Knowledge Engineering Tools in Planning: State-of-the-art and Future Challenges

M.M.S. Shah and **L. Chrpa** and **F. Jimoh** and **D. Kitchin**

T.L. McCluskey and **S. Parkinson** and **M. Vallati**
School of Computing and Engineering
University of Huddersfield
United Kingdom

Abstract

Encoding a planning domain model is a complex task in realistic applications. It includes the analysis of planning application requirements, formulating a model that describes the domain, and testing it with suitable planning engines. In this paper we introduce a variety of new planning domains, and we then use and evaluate three separate strategies for knowledge formulation, encoding domain models from a textual, structural description of requirements using (i) the traditional method of a PDDL expert and text editor (ii) a leading planning GUI with built in UML modelling tools (iii) a hierarchical, object-based notation inspired by formal methods.

We distill lessons learned from these experiences. The results of the comparison give insights into strengths and weaknesses of the considered approaches, and point to needs in the design of future tools supporting PDDL-inspired development.

Introduction

Knowledge Engineering for automated planning is the process that deals with acquisition, formulation, validation and maintenance of planning knowledge, where a key product is the *domain model*. The field has advanced steadily in recent years, helped by a series of international competitions[1], the build up of experience from planning applications, along with well developed support environments. It is generally accepted that effective tool support is required to build domain models and bind them with planning engines into applications. There have been reviews of such knowledge engineering tools and techniques for AI Planning (Vaquero, Silva, and Beck 2011). While these surveys are illuminating, they tend not to be founded on practice-based evaluation, in part, no doubt, because of the difficulty in setting up evaluations of methods themselves. Given a new planning domain, there is little published research to inform engineers on which method and tools to use in order to effectively *engineer a planning domain model*. This is of growing importance, as domain independent planning engines are now being used in a wide range of applications, with the consequence that operational problem encodings and domain models have to be developed in a standard language such as PDDL (Ghallab et al. 1998).

[1] for the most recent see http://icaps12.poli.usp.br/icaps12/ickeps

In this paper we explore the deployment of automated planning to assist a variety of real world applications: machine tool calibration, road traffic accident management, and urban traffic control. In introducing these new planning domains, we take the opportunity to employ and hence evaluate three separate methods for knowledge formulation (i) the traditional method of hand-coding by a PDDL expert, using a text editor and relying on dynamic testing for debugging (ii) itSIMPLE (Vaquero et al. 2007), an award-winning GUI, utilising a method and tool support based on UML (iii) a rigorous method utilising a hierarchical, object-based notation OCL_h, with the help of tool support from GIPO (Simpson, Kitchin, and McCluskey 2007). Evaluating these three approaches gives a range of interesting insights into their strengths and weaknesses for encoding new domains, and point to needs in the design of future tools supporting PDDL-inspired development. Evaluation measures used are based on several criteria that describes the quality of the engineering process and the quality of the product.

This paper is organized as follows. We first provide an overview of existing KE tools for supporting the task of encoding planning domain models. Next we introduce the real-world domains that have been considered in this analysis. Then we introduce the features that are used for comparing the different encoding methods. Finally, we summarize the lessons learned and we provide some guidelines for future tools.

Overview of existing KE tools

In this section we will provide an overview of KE tools that can be used for producing planning domain models. Tools are listed in alphabetical order.

EUROPA

The Extensible Universal Remote Operations Planning Architecture (EUROPA) (Barreiro et al. 2012), is an integrated platform for AI planning & scheduling, constraint programming and optimisation. The main goal of this platform is to deal with complex real-world problem. It is able to handle two representation languages, NDDL and ANML (Smith, Frank, and Cushing 2008). The latter has been used in various missions by NASA. EUROPA provides modelling support, result visualisation and an interactive planning process.

GIPO
The Graphical Interface for Planning with Objects (GIPO) (McCluskey and Simpson 2006; Simpson, Kitchin, and McCluskey 2007) is based on its own object-centred languages *OCL* and OCL_h. These formal languages exploit the idea that the universe of potential states of objects are defined first, before operator definition (McCluskey and Kitchin 1998). GIPO is centered on the precise definition of a planning state as an amalgam of object's individual states. This gives the concept of a *world state* as one being made up of a set of states of objects, satisfying certain types of constraints. Operator schemas are constrained to be consistent with respect to the state, giving the opportunity for using tools to do consistency checking. GIPO uses a number of consistency check, like if the object's class hierarchy is consistent, object state descriptions satisfy invariants, predicate structures and operator schema are mutually consistent and task specifications are consistent with the domain model. Such consistency checking guarantees that several classes of errors are prevented, in contrast to ad hoc methods such as hand crafting.

itSIMPLE
itSIMPLE (Vaquero et al. 2007; 2012) provides an environment that enables knowledge engineers to model a planning domain using the Unified Modelling Language (UML) standard (OMG 2005). itSIMPLE focuses on the initial phases of a disciplined design cycle, facilitating the transition of requirements to formal specifications. Requirements are gathered and modeled using UML to specify, visualize, modify, construct and document domains in an object-oriented approach. A second representation is automatically generated from the UML model, and it is used to analyze dynamic aspects of the requirements such as deadlocks and invariants. Finally, a third representation in PDDL is generated in order to input the planning domain model and instance into an automated planner.

JABBAH
JABBAH (González-Ferrer, Fernández-Olivares, and Castillo 2009) is an integrated domain-dependent tool that aims to develop process transformation to be represented in a corresponding HTN planning domain model. The system mainly deals with business processes and workflows. The processes are represented as Gantt charts or by using an open source workflow engine. The tool provides support for transforming Business Process Management Notation (BPMN) (graphical notation) to HTN-PDDL. Such HTN-PDDL (Castillo et al. 2006) domain model is used in HTN planners to obtain a solution task network.

MARIO
Mashup Automation with Runtime Invocation and Orchestration (MARIO) (Bouillet et al. 2009; Feblowitz et al. 2012) is an integrated framework for composing workflow for multiple platforms, such as Web Services and Enterprise Service Bus. This tool provides a tag-based knowledge representation language for composition of planning problems and goals. It also provides a web-based GUI for AI planning system so that the user can provide software composition goals, views and generated flow with parameter to deploy them into other platform.

PDDL Studio
PDDL Studio (Plch et al. 2012) is a recent PDDL editor that allows the user to write and edit PDDL domain and problem files. The main goal of the tool is to provide knowledge engineers the functionality to edit and inspect PDDL code, regardless of how the PDDL code was created. The tool supports the user by identifying syntactic errors, highlighting PDDL components and integrating planners. PDDL Studio does not require the user to draw any diagram, it is more like writing traditional programming language code by using an Integrated Development Environment (IDE). The current version of this tool can help editing basic PDDL and also provides error checking.

VIZ
VIZ (Vodrážka and Chrpa 2010), is a knowledge engineering tool inspired by GIPO and itSIMPLE. It shares many characteristics of those systems (GIPO and itSIMPLE) with the addition of a simple, user friendly GUI by allowing inexperienced knowledge engineers to produce PDDL domain models. This tool uses an intuitive design process that makes use of transparent diagrams to produce a PDDL domain model. The tool does not support any third party planner integration. However, the tool is still being developed.

Considered applications

We considered three real-world domains that have been encoded in planning domain models. Namely, the machine tool calibration (Parkinson et al. 2012), the road traffic accident management (Shah, McCluskey, and Chrpa 2012), and the urban traffic control (Jimoh et al. 2012)

Machine tool calibration
Engineering companies working with machine tools will often be required to calibrate those machines to international standards. The requirement to manufacture more accurate parts and minimise manufacturing waste is resulting in the continuing requirement for machine tools which are more accurate. To determine a machine's accuracy, frequent calibration is required. During calibration, the machine will not be available for normal manufacturing use. Therefore, reducing the time taken to perform a calibration is fundamental to many engineering companies.

The calibration process requires various errors in the machine to be measured by a skilled expert. In addition to conducting the tests, the engineer must also plan the order in which the tests should take place, and also which instruments should be used to perform each test. It is critical to find the optimal sequence of measurements so that the machine is not out of service for too long.

An example PDDL2.2 (Edelkamp and Hoffman 2004) operator taken from the machine tool calibration domain model can be seen in Figure 1. This operator can be considered as

```
(:durative-action setup
:parameters
(?er - error ?ax - axis ?in - instrument)
:duration (= ?duration (setup-time ?in ?ax))
:condition
(and (over all (not (blocked ?in ?ax)))
    (over all (axis-error ?ax ?er))
    (over all (measures   ?in ?er))
    (over all
        (forall (?a - axis ?i - instrument)
           (imply (setup ?i ?a) (= ?a ?ax))))
    (over all (forall (?i - instrument)
           (imply (operating ?i)
            (compatible ?i ?in))))
    (at start (<= (using ?in) 0))
    (at start (>= (using ?in) 0))
    (at start (not (measured ?ax ?er)))
    (at start (not (operating ?in)))
    (over all (>= (working-range ?in)
        (travel-length ?ax)))
    (over all (working-day))
  )
 :effect
   (and
(at end   (setup      ?in ?ax))
(at end   (setup-for  ?in ?er))
(at end   (operating  ?in))
(at start (increase (using ?in) 1))
   )
)
```

Figure 1: A sample planning operator from the machine tool calibration domain model, encoded in PDDL 2.2.

one of the most complex that we have dealt with in this work. It includes quantification, timed-initial literal and ADL features of PDDL.

Figure 2 illustrates an excerpt taken from a valid, optimal calibration plan produced from using the PDDL2.2 domain and LPG-td planner (Gerevini, Saetti, and Serina 2006). From the excerpt it can be seen that the planner has scheduled measurements that use the same equipment together, and that where possible, measurements are taken concurrently. This plan allows the calibration process to be completed as quickly as possible, minimizing the down-time of the machine.

Road traffic accident management

Accidents cause traffic congestion, injury, increase environment pollution, and cost millions of pounds every year because of delay and damage. This has lead to highway agencies needing more appropriate solutions to manage accidents. Accidents are a particular type of road traffic incident which can be defined as irregular or unplanned events that reduce road capacity, increase congestion and travel time. Incidents increase traveler delay which may lead to more serious problems such as further accidents (Owens et al. 2000). The consequence of this problem is often severe since accidents limit the operation of the road networks and put all road users at risk. The main

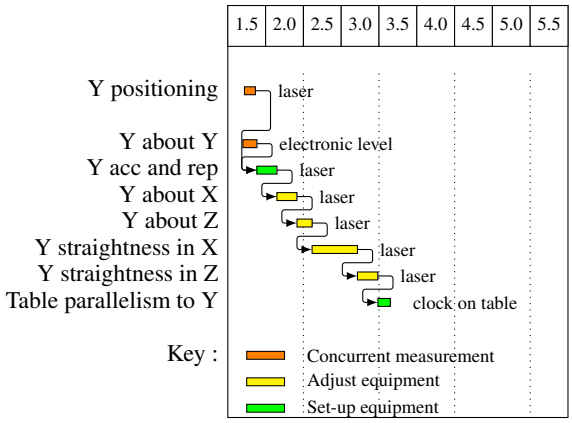

Figure 2: Excerpt from a produced calibration plan from the machine tool calibration application

responsibility for managing and dealing with the incident lies with the highway agencies which are serving on that area. It is a top priority for highway agencies around the world to manage accidents more effectively, efficiently and as fast as possible to save time, money and most crucially life. Utilising automated planning capabilities in real applications is a current topic with great potential to help in speed, accuracy, and co-ordination of tasks to be carried out.

Urban Traffic Control

Traffic in urban areas (e.g. town centers) tends to be dense, especially in rush hours, which often leads to traffic jams which can significantly increase travel time. Therefore, a need for efficient Urban Traffic Control in such exposed areas is becoming more important. It is necessary to minimize travel time by efficiently navigating road vehicles throughout the road network, while avoiding road congestion and diverting traffic when a road is blocked. Traditional Urban Traffic Control methods are based on reactive acting, they operate, for instance, on the basis of adaptive green phases and flexible co-ordination in road (sub)networks based on measured traffic conditions (Dusparic and Cahill 2009; sheng Yang et al. 2005; Salkham et al. 2008; Daneshfar et al. 2009; Bazzan 2005). However, these approaches are still not very efficient during unforeseen situations such as road incidents, when changes in traffic are requested in a short time interval (Dusparic and Cahill 2012). The role of AI planning in this case is to come with deliberative reasoning. In contrary to traditional reactive control we can reason about the road network globally that is useful while dealing with unexpected situations. Consequently, by exploiting AI planning in traffic control, we can reduce cost and pollution which is often a serious issue in town/city centers.

Criteria for evaluating approaches

We identified several criteria that are useful for evaluating the considered approaches for encoding domain models.

Operationality. How efficient are the models produced? Is

```
(:durative-action DRIVE
   :parameters (?r - road ?n - num)
   :duration (= ?duration (length ?r))
   :condition
    (and
      (at start (>= (head ?r) (val ?n)))
      (over all (operational ?r))
    )
   :effect
    (and
     (at start (decrease (head ?r) (val ?n)))
     (at end (increase (tail ?r) (val ?n)))
    )
)
```

Figure 3: A sample planning operator from the Urban Traffic Control application encoded in PDDL 2.1.

the method able to improve the performances of planners on generated models and problems?

Collaboration. Does the method/tool help in team efforts? Is the method/tool suitable for being exploited in teams or is it focused on supporting the work of a single user?

Maintenance. How easy is it to come back and change a model? Is there any type of documentation that is automatically generated? Does the tool induce users to produce documentation?

Experience. Is the method/tool indicated for inexperienced planning users? Do users need to have a good knowledge of PDDL? Is it able to support users and to hide low level details?

Efficiency. How quickly are acceptable models produced?

Debugging. Does the method/tool support debugging? Does it cut down the time needed to debug? Is there any mechanism for promoting the overall quality of the model?

Support. Are there manuals available for using the method/tools? Is it easy to receive support? Is there an active community using the tool?

Evaluation of the approaches with respect to stated criteria

In this paper we employ and hence evaluate three separate methods for knowledge formulation: (A) the traditional method of hand-coding by a PDDL expert, using a text editor and relying on dynamic testing for debugging; (B) using state-of-the-art KE tool itSIMPLE; (C) using transparency and consistency checkers in GIPO III.

In the following we will evaluate each method with respect to the criteria stated in the previous Section.

Method A

This method involves a PDDL expert that uses a text editor (in this paper, Gedit) for generating a planning domain model, given the description of the real world domain. For illustration, a handcoded planning operator is depicted in Figure 3.

Operationality. Even if this is the most exploited method for generating new planning domain models, there is no evidence that it leads to models that are more efficient than those generated by other methods. The quality of models depends on the expertise of the person that encodes it, which is very hard to predict a-priori. In fact, we have experimentally observed that often, the models generated by this method reduce the performances of planning algorithms.

Collaboration. This method does not support any type of collaboration. Usually the model is produced by a single expert, that eventually discusses issues or improvements with domain experts rather than with other planning experts.

Maintenance. It is usually easy, for the expert that encoded the domain model, to come back and maintain or modify it. On the other hand, models are usually not documented. This means that the maintenance is potentially hard, with regard to the complexity of the model, for people that were not involved in the encoding process.

Experience. This method is applicable only for PDDL experts. PDDL experts know the ways for handling some common issues and are able to interpret the planners output in order to identify bugs.

Efficiency. Usually, the first version of the model is quickly produced. This leads users to perceive this method as a very efficient one. On the other hand, the first version requires a lot of dynamic tests and improvements to become acceptable.

Debugging. Debugging while hand-coding a model is a critical task. The only way for debugging is dynamic testing. This involves the use of one (or more) planners for solving some toy instances. The produced plans are then analysed for identifying bugs that can be fixed by modifying the model. This cycle is repeated until no bugs are found. Omitting some important constraints is often a source of bugs for this encoding method.

Support. Since this is a traditional method, there are many guides available online for generating new domain models. However, these guides are usually technically written and are very difficult to follow for non-experts of automated planning.

Method B

This method involves a user that exploits itSIMPLE for generating new planning domain models. The steps of the method follow the use of UML in software engineering: (i) design of class diagrams; (ii) definition of state machines; (iii) translation to PDDL; (iv) generation of problem files.

Operationality. From our experience, it seems that domain models generated by itSIMPLE can often improve the performances of planners. This is probably due to a domain description that is less constrained than the one developed by exploiting method A. The quality of the models generated does not depend on the expertise of the users; itSIMPLE guides users in the design process.

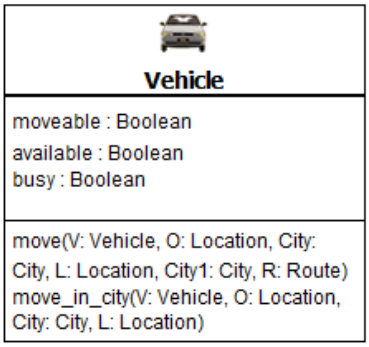

Figure 4: An example of a Class Diagram designed in itSIMPLE.

```
...
Checking method carry_direct(P,O,D)
found an unrecognised decomposition item: unload_subject(P,D,V)'.
Check failed
Checking method carry_direct(P,O,D)
found an unrecognised decomposition item: unload_subject(P,D,V)'.
Check failed
Checking method transport(Subject,Org,Dest)
The static predicate in_region(Org,Region) has no prototype
The static predicate in_region(Dest,Region) has no prototype
Check failed
Doing task checks.
....
```

Figure 5: Part of output from GIPO: here the transparency of HTN methods is checked and found to fail, with the likely faulty components identified.

Collaboration. itSIMPLE has not been designed for team work. This means that usually, the model is developed by a single user. However, it is possible to import models (projects) developed by different users. This helps to exchange ideas and comments among users. Moreover, the UML diagrams generated are useful for sharing and discussing issues with experts.

Maintenance. The itSIMPLE tool is designed for supporting a disciplined design cycle. The UML diagrams can also be used as documentation. From this point of view it is easy to maintain a generated domain model also for people that were not involved in the design process. However, if a model generated by itSIMPLE is modified using a different tool (or even a text editor), then it is not possible to import it back to itSIMPLE.

Experience. The typical itSIMPLE user does not have to be a PDDL expert. However, he should have some basic experience in software engineering and especially in UML.

Efficiency. Most of the time is spent in designing classes of objects and defining legal interactions between them in UML. After that, only a short time is required for debugging. This method is usually slower than method A, but faster than method C, to generate a first version model.

Debugging. Even if itSIMPLE provides dynamic analysis by simulation of Petri Nets created from UML models, most debugging initiates through dynamic testing done by running planners on some toy instances. While the UML description of the models helps in development and maintenance, and "designs out" some sources of error, it is the failure of a planning engine to solve a given problem that, in most cases, alerts the user to bug presence.

Support. itSIMPLE provides a complete documentation which includes a description of the tool, a tutorial and an online FAQ section. It is easy to find information and solutions for most of the common issues.

Method C

We focus on domain models encoded in OCL_h with the help of tools in GIPO-III, using basic consistency checkers as well as the more complex transparency property checker. Hierarchies of classes are used to capture state: for example, in the Road Traffic Accident domain, in a particular state an ambulance may have a position, be in service and available. The set of constraints are added using GIPO-III to encode the behaviour of each of the dynamic object classes, that is, the range of states that each object can occupy.

Operationality. The size of the OCL_h model is larger than the size of the PDDL models, because state constraints are encoded explicitly, and HTN methods are specified in addition to primitive ones. The structure of the solutions is similar to the structure of solutions generated by the LPG planner on the PDDL model developed in itSIMPLE. However, given the different nature of OCL_h, it seems to be impossible to provide a rigorous comparison with PDDL.

Collaboration. GIPO has been designed for a single user. However, it is possible to import models (projects) developed by different users. This helps the exchange of ideas and comments among users.

Maintenance. Domain models generated by GIPO are generally easier to maintain than the hand-encoded ones, due to the consistency checking opportunities during each stage of the modelling process.

Experience. The typical user is not required to be an OCL expert, but he should have some basic knowledge of the language or the tool.

Efficiency. GIPO does not require creating UML diagram like itSIMPLE, but it requires to explicitly encode the constraints of the domain as transition diagrams, that are then used to create operators. This method is usually slower then the others, but the first model generated is very close to the final one.

Debugging. The creation of a dynamic hierarchy of object classes encoded constraints of the domain explicitly, and this is what is used by GIPO's tool support to check operator schema, states, predicates, etc., and identify bugs prior to dynamic testing. Part of the output of the consistency checking tools is shown in Figure 5.

Support. GIPO comes with complete documentation which includes a user manual, tutorial and OCL manual. It is easy to find information and solutions for most of the common issues.

Summary of lessons learned

We can now summarize the lessons that we learned by using the three outlined methods for formulating requirements into domain models.

We observed that creating different models does not take very different amounts of time (taking into account the developers expertise) while exploiting method A and B. Method C usually requires significantly more time resource than the others, because in addition the users have to provide object hierarchies and invariants (for consistency check), and also users have to encode HTN methods. As mentioned before, in method C creation of a dynamic hierarchy of object classes encoded constraints of the domain explicitly, and this is what is used by GIPO's tool support to check operator schema, states, predicates, etc., and identify bugs prior to dynamic testing. While one could argue that dynamic testing will pick bugs up anyway, there may be behaviours that have not been picked up in the tests done in method A and B, that result from hidden bugs. On the other hand the UML description of the domain, that is required by itSIMPLE, helps to prevent many unwanted behaviours. Method A is the most sensitive to bugs, and the quality of the produced model completely depend on the expertise of the user.

Considering the maintenance of the generated models, method B provides the better instruments for changing a model. The UML description provides a sort of documentation that can be exploited for quickly understanding the domain and for applying changes. An issue that we noticed while working with itSIMPLE is that it is not possible to import a model that, even if originally generated by using itSIMPLE, has been slightly modified with a different tool. This force users to make several steps in the framework also for very small changes. In method C, the complex representation of domain models makes it more difficult for a non-expert user to come back and maintain the model.

Regarding the generated models, there are several interesting aspects to consider. Models generated by method A are usually very compact in terms of numbers of lines, predicates and types, but they are usually over-constrained in order to avoid unwanted behaviours. The iterative process of analyzing produced plans, identifying bugs and removing them from the model leads to incrementally add constraints in the form of pre- and/or post- conditions. The structured and principled process of encoding the requirements of Method B usually leads to domain encodings that are clear and easy to understand, even if less compact (around 10% longer) then the ones generated by method A. It is worth mentioning that the good quality of the encoded domains leads to good quality generated plans. The quality of models generated by method C is harder to understand due to the different language used. The models are significantly larger than PDDL ones (about two times longer in terms of number of lines) and need a HTN planner to solve corresponding problems. We observed that the structures of the solutions are similar to solutions generated by domain-independent planners on models developed by other methods. We believe that a possible advantage of this approach might be the better scalability than the PDDL models. However, current OCL_h techniques do not support durative actions, which makes it less interesting in domains where time optimization is critical.

Needs in the design of future tools

The comparison of the three methods for encoding different domain models was fruitful. It gave us the opportunity for understanding the strengths and weaknesses of them, and to highlight the needs that future tools should meet.

Expertise

A main issue of current KE approaches for encoding domain models is that they require a specific expertise. Method A (and some approaches based on existing KE tools such as PDDL Studio) requires a PDDL expert, Method B requires some expertise in UML language, which is common knowledge mainly in software engineering. Finally, method C requires some expertise in the OCL_h language, which is not a widely known language in the AI Planning community. This requirement might significantly reduce the number of potential users of the KE tools. Since users with different research background usually do not have the required expertise, they are not able to exploit existing approaches for encoding domain models. They require an expert that, due to his limited knowledge of the real world domain, will introduce some noise in the encoding. Moreover, given the hardness of generating domain models for planning, many users are not exploiting automated planning but use easier approaches, even if they are less efficient. It is also worth considering that KE tools for encoding domain models are, usually, not very well known outside the planning community. This, again, reduces the number of potential users that could exploit them.

Team work

Current KE tools are designed for a single user. This is usually fine; actually, the generated domain models are encoding easy domains or significantly simplified versions of complex domains. On the other hand, the number of efficient KE tools is growing, especially in the last few years. Hopefully, in coming years, we will be able to encode very accurate models of complex real-world domains. In this scenario, it seems reasonable that many experts will have to cooperate for generating a domain model. From this perspective it is straightforward to consider the need of tools explicitly designed for team work as a critical requirement for future KE tools.

Maintenance

Users are not supported by existing KE tools in writing documentation related to the generated model. As a result, users are usually not writing any sort of documentation. Given this, it is often quite hard to change an existing domain model a few months after its generation. Providing support for writing documentation would make changes

easier and would also help the users while encoding the model. The process of describing what has been done is a first test for the model. Furthermore, some tools are not able to handle domain models that have been changed manually, or by using a different tool. This limits the support that such tools could give to the life cycle of domain models.

Debugging
We noticed that the checking tools provided by GIPO are very helpful for minimizing the time spent on dynamic debugging. Moreover, exploiting the automatic debugging is a strategy for reducing the number of bugs that remain in the domain model, since many problems are usually not easy to find by dynamic debugging. A significant improvement in the techniques for automatic debugging of static/dynamic constraints will lead to significantly better encoded domain models.

Language support
Finally, existing KE tools for generating domain models for planning have a very limited support of the features of PDDL language. Most of them are supporting only PDDL, while a few of them are also able to handle some structures of PDDL2.1 (Fox and Long 2001). It is noticeable that the latest versions of PDDL have some features (e.g. durative actions, actions costs, ...) that are fundamental for a correct encoding of real world domains. Furthermore, none of the existing tools support PDDL+ (Howey, Long, and Fox 2004). PDDL+ provides features for dealing with continuous planning, which is needed in systems working in real-time and that must be able to react to unexpected events. This is the case for the machine tool calibration domain that we considered in this paper. In addition, during the implementation of this domain it was noticed that it can be difficult to model and debug multiple, interacting equations in PDDL. The only way of currently evaluating the implementation is by using VAL with the domain, problem and solution. A KE tool with support for PDDL+ with strong numeric domains would be highly beneficial.

Conclusions

In this paper we have presented the state-of-the-art of Knowledge Engineering tools for encoding planning domain models. We introduced three real world domains that we have encoded: the machine tool calibration, the road traffic accident management and the urban traffic control. We used and evaluated three different strategies for knowledge formulation, encoding domain models from a textual, structural description of requirements using: (i) the traditional method of a PDDL expert and text editor (ii) a leading planning GUI with built in UML modelling tools (iii) a hierarchical, object-based notation inspired by formal methods. We evaluate these methods using a set of criteria, i.e., operationality, collaboration, maintenance, experience, efficiency, debugging and support. We observed that creating different models does not take very different amounts of time. We highlighted weaknesses of existing methods and tools and we discussed the needed in the design of future tools support for PDDL-inspired development.

Future work will involve a simulation framework for evaluating plan execution, where we can couple model design and plan generation more tightly. We are also interested in improving the KE tools comparison by considering also other existing tools and a larger set of features to compare, such as quality of the solutions found and runtimes of different planners on generated domain models.

Acknowledgements

The research was funded by the UK EPSRC Autonomous and Intelligent Systems Programme (grant no. EP/J011991/1).

References

Barreiro, J.; Boyce, M.; Do, M.; Jeremy Frank, M. I.; Kichkayloz, T.; Morrisy, P.; Ong, J.; Remolina, E.; Smith, T.; and Smithy, D. 2012. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12) – The 4th International Competition on Knowledge Engineering for Planning and Scheduling*.

Bazzan, A. L. 2005. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems* 10(1):131–164.

Bouillet, E.; Feblowitz, M.; Feng, H.; Ranganathan, A.; Riabov, A.; Udrea, O.; and Liu, Z. 2009. Mario: middleware for assembly and deployment of multi-platform flow-based applications. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09, 26:1–26:7. New York, NY, USA: Springer-Verlag New York, Inc.

Castillo, L.; Fdez-olivares, J.; scar Garca-prez; and Palao, F. 2006. Efficiently handling temporal knowledge in an htn planner. In *In Sixteenth International Conference on Automated Planning and Scheduling, ICAPS*, 63–72. AAAI.

Daneshfar, F.; RavanJamjah, J.; Mansoori, F.; Bevrani, H.; and Azami, B. Z. 2009. Adaptive fuzzy urban traffic flow control using a cooperative multi-agent system based on two stage fuzzy clustering. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, 1–5.

Dusparic, I., and Cahill, V. 2009. Distributed w-learning: Multi-policy optimization in self-organizing systems. In *Proceedings of the 2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, SASO '09, 20–29. Washington, DC, USA: IEEE Computer Society.

Dusparic, I., and Cahill, V. 2012. Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *ACM Trans. Auton. Adapt. Syst.* 7(1):11:1–11:25.

Edelkamp, S., and Hoffman, J. 2004. PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. Technical report, Albert-Ludwigs-Universität Freiburg.

Feblowitz, M. D.; Ranganathan, A.; Riabov, A. V.; and Udrea, O. 2012. Planning-based composition of stream processing applications. *and Exhibits* 5.

Fox, M., and Long, D. 2001. PDDL2.1: An extension to PDDL for expressing temporal planning domains . In *Technical Report, Dept of Computer Science, University of Durham*.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An Approach to Temporal Planning and Scheduling in Domains with Predictable Exogenous Events. *JAIR* 25:187–231.

Ghallab, M.; Howe, A.; Knoblock, C.; McDermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL - the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

González-Ferrer, A.; Fernández-Olivares, J.; and Castillo, L. 2009. JABBAH: A java application framework for the translation between business process models and htn. In *Working notes of the 19th International Conference on Automated Planning & Scheduling (ICAPS-09) – Proceedings of the 3rd International Competition on Knowledge Engireeng for Planning and Scheduling(ICKEPS)*, 28–37.

Howey, R.; Long, D.; and Fox, M. 2004. Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *Proceedings of the Sixteenth International Conference on Tools with Artificial Intelligence*, 294 – 301.

Jimoh, F.; Chrpa, L.; Gregory, P.; and McCluskey, T. 2012. Enabling autonomic properties in road transport system. In *The 30th Workshop of the UK Planning And Scheduling Special Interest Group, PlanSIG 2012*.

McCluskey, T. L., and Kitchin, D. E. 1998. A tool-supported approach to engineering htn planning models. In *In Proceedings of 10th IEEE International Conference on Tools with Artificial Intelligence*.

McCluskey, T. L., and Simpson, R. 2006. Combining constraint-based and classical formulations for planning domains: GIPO IV. In *Proceedings of the 25th Workshop of the UK Planning and Scheduling SIG (PLANSIG-06)*, 55–65.

Owens, N.; Armstrong, A.; Sullivan, P.; Mitchell, C.; Newton, D.; Brewster, R.; and Trego, T. 2000. Traffic Incident Management Handbook. Technical Report Office of Travel Management, Federal Highway Administration.

Parkinson, S.; Longstaff, A.; Crampton, A.; and Gregory, P. 2012. The application of automated planning to machine tool calibration. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012*.

Plch, T.; Chomut, M.; Brom, C.; and Barták, R. 2012. Inspect, edit and debug pddl documents: Simply and efficiently with pddl studio. *ICAPS12 System Demonstration* 4.

Salkham, A.; Cunningham, R.; Garg, A.; and Cahill, V. 2008. A collaborative reinforcement learning approach to urban traffic control optimization. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '08, 560–566. Washington, DC, USA: IEEE Computer Society.

Shah, M.; McCluskey, T.; and Chrpa, L. 2012. Symbolic representation of road traffic domain for automated planning to manage accidents. In *The 30th Workshop of the UK Planning And Scheduling Special Interest Group, PlanSIG 2012*.

sheng Yang, Z.; Chen, X.; shan Tang, Y.; and Sun, J.-P. 2005. Intelligent cooperation control of urban traffic networks. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 3, 1482–1486 Vol. 3.

Simpson, R.; Kitchin, D. E.; and McCluskey, T. 2007. Planning domain definition using gipo. *Knowledge Engineering Review* 22(2):117–134.

Smith, D. E.; Frank, J.; and Cushing, W. 2008. The anml language. *Proceedings of ICAPS-08*.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE2.0: An integrated tool for designing planning domains. In *Proceedings of the 17th International Conference on Automated Planning & Scheduling (ICAPS-07)*, 336–343. AAAI Press.

Vaquero, T. S.; Tonaco, R.; Costa, G.; Tonidandel, F.; Silva, J. R.; and Beck, J. C. 2012. itSIMPLE4.0: Enhancing the modeling experience of planning problems. In *System Demonstration – Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12)*.

Vaquero, T. S.; Silva, J. R.; and Beck, J. C. 2011. A brief review of tools and methods for knowledge engineering for planning & scheduling. In *Proceedings of the Knowledge Engineering for Planning and Scheduling workshop – The 21th International Conference on Automated Planning & Scheduling (ICAPS-11)*.

Vodrážka, J., and Chrpa, L. 2010. Visual design of planning domains. In *KEPS 2010: Workshop on Knowledge Engineering for Planning and Scheduling*.