of components from which to assemble planning knowledge bases and planning systems seems very desirable (Gill & Blythe 2000), but how do we go about funding and evaluating this work? Hertzberg in the last section of reference (Hertzberg 1996) declares that there is a lack of a 'vocabulary for describing the characteristics of domains, plans ...'. In the context of Knowledge Engineering, the pursuit of such a classification system and/or vocabulary is still on the to-do list, and well worthy of action.

The Workshop Papers

Aylett and Doniat tackle the very difficult area of knowledge acquisition for planning, using an approach inspired by the KBS community. Their focus is on helping a domain expert rather than a planning expert perform such a task. The aim of Simpson et al's work is also knowledge acquisition, but at a more detailed specific level where libraries of generic types could be used to aid the acquisition of new domains. Murray's paper too concerns generic types, but rather than for domain structure, he attempts to use them as abstract control rules that could form a generic control rule library.

The papers by Cresswell et al and Varrentrapp et al both concern support tools. The first deals with a much needed extension to an existing domain analyser, while the second postulates an open environment specifically for evaluating planners using dynamic testing.

The work of Fernandez et al falls into the category of using learning techniques to tune planning heuristics. Of note is their use of a Neural Network as the learning technique, resulting in interesting coding issues centering on the representation of states and goals as inputs to such a network.

Bartak's paper is ambitious in that it proposes the creation of a modelling framework which spans both planning and scheduling, and which regards resources and activities with durations as fundamental. Influenced by scheduling applications, Bartak's work provides a good counterpoint to emerging modelling platforms aimed at AI planning.

Finally, Jarvis's paper calls for a change in AI Planning's research direction away from the easily evaluated 'stand-alone' knowledge sparse planner (of the AIPS competition variety), to the more embedded, mixed-initiative expressive kind. He introduces the idea of 'computer aided planning' rather than 'computer replaced planning' and argues most convincingly that this is both a more feasible and useful direction for main-

stream planning research. Perhaps the AI Planning Community will split up along these lines, with a gap emerging between AI scientists, interested in planning capabilities per se, and AI Planning engineers, interested in exploiting the technology. These issues will no doubt be discussed at the Panel session!

References

AIPS-98 Planning Competition Committee. 1998. PDDL - The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Cesta, A., and Oddi, A. 1996. DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains. In Ghallab, M., and Milani, A., eds., *New Directions in AI Planning*. IOS Press. 341–352.

Gill, Y., and Blythe, J. 2000. PLANET: a shareable and reusable ontology for representing plans. In *Proceedings 17th International Conference on AI*.

Hertzberg, J. 1996. On Building a Planning Tool Box. In Ghallab, M., and Milani, A., eds., *New Directions in AI Planning*. IOS Press. 3–18.

McCluskey, T. L., and Kitchin, D. E. 1998. A Tool-Supported Approach to Engineering HTN Planning Models. In *Proceedings of 10th IEEE International Conference on Tools with Artificial Intelligence*.

McCluskey, T. L.; Aler, R.; Borrajo, D.; Haslum, P.; Jarvis, P.; and Scholz, U. 2000. Knowledge Engineering for Planning ROADMAP. *http://scom.hud.ac.uk/planet/*.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. C. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence* 103(1-2):5–48.

Myers, K., and Wilkins, D. 1997. The Act-Editor User's Guide: A Manual for Version2.2. SRI International, Artificial Intelligence Center.

Newell, A. 1982. The Knowledge Level. *Artificial Intelligence* 18:87 – 127.

Planform. 1999. An open environment for building planners. http://scom.hud.ac.uk/planform.

Tate, A.; Drabble, B.; and Dalton, J. 1996. O-Plan: a Knowledged-Based Planner and its Application to Logistics. AIAI, University of Edinburgh.

Tate, A.; Polyak, S. T.; and Jarvis, P. 1998. TF Method: An Initial Framework for Modelling and Analysing Planning Domains. Technical report, University of Edinburgh.

Tate, A. 1998. Roots of spar - shared planning and activity representation. *Knowledge Engieering Review* 13:121 – 128.

Wilkins, D. 1999. Using the SIPE-2 Planning System: A Manual for SIPE-2, Version5.0. SRI International, Artificial Intelligence Center.