A Brief Review of Tools and Methods for Knowledge Engineering for Planning & Scheduling

Tiago Stegun Vaquero[1] and **José Reinaldo Silva**[1] and **J. Christopher Beck**[2]
[1]Department of Mechatronics Engineering, University of São Paulo, Brazil
[2]Department of Mechanical & Industrial Engineering, University of Toronto, Canada
tiago.vaquero@usp.br, reinaldo@usp.br, jcb@mie.utoronto.ca

Abstract

In this paper we present a brief overview of the Knowledge Engineering for Planning and Scheduling (KEPS) area in the light of a prospective design process of planning application models. The main discussion is based on the fact that KE is better introduced in the planning world through the design process, more than through the planning techniques. Thus, we examine the fundamental steps in the design process of AI planning domain models considering techniques and methods that have appeared in the research literature. We analyze design phases that have not been received much attention in practical planning literature.

Introduction

The 25th. anniversary special issue of The Knowledge Engineering Review (KER) brings some reflection on the development of KE during this years and what would be the alternatives for the future (McBurney and Parsons 2011). We think that it would be interesting to briefly analyze in this paper the relationship between KE and planning and scheduling, specially, in the capability of the combined area (KE and P&S) of treating complex real problems. First of all it would be interesting to ask "to which point KE would be directed to better contribute to solve a planning problem?". Clearly we can identify two main points were knowledge is involved in planning and scheduling: i) in the support of the planning algorithm, that is, in the underlying knowledge system used as problem solver; ii) in the modeling of the surrounding domain, and the explicit enunciation of the planning problem. Besides, expert knowledge about the domain can be used to provide better answers.

While much of the mainstream of AI planning has focused on developing and improving planning techniques, for almost twenty years there existed some research on design processes for planning that considers the special characteristics of this class of problem in which the knowledge management is particularly important (Allen, Hendler, and Tate 1990). The integration of the planning algorithms with the design processes is clearly a vital, strategic goal for planning research. Both lines of work are essential, specially if real-life application is the final objective.

Knowledge engineering for planning has not yet reached the maturity of other traditional engineering areas (e.g., Software Engineering (Sommerville 2004)) to define a common sense design process for planning applications. Nevertheless, research in the planning literature has shown some discussion about the needs and singularities of such design process and life cycle (McCluskey et al. 2003; Simpson 2007; Vaquero et al. 2007). Some of these initiatives introduce techniques, methods and tools to support designers during the design life cycle.

In this paper we present an overview of the Knowledge Engineering for Planning and Scheduling (KEPS) area in the light of a hypothetical design process of planning application models. We present a review of tools and methods that address the challenges encountered in each phase of a design process. While examining reviewing the literature about the design cycle, we pinpoint those phases and aspects that have not been received much attention in practical planning literature. Our goal is to provide some inputs for new upcoming research on KEPS in order to address the challenges that have receiving least attention in the AI planning community. They will be important to the development of real planning and scheduling applications.

Design Process of Planning Domain Models

Design process principles have become important to the success of the development and maintenance of real world planning applications. A well-structured design process increases the chances of building an appropriate planning application while reducing possible costs of fixing errors in the future. In this section we examine existing research on knowledge engineering for planning in the light of an prospective design process which derive some features from Software Engineering and Design Engineering fields and expert knowledge from the experience from real planning domain modeling (Vaquero et al. 2007). Such process follows

a partially ordered sequence of phases.The baseline phases are as follows:

1. **Requirements Specification**: the elicitation, analysis, and validation of requirements, potentially using a semi-formal approach and viewpoint analysis (Sommerville and Sawyer 1997).
2. **Knowledge Modeling**: the abstraction, modeling and reuse of the domain definition and the basic relationships within the planning problem.
3. **Model Analysis**: verification and validation of the domain model and the planning problem, as well as model enhancement.
4. **Deploying Model to Planner**: translation of the problem specification into a communication language understood by automated planners.
5. **Plan Synthesis**: interaction with one or more automated planning systems to create potential solutions to the planning problem.
6. **Plan Analysis and Post-Design**: analysis of the generated plans according to some metrics. New insights may be generated and added to the requirements as part of the overall, iterative design process.

Designing a real planning application following a pure theoretical approach can be often impractical and therefore research on real planning application has followed a more practical approach: developing tools to support the design process. Most of the work on KE for planning refers to tools that cover some of the above phases of the design (specific tools) while few of them try to cover the whole process (general tools). In what follows we will explain and analyze how the available tools approach each of the design phases.

In such analysis, we emphasize the characteristics of two of the most important general tools for KE for planning: GIPO (Simpson 2007) and itSIMPLE (Vaquero et al. 2007). GIPO is the pioneer tool for KE for planning that explicitly focus on the challenges of building a planning domain model and itSIMPLE has focused on a disciplined design process of real planning applications. itSIMPLE integrates a set of languages and tools to support the cyclic design process of a domain model, from a informal representation to a formal model. The KE tool itSIMPLE is the winner of the 3rd International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS) and has been designed by the author of this thesis. We also include other tools and methods that have been active on the planning community, especially in KE research. For example, we include tools such as ModPlan (Edelkamp and Mehler 2005) for modeling and analysis with PDDL, MrSPOCK (Cesta et al. 2008) verification of temporal and causal constraints, TIM (Fox and Long 1998; Cresswell, Fox, and Long 2002) and DISCOPLAN (Gerevini and Schubert 1998) for model analysis, VAL (Howey, Long, and Fox 2004) for plan validation, and a number of application-specific tools such as JABBAH (González-Ferrer, Fernández-Olivares, and Castillo 2009) for modeling and planning in business process.

Requirements Specification

It is well-known in the software and systems world that the lack of a requirements phase can be a primary cause of difficulties in a project ranging from budget or schedule overruns to outright failure (Kotonya and Somerville 1996; Sommerville and Sawyer 1997). In real-life projects a clear identification and analysis of requirements is a key issue to the success of the project.

Real-life projects also have distinctive classes of users, stakeholders whose viewpoints (Sommerville and Sawyer 1997) must be combined and made consistent with the goals of the developer or designer. Therefore, a phase of requirement elicitation and analysis must not be missed. In the context of complex systems, the specifications are very unlikely to be provided at once, as components of the new system and even some basic requirements may not be known in the initial phase. Therefore, a requirements specification phase is generally divided in two steps:

1. knowledge acquisition or requirements elicitation.
2. analysis of the requirements to spot conflicts, omissions or misconceptions about the interaction between the planning systems and its surrounding environment.

The two pioneering works on knowledge acquisition in planning were O-Plan (Tate, Drabble, and Dalton 1996) and SIPE (Myers and Wilkins 1997). Both projects have developed tools that help in the knowledge engineering process. Designed for some specific applications, O-Plan introduced the "Common Process Method" while SIPE introduced the Act Editor.

One of the first domain-independent tools in literature for supporting knowledge acquisition is the system GIPO (Simpson 2007). Knowledge acquisition is performed in GIPO with visual interface that has been designed to assist the user by taking care of simple syntax details in order to avoid syntactically ill-formed model specifications (Simpson 2007). GIPO treats this phase as a large knowledge acquisition process and does not distinguish non-functional requirements from functional or operational aspects.

The system itSIMPLE (Vaquero et al. 2007; 2009) focuses not only on the initial phases of a design process but also on the design life cycle of planning applications. The tool provides an integrated environment that supports knowledge acquisition in its early stages. The work was one of the first to introduce the principles of requirements engineering to AI planning. itSIMPLE is more pragmatic with the use of a visual interface for *Elicitation* and *Analysis of Requirements*; it goes directly to the operational and functional aspects, as well as non-functional requirements, using initially a semi-formal representation language, the Unified Modeling Language (UML) (OMG 2005). UML is the most commonly used language for requirements representation in software engineering. Many engineers, working in different application areas, are familiar with this representation.

Because of its pragmatic approach, itSIMPLE is currently closer to users and stakeholders while GIPO embodies a more designer-oriented approach. The former environment goes directly to requirements analysis and validation while the latter encompasses all that in the knowledge analysis

and representation in Object Centered Language (OCL), a language created as part of the GIPO project to better capture the semantics inherent in planning applications (Simpson et al. 2000). Therefore, no external analysis is considered in GIPO while such analysis is an important feature in itSIMPLE via Petri Nets techniques (Silva and Santos 2003). Viewpoint analysis and requirements engineering approaches are lacking in GIPO whereas a sound manipulation for knowledge that are strong in GIPO are missing in itSIMPLE. Certainly both are necessary.

Regarding domain-dependent tools, the work from Bonasso and Boddy (2010) describes an ongoing project for eliciting planning information from the domain experts in order to support NASA operations personnel in planning and executing activities on the International Space Station (ISS). Aiming at the initial phase of the design, this work introduces a tool for gathering procedural requirements, including a) time, for both task duration and for temporal constraints among procedures, b) resources that are required, produced or consumed by a procedure, c) preconditions, post-conditions and other constraints for both a given procedure and among concurrently executing procedures, and d) the decomposition of large procedures into the fundamental actions used to build up a mission plan (Bonasso and Boddy 2010). With such planning information acquired using template forms, the goal is to generate actions specifications in a standard planning languages that automated planners can use. Even though the tool has a particular application, its method can be generalized to other domains.

Knowledge Modeling

No real-life system is truly isolated. Any planning system is embedded in a real domain: a myriad of "non-system" tools, objects, people, and processes with which it must interact. It is necessary to have a model for this environment – and it is important that this model be developed independently of the planning system (McDermott 1981). The term model implies that we have a representation or a formalism that mirrors behaviors in the real domain. Such model representation must provide semantics, implying that named elements within it correspond directly to named elements in the real domain (McCluskey 2002). Moreover, since several planning problems could be related to the same or similar domains, reusability is a key issue in real-life planning systems and is a fundamental part of the modeling process. Reuse can speed up the design process by using well-tested model structures and elements from other successful applications.

GIPO utilizes the Object Centered Language (OCL) to model domain ontology, objects and actions. GIPO's GUI allows the use of diagrams to support users defining such domain elements. For the domain ontology representation, GIPO (version IV) provides an editor to create "Concept Diagrams" in the style of UML class diagrams to define the kinds of objects and concepts, as well as their relationships. These diagrams also provide the opportunity to define properties that are common to all object instances of the various concepts (Simpson 2007).

The Life History editor from GIPO allows the user to draw state machines that describe the dynamics of an object class. It can be used to name the states that object instances can occupy and to show the possible transitions between the states. In addition to the manual input of the action representation, users can be assisted by induction techniques to aid the acquisition of detailed operator descriptions. The operator description assistant, called *opmaker*, requires as input an initial structural description of the domain along with a training instance and a valid plan for that instance (Simpson 2007).

GIPO is the only tool that currently provides support for knowledge re-use in which the development of the action representation may involve the use of common design patterns of planning domain structures (called "Generic Types"). Of course, domain knowledge re-use and storage raise a number of issues such as when to create new reusable knowledge to avoid storing too many similar ones, or what is the impact on planner performance of a reusable component. At the moment a base of cases does not exist and these issues have consequently not yet been demonstrated in practice.

itSIMPLE provides a tool-set and methods to support designers during domain model creation through an object-oriented approach (Vaquero et al. 2007). The system uses the diagrammatic language UML (OMG 2005) for *Knowledge Acquisition* and *Modeling* processes. Modeling is made through UML diagrams, from a high level of abstraction (such as use case diagrams) to lower levels (e.g., class diagrams or state machine diagrams). The visual components provided by UML can make the planning domain modeling process friendly and can facilitate communication and analysis of requirements belonging to different viewpoints (e.g., stakeholders, planning domain experts, users).

Classes, properties, relationships, and constraints are defined by using class diagrams which represent most of static characteristics of a domain. Operators' parameters and durations are also specified in the class diagram. The dynamics of operators (actions) are modeled by using UML state machine diagrams. These diagrams represent the states that a class object can enter during its life. One diagram is built for each class that has dynamic features. An action's pre- and post-conditions are defined by using a formal constraint language of UML, called Object Constraint Language (OCL – a different "OCL" than used in GIPO) (OMG 2003), as part of operator representation. Planning problems are created by using object diagrams which represent snapshots of a planning domain, most commonly the initial state and the goal state. Users can also create preferences and constraints with object diagrams, for instance on the plan trajectory, which can capture either desirable or undesirable situations (Vaquero et al. 2007).

ModPlan (Edelkamp and Mehler 2005) is a planning workbench that provides some knowledge acquisition and modeling functionalities. The tool uses PDDL (version 2.2) as the base representation of the knowledge and it is aimed more at planning experts than designers with domain knowledge.

Bouillet et al. (2007) describe an ongoing knowledge engineering and planning framework that supports designers during construction of planning domains and problems

based on OWL ontologies (McGuinness and van Harmelen 2004). The state of the world is represented as a set of OWL facts, using a Resource Description Framework (RDF) graph, while actions are described as RDF graph transformations. Planning goals are described as RDF graph patterns. The framework allows the creation of planning domain through OWL ontologies extension in a collaborative manner. The framework (2007) provides a certain re-use capability by the general concept of ontology, but not as explicitly as GIPO. As mentioned by Bouillet et al. (2007), the use of OWL ontologies as a basis for modeling domains allows the re-use of existing knowledge in the Semantic Web. While the framework has been applied towards composing workflows in stream processing systems, it can be seen as a general tool and could, therefore, be applied in other planning domains (Bouillet et al. 2007).

Inspired by GIPO and itSIMPLE, Vodrazka and Chrpa (2010) created a compact modeling tool for planning as an attempt to simplify the model construction process to non-planning experts. Unlike GIPO and itSIMPLE, the tool called VIZ uses straightforward approach to model simple STRIPS-like domains. The tool provides a graphical interface that uses uniquely simple (non-standard) diagrams to capture action specifications, objects and their relations.

Besides the general purpose tools, as those considered above, there are also research on specific knowledge applications. For example, JABBAH (González-Ferrer, Fernández-Olivares, and Castillo 2009) is a KE tool dedicated to Business Process Modeling (BPM). This tool is able to support modeling and representation of business process models in order to use planners to obtain action plans for task management.

Model Analysis

The definition of a suitable planning domain is related to the possibility of analyzing the model during the design phases. Contrasting features of the model being built and the acquired requirements becomes very important in non-trivial planning applications. Model analysis encompasses verification, validation, knowledge enhancement and refinement of the entire model (McCluskey 2002). Generally, the analysis, performed manually, automatically or system-assisted, focuses on two main aspects: the static and dynamic properties. Fining errors, inconsistencies and incoherences in such properties can save time and resources in posterior phases.

Static analysis essentially investigates whether the model is self-consistent. Such analysis can range from simple syntax checkers and debuggers to cross-validation of different parts of the model, particularly for those models containing a set of diagrams or representation schemes. For example, static analysis can be applied to verify the definition of types of objects, constraints, state definition, and other static model elements.

Dynamic analysis entails validating whether the behavior of modeled actions is consistent to the requirements and to what is expected by humans. That involves the examination of how actions interact with each other and how they are executed. Both static and dynamic analysis can be made independently of the planner.

Most real-life planning problems require the investigation and enhancement of specific knowledge, acquired during analysis, in order to achieve reliable planner performance and high plan quality. Some of this specific knowledge may take the form of heuristics or domain control knowledge that can be used to guide planners in finding an efficient plan. Moreover, knowledge enhancement may be concerned with the inclusion of design decisions, reasons, and justifications (i.e. rationales) in the specification process and documentation, which supports the maintenance of complex projects. Klein (1993) explains how rationale are important to engineering design projects for airplane parts.

Few methods and tools are available in the planning literature that can deal with domain analysis. As described by McCluskey (2002), because AI planning has been largely in the realm of research, many researchers in the past used nothing more than basic syntax checkers in support of their model analysis process. However, this approach is neither sufficient nor efficient in large models. Inspired by these large applications, recent research has introduced more elaborated domain analysis techniques.

GIPO (Simpson 2007) checks local and global model consistency such as object class hierarchy consistency, object state descriptions invariants satisfaction, mutual consistency of predicate structures, and others. This static validation can uncover potential errors within a domain specification. In addition to static analysis, GIPO provides a visual representation of dynamic behavior for analysis, a combination of state-machine-like diagrams to show how objects of two or more concept types coordinate their dynamic movements (Simpson 2007). GIPO allows designers to check the model against a set of problems by using a *stepper*. The *stepper* provides the manual selection of actions state-by-state to verify their applicability and to validate the dynamic part of domain model (Simpson 2007). Knowledge discovery and enhancement during domain analysis is not provides by GIPO.

itSIMPLE (Vaquero et al. 2007) provides a rich graphical interface where different viewpoints can be used to validate or criticize a model. Users are supported while creating coherent diagrams to avoid modeling mistakes. For example, snapshots are created based on class diagrams and all constraints defined on them. The tool can check each snapshot for coherence in order to avoid inconsistent states. For dynamic analysis, the environment uses Petri Nets (Murata 1989), a formal representation for dynamic domain validation deploying visual and animated information of the entire system based on the UML state machine diagrams. However, the approach with Petri Net is not fully implemented and tested. itSIMPLE has no support for knowledge enhancement, trusting the requirement validation process to provide insight into improving the problem description and also, based on the direct intervention of the designer, to provide heuristics to guide the planner.

In the literature there is a large variety of techniques for knowledge extraction during domain analysis, but most of them are dependent of the planning system (McCluskey 2002). The extraction of properties such as types, invariants, strategies, heuristics, or subproblems can be a way to en-

hance models with essential information to be used during the planning process. Systems such as TIM (Fox and Long 1998; Cresswell, Fox, and Long 2002) and DISCOPLAN (Gerevini and Schubert 1998) find types and state invariants while RSA (Scholz 1999) and RedOp (Haslum and Jonsson 2000) find different types of constraints on which action sequences are necessary or relevant for solving a given problem. Moreover, the work described in (Fox and Long 1999; Crawford et al. 1996) introduces detection of symmetry as additional knowledge to improve planners' performance.

Deploying Model to Planner

Standard planners cannot directly parse arbitrary specification languages. Enabling them to do so would require a large amount of effort which is, at best, peripheral to the interests of the researchers, mainly directed to the implementation of AI planning algorithms. Therefore, it is reasonable to develop a unified communication language embodying a convergence of goals and computational effort.

At present, PDDL (Fox and Long 2003) works as such a language even though it was not explicitly designed with that purpose in mind. Thus, existing tools and integrated environments should be able to translate specifications to a communication language without any (or minimum) loss in the problem specification. The communication language must, therefore, have the same expressive power as the specification language. Since expressiveness issues are not in the scope of this thesis, we do not go into that discussion.

Considering general purpose tools, GIPO, itSIMPLE and VIZ have sound and efficient mechanisms to translate their respective front-end languages to PDDL. GIPO translates its OCL domain model to PDDL 2.2 (Simpson 2007) while itSIMPLE transfers the knowledge in UML to a solver-ready PDDL model up to version 3.1 (the latest version of PDDL) (Vaquero, Silva, and Beck 2010). VIZ translates simple diagrams into a STRIPS-like PDDL model (Vodrazka and Chrpa 2010).

Regarding specific tools, JABBAH translates a business process model into a solver-ready representation, in this case the Hierarchical Task Network (HTN) (González-Ferrer, Fernández-Olivares, and Castillo 2009). Fernández et al. (2009) describe an approach to represent data-mining processes, using Predictive Model Markup Language (PMML), in terms of automated planning and translate the data mining tasks into PDDL. The tool PORSCE II (Hatzi et al. 2009), while focusing on semantic description of web services, provides a translation process from OWL-S to PDDL to make the domain available for planners.

Plan Development

In this phase, plans are produced by planning algorithms based on knowledge specified and modeled in the domain model. This phase is where most research work on AI planning are focused. However, instead of focusing on planning techniques (see (Ghallab, Nau, and Traverso 2004) for details on techniques) we look at the research on KE tools that give support and facilitate the use of planning algorithms, by an integrated environment that support different planners - using distinct AI planning approaches - and by including features to communicate and visualize the resulting plans.

The most significant work on this phase is itSIMPLE (Vaquero et al. 2009; Vaquero, Silva, and Beck 2010). As an integrated environment, itSIMPLE uses PDDL to communicate automatically with solvers, including Metric-FF, FF, SGPlan, MIPS-xxl, LPG-TD, LPG, hspsp, SATPlan, Plan-A, Blackbox, MaxPlan, LPRPG, and Marvin. In fact, the tool allows new planners to be easily added. This feature gives to itSIMPLE the flexibility to exploit recent advances in solver technology. Designers can test different planning approaches on their model and identify the most promising one. Other tools like GIPO, ModPlan, JABBAH and VIZ do not have such extensive connection with planners.

Plan Analysis and Post-Design

Because of the characteristics of models, it is likely that some problem instances, domains, and models will be better suited to the one planning algorithm rather than another. Furthermore, for complex problems, the lack of knowledge or ill-defined requirements and metrics could propagate to specifications and from there to the problem submitted to the planner. Either of these scenarios (and others) may lead to the generation of poor quality plans. Regardless, bad plans must be spotted and fixed.

A last fundamental step in the design cycle is the analysis of generated plans with respect to the requirements and quality metrics. Plan analysis naturally leads to feedback and the discovery of hidden requirements for refining the model, giving consequently the capacity of improving the quality of generated plans. We call '*post-design analysis*' the process performed after plan generation, in which we have a base model and a set of planners to investigate the solutions generated by them. Such a post-design process requires approaches that range from simple plan validation and visualization to a more sophisticated treatment based on metrics. Such a treatment should be able to evaluate the plan and to relate defects to a set of requirements or even to a lack of such requirements. What may be produced is a new insight and a need to change requirements which can be used in the next design iteration.

AI planning research on plan analysis has developed tools and techniques mostly for plan validation, plan visualization (e.g., diagrams and Gantt charts), animation, plan querying and summarization. In (Smith and Holzmann 2005), formal verification is used in order to check the existence of undesirable plans with respect to the domain model. The work from Howey, Long, and Fox (2004) describes the system VAL, a plan validation tool for PDDL. Given an input plan in PDDL syntax and its respective domain model and problem, VAL validates action execution while recognizing if the plan does not reach the specified goal or if it contains invalid sequence of actions. VAL has recently been extended to capture most of PDDL features (up to version 3.0) such as continuous effects, exogenous events and process handling.

GIPO provides a simple plan visualizer (*animator*) to allow a graphical view of successful plans. Recent publications (McCluskey and Simpson 2006) describes such visualization tool. In contrast with GIPO, itSIMPLE starts from

basic visualization and simulation of plans to a more sophistiThis information is the base for discovering hidden requirements, constraints, preferences and the real intentions of design actors (knowledge that was not identified during the virtual prototyping phase).cated simulation interaction and analysis of domain variable (Vaquero et al. 2007). With a flexible interface to a large set of planners, it is possible to analyze plans produced by different planning techniques. The tool supports plan evaluation through a functionality called "Variable Tracking", which allows analysis based on variable observation or quality metrics displayed on charts. The functionality called "Movie Maker" provides a simulation and a visualization of plans through a sequence of UML object diagrams, snapshot-by-snapshot. A minimal interaction with the simulator is allowed where new actions can be added or removed to check different situations. The plan analysis tools provided by itSIMPLE aim to help designer adjust models by observing plans being executed in diagrammatic form. However, the use of these diagrams prevent the proper analysis of large-scale problems.

ModPlan integrates VAL for plan validation (Edelkamp and Mehler 2005) and, for plan visualization, it includes the animation system Vega (Hipke 2000) allowing a magnification to an arbitrary part of the plan. Gantt charts are plotted for temporal plans, in which a horizontal open oblong is drawn against each activity indicating estimated duration. Plan animation is assisted by users and is provided for some benchmark domains. JABBAH also shows output plans for business processes using Gantt diagram (González-Ferrer, Fernández-Olivares, and Castillo 2009).

The work from Haas and Havens (2008) introduces a specific dynamic plan simulator for the Canadian CoastWatch project. CoastWatch is an oversubscribed dynamic multimode problem with unit resources and lies in the Search & Rescue domain. CoastWatch datasets simulate a typical day for the Canadian Coast Guard, where officers assign resources (planes, helicopters, ships) to execute several different kinds of missions (rescue, patrol, transport). The dynamic simulator includes a visualization tool which creates an animation of the planning and scheduling problem on GoogleEarth™. The animation steps through the scheduling horizon and visualizes the different entities in action. Such application-dependent simulator creates a good communication channel between project participants.

Aiming at plans with rich sophistication and complexity, Myers (2006) describe a domain-independent framework for plan summarization and comparison that can help a human user to understand both key strategic elements of an individual plan and important differences among plans. The goal of such summarization and comparison is to analyze the relative merits of various plan candidates before deciding on a final option. The approach is grounded in a domain metatheory, which specifies important semantic properties of tasks, actions, planning variables, and plans. This work defines three capabilities grounded in the metatheoretic approach: (1) summarization of an individual plan, (2) comparison of pairs of plans, and (3) analysis of a collection of plans. The approach has the benefit of framing summaries and comparisons in terms of high-level semantic concepts, rather than low-level syntactic details of plan structures and derivation processes. As reported by Myers (2006), application of these capabilities within a rich application domain facilitates user understandability of complex plans.

Giuliano and Johnston (2010) proposes a visualization tool for multi-objective problems in space telescope control systems that helps users while selecting schedules to be executed. The tool supports schedule analysis by keeping user objectives separated instead of combined to make trade-offs between competing objectives. The analysis is done through charts and graphs to explore the different aspects of distinct schedules. In a similar direction, Cesta et al. (2008) describe the MrSPOCK system able also to validate schedules and illustrate trade-offs of space mission plans. These two works emphasize how important and difficult it is to work with different criteria coming from distinct groups in real problems.

The main focus of works like Myers (2006), Cesta et al. (2008) and Giuliano and Johnston (2010) is on helping users to (1) better understand the underlying properties of generated plans and schedules and (2) to select the most appropriated solutions to be executed. In fact, re-modeling, or the refinement cycle, seems not be the main target for most of plan analysis tools in AI planning literature. In addition, acquiring valuable information during analysis process itself is not the main goal either.

Discussion

As defined by Fox (2011), knowledge engineering is "a discipline that involves integrating knowledge into computer systems in order to solve complex problems, normally requiring a high level of human expertise". Therefore, KE is connected with solving real problems. In what follows we look at the tools and its underlying knowledge systems and methods in this perspective, that is, by the contribution they might have to the integration of knowledge into computer systems that solve real problems.

As we mentioned before, the contribution of these tools could appear: i) in the knowledge system underlying algorithms encapsulated in planners; ii) in the knowledge representation methods applied to the domain environment; iii) in the knowledge about the design process. Here, we focus the discussion on the last two kinds of contribution (since the present analysis does not include planners).

Concerning the knowledge representation methods applied to the domain we should point the following:

1. none of the tools treat differently the knowledge encapsulated in the planning problem and in the surrounding domain. Such a distinction could be valuable since in several cases it is required to solve different instances of the planning problem. All of those instances have the same surrounding domain and such knowledge could be reused.
2. the general problem of reusing knowledge is not explored besides some tools as GIPO where an attempt were made by using design patterns.
3. there is no attempt in any tool to investigate the relationship between knowledge domain and the underlying knowledge system that supports planners. That implies

that there is no way to find a "best planner" to a given planning domain. Some tools (like itSIMPLE) address at least a pool of planners where the result could be compared, but that is far from solving the matching issue.

Regarding the design process, the challenging points are:

1. there is not a referential design process for planning and scheduling applications. At most, general phases are adapted from a General Design Theory (Tomiyama 1994) or more recent developments of that. Therefore, it is not clear if formal methods such as Process Algebra, Petri Nets, and others would help or improve the process.
2. in all tools, few attention was given to requirement analysis which is important to any real problem. Certainly it is not possible to assume that the planning problem could be synthesized ad hoc, without a requirement gathering and analysis.
3. once the planning problem is defined, it is necessary to provide a model of the overall domain that could be formally verified. In that point it is important to notice that to cover scheduling a formal time verification is required. Unfortunately there is not a long list of time formalisms that could be applied for that.
4. all previous topics are concerned with what could be called the design phases, that is, all phases that culminate with a design model for the domain and planning problem. However the design process could be accelerated if a post-design process is added to feed the refinement cycle, just when the top down refinement no longer improves the model significantly.

Besides all these points another important issue is that there is not enough effort to support scheduling for several reasons, where the most significant is the lack of a time formalism that could be more practical to be inserted in software tools. In addition, there has been not enough effort to support resource reasoning.

Conclusion

In this paper, we presented a brief overview of tools and methods on knowledge engineering for planning considering a design process of planning domain models. We reviewed existing tools that support each phase of such design process, especially the plan analysis and post-design which drive re-modeling and refinement. The intention of this paper is to raise a discussion about where we are on the use of knowledge engineering for planning and scheduling and to give some inputs about were we want to go as research community in this area.

References

Allen, J. F.; Hendler, J.; and Tate, A. 1990. *Readings on Planning*. San Mateo, Ca., USA: Morgan-Kaufman.

Bonasso, P., and Boddy, M. 2010. Eliciting Planning Information from Subject Matter Experts. In *Proceedings of ICAPS 2010 Workshop on Scheduling and Knowledge Engineering for Planning and Scheduling (KEPS)*, 5–12.

Bouillet, E.; Feblowitz, M.; Liu, Z.; Ranganathan, A.; and Riabov, A. 2007. A Knowledge Engineering and Planning Framework based on OWL Ontologies. In *Proceedings of the Second International Competition on Knowledge Engineering*.

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2008. Validation and verification issues in a timeline-based planning system. In *Proceedings of the ICAPS 2008 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Crawford, J.; Ginsberg, M.; Luks, E.; and Roy, A. 1996. Symmetry-breaking predicates for search problems. In *Fifth International Conference on Principles of Knowledge Representation and Reasoning*, 148–159. Cambridge, Massachusetts: Morgan Kaufmann.

Cresswell, S.; Fox, M.; and Long, D. 2002. Extending tim domain analysis to handle adl constructs. In *Knowledge Engineering Tools and Techniques for AI Planning: AIPS'02 workshop*.

Edelkamp, S., and Mehler, T. 2005. Knowledge acquisition and knowledge engineering in the modplan workbench. In *Proceedings of the First International Competition on Knowledge Engineering for AI Planning*.

Fernández, S.; Fernández, F.; Sánchez, A.; de la Rosa, T.; Ortiz, J.; Borrajo, D.; and Manzano, D. 2009. On Compiling Data Mining Tasks to PDDL. In *Proceedings of the Third International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS), ICAPS 2009*, 8–17.

Fox, M., and Long, D. 1998. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research* 9:367–421.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In *Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 956–961. Stockholm, Sweden: Morgan Kaufmann.

Fox, M., and Long, D. 2003. PDDL2.1: An extension of PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)* 20:61–124.

Fox, J. 2011. Formalizing knowledge and expertise: where have we been and where are we going? *The Knowledge Engineering Review* 26(1):5–10.

Gerevini, A., and Schubert, L. 1998. Inferring state constraints for domain-independent planning. In *Proceedings of 15th National Conference on Artificial Intelligence*, 905–912. Madison, USA: AAAI Press/MIT Press.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. San Francisco, CA, USA: Morgan Kaufman.

Giuliano, M. E., and Johnston, M. D. 2010. Visualization Tools for Multi-Objective Scheduling Algorithms. In *Proceedings of ICAPS 2010 System Demostration*, 11–14.

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 *Proceedings of the Third International Competition on*

Knowledge Engineering for Planning and Scheduling (ICKEPS), ICAPS 2009, 28–37.

Haas, W., and Havens, W. S. 2008. Generating Random Dynamic Resource Scheduling Problems. In *ICAPS 2008 Workshop on Knowledge Engineering for Planning and Scheduling*.

Haslum, P., and Jonsson, P. 2000. Planning with reduced operator sets. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems (AIPS)*, 150–158. Breckenridge, CO: AAAI Press.

Hatzi, O.; Meditskos, G.; Vrakas, D.; Bassiliades, N.; Anagnostopoulos, D.; and Vlahavas, I. 2009. PORSCE II: Using Planning for Semantic Web Service Composition. In *Proceedings of the Third International Competition on Knowledge Engineering for Planning and Scheduling*, 38–45.

Hipke, C. A. 2000. *Distributed Visualization of Geometric Algorithms*. Phd thesis, University of Freiburg.

Howey, R.; Long, D.; and Fox, M. 2004. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *ICTAI'04: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301. Washington, DC, USA: IEEE Computer Society.

Klein, M. 1993. Capturing design rationale in concurrent engineering teams. *IEEE Computer* 26(1):39–47.

Kotonya, G., and Somerville, I. 1996. Requirements engineering with viewpoints.

McBurney, P., and Parsons, S. 2011. *The Knowledge Engineering Review*, volume 26 - Special Issue 01 (25th Anniversary Issue). Cambridge University Press.

McCluskey, T. L., and Simpson, R. M. 2006. Tool support for planning and plan analysis within domains embodying continuous change. In *Proceedings of the ICAPS 2006 Workshop on Plan Analysis and Management*.

McCluskey, T.; Aler, R.; Borrajo, D.; Haslum, P.; Jarvis, P.; Refanidis, I.; and SCHOLZ. 2003. Knowledge engineering for planning roadmap.

McCluskey, T. L. 2002. Knowledge Engineering: Issues for the AI Planning Community. *Proceedings of the AIPS-2002 Workshop on Knowledge Engineering Tools and Techniques for AI Planning. Toulouse, France* 1–4.

McDermott, J. 1981. Domain knowledge and the design process. In *Proceedings of the 18th Conference on Design Automation*, 580–588. Piscataway, NJ, USA: IEEE Press.

McGuinness, D. L., and van Harmelen, F. 2004. OWL Web Ontology Language Overview. W3C recommendation.

Murata, T. 1989. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, 541–580.

Myers, K. L., and Wilkins, D. 1997. The Act-Editor User's Guide: A Manual for Version 2.2.

Myers, K. L. 2006. Metatheoretic Plan Summarization and Comparison. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS-06)*. Cumbria, UK: AAAI Press.

OMG. 2003. *OMG Unified Modeling Language Specification - Object Constraint Language, Version 2.0*.

OMG. 2005. *OMG Unified Modeling Language Specification, Version 2.0*.

Scholz, U. 1999. Action constraints for planning. In *Biundo & Fox*, 148–160. Berlin, Heidelberg: Springer Verlag.

Silva, J. R., and Santos, E. A. 2003. Viewpoint requirements validation based on petri nets. In *Proceedings of the 17th Int. Conf. of Mechanical Engineering, Brazilian Mechanical Eng. Society*.

Simpson, R. M.; Mccluskey, T. L.; Liu, D.; and Kitchin, D. 2000. Knowledge Representation in Planning: A PDDL to OCLh Translation. In *In Proceedings of the 12th International Symposium on Methodologies for Intelligent Systems*. Charlotte, North Carolina, USA: Springer.

Simpson, R. M. 2007. Structural Domain Definition using GIPO IV. In *Proceedings of the Second International Competition on Knowledge Engineering for Planning and Scheduling*.

Smith, M. H., and Holzmann, G. J. 2005. Model Checking Autonomous Planners: Even the best laid plans must be verified. In *Aerospace, 2005 IEEE Conference*, 1–11. IEEE Computer Society.

Sommerville, I., and Sawyer, P. 1997. Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering. *Annals of Software Engineering* 3:101–130.

Sommerville, I. 2004. *Software Engineering (7th Edition)*. Pearson Addison Wesley.

Tate, A.; Drabble, B.; and Dalton, J. 1996. O-Plan: a Knowledged-Based Planner and its Application to Logistics. In *Advanced Planning Technology ARPI*, 259–266. AAAI Press.

Tomiyama, T. 1994. From general design theory to knowledge-intensive engineering. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing* 8:319–333.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE2.0: An integrated Tool for Designing Planning Environments. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS 2007)*. Providence, Rhode Island, USA: AAAI Press.

Vaquero, T. S.; Silva, J. R.; Ferreira, M.; Tonidandel, F.; and Beck, J. C. 2009. From Requirements and Analysis to PDDL in itSIMPLE3.0. In *Proceedings of the Third International Competition on Knowledge Engineering for Planning and Scheduling, ICAPS 2009*, 54–61.

Vaquero, T. S.; Silva, J. R.; and Beck, J. C. 2010. Improving Planning Performance Through Post-Design Analysis. In *Proceedings of ICAPS 2010 workshop on Scheduling and Knowledge Engineering for Planning and Scheduling (KEPS)*, 45–52.

Vodrazka, J., and Chrpa, L. 2010. Visual design of planning domains. In *Proceedings of ICAPS 2010 workshop on Scheduling and Knowledge Engineering for Planning and Scheduling (KEPS)*, 68–69.