Journal of Artificial Intelligence Research 1 (2013) 1-37 Submitted 3/13; published 6/13

Knowledge Engineering for Planning & Scheduling: Tools and Methods

Tiago S. Vaquero TVAQUERO@MIE.UTORONTO.CA
J. Christopher Beck JCB@MIE.UTORONTO.CA
Department of Mechanical & Industrial Engineering,
University of Toronto,
5 King's College Rd. Toronto, Ontario, M5S 3G8, Canada

Thomas L. McCluskey T.L.MCCLUSKEY@HUD.AC.UK
Department of Informatics, School of Computing and Engineering,
University of Huddersfield,
Huddersfield, West Yorkshire, HD1 3DH, United Kingdom

José Reinaldo Silva REINALDO@USP.BR
Department of Mechatronics Engineering,
University of São Paulo,
Av. Prof. Mello Moraes, 2231, Cidade Universitária, São Paulo, SP, 05508-030, Brazil

Abstract

Artificial Intelligence Planning and Scheduling is a promising technology for providing intelligent behavior in complex autonomous systems. The design of these systems requires advanced knowledge engineering tools, dedicated to support the process of creating the knowledge model and its respective integration with algorithms that reason about the world to plan intelligently. In this paper, we review the existing knowledge engineering tools and methods that support the design of knowledge models for AI Planning and Scheduling applications. We examine the state-of-the-art tools and methods of *Knowledge Engineering for Planning & Scheduling* (KEPS) in the context of a hypothetical design process for constructing knowledge models representing domain requirements. While examining the literature, we analyze the design phases that have not received much attention and tighten up the nomenclature of the area.

1. Introduction

The research area of *Knowledge Engineering for Planning & Scheduling* (KEPS) is concerned with the design process and support tools for the development of knowledge models and their respective integration with Artificial Intelligence Planning and Scheduling (P&S) algorithms. KEPS was defined in the 2003 PLANET Roadmap as the processes involving (i) the acquisition, validation and verification, and maintenance of planning domain models, (ii) the selection and optimization of appropriate planning machinery, and (iii) the integration of (i) and (ii) to form a planning and scheduling application (McCluskey, Aler, Borrajo, Haslum, Jarvis, Refanidis, & Scholz, 2003). The area can be seen as a special case of Knowledge-based Systems (KBS), where the need for methodologies for acquiring, modeling and managing knowledge at the conceptual level has long been accepted. However, the peculiarities of P&S applications clearly distinguish KEPS from general knowledge-based systems, chiefly in the area of the principled acquisition and representation of knowledge about actions (McCluskey & Simpson, 2004; McCluskey et al., 2003).

KEPS research focuses on the *design process* for creating reliable, high quality knowledge models of real domains (McCluskey, 2002; Vaquero, Romero, Tonidandel, & Silva, 2007). It includes the investigation of methods, tools and representation languages to support and organize the phases of the design life cycle. Using a well-structured life cycle to guide design increases the chances of building an appropriate application while reducing the costs of encountering and fixing errors in the future (van Moll, Jacobs, Freimut, & Trienekens, 2002; Bender RBT Inc, 2003). A simple design life cycle is feasible for the development of a small prototype system, but is likely to fail to produce large, knowledge-intense applications that are reliable and maintainable (Studer, Benjamins, & Fensel, 1998).

In addition to investigating new techniques to support the design of knowledge models and their integration with planners, research on KEPS has another fundamental purpose. The accessibility of current P&S technology is limited for non-experts, such as someone with substantial domain knowledge but no understanding of automated planning and scheduling. The use of planners as an off-the-shelf technology by non-experts is not realistic, given the current state of AI P&S, as a deep understanding of automated planning and scheduling techniques is needed to adequately utilize them (McCluskey, 2002; Barták, Jaška, Novák, Rovenský, Skalický, Cully, Sheahan, & Thanh-Tung, 2012a). Research on KEPS has the goal of bridging the gap between P&S non-experts and the AI P&S technology, making it accessible for practical problems in the real world.

Studies on KEPS have led to the creation of tools and techniques to support the design of knowledge models and the use of planners for real-world problems. Most of these tools have been presented in specialized workshops and competitions such as the *International Competition on Knowledge Engineering for Planning & Scheduling* (ICKEPS).[1] The competition has motivated the development of powerful KEPS systems and advances in modeling techniques, languages and analysis approaches.

KEPS has not yet reached the maturity of other traditional engineering areas (e.g., software engineering (Sommerville, 2004)) in having established standard design processes. Nevertheless, research in the KEPS literature has developed design tools and identified the needs and singularities of the design process and life cycle of AI P&S applications (McCluskey et al., 2003; Simpson, 2007; Vaquero et al., 2007). For the KEPS researcher or developer, however, there is currently little published work on how to proceed when engineering planning domain models, and which design steps and tools to use to support that process. In this paper, we review the tools and methods developed in KEPS in the context of a hypothetical design process for domain knowledge models. This process aims to provide an organization for the research and a framework within which the progress in the area can be understood. We investigate existing tools and methods that address the challenges encountered in each phase of the design process. While examining the design literature, we pinpoint phases and aspects that have not received much attention and tighten up currently non-standardized nomenclature.

This paper is organized as follows. First, we provide a short background on AI Planning and Scheduling and the concepts and roles of knowledge engineering in P&S system design.

1. http://icaps-conference.org/index.php/Main/Competitions

Then, we describe a hypothetical design process as a framework for organizing the research in the KEPS area. Next, we examine and review the KEPS tools and methods according to the phases of the hypothetical design process. We conclude with a discussion of the current state of KEPS and the knowledge engineering challenges that deserve more attention.

2. Artificial Intelligence Planning & Scheduling

In this section we present a short overview of Artificial Intelligence Planning and Scheduling, and provide the preliminaries for the review of available KEPS tools and methods.

2.1 Basic Definitions

Automated P&S is a branch of AI that investigates computational techniques and systems capable of performing automated reasoning about actions (Ghallab, Nau, & Traverso, 2004). Such automated reasoning is considered an essential component of intelligent behavior and, consequently, an essential part of both autonomous systems and decision support frameworks (Wilkins, 1988).

The term *planning* usually refers to the process of creating an organized set of actions, a *plan*, that can be executed to achieve a desired goal from a given initial situation (*initial state*). The term *scheduling* refers to the process of assigning a set of actions to resources over time. AI P&S techniques can focus on both planning and scheduling mechanisms, or only one of them. An algorithm capable of performing automated planning and scheduling is called *planner*. Even though the term *scheduler* is used for algorithms that perform automated scheduling only, we will utilize the term *planner* to represent all P&S algorithms. More details about AI P&S terminology can be found in the works from Ghallab et al. (2004) and Hendler, Tate, and Drummond (1990).

A *planner* generates a plan that is one possible solution to a specified *problem* in a particular *domain* (Hendler et al., 1990). A *domain* is an application area in which we want the planner to operate – for example, block stacking, work-flow generation, or robot navigation. We focus on *domain independent planners*, where domain information is not "built in" to the algorithms embodied in the planner. Rather, the domain information is accepted by the planner as a logically separate input.

2.2 The P&S Knowledge Model

Let us assume that some specific set of overall requirements for the planning and scheduling task are available, related to a domain, within some wider project. Such requirements would naturally contain descriptions of the kind of input goals that the planner needs to solve and the kind of plans that need to be output. For example, it might be essential that resource consumption is taken into consideration and so plans need to be generated which achieve goals while minimizing resource consumption. Before a domain independent planner can be chosen and used, the domain information needs to be conceptualized and formalized. During this process (elaborated in the sections below) the assumptions and features that are essential to represent are derived from the overall requirements.

Within this context, following McCluskey (2002), we define a *domain model* to be a formal description of the application domain which explicitly models entities such as the objects, functions, properties, relations, and actions in the domain. In particular, the language in which the model is written has to have a well-defined operational semantics: independent of planner and domain, there is a defined process for executing actions in the model which represent actions in the domain. This results in an interpretation of domain dynamics for any well-formed domain model (McCluskey, 2002).

Traditionally, the central part of a domain model is the representation of the set of actions that a planner can reason about and the elements that support the specification of actions (Ghallab et al., 2004). A typical logic-based model contains predicates and functions that support the representation of states; the actions, characterized in terms of their preconditions and effects, that define how the state of the world can be changed; ordering constraints over the actions; and temporal constraints, including action durations. As opposed to explicitly representing every possible ground action in the domain,[2] operator schemata are often used to create a compact specification of actions. Each operator schema characterizes a class of possible actions by specifying a set of variables that can be replaced by constants to derive instances of the operator schema. Such instances of operator schema describe specific, ground individual actions (Hendler et al., 1990). For example, in the Logistics domain (Ghallab et al., 2004), the operator schema *drive(t,a,b)* (where t, a and b are variables) represents a class of actions that drives a truck t from location a to location b, while *drive(T1,L1,L2)* represents the ground action (an instance of the *drive(t,a,b)* operator schema) that makes the specific truck T1 go from location L1 to L2.

In general, a planner needs two inputs: the domain model and the *problem description* (i.e. the description of the problem instance to be solved). When the description of the problem instance is specified by an initial state, S_0, and a goal condition, S_G, this characterizes what is called *goal-achievement planning.* The *initial state* is a representation of the state of the world (a set of facts) at the outset of planning while the *goal condition* is a description of the (partial) state the world should be in when the plan has been executed. The goal condition is usually referred to as simply the *goal.* Collectively, the domain model and the problem description, represent all the domain knowledge and associated requirements that is used by the planner to solve the posed problem, hence we call this input information the *knowledge model.*

The *knowledge model* must be represented in a language so it can be processed by planners. The most common language for P&S problems is the *Planning Domain Definition Language* (PDDL) (McDermott, 2000; Fox & Long, 2003; Edelkamp & Hoffmann, 2004; Gerevini & Long, 2006).[3] While many state-of-the-art planners have PDDL as a primary input language, others also use different formal languages including classical STRIPS (Fikes & Nilsson, 1971) (a predecessor of PDDL), Simplified Action Structures (SAS and SAS+) (Bäckström & Nebel, 1995), Hierarchical Task Networks (HTN) (Ghallab et al., 2004), and timeline-based representations (Fratini, Pecora, & Cesta, 2008). Introductory information about representation languages and classical formalisms can be found in the work of Ghallab et al. (2004).

The knowledge model forms a potentially complex knowledge base, and there is a need to use the tools of validation and verification (V&V) to identify and remove bugs during

2. The number of actions can be very large even for simple, small domains.
3. The PDDL 3.1 specification can be found at http://ipc.informatik.uni-freiburg.de/PddlExtension.

its design. This need is similar to the need for V&V in Knowledge-based Systems (Bench-Capon, Coenen, Nwana, Paton, & Shave, 1993). In our context, V&V are defined as follows:

- validation - assuming there is some (informal) statement of requirements, validation is the process of answering the question "does the product we have made satisfy the requirements?". We cannot, in general, formally prove a knowledge model correct with respect to an informal statement of requirements, but we can increase its accuracy of the model by identifying and removing bugs in it.

- verification - when knowledge is captured in one language and needs to be refined into another, it is necessary to guarantee (i.e., verify) the correctness of the translation. For example, if we produce a model, *M1*, in UML, then from that we produce model, *M2*, in PDDL, a verification step (a "proof obligation" in formal methods parlance) demonstrates that *M2* is correct with respect to *M1*. Depending on the formal nature of the languages involved, it may be possible to make this check formal.

In this area, Preece equated Knowledge Engineering to Method plus Measurement, where the latter entails using the processes of V&V (Preece, 2001). The Design Process we use below is an example of the "Method" in Preece's equation. There are parallels with Software Engineering also; the acquisition of a knowledge model is a similar process to acquiring a formal requirements model from a set of informal requirements (Shaw & Gaines, 1996). We can thus elaborate on some properties of the knowledge model that V&V seeks to promote, and use these in the discussion below. With respect to a set of requirements within some domain, we can say that a knowledge model is:

- *accurate* if the features it contains conform to the requirements. In particular, accuracy means relationships depicted in the model are deemed true in the requirements and the effect of actions in the environment faithfully represent the real effects of actions. Accuracy is therefore related to correctness, but while the latter quality is generally considered a relationship between two formal expressions, accuracy relates a formal expression (the knowledge model) with an informal statement (the requirements). For example, consider a *Blocks World* domain model wherein the operator schema *(puton ?x ?y)* did not contain the precondition *(clear ?y)*. We would say that this model was inaccurate, as the requirements demanded that a block be clear before another block was put on it.

- *adequate* if it represents the requirements in sufficient detail. Hence adequacy is related to the level or granularity of the model, and derives from the idea of representational or expressive adequacy of a knowledge representation. Returning to our *Blocks World* example, assume that in the requirements, the duration of *(puton ?x ?y)* operators varied, and required goals to be solved had the additional constraints of minimizing makespan. Then a STRIPS-like model of the *Blocks World*, with its implicit assumption that actions operate instantaneously, would not be adequate.

- *complete* if it is adequate and accurate, and it contains sufficient features to satisfy the requirements. In the durative *Blocks World* above, assume that the operator schema were represented adequately (they had timings) and all the features (relations, conditions and so on) conformed to the requirements. Subsequent automated planning failed because we forgot to represent a *(pickup ?x y)* operator in the model! While the model may have been accurate and adequate, it was not complete.

Knowledge models can be adequate but not accurate (they represent the task at the required level of abstraction, but some of the features are not represented faithfully) or accurate but not adequate (all the features present conform to the requirements, but some requirements cannot be represented at all). A model can be accurate and adequate, but not complete: this is the case where the model does not contain all the features required – for example some parts of the requirements may be missing.

2.3 Solving P&S Problems

Since its origin in the 1970s, research in AI P&S has focused on the development of domain-independent planning algorithms and techniques (Ghallab et al., 2004; ICAPS, 2012). The relative performance of planners varies according to the type of search or inference mechanisms and strategies implemented (see, for example, the varying performance of planners in the *International Planning Competition* (IPC)[4]). Traditionally, given the domain model and problem instance descriptions, these algorithms apply search, general heuristics and inference techniques to generate a solution to the problem.

Most work in AI planning is undertaken under the assumption that a deterministic model of actions is adequate, that is a model where actions always produce the same result on execution. Within such a model, a *plan* is considered a solution to a given problem instance if, when executed starting from the problem's initial state (using the domain model's operational semantics), the result is a state that satisfies the goal. In such an execution, the first action of the plan must be applicable in the initial state (i.e., all the preconditions for this first action must hold in the initial state) and each of the subsequent actions in the plan must be applicable in the state arising from execution of the previous action (Hendler et al., 1990). Repeated analysis of the actions' applicability can determine whether they can be executed in the order specified by the plan.

The process of finding a plan that is actually usable in an application depends not only on the strategy used by the planner, but also the knowledge model (McCluskey et al., 2003). For example, if the model is not accurate, then the planner will generate flawed plans or no plans at all (McCluskey et al., 2003). Even the planner's speed can be affected under such circumstances. For instance, case studies have shown that fixing and refining the model itself (e.g., adding additional relevant knowledge) can improve the performance of planners, without modifying the planners and their search mechanism (Vaquero, Silva, Beck, et al., 2010). In addition, works like (Huang, Selman, & Kautz, 1999; Bacchus & Kabanza, 2000; de la Rosa & McIlraith, 2011; Doherty & Kvarnström, 2001) show that adding relevant, redundant constraints (in the form of control knowledge and rules) in the knowledge model can also speed up planners. An example of these redundant constraints would be the following: "do not move a truck if there is an object in the truck that needs to be unloaded at that location" (Huang et al., 1999).

This leads us to the fourth property of a knowledge model. With respect to a set of requirements within some domain, we can say that a knowledge model is:

4. http://ipc.icaps-conference.org/

- *operational* if a planner can be used with it to generate plans which meet the requirements.

A model can be complete but not operational: given a complete model exists, there will always be ways of re-representing the model without compromising completeness. These models may give different results when input to a planner: some may not satisfy some real time constraint in the requirements. Operationality is thus a different type of property from the three above since it is partly dependent on the choice of the planner. It is also possible that two distinct domain models are operational, but one leads to a more efficient implementation, or better quality plans.

2.4 Applied P&S

During its early history, AI P&S technology was largely restricted to small or simple domains (i.e., 'toy' problems) due to the lack of efficient planning algorithms. More recently, research has enabled the technology to be applied to challenging real-world problems, such as navigation and movement of robots on Mars (Gaines, Estlin, Chouinard, Castano, Castano, Bornstein, Anderson, Judd, Nesnas, & Rabideau, 2006; Jain, Guineau, Lim, Lincoln, Pomerantz, Sohl, & Steele, 2003), control of spacecraft and satellites (Ghallab et al., 2004; Frank, 2008), control of instruments in space stations (Reddy, Ai-Chang, Iatauro, Kurklu, Boyce, Frank, & Jonsson, 2008), oncology therapy planning (Fdez-Olivares, Cózar, & Castillo, 2009), car sequencing in assembly lines (Vaquero, Tonidandel, Barros, & Silva, 2006), and control activities at oil terminals (Sette, Vaquero, Park, & Silva, 2008). These applications have motivated not only the development of new planners, but also improvements and extensions of existing formalisms. For example, the representation language PDDL has been extended to express numeric fluents, metric optimization, durative-actions, timed-based elements, preferences, and constraints (Edelkamp & Hoffmann, 2004; Gerevini & Long, 2006).

The current state of planning and scheduling applications is encouraging for those who need machinery for real-world problems. However, as discussed by Vaquero, Silva, and Beck (2012) there is still a significant gap between the problems typically studied in the research literature and these realistic applications. The major challenges in studying these real problems are found in two main areas: 1) the design and development of an adequate knowledge model, and of the application itself, including processes such as knowledge acquisition, modeling, and analysis of the domain knowledge model; and 2) the development of high performance planners in these complex domains. Challenges encountered in the first area are addressed by the *Knowledge Engineering for Planning & Scheduling.* Challenges encountered in the second area have been the mainstream topic in the AI P&S literature.

3. A Process for Engineering AI Planning Applications

An engineering design process is a sequence of steps (often iterative) that are intended to lead to the creation of a system. As described by Ertas and Jones (1996), the fundamental elements in a design process are the precise definition of the problem, the establishment of objectives and criteria, and the synthesis, analysis, construction, testing and evaluation of the system. Depending on the target product, different design approaches, with the same core elements, exist in the literature. For example, there are distinct design methodologies for software development (Sommerville, 2004) and for knowledge-based system development (Schreiber, Wielinga, & Breuker, 1993).

While the scope of KEPS is to support the process of creating reliable, high quality planning and scheduling applications, current research tends to focus more specifically on the design processes which deliver the knowledge model, ones that consider the challenges and peculiarities of AI P&S applications. By definition, KEPS is concerned with knowledge engineering aspect, hence in this paper we *do not consider the development of the planning and scheduling engine itself.*

Since there is no such established design process, we propose one in this section as a framework for our review. The process has a minimal set of phases to support the knowledge model development. It derives some features from Software Engineering (Sommerville, 2004), Knowledge Engineering (Studer et al., 1998), and Design Engineering (Ertas & Jones, 1996; Eggert, 2010) fields and is based on practical experience from real P&S domain modeling (Vaquero et al., 2007; Vaquero, Costa, Tonidandel, Igreja, Silva, & Beck, 2012).

The phases of the design process are as follows:

1. **Requirements Specification**: the knowledge acquisition and requirements elicitation, analysis and documentation, potentially using a semi-formal approach and viewpoint analysis (Sommerville & Sawyer, 1997). Where AI planning and scheduling is to be an embedded component within a larger system, rather than stand alone, this phase will include extracting requirements such as interface specifications and plan properties from a wider set of requirements and/or design specifications.

2. **Knowledge Modeling**: the abstraction, conceptualization, formulation, modeling and re-use of the domain definition and the basic relationships within the planning and scheduling problem.

3. **Model Analysis**: verification, validation, and enhancement of the domain and problem models.

4. **Model Preparation**: translation of the problem specification into a language understood by automated planners.

5. **Plan/Schedule Synthesis**: choice of automated planning algorithms, and interaction with one or more planning systems to create candidate solutions to the problem.

6. **Plan/Schedule Analysis and Post-Design**: analysis of the generated plans and schedules according to some metrics. New insights may be generated and added to the requirements as part of the overall, iterative design process.

With respect to point (1), is worth noticing that the design of a planning application is usually embedded in a larger design process that encompasses the development and integration of several other components (e.g., software and hardware, robotic systems, sensors, actuators, external applications). This higher level of the design and integration is out of the scope of this paper.

4. KEPS Tools and Methods

Most of the work in KEPS refers to tools and methods that cover some of the phases of the design process in a particular domain (domain-dependent, specific tools), while a few tools try to cover the whole process in several domains (domain-independent, general tools). In this section, we explain and analyze how the available tools approach each of the design phases. The main systems reviewed are the following:

- **GIPO**.[5] The *Graphical Interface for Planning with Objects* (GIPO) is one of the first domain-independent tools in the literature for supporting knowledge acquisition and modeling using visualization and diagrammatic approaches (Simpson, 2007). The system is the winner of the first International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS) as the best general tool.

- **itSIMPLE**.[6] This domain-independent tool focuses on a disciplined design process for real planning and scheduling applications (Vaquero et al., 2007; Vaquero, Silva, Tonidandel, & Beck, 2011). The tool provides an integrated environment that supports knowledge acquisition, modeling and plan analysis. itSIMPLE integrates a set of languages and tools to support the iterative design of a knowledge model (e.g., Unified Modeling Language (UML), eXtensible Markup Language (XML), PDDL, and Petri Nets), from an informal representation to a formal model. The system is the winner of the third ICKEPS as the best general tool. The tool has been tested and used in several real applications such as manufacturing (Vaquero et al., 2006), petroleum supply management (Sette et al., 2008), logistics (Vaquero et al., 2012) and software development management (Udo, Vaquero, Silva, & Tonidandel, 2008).

- **EUROPA**.[7] The *Extensible Universal Remote Operations Planning Architecture* (EUROPA) is an open-source and extensible platform and tool set for building knowledge models and analyzing constraint-based temporal planners (Frank & Jónsson, 2003; Barreiro, Boyce, Do, Frank, Iatauro, Kichkaylo, Ong, Remolina, Smith, & Smith, 2012). The EUROPA platform integrates modeling, constraint reasoning, search and plan/schedule analysis capabilities to solve and address complex planning and scheduling problems. The main input representation languages are the New Domain Definition Language (NDDL)[8] and its successor the Action Notation Modeling Language (ANML) (Smith, Frank, & Cushing, 2008). These representations include state and activity descriptions, common features of traditional planning languages such as PDDL. However, NDDL and ANML also use a state-variable-based formalism called *timelines* (Barreiro et al., 2012). The framework and languages have been successfully applied in several NASA applications such as observation scheduling for the Hubble Telescope (Muscettola, Nayak, Pell, & Williams, 1998) and ground-based activity planning for Mars Exploration Rover project (Ai-Chang, Bresina, Charest, Chase, jung Hsu, Jonsson, Kanefsky, Morris, Rajan, Yglesias, Chafin, Dias, & Maldague,

5. http://scom.hud.ac.uk/planform/gipo/
6. http://code.google.com/p/itsimple/
7. http://code.google.com/p/europapso/
8. A reference manual is available at http://code.google.com/p/europa-pso/wiki/NDDLReference.

2004). The system is the winner of fourth ICKEPS as the best tool in the design process track.

- **ModPlan.**[9] This system is an interactive knowledge acquisition, modeling and engineering tool for AI planning (Edelkamp & Mehler, 2005). Its models are elicited in PDDL, and it is intended for planning experts rather than designers and domain experts. It provides automated domain analysis tools and PDDL learning capabilities.
- **VIZ.**[10] This is a lightweight system which uses straightforward approach to model pure planning domains (Vodrazka & Chrpa, 2010). VIZ provides a graphical user interface for description of planning domains and problems using (non-standard) diagrams. VIZ is a simple tool and can be used to model well-structured domains.
- **MrSPOCK.** The *Mars Express Science Plan Opportunities Coordination Kit* (MrSPOCK) is a timeline-based planning system that has been developed for the European Space Agency (ESA) (Cesta, Finzi, Fratini, Orlandini, & Tronci, 2008). The tool is based on the Timeline-based Representation Framework which aims at supporting the development of applications for different ESA missions. MrSPOCK uses a symbolic model to drive the generation of plans and schedules and then provides formal, automated validation and verification of the generated solutions against the temporal and causal constraints in the symbolic model.
- **TIM.**[11] This tool provides domain-independent techniques for extracting knowledge that is implicit in a PDDL domain description and that can assist domain designers in debugging models (Fox & Long, 1998; Cresswell, Fox, & Long, 2002). As a model analyzer, the system extracts state invariants and type structures from a domain model and makes this information available for designers and planners such as STAN (Fox & Long, 1998).
- **DISCOPLAN.**[12] This is a domain analysis system developed for discovering state invariants in planning domain models, represented in terms of a set of initial conditions and operator schemata expressed in PDDL (Gerevini & Schubert, 1998). The system can find, for example, implied constraints that relate fluents, predicates, single-valuedness constraints and exclusiveness constraints (Gerevini & Schubert, 2001).
- **VAL.**[13] The system is a plan validation tool for PDDL models that has been successfully used during the International Planning Competition (IPC) since 2002 (Howey, Long, & Fox, 2004). VAL can verify whether the PDDL definition of the domain model and the problem instance along with the given plan are syntactically correct and valid.
- **JABBAH.**[14] This system provides a tool for resource allocation analysis of business processes and work-flows (González-Ferrer, Fernández-Olivares, Castillo, et al.,

9. http://www.tzi.de/∼edelkamp/modplan/
10. http://clp.mff.cuni.cz/Viz.html
11. http://planning.cis.strath.ac.uk/TIM/
12. http://www.cs.rochester.edu/ schubert/projects/discoplan.html
13. http://planning.cis.strath.ac.uk/VAL/
14. https://sites.google.com/site/bpm2hth/

2009). It provides automated support for the analysis, allowing engineers to exploit the Business Process Management Notation (BPMN) standard for work-flow specification. The tool provides a non-trivial transformation of BPMN-expressed work-flows to Hierarchical Task Networks (HTN-PDDL). The HTN-PDDL representation is used as input to HTN planner to obtain plans for management tasks (González-Ferrer et al., 2009). JABBAH is the winner of the third ICKEPS as the best domain specific tool.

- **FlowOpt**. This tool combines modeling production work-flows with optimization and scheduling technology based on constraint satisfaction techniques and interactive visualization (Barták et al., 2012a). FlowOpt is built on top of the commercial system MAK€, a performance prediction and optimization system for small and medium enterprises (Barták, Sheahan, & Sheahan, 2012b). One of the main goals of FlowOpt is to simplify work-flow modeling for users, guiding them through the creation of correct models. The tool supports 1) visual design of production work-flows with alternative processes, 2) generation of feasible schedules, 3) visualization of the generated schedules, and 4) analysis of schedules to find potential improvements.
- **MARIO**. This tool is a software composition framework from IBM that uses AI planning to support designers in composing software components into data flows (Feblowitz, Ranganathan, Riabov, & Udrea, 2012; Bouillet, Feblowitz, Liu, Ranganathan, & Riabov, 2008). These flows are used by domain experts to transform raw data into useful information. Developers describe the software composition planning problem and composition goals through the Cascade language (Ranganathan, Riabov, & Udrea, 2009), a tag-based knowledge representation language. The description can include commonly used flow patterns which cover different variations of the flows, the components in the flow and the possible parameterizations of the software components. The system provides a web-based interface for automated planners in which users can input software composition goals, view and parametrize the generated flows, and deploy them onto other platforms.

For each phase of the design process, we describe how the above tools and other works address the embedded knowledge engineering challenges. We emphasize the characteristics of two of the most important general tools for KEPS: GIPO and itSIMPLE. They provide features in almost all phases and, due to their generality and flexibility, are currently the main references in the KEPS literature.

4.1 Requirements Specification

The requirements specification phase refers to the identification, acquisition, analysis and documentation of 1) relevant knowledge about the domain and problem to be solved, and 2) the characteristics and behavior of the P&S system to be developed. The result of this phase is a structured requirements specification document that includes not only a detailed (informal or semi-formal) description of what problem the system should solve, but also a clear description of the quality requirements (e.g., performance and usability constraints) (Kotonya & Somerville, 1996; Kotonya & Sommerville, 1998).

One of the main challenges in this phase is to elicit and combine the requirements and goals of multiple stakeholders (Sommerville & Sawyer, 1997) to prevent an inconsistent and incomplete requirements specification document. To prevent that, it is necessary to analyze, detect and fix conflicts, omissions and misconceptions about the problem and the interaction between the P&S system and its surrounding environment. A disciplined knowledge acquisition and requirements analysis are key factors to precise documentation that is the foundation of the modeling phase.

As a general tool, itSIMPLE (Vaquero et al., 2007; Vaquero, Silva, Ferreira, Tonidandel, & Beck, 2009) was one of the first to introduce the principles of requirements engineering to AI P&S. Requirements and relevant knowledge from different viewpoints are represented as use cases in a semi-formal representation language, the Unified Modeling Language (UML) (OMG, 2005). Analysis and validation of the requirements is done using UML use case diagrams visualization and Petri Nets techniques (Silva & Santos, 2003) to support conflict and inconsistency detection.

Regarding domain-dependent tools, the work of Bonasso and Boddy (2010) describes an ongoing project for eliciting planning and scheduling information and requirements to support NASA operations personnel in planning and executing activities on the International Space Station (ISS). Aiming at the initial phase of the design, the work introduces predefined action template forms in a visual interface for gathering procedural requirements, including a) time, for both task duration and for temporal constraints among procedures, b) resources 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 a mission plan. From the information provided by users through the template forms, the tool generates actions specifications in PDDL and ANML. While the tool has a particular application, its method of using predefined forms can be generalized to other domains.

4.2 Knowledge Modeling

The key stage of our design process is where the informal requirements developed in the step above are translated into a *formal model*. The choices made within this step (e.g., choice of predicates, granularity of representation) are crucial to the success of the whole process. With this in mind, it is important that the model is developed to a large extent independently of the P&S algorithm itself (McDermott, 1981). The term *model* implies that we have a representation that mirrors the real world, for example, operators represent real-world actions and named elements correspond directly to identifiable objects in the real domain (McCluskey, 2002). Most importantly, the model must respect the specified requirements, in terms of accuracy, adequacy and completeness. Moreover, since several planning and scheduling problems could be related, reusability is a key issue in real-life P&S systems and is a fundamental part of the modeling process (Krueger, 1992; Van Vliet, 2002).

In addition to supporting the creation of the knowledge model, most of the available tools in the KEPS literature assist knowledge acquisition as part of the modeling process. Combining knowledge acquisition and modeling in a single step means that relevant knowledge and requirements are acquired as the model progresses as opposed to having a dedicated phase of requirements specification.

Tool sets for knowledge modeling generally contain a helpful GUI supporting the formulation process, but leave the main design decisions to the developer. We will deal with these tool sets in the next subsection. There is a growing amount of research into using automated model building tools, however, and we dedicate the subsequent section to these. The latter have the potential to decrease modeling effort required, and minimize the design bias inherent in the knowledge engineer carrying out the encoding.

4.2.1 Knowledge Modeling: GUI-based Manual Tools

The two pioneering works on knowledge acquisition and modeling in AI planning were O-Plan (Currie & Tate, 1991; Tate, Drabble, & Dalton, 1996) and SIPE (Myers & Wilkins, 1997), both designed to be applicable to a wide set of domains. The O-Plan framework introduced the Task Formalism (Currie & Tate, 1991) while SIPE framework introduced the Act Formalism (Wilkins & Myers, 1995) for representing the model in a HTN format. SIPE has a specific graphical and textual editor, called the “Act Editor”, for developing the knowledge model. Both systems were used for several applications, although the knowledge models produced were tailored to be used with dedicated planners for solving planning problems.

These earlier works inspired the development of GIPO as one of the first planner- and domain-independent tools appearing in the KEPS literature for supporting knowledge acquisition and modeling (Simpson, Mccluskey, Zhao, Aylett, & Doniat, 2001). The interface was built using an object metaphor which had inherent syntax and state consistency checks designed primarily to improve the accuracy of a model using static analysis. The *Life History Editor* interface was a significant advance on the standard GUI interface introduced in the third version of GIPO (Simpson, 2007, 2005), GIPO III. It enabled users, without any expertise in languages such as PDDL, to represent the dynamics of an object class by constructing and annotating graphical state machines; the interface then automatically builds a formal specification of the domain and outputs this in the form of a model in PDDL.

Like GIPO, the itSIMPLE tool-set supports designers during domain model creation through an object-oriented approach (Vaquero et al., 2007). Its principal innovation is to utilize Universal Modeling Language (UML) and its associated method, from the field of software engineering. Developers follow the method from a high level of abstraction (using use case diagrams) to lower levels (creating class diagrams, state machine diagrams, timing diagrams, and object diagrams). Classes, properties, relationships, and constraints are defined in class diagrams which represent most of the static characteristics of a domain. Operator parameters and action durations are also specified in the class diagram, then the dynamics are modeled with UML state machine diagrams to represent the states that a class object can enter during its life (similar, again to GIPO’s inbuilt method). An operator’s pre- and post-conditions are defined using UML’s Object Constraint Language (OMG, 2003), with time constraints modeled using the UML timing diagram or annotated in OCL expressions (Vaquero, Tonaco, Costa, Tonidandel, Silva, & Beck, 2012). Planning problems are created using object diagrams which represent snapshots of a planning domain, most commonly the initial state and the goal state. Desirable or undesirable intermediate (partial) states can be modeled with object diagrams as well (Vaquero et al., 2007). As a general modeling environment, itSIMPLE also provides a PDDL editor as an alternative to the UML editor.

Both GIPO and itSIMPLE provide support for knowledge re-use. In GIPO, the development of the domain ontology and the action representation may involve the use of common design patterns of planning domain structures (called "Generic Types") (Simpson, McCluskey, Long, & Fox, 2002). As an extension of this idea, itSIMPLE provides a set of design patterns for planning, called *modeling patterns* (Vaquero et al., 2012), in a format similar to design patterns in Software Engineering (Gamma, Helm, Johnson, & Vlissides, 1995). In itSIMPLE, these UML template models contain classes, relations, and action definitions for common planning problems such as transportation, assembling, and autonomous rovers. In both GIPO and itSIMPLE, the patterns can be imported, reused and extended by the designer.

Rather than using the object metaphor, ModPlan (Edelkamp & Mehler, 2005) aimed to be a very comprehensive tools environment for knowledge acquisition based firmly around the PDDL language itself. It comprised a text editor for knowledge domain encoding and wizards to specify action preconditions, effects and duration. The wizards generate PDDL code for the actions and place them on the domain model being build through the PDDL editor. In a similar way VIZ (Vodrazka & Chrpa, 2010) was built to help the direct construction of PDDL models. It comprised a simple modeling tool for non-planning experts enabling the user to graphically describe the domain, with the system generating the PDDL.

In contrast to the systems discussed above, there have been some significant planning modeling support systems designed with a particular class of applications in mind. EUROPA (Barreiro et al., 2012), developed at NASA, was aimed at *space applications.* The environment's representation language (NDDL) is fundamentally different from PDDL in that encodings are based around representations of objects and object instances which persist in predefined timelines of continuous activities. Each activity has a start and end time interval (to represent uncertainty of duration), and the distinction between action and state is effectively blurred. Plan generation and execution are therefore linked to a much greater degree than with PDDL-based planners, which typically just perform plan generation. ANML, a higher level language than NDDL, has been introduced to help developers take a more abstract view of modeling, along with graphical visualization of the object type hierarchy and the relationships between actions, fluents, and objects. However, this is ongoing work, and only a subset of ANML can be translated down to NDDL, and then input to EUROPA for operational use.

Work at IBM T.J. Watson has concentrated on knowledge modeling for performing automated *stream processing.* Their early work (Bouillet, Feblowitz, Liu, Ranganathan, & Riabov, 2007) describes a framework for modeling planning knowledge using the Web Ontology Language (OWL) (McGuinness & van Harmelen, 2004). The state of the world is represented as a set of OWL facts, using a Resource Description Framework (RDF) graph, actions are described as RDF graph transformations, and planning goals are described as RDF graph patterns. Reminiscent of Vaquero et al.'s use of UML, the framework offers many of the advantages of using a well-developed existing language. As well as drawing on OWL toolsets, the inherent principle of shared conceptualization in ontologies can give the framework potential to re-use knowledge, although perhaps not in such an action-focused way as the design patterns of GIPO and itSIMPLE offer. The work of this group in this area is ongoing: MARIO, a set of Eclipse plug-ins to support the modeling of data flows (characterized by directed graphs) using the Cascade language (Ranganathan et al., 2009) competed in the 2012 ICKEPS competition.

Besides the general purpose modeling tools, there is also research on specific domain applications. Here the modeling of action knowledge is almost implicit within the application area. Typically a translator is used to translate the application encoding into an explicit PDDL model. For example, JABBAH provides a toolbench for the modeling of business process models, and these are translated into PDDL-like models in order to use a planner to obtain action plans for task management. FlowOpt is an environment for modeling work-flow, but in this case in production planning. It uses a slightly modified Temporal Network with Alternatives formalism (Barták & Cepek, 2008) to allow users to create a hierarchical representation of processes and activities, and model production work-flow structures graphically through the FlowOpt editor by dragging and dropping predefined activities. These activities are placed on a virtual board and connected to describe the flow of material and other constraints between the activities.

4.2.2 Knowledge Modeling: Automated Tools and Techniques

There has been significant progress in the automated or semi-automated acquisition of knowledge models, aimed at learning representations of actions in enough detail to be used as inputs to a planning engine. Using techniques from the field of *Machine Learning*, researchers have experimented with processes that input training or observation inputs, and output solver-ready models in languages such as PDDL. In the context of domain independent planning, as well as research aimed at learning a domain model representing the structure or physics of the world, much machine learning work is aimed at learning domain-specific heuristics to make the use of a planning engine more efficient: here we restrict our discussion to the former area. Machine Learning applied to automated P&S has attracted a long history of research, and we point the reader to a recent survey for a full account (Jiménez, de la Rosa, Fernández, Fernández, & Borrajo, 2012).

Few of the GUI-based approaches discussed include machine learning tools. An exception is GIPO III, which embodies an induction technique to aid the acquisition of operator descriptions, called *OpMaker*. The tool requires an initial structural description of the static parts of the domain comprising knowledge about states of objects and their relations. Given a training problem instance and a valid solution plan for that instance, *OpMaker* outputs a full PDDL model (McCluskey, Cresswell, Richardson, & West, 2010). This kind of domain model learning can be separated into three concerns:

(i) What language is the learned domain model going to be expressed in?

(ii) What inputs (training plans, observations, constraints, partial models, etc.) are there to the learning process?

(iii) What stage is the learning taking place (initial acquisition, or incremental, online adaptation)?

In the case of *OpMaker*, (i) is PDDL version 2.1, (ii) is a partial model and one example, and (iii) is initial acquisition. In fact, most current learning systems aim to output some variant of PDDL within an initial acquisition phase. However, adaptation can be viewed as a non-monotonic special case of initial acquisition, where input to the learning process includes the current domain model as well as training examples and output is the updated model. Wang's OBSERVER system was a seminal example of this, as it learns an initial model and continues to perform repairs to the model during operation (Wang, 1996).

Regarding (ii), systems that learn very expressive domain models tend to demand the most detailed input, often requiring detailed state information before and after action execution within each training plan. Past work on learning domain models for robotic agents in uncertain environments assumes such detailed input and substantial *a priori* knowledge (Amir, 2005; Benson, 1995). These systems are input with predicate descriptions of states and partial or total state information before and after action execution. With such rich inputs, systems such as Amir's SLAF (Amir, 2005) can learn actions within an expressive action schema language. There have been several other notable developments in learning in uncertain or partially known domains. *Reinforcement learning*, traditionally used in single goal or policy learning planners, has recently been developed for symbolic or relational learning, though its potential for learning full models of the PDDL variety is not yet proven (Jiménez et al., 2012). A promising approach towards learning incomplete and uncertain domain models is ongoing in the *Model-lite* project (Yoon & Kambhampati, 2007). Here the authors use probabilistic logic as the basis for the language of the learned domain model, with probabilities of certainties of axioms being refined after testing.

In comparison, significant recent work on learning domain models within a deterministic and totally observable framework has concentrated on learning from example plans but with little or no pre-engineered domain knowledge. The LAMP system (Zhuo, Yang, Hu, & Li, 2010) can form PDDL domain models from example plan scripts and associated initial and goal states only. It inputs object types, predicate specifications, and action headings, and from plan scripts taken from planning solutions, it learns a domain model. The domain model is synthesized using a constraint solver, inputting two sets of constraints. One set is based on assumed physical, consistency and teleological constraints – for example, every action in the example plan script adds at least one precondition for a future action, actions must have non-empty effects, and so on. The other set of constraints is generated using a type of associative classification algorithm which uses each plan script as an itemset, and extracts frequent itemsets to make up constraints. LAMP is aimed at helping knowledge engineers create a new domain model, as the authors maintain that, after learning, the model needs to be hand-crafted to remove bugs. Another system, LOCM, exploits the assumption of an object-centred domain to enable it to learn from plan scripts only (Cresswell, McCluskey, & West, 2010). As with LAMP, LOCM outputs a model in a PDDL format but it inputs *only* training plan scripts: it does not require representations of initial and goal states, or any descriptions of predicates, object classes, states etc. LOCM assumes that the objects referred to in the training plans can be clustered into classes, and each class has a behavior defined by a parameterized state machine, which it constructs using implicit physical constraints on the state change of objects.

To be successful, automated tools that learn domain models have to embody general properties and constraints about actions and objects, and in most cases the kind of domain in which they are learning. The key idea within these approaches is that of *inductive generalization* – using examples of behaviors of a class of objects and generalizing these examples to a theory about the whole class of objects. In the case of planning, a set of plans that are observed from the domain itself is a natural training input. In the context of our engineering process, these training plans would therefore be considered part of the requirements specification: they would be examples of the kinds of plans that the system would be required to generate. Potential training plans could be harvested from a wide range of applications, and examples include logs of commands such as operating system instructions, moves made in a game or traces of work-flow or business process execution. Learning tools have been used successfully in isolation, e.g. LOCM has been used in a system that learns to play the Freecell game by observation, with no *a priori* knowledge of the game (Cresswell et al., 2010). The utility of these approaches within a knowledge engineering process, however, has still to be proved.

4.3 Model Analysis

Model analysis encompasses validation (measuring the accuracy, adequacy and completeness of the model as judged against the requirements), as well as enhancement and refinement of the entire model to make it operational. Generally, the analysis, performed manually, automatically or semi-automatically, focuses on finding errors, inconsistencies, and incoherencies in the *static* and *dynamic* properties of the model.

Static analysis investigates whether the model is well formed and self-consistent and 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. This often leads to the removal of bugs, which improves the *accuracy* of the model. For example, Wickler (2011) defines four features of a (PDDL) domain model that one can query to investigate the overall validity of a domain model: domain types, relation fluency, inconsistent effects and reversible actions. For any domain model, one can use tools based on these properties to investigate the type structure of the model and the set of fluents. The last two properties concern the construction of operators, and help the engineer investigate accuracy (by eliminating inconsistencies in operators), and completeness (by checking whether operators have an inverse, if they need one).

Dynamic analysis entails validating whether the behavior of modeled actions and operator schemes is consistent with the requirements and with what is expected by stake-holders. Its chief benefit is uncovering bugs relating to *completeness* and *operationality* of the model through the examination of how actions are executed and how they interact. For example, dynamic analysis can show the absence of some actions that are necessary to form the solution of a desired goal. Both static and dynamic analysis can be done independently of a planner.

Most real-life planning and scheduling problems require the investigation and enhancement of specific knowledge, acquired during analysis, in order to achieve operationality, in the form of 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 (Huang et al., 1999; Bacchus & Kabanza, 2000; Doherty & Kvarnström, 2001). Moreover, knowledge enhancement may be concerned with the inclusion of design decisions, reasons, and justifications (i.e. rationales) in the specification process and documentation to support the maintenance of complex projects. For example, Klein (1993) explains how rationales are important to engineering design projects for airplane parts.

Few methods and tools are available in the P&S literature for domain analysis. As described by McCluskey (2002), because AI P&S algorithms has been largely in the realm of research, many researchers used nothing more than basic syntax checkers in support of their model analysis process. However, this approach is neither sufficient nor efficient for large models. Inspired by such real-world applications, recent research has introduced more elaborated domain analysis techniques.

GIPO checks include local and global model consistency such as object class hierarchy consistency, satisfaction of object state invariants by operators, and mutual consistency of predicate structures. In addition to static analysis, GIPO provides a visual representation of dynamic behavior, a combination of state-machine-like diagrams to show how objects of two or more concept types coordinate their dynamic movements. Designers can check the model with a set of problem instances by using a *stepper* that provides the manual selection of actions state-by-state to verify their applicability and to validate the dynamic part of domain model.

Similarly, itSIMPLE provides a graphical interface where different viewpoints can be used to validate or criticize a model. Users are supported while creating diagrams to avoid modeling mistakes. For example, snapshots (object diagrams) 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 Petri Net approach is not, as yet, fully implemented and tested.

The model analysis supported in EUROPA follows the approach of providing programming and coding environments such as Eclipse. The NDDL and ANML editors support error detection in the text-based representation. The identified errors are marked using the traditional wavy red underlines. The Eclipse plug-in version of EUROPA inherits the programing support of Eclipse environment (e.g., highlights, hints, warnings, errors) and debuggers for the models.

Considering specific tools, the FlowOpt Editor includes a work-flow verification mechanism. Given that user can insert flaws and inconsistencies in the models, especially in the activity constraints, the system can detect some of them and prevent their addition to the work-flow. When a flaw is found, the tool explains the error and highlights the flaw in the work-flow.

None of the general tools above (GIPO, itSIMPLE, EUROPA) provide routes for knowledge enhancement, although there is a large variety of specific techniques for knowledge extraction during domain analysis. Most of them assume a particular class of planning algorithm: extraction of properties such as types, invariants, strategies, heuristics, or subproblems can be a way to enhance models with essential information to be used during the planning process, but planners must be written to take advantage of them. Systems such as TIM (Fox & Long, 1998; Cresswell et al., 2002) and DISCOPLAN (Gerevini & Schubert, 1998) find types and state invariants while RSA (Scholz, 1999) and RedOp (Haslum & Jonsson, 2000) find conditions which imply that certain action sequences are necessary or relevant for solving a given problem. Moreover, the work described by Fox and Long (1999) and Crawford, Ginsberg, Luks, and Roy (1996) introduces detection of symmetry as additional knowledge to improve planner performance.

There is clearly a link between the creation and analysis of domain models, but research into learning tools which can build both structurally adequate and heuristically efficient domain models has seen little progress. Work on domain analysis has invariably *started* with the input of a complete and correct domain model and is aimed at teasing out heuristics or reforming operators. Domain analysis has had considerable effect on the efficiency of recent planners, and lately has been used to explore the h+ heuristic in benchmark domains (Hoffmann, 2011).

As an example of an operational, planner independent tool, Chrpa's "entanglements" help to reformulate a knowledge model by removing ground actions that look likely not to be used in a solution. Although this can be classed as domain and planner independent knowledge enhancement, in fact it is chiefly aimed at planners that ground operator schema into an action set as an initial step, since it restricts the size of this set. It can be used with any planner inputting PDDL, and empirical tests show that under certain circumstances it will result in plan generation speed up, but currently is a heuristic method with no guarantee of success (Chrpa, McCluskey, & Osborne, 2012b).

Another direction on domain reformulation methods refers to identifying a set of macro actions, i.e. a sequences of the existing domain actions, and adding them to the model as atomic actions. For example, the two actions *pickup* and *stack* in the classical Blocks World domain could be combined as one macro action (*pickup-stack*) in which a block would be moved directly from the table to be on top of another block in a pile. This domain model reformulation approach has been shown to be effective in improving the runtime of planners (Coles, Fox, & Smith, 2007; Botea, Enzenberger, Müller, & Schaeffer, 2005; Newton, Levine, Fox, & Long, 2007). These works have traditionally focused on finding and learning the set of macro actions that can improve average problem solving performance in a given domain. The work from Alhossaini and Beck (2012, 2009), on the other hand, focuses on identifying set of macro actions to specific problem instances. The work uses a machine learning mechanism to develop a prediction model relating the problem instance features to the performance of a planner on domains augmented with the different subset of macro actions. Therefore, the domain model is reformulated and adapted based on the problem instance to be solved.

4.4 Model Preparation

It is unreasonable to expect all automated planners to be written so they can input knowledge models in a range of specification languages. It makes more sense to have a common "standard" family of representation languages that communicates the available knowledge model to the planners. If the knowledge model is not already represented in a standard language, a translation process must take place. A model preparation phase refers to such a translation process, allowing planners to parse the knowledge model.

Tools and integrated environments need to be able to translate specifications to a unified language without any (or minimum) loss in the problem specification. The target language must, therefore, have at least the same expressive power as the source specification language. Hence, if the model is accurate, adequate and complete in the source language, a faithful translation will preserve these properties when the model is represented in the target language. At present, languages from the PDDL family are used as targets, with the various expressive levels of PDDL giving scope for choice, depending on expressivity of the particular encoding.

Considering general 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 while itSIMPLE transfers the knowledge in UML to PDDL 3.1. VIZ translates simple diagrams into a STRIPS-like PDDL model. In EUROPA, NDDL and ANML models are sent directly to the integrated constraint-based planners. EUROPA currently does not allow a translation from NDDL/ANML to PDDL and vice versa; however, the work of Bernardini and Smith (2011) shows that translating PDDL 2.2 into NDDL or ANML models is possible.

Regarding tools that are more planner-specific, JABBAH translates a business process model into a solver-ready representation, in this case HTN-PDDL. Fernández, Fernández, Sánchez, de la Rosa, Ortiz, Borrajo, and Manzano (2009) describe an approach to represent data-mining processes, using Predictive Model Markup Language (PMML), in terms of automated planning and translate to PDDL. The tool PORSCE II (Hatzi, Meditskos, Vrakas, Bassiliades, Anagnostopoulos, & Vlahavas, 2009), while focusing on semantic description of web services, provides a translation process from OWL-S to PDDL. MARIO translates the model represented in Cascade into a planning domain description language called Stream Processing Planning Language (SPPL) (Riabov & Liu, 2006), a language designed on top of PDDL for stream processing planning and other related domains. The generated SPPL model is sent to a specific planner that can handle such formalism. FlowOpt represents the graphical work-flow models in a nested work-flow formalism which is then sent direct to the planner available in the MAK€ platform. For further study in this area, the reader can consult the proceedings of the third ICKEPS,[15] which focused on tools supporting this phase of knowledge engineering.

4.5 Plan/Schedule Synthesis

In this phase, plans and schedules are synthesized by automated P&S algorithms based on the knowledge specified and represented in the knowledge model. Most research effort in AI P&S has focused on this particular phase and a standard overview of the work can be found in Ghallab et al. (2004).

Here we look at the research on KEPS tools that support and facilitate the use of planning and scheduling algorithms. We concentrate on integrated environments that support the use of different planners and the communication of the synthesized plans and schedules back from the planners. Systems where domain analysis or knowledge enhancement is tightly integrated and performed within the planner itself (e.g., (Coles & Smith, 2007)), we consider outside our scope.

Given a knowledge model, it is desirable to be able to apply a range of planners to test for model operationality as well as to identify a specific planner that is likely to perform well. For such a flexible use of a range of planners with a knowledge model, itSIMPLE

15. ICKEPS 2009. http://kti.mff.cuni.cz/ bartak/ICKEPS2009/

is undoubtedly the leading system. Within the itSIMPLE environment designers can use Metric-FF (Hoffmann, 2003), FF (Hoffmann & Nebel, 2001), SGPlan (Hsu, Wah, Huang, & Chen, 2006; Hsu & Wah, 2008), MIPS-xxl (Edelkamp, Jabbar, & Nazih, 2006; Edelkamp & Jabbar, 2008), LPG-TD (Gerevini, Saetti, Serina, & Toninelli, 2004; Gerevini, Saetti, & Serina, 2006), LPG (Gerevini & Serina, 2002), hspsp (Haslum, 2008), SATPlan (Kautz, Selman, & Hoffmann, 2006), Plan-A (Chen, Lv, & Huang, 2008), Blackbox (Kautz & Selman, 1998), MaxPlan (Xing, Chen, & Zhang, 2006), LPRPG (Coles, Fox, Long, & Smith, 2008), POPF (Coles, Coles, Fox, & Long, 2010), and Marvin (Coles & Smith, 2007). This integration feature allows designers to test and explore different P&S approaches on their model and identify the most promising one.

Tools like GIPO, ModPlan, JABBAH, PORSCE II, FlowOpt and MARIO have automated planners (general or specific to the representation used) integrated to their platforms, but are not as extensible and flexible as itSIMPLE. Like these systems, EUROPA provides its own integrated planners, but additionally enables users to create their own customized planners using its interface.

4.6 Plan/Schedule Analysis and Post-Design

The final phase in the design cycle is the analysis of generated plans and schedules with respect to the requirements and quality metrics. Plan/Schedule analysis naturally leads to feedback, trade-off discussions and the discovery of hidden requirements, giving the opportunity to spot possible model improvements and, consequently, increase the quality of generated plans. *Post-design analysis* is the process performed after plan generation in which a base model, a set of planners, and a set of solutions generated by them are analyzed. Careful post-design phase may produce new insights and a need to change requirements which can be used in the next design iteration.

The post-design process requires approaches that range from simple plan/schedule validation and visualization to a more sophisticated treatment based on metrics. KEPS research on plan/schedule analysis has developed tools and techniques mostly for plan validation and verification, plan visualization (e.g., diagrams, charts and Gantt views), animation, plan querying and summarization, and plan refinement.

The work of Howey et al. (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 instance, VAL validates and verifies the execution of the actions in the plan to determine if the goal is reached. VAL has recently been extended to handle most PDDL features (up to version 3.0) including continuous effects, exogenous events and process handling. In (Smith & Holzmann, 2005), formal verification is also used in order to check the existence of undesirable plans with respect to the domain model.

Aiming to create plan visualizers within domain independent tool environments is problematic, given that the wide range of potential planning applications will have differing visual interpretations. However, GIPO's plan visualizer (called *animator*) gives a graphical view of plans using the object metaphor. It animates the action execution in the plan, by showing the state change of objects after each step (McCluskey & Simpson, 2006). itSIMPLE also provides basic visualization of plans as well as more sophisticated simulation of plans and analysis of domain variables. 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 tool uses VAL for extra plan analysis capabilities.

itSIMPLE also has an interface to a 3D content modeling tool that allows user to simulate and execute the generated plans (Vaquero et al., 2010). Such a 3D simulation provides a more realistic validation of the execution of actions and their effects. The goal of the simulator is to provide inputs and knowledge discovery opportunities for the re-modeling cycles. The tool also addresses the acquisition of users feedback raised during the simulations (either with the 3D environment or with the other available analysis tools). Users can input plan evaluation rationales using an ontology-based representation to express their feedback (i.e., whether a plan is good or bad) and what properties of the plan are responsible for their evaluation (Vaquero, Silva, Beck, et al., 2011). The acquired feedback is stored in a database and reused when a new plan is generated. Using the information stored in the database and applying automatic reasoning mechanisms, users can have an automatic initial evaluation (and the corresponding justifications) of a new plan.

EUROPA integrates a set of tools for plan/schedule visualization and analysis of planner statistics. The main visualization mechanism is the Gantt chart, used to analyze resource allocation over time and the changes of the fluent values in a given plan. Line charts are also used for representing the resource usage during the plan execution and for showing planner statistics (e.g., time spent per step in the search process, open decision count per step, number of decisions per step). The framework allows users to interact with planner during the planning process for the investigation of different planning scenarios.

ModPlan integrates VAL for plan validation and, for plan visualization, it includes the animation system Vega (Hipke, 2000) in which Gantt charts are plotted for temporal plans. Plan animation is provided for some benchmark domains only.

JABBAH and FlowOpt show output plans/schedules using Gantt charts. The Gantt viewer from FlowOpt allows users to analyze alternatives not produced by the planner by moving activities to different time slots and resources. When changes are made in the schedule the viewer 1) highlights constraint violations and 2) provides a plan repair functionality that can automatically shift activities while trying to minimally change the original plan. Moreover, FlowOpt includes an analyzer that is able to suggest improvements to the production process by identifying bottlenecks in the generated plan and proposing how to solve them. Users can modify the production based on the given suggestion and interactively call the analyzer.

Daley, Frank, Iatauro, McGann, and Taylor (2005) describe the system PlanWorks, an environment designed at NASA for debugging constraint-based planning and scheduling systems. The tool supports users in the visualization and analysis of logs generated by constraint-based planners while synthesizing plans/schedules. The visualization includes timeline views of resource usage and diagrams to represent the decisions made by the planners. The tool was specially developed as part of the EUROPA project.

The work of Haas and Havens (2008) introduces a domain-specific plan simulator for the Canadian CoastWatch project. CoastWatch is an oversubscribed dynamic multi-mode problem with unit resources in the Search & Rescue domain. 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 simulator includes a visualization tool that creates an animation of the planning and scheduling problem on GoogleEarth^{TM}. The animation steps through the scheduling horizon and visualizes the different entities in action.

Aiming at plans with rich sophistication and complexity, Myers (2006) describes a domain-independent framework for plan summarization and comparison that can help a user understand both key strategic elements of an individual plan and important differences among plans. The approach is grounded in a domain meta-theory, which specifies important semantic properties of tasks, actions, planning variables, and 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) propose a visualization tool for multi-objective problems in space telescope control systems that helps users to select schedules to be executed. The tool supports schedule analysis by keeping user objectives separated 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 functionality of the MrSPOCK system to validate schedules and illustrate trade-offs of space mission plans. These two works emphasize how important and difficult it is to manage different criteria coming from distinct groups.

Sometimes finding an optimal plan for a particular problem is intractable or might take too much time. In many real applications (e.g., robotic rescue systems), planners have to compromise between planning speed and plan quality to achieve satisfactory behavior. To handle these issues, sub-planners seek reasonable solutions as opposed to optimal solutions. These approaches might insert unnecessary and replaceable actions in the resulting plan, therefore requiring *a posteriori* plan refinement and optimization. Different from visualization and simulation approaches, recent works such as (Chrpa, McCluskey, & Osborne, 2012a; Nakhost & Muller, 2010) have been studying automatic processes of plan refinement and optimization as a post-processing step. Based on the domain model, some systems post-process the generated plan while analyzing the action dependencies in order to make it better or optimal, for example, by removing redundant actions (actions that do not contribute to establishing the goal) or replacing some existing actions in the plan (Chrpa et al., 2012a). It is worth noting that these methods rely on accurate and adequate knowledge models, developed, for example, using the above techniques.

4.7 Summary

Table 1 provides a summary of our review. The table presents the examined systems and works considering several dimensions: the design phases they support; the extent they can address different domains (tools that can be used on a wide range domains (domain-independent) and tools that focus on one particular domain or domain family); the extent to which they can feed into a range of planners; and whether aimed at a planning expert, a domain expert, or both. Most of the systems have been developed in last 10 years, indicating the rise in interest in this area. However, the table illustrates the lack of wide spectrum tools, and those that can support requirements specification.

KEPS tools	**Design Phases and Properties**									
	Design Phases						Domain	Planner	Intended Expert	
	1	2	3	4	5	6	Independent	Independent	Domain	Planning
O-Plan		✓				≈	✓			✓
SIPE		✓					✓			✓
GIPO		✓	✓	✓	✓	✓	✓	≈	≈	✓
itSIMPLE	✓	✓	≈	✓	✓	✓	✓	✓	✓	✓
EUROPA		✓	✓		✓	✓	✓			✓
ModPlan		✓	≈		✓	✓	✓			✓
VIZ		✓		✓			✓		✓	
TIM			✓				✓			✓
DISCOPLAN			✓				✓	≈		✓
RSA			✓				✓			✓
RedOp			✓				✓	✓		✓
VAL						✓	✓	✓		✓
PlanWorks						✓	✓			✓
MrSPOCK						✓			✓	
JABBAH		✓		✓	✓	✓			✓	
PORSCE II				✓	✓			✓	✓	
CoastWatch						✓			✓	
FlowOpt		✓	✓		✓	✓			✓	
MARIO		✓		✓	✓				✓	
SLAF		✓					✓	✓		✓
LAMP		✓					✓	✓		✓
LOCM		✓					✓	≈		✓
ARMS		✓					✓	✓		✓
Bonasso & Boddy, 2010	✓			✓					✓	✓
Bouillet et al., 2007		✓					✓			✓
Fox & Long, 1999			✓				✓			✓
Crawford et al., 1996			✓				✓			✓
Fernández et al., 2009				✓				✓	✓	
Giuliano & Johnston, 2010						✓			✓	
Myers, 2006						✓	✓		✓	
Chrpa et al., 2012a						✓	✓	✓		✓
Chrpa et al., 2012b			✓				✓	✓		✓
Nakhost & Muller, 2010						✓	✓	✓		✓

Table 1: Summary of available tools and methods in the Knowledge Engineering for Planning and Scheduling literature. Design phases: (1) Requirements, (2) Knowledge Modeling, (3) Model Analysis, (4) Model Preparation, (5) Plan/Schedule Synthesis, (6) Plan/Schedule Analysis and Post-Design. '✓' means that the feature is present in the tool, '≈' means that it is to some degree present, and *blank* means that it is not present.

5. Discussion

5.1 Challenges and Future Research Directions

The general area of knowledge engineering has been defined as "a discipline that involves integrating knowledge into computer systems in order to solve complex problems, normally requiring a high level of human expertise" (Fox, 2011, p. 5). In what follows we analyze the KEPS tools and methods with a similar perspective of integrating knowledge and *general solvers* into computer systems, where in the case of AI planning, the solver is a planning engine. Hence we evaluate approaches on the basis of the contribution they make to the integration of knowledge and planning engines into computer systems that solve real problems.

Points (1) through to (4) in the discussion relate to knowledge-based aspects, while points (5) through (7) relate to the engineering and design process.

1. There is no agreement on the representation language to use for initial knowledge acquisition and modeling. Most of the tools use some sort of graphical representation for the initial phases of knowledge modeling and formal languages for communicating with planners. Graphical approaches seem to be a trend, making requirements acquisition and modeling easier for designers and better understood by domain experts and non-planning experts. However, there is no work on defining a common graphical language for AI Planning and Scheduling; languages have been borrowed and adapted from other areas (such as UML) to fulfill this role. Regardless of the language used in the initial phases, the model must be read by an automated planner which requires the execution of a translation process to a representation such as PDDL.

2. The general problem of reusing knowledge is not explored outside of tools like GIPO and itSIMPLE where attempts have been made by using design and modeling patterns. In the last 20 years knowledge engineering has embraced the view of *sharing* formalized knowledge with the introduction of globally accessible and community agreed ontologies for representing specific applications as well as representing common knowledge (Uschold & Gruninger, 1996). With few exceptions, this innovation in knowledge re-use has not affected the engineering of planning applications. This may be partly explained in that ontologies tend to contain static, taxonomic or descriptive knowledge about objects, rather than changes effected by actions. More research is needed on extracting planning-related knowledge from ontological descriptions, and on how existing action descriptions can be "marked up" for sharing and re-use.

3. None of the tools surveyed provide explicit help to the engineer to investigate the relationship between knowledge models and the planners: there is no support to find the "best planner" for a given planning and scheduling domain. Some tools (like itSIMPLE) can be linked to a pool of planners, so the results can be compared, but that is far from solving the matching issue.

4. When hand-coding a knowledge model, a designer often adjusts the model based on the performance of planners when input with that model. The behavior of planners changes (for better or for worse) when small modifications and tricks are added to the knowledge model. P&S experts usually know what would affect the performance of their planners but such expertise is not available for domain experts. None of the tools surveyed provided explicit help for this aspect of encoding.

5. 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

Software Engineering. Within this, there are currently no *metrics* for measuring the process and the product of KEPS. How do we measure aspects of the design process such as the removal of flaws in a domain model? How do we measure the quality of the product i.e. the knowledge model? How do we use other insights from software engineering, e.g., the use of formal methods such as Process Algebra, Petri Nets, and others, to help or improve the process?

6. The survey shows that very few tools address the requirement specification and analysis stage. High level representations of requirements are particularly important if we are to address the issue alluded to above, of seeking to make changes in the model in order to assist in increasing the operationality of a model. The changes made must hold the requirements invariant, hence the value in capturing requirements within the tool. Further, if within the captured requirements, formal models could be established (for example in some kind of process algebra), then it would be feasible to formally verify parts of the domain model. For example, in scheduling a formal time verification is required.

7. During the post-design process many tools are available for plan analysis, with opportunities for engineers to visualize a simulation of plan execution. However, features to assist in re-modeling and acquisition of valuable information during this analysis process are missing from all the tools surveyed. These features could contribute to the challenge referred to in points (3) and (4) above.

In addition, another important issue is that there is little effort to support scheduling. There are several reasons for this, with the most significant being the lack of a time formalism that could be inserted in software tools. There has also not been enough effort to support resource reasoning.

5.2 KEPS: Past and Future

Has KEPS progressed in the last 10 years? Considering the section of the PLANET Roadmap (McCluskey et al., 2003) covering KEPS, first produced in 2001 and finally published is 2003, there has not been much progress on the actions suggested at the end of that document. This includes the investigation of engineering and evaluation methods. The exception is the development of tools environments: as shown above, there has been an array of tools produced and used for many applications.

The development of an design process such as the one described in this paper, is important for the principled production of an application embodying a planner. However, there is another aspect as well as engineering soundness: the planning function itself is well known to be intractable in general, and hence a planner cannot be used as a black box as with most software components. Given a statement of requirements, there will be a set of distinct, complete knowledge models which satisfy them. As a corollary to our discussion on the challenges faced, more research is required into engineering methods which aim to find knowledge model(s) from this set which optimize operationality when matched with a particular planning engine. In particular, while one knowledge model may result in an intractable planning problem, another may be operational, while *both* satisfy the requirements. Knowledge model reformulation, such as that reported early in the paper, keeps invariant the input/output language of the planner, and is but a first step to considering this challenge. More generally, we need to create techniques which can reformulate a model while keeping invariant the requirements specification. A better understanding of such aspects as planner-domain matching, of model re-use, metrics and reformulation (combining the research directions in discussion points 1-7 above) will contribute to the aim of developing engineering methods that ensure the production of both a complete and operational knowledge model within a planning application.

A parallel and related development, and one that supports the flexibility of general planning agents, is in the automated learning of domain models. Here work on learning complete domain models (by learning domain physics) is complemented by a long history of learning knowledge to promote operationality (by learning domain specific heuristics). A more fruitful co-operation of research into the learning and engineering approaches would help accelerate developments in the field.

6. Conclusion

In this paper, we presented a review of KEPS tools and methods considering a hypothetical design process of domain knowledge models. Such a hypothetical design process was used as a framework to organize the work in the literature. We examined existing tools that support each phase of the design process. The main goal of this paper is to show the state-of-the-art techniques in KEPS and understand the progress in the field. Moreover, this paper aims at tightening up the nomenclature of the area and at giving some inputs to the AI Planning and Scheduling community for new research directions on supporting the creation of high quality knowledge models.

References

(2004). *LPG-TD: a fully automated planner for PDDL2.2 domains.*

(2008). *Additive and Reversed Relaxed Reachability Heuristics Revisited.*

(2008). *MIPS-XXL: Featuring External Shortest Path Search for Sequential Optimal Plans and External Branch-And-Bound for Optimal Net Benefit*, Proceedings of the 6th International Planning Competition Booklet (IPC 2008).

(2008). *The SGPlan Planning System in IPC-6.*

Ai-Chang, M., Bresina, J., Charest, L., Chase, A., jung Hsu, J. C., Jonsson, A., Kanefsky, B., Morris, P., Rajan, K., Yglesias, J., Chafin, B. G., Dias, W. C., & Maldague, P. F. (2004). MAPGEN: Mixed-Initiative Planning and Scheduling for the Mars Exploration Rover Mission. *IEEE Intelligent Systems*, *19*(1).

Alhossaini, M., & Beck, J. C. (2009). Learning Instance-Specific Macros. In *Proceedings of ICAPS 2009 Workshop on Planning and Learning. Thessaloniki, Greece.*

Alhossaini, M. A., & Beck, J. C. (2012). Macro Learning in Planning as Parameter Configuration. In Kosseim, L., & Inkpen, D. (Eds.), *Canadian Conference on Artificial Intelligence*, Vol. 7310 of *Lecture Notes in Computer Science*, pp. 13–24. Springer.

Amir, E. (2005). Learning Partially Observable Deterministic Action Models. In Kaelbling, L. P., & Saffiotti, A. (Eds.), *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1433–1439. Professional Book Center.

Bacchus, F., & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, *116*(1-2), 123–191.

Bäckström, C., & Nebel, B. (1995). Complexity Results for SAS+ Planning. *Computational Intelligence*, *11*, 625–656.

Barreiro, J., Boyce, M., Do, M., Frank, J., Iatauro, M., Kichkaylo, T., Ong, P. M. J., Remolina, E., Smith, T., & Smith, D. (2012). EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *Proceedings of the 4th International Competition on Knowledge Engineering for Planning and Scheduling, ICAPS 2012, Atibaia, Sao Paulo, Brazil.*

Barták, R., & Cepek, O. (2008). Nested Precedence Networks with Alternatives: Recognition, Tractability, and Models. In Dochev, D., Pistore, M., & Traverso, P. (Eds.), *AIMSA*, Vol. 5253 of *Lecture Notes in Computer Science*, pp. 235–246. Springer.

Barták, R., Jaška, M., Novák, L., Rovenský, V., Skalický, T., Cully, M., Sheahan, C., & Thanh-Tung, D. (2012a). FlowOpt: Bridging the Gap Between Optimization Technology and Manufacturing Planners. In *Proceedings of the 4th International Competition on Knowledge Engineering for Planning and Scheduling, ICAPS 2012, Atibaia, Sao Paulo, Brazil.*

Barták, R., Sheahan, C., & Sheahan, A. (2012b). MAKE – A System for Modelling, Optimising, and Analyzing Production in Small and Medium Enterprises. In *Proceedings of 38th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM). LNCS 7147*, pp. 600–611. Springer Verlag.

Bench-Capon, T. J. M., Coenen, F., Nwana, H. S., Paton, R., & Shave, M. J. R. (1993). Two Aspects of the Validation and Verification of Knowledge-Based Systems. *IEEE Expert*, *8*(3), 76–81.

Bender RBT Inc (2003). Systems Development Life Cycle: Objectives and Requirements..

Benson, S. (1995). Inductive Learning of Reactive Action Models. In Prieditis, A., & Russell, S. J. (Eds.), *Proceedings of 12th International Conference on Machine Learning*, pp. 47–54. Morgan Kaufmann.

Bernardini, S., & Smith, D. E. (2011). Finding mutual exclusion invariants in temporal planning domains. In *Proceedings of 7th International Workshop on Planning and Scheduling for Space (IWPSS).*

Bonasso, P., & 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)*, pp. 5–12, Toronto, Canada.

Botea, A., Enzenberger, M., Müller, M., & Schaeffer, J. (2005). Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. *Journal of Articial Intelligence Research*, *24*, 581–621.

Bouillet, E., Feblowitz, M., Liu, Z., Ranganathan, A., & Riabov, A. (2007). A Knowledge Engineering and Planning Framework based on OWL Ontologies. In *Proceedings of the Second International Competition on Knowledge Engineering*, Providence, Rhode Island, USA.

Bouillet, E., Feblowitz, M., Liu, Z., Ranganathan, A., & Riabov, A. (2008). A tag-based approach for the design and composition of information processing applications. In Harris, G. E. (Ed.), *Proceedings of the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications (OOPSLA)*, pp. 585–602. ACM.

Cesta, A., Finzi, A., Fratini, S., Orlandini, A., & 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). Sydney, Autralia.*

Chen, Y., Lv, Q., & Huang, W. R. (2008). Plan-A: A Cost Optimal Planner Based on SAT-Constrained Optimization. In *Proceedings of the Sixth International Planning Competition, International Conference on Automated Planning and Scheduling (ICAPS'08).*

Chrpa, L., McCluskey, T. L., & Osborne, H. (2012a). Optimizing Plans through Analysis of Action Dependencies and Independencies. In *Proceedings of the 22th International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, Sao Paulo, Brazil*, pp. 338–342.

Chrpa, L., McCluskey, T. L., & Osborne, H. (2012b). Reformulating Planning Problems: A Theoretical Point of View. In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference. AAAI Press.*

Coles, A. I., & Smith, A. J. (2007). Marvin: A Heuristic Search Planner with Online Macro-Action Learning. *Journal of Artificial Intelligence Research*, *28*, 119–156.

Coles, A. J., Coles, A. I., Fox, M., & Long, D. (2010). Forward-Chaining Partial-Order Planning. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10).*

Coles, A., Fox, M., Long, D., & Smith, A. (2008). A Hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pp. 52–59.

Coles, A., Fox, M., & Smith, A. (2007). Online Identification of Useful Macro-Actions for Planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007*, pp. 97–104.

Crawford, J., Ginsberg, M., Luks, E., & Roy, A. (1996). Symmetry-Breaking Predicates for Search Problems. In *Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 148–159, Cambridge, Massachusetts. Morgan Kaufmann.

Cresswell, S., McCluskey, T., & West, M. M. (2010). Acquiring planning domain models using LOCM. In *Knowledge Engineering Review*, pp. 1–18. Cambridge University Press.

Cresswell, S., Fox, M., & Long, D. (2002). Extending TIM domain analysis to handle ADL constructs. In *Knowledge Engineering Tools and Techniques for AI Planning: AIPS'02 workshop*, Toulouse, France.

Currie, K., & Tate, A. (1991). O-Plan: The open Planning Architecture. *Artificial Intelligence*, *52*(1), 49–86.

Daley, P., Frank, J., Iatauro, M., McGann, C., & Taylor, W. (2005). PlanWorks: A debugging environment for constraint based planning systems. In *Proceedings of the First International Competition on Knowledge Engineering for AI Planning, Monterey, Califormia, USA*.

de la Rosa, T., & McIlraith, S. A. (2011). Learning Domain Control Knowledge for TLPlan and Beyond. In *Proceedings of the ICAPS-11 Workshop on Planning and Learning (PAL)*.

Doherty, P., & Kvarnström, J. (2001). TALplanner: A Temporal Logic-Based Planner. *AI Magazine*, *22*(3), 95–102.

Edelkamp, S., & Hoffmann, J. (2004). PDDL 2.2: The Language for Classical Part of the 4th International Planning Competition. Tech. rep., Fachbereich Informatik and Institut für Informatik, Germany.

Edelkamp, S., & Mehler, T. (2005). Knowledge acquisition and knowledge engineering in the ModPlan workbench. In *Proceedings of the 1st International Competition on Knowledge Engineering for AI Planning, Monterey, California, USA*.

Edelkamp, S., Jabbar, S., & Nazih, M. (2006). Cost-Optimal Planning with Constraints and Preferences in Large State Spaces. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Preferences and Soft Constraints in Planning*, pp. 38–45. AAAI Press.

Eggert, R. (2010). *Engineering Design*. High Peak Press.

Ertas, A., & Jones, J. (1996). *Engineering Design Process* (2nd edition). John Wiley & Sons, Inc.

Fdez-Olivares, J., Cózar, J., & Castillo, L. (2009). OncoTheraper: Clinical Decision Support for Oncology Therapy Planning Based on Temporal Hierarchical Tasks Networks. In Riaño, D. (Ed.), *Knowledge Management for Health Care Procedures*, Lecture Notes in Computer Science.

Feblowitz, M. D., Ranganathan, A., Riabov, A. V., & Udrea, O. (2012). Knowledge Engineering for Planning-based Data-flow Composition. In *Proceedings of the 4th International Competition on Knowledge Engineering for Planning and Scheduling, ICAPS 2012, Atibaia, Sao Paulo, Brazil.*

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

Fikes, R., & Nilsson, N. J. (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In Cooper, D. C. (Ed.), *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 608–620. William Kaufmann.

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

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

Fox, M., & Long, D. (1999). The Detection and Exploitation of Symmetry in Planning Problems. In *Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 956–961, Stockholm, Sweden. Morgan Kaufmann.

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

Frank, J. (2008). An Intelligent Agent for Autonomous Lunar Exploration. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008) Scheduling and Planning Application Workshop. Sydney, Australia.*

Frank, J., & Jónsson, A. K. (2003). Constraint-Based Attribute and Interval Planning. *Journal of Constraints Special Issue on Constraints and Planning*, *8*(4), 339–364.

Fratini, S., Pecora, F., & Cesta, A. (2008). Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences*, *18*(2), 231–271.

Gaines, D. M., Estlin, T., Chouinard, C., Castano, R., Castano, A., Bornstein, B., Anderson, R. C., Judd, M., Nesnas, I., & Rabideau, G. (2006). Opportunistic Planning and Execution for Planetary Exploration. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006) Software Demostration. Cumbria, UK.*

Gamma, E., Helm, R., Johnson, R. E., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA.

Gerevini, A., & Long, D. (2006). Preferences and Soft Constraints in PDDL3. In Gerevini, A., & Long, D. (Eds.), *Proceedings of ICAPS Workshop on Planning with Preferences and Soft Constraints*, pp. 46–53.

Gerevini, A., Saetti, A., & Serina, I. (2006). An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research (JAIR)*, *25*(1), 187–231.

Gerevini, A., & Schubert, L. (2001). DISCOPLAN: An Efficient On-Line System for Computing Planning Domain Invariants. In *Proceedings of the Sixth European Conference on Planning (ECP-01)*, pp. 12–14.

Gerevini, A., & Schubert, L. (1998). Inferring State Constraints for Domain-Independent Planning. In *Proceedings of 15th National Conference on Artificial Intelligence*, pp. 905–912, Madison, USA. AAAI Press/MIT Press.

Gerevini, A., & Serina, I. (2002). LPG: A Planner Based on Local Search for Planning Graphs with Action Costs. In *Proceedings of the Conference on Artificial Intelligence Planning Systems (AIPS)*, pp. 13–22.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning: Theory and Practice.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Giuliano, M. E., & Johnston, M. D. (2010). Visualization Tools for Multi-Objective Scheduling Algorithms. In *Proceedings of ICAPS 2010 System Demostration*, pp. 11–14, Toronto, Canada.

González-Ferrer, A., Fernández-Olivares, J., Castillo, L., et al. (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 (KEPS), ICAPS 2009*, pp. 28–37, Thessaloniki, Greece.

Haas, W., & Havens, W. S. (2008). Generating Random Dynamic Resource Scheduling Problems. In *Workshop on Knowledge Engineering for Planning and Scheduling - ICAPS 2008*, Sydney, Australia.

Haslum, P., & Jonsson, P. (2000). Planning with Reduced Operator Sets. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems (AIPS)*, pp. 150–158, Breckenridge, CO. AAAI Press.

Hatzi, O., Meditskos, G., Vrakas, D., Bassiliades, N., Anagnostopoulos, D., & 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 (KEPS), ICAPS 2009*, pp. 38–45, Thessaloniki, Greece.

Hendler, J., Tate, A., & Drummond, M. (1990). AI planning: systems and techniques. *AI Magazine*, *11*, 61–77.

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

Hoffmann, J. (2003). The Metric-FF Planning System: Translating Ignoring Delete Lists to Numerical State Variables. *Journal of Artificial Intelligence Research (JAIR)*, *20*.

Hoffmann, J. (2011). Analyzing Search Topology Without Running Any Search: On the Connection Between Causal Graphs and h+. *Journal of Artificial Intelligence Research (JAIR)*, *41*, 155–229.

Hoffmann, J., & Nebel, B. (2001). The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR)*, *14*, 253–302.

Howey, R., Long, D., & 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*, pp. 294–301, Washington, DC, USA. IEEE Computer Society.

Hsu, C.-W., Wah, B., Huang, R., & Chen, Y. (2006). Handling Soft Constraints and Goals Preferences in SGPlan. In *Proceedings of the 5th International Planning Competition Booklet (IPC 2006)*, pp. 39–41.

Huang, Y.-C., Selman, B., & Kautz, H. A. (1999). Control Knowledge in Planning: Benefits and Tradeoffs. In Hendler, J., & Subramanian, D. (Eds.), *AAAI/IAAI*, pp. 511–517. AAAI Press / The MIT Press.

ICAPS (2003–2012). Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS). Available at http://www.aiconferences.org/ICAPS/icaps.html.

Jain, A., Guineau, J., Lim, C., Lincoln, W., Pomerantz, M., Sohl, G., & Steele, R. (2003). ROAMS: Planetary Surface Rover Simulation Environment. In *Proceedings of the International Symposium on Artificial Intelligence Robotics and Automation in Space, Nara, Japan.*

Jiménez, S., de la Rosa, T., Fernández, S., Fernández, F., & Borrajo, D. (2012). A review of machine learning for automated planning. *Knowledge Eng. Review*, *27*(4), 433–467.

Kautz, H., & Selman, B. (1998). BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. In *Working notes of the Workshop on Planning as Combinatorial Search, held in conjunction with AIPS-98, Pittsburgh, PA*, pp. 58–60.

Kautz, H. A., Selman, B., & Hoffmann, J. (2006). SatPlan: Planning as Satisfiability. In *The 5th International Planning Competition, 16th International Conference on Automated Planning and Scheduling (ICAPS-06), abstract booklet of the competing planners, Cumbria, UK.*

Klein, M. (1993). Capturing Design Rationale in Concurrent Engineering Teams. *Computer*, *26*(1), 39–47.

Kotonya, G., & Somerville, I. (1996). Requirements engineering with viewpoints..

Kotonya, G., & Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques.* John Wiley & Sons.

Krueger, C. W. (1992). Software reuse. *ACM Comput. Surv.*, *24*(2), 131–183.

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

McCluskey, T. L., & 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. Cumbria, UK.*

McCluskey, T. L., & Simpson, R. M. (2004). Knowledge Formulation for AI Planning. In *Knowledge Acquisition, Modeling and Management (EKAW)*, pp. 449–465.

McCluskey, T., Aler, R., Borrajo, D., Haslum, P., Jarvis, P., Refanidis, I., & Scholz, U. (2003). Knowledge Engineering for Planning Roadmap.. Available at http://scom.hud.ac.uk/planet/home/.

McCluskey, T., Cresswell, S., Richardson, N., & West, M. M. (2010). Action Knowledge Acquisition with Opmaker2. In Filipe, J., Fred, A., & Sharp, B. (Eds.), *Agents and Artificial Intelligence*, Vol. 67 of *Communications in Computer and Information Science*, pp. 137–150. Springer. International Conference, ICAART 2009, Porto, Portugal, January 19-21, 2009. Revised Selected Papers.

McDermott, D. (2000). The 1998 AI planning competition. *AI Magazine*, *20(2).*

McDermott, J. (1981). Domain knowledge and the design process. In *DAC '81: Proceedings of the 18th conference on Design automation*, pp. 580–588, Piscataway, NJ, USA. IEEE Press.

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

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, *77*(4), 541–580.

Muscettola, N., Nayak, P. P., Pell, B., & Williams, B. C. (1998). Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, *103*(1-2), 5–47.

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

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

Nakhost, H., & Muller, M. (2010). Action Elimination and Plan Neighborhood Graph Search: Two Algorithms for Plan Improvement. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada*.

Newton, M. A. H., Levine, J., Fox, M., & Long, D. (2007). Learning macro-actions for arbitrary planner and domains. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*.

OMG (2003). *UML 2.0 OCL Specification m Version 2.0*.

OMG (2005). *OMG Unified Modeling Language Specification, m Version 2.0*.

Preece, A. (2001). Evaluating Verification and Validation Methods in Knowledge Engineering. *Micro-Level Knowledge Management*, 123–145.

Ranganathan, A., Riabov, A., & Udrea, O. (2009). Mashup-based information retrieval for domain experts. In Cheung, D. W.-L., Song, I.-Y., Chu, W. W., Hu, X., & Lin, J. J. (Eds.), *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp. 711–720. ACM.

Reddy, S. Y., Ai-Chang, M., Iatauro, M. J., Kurklu, E., Boyce, M. E., Frank, J. D., & Jonsson, A. K. (2008). Planning and Monitoring Solar Array Operations on the ISS. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008) Scheduling and Planning Application Workshop. Sydney, Australia*.

Riabov, A., & Liu, Z. (2006). Scalable Planning for Distributed Stream Processing Systems. In Long, D., Smith, S. F., Borrajo, D., & McCluskey, L. (Eds.), *Proceeding of the 16th International Cconference on Automated Planning and Scheduling (ICAPS)*, pp. 31–41. AAAI.

Scholz, U. (1999). Action Constraints for Planning. In *In BIUNDO & FOX*, pp. 148–160, Berlin, Heidelberg. Springer Verlag.

Schreiber, G., Wielinga, B., & Breuker, J. (Eds.). (1993). *KADS: A Principled Approach to Knowledge-Based System Development*, Vol. 11 of *Knowledge Based Systems*. Academic Press, London.

Sette, F. M., Vaquero, T. S., Park, S. W., & Silva, J. R. (2008). Are Automated Planners up to Solve Real Problems?. In *Proceedings of the 17th World Congress The International Federation of Automatic Control (IFAC'08), Seoul, Korea*, pp. 15817–15824.

Shaw, M. L. G., & Gaines, B. R. (1996). Requirements Acquisition. *IEE Software Engineering Journal*, *11*(3), 149–165.

Silva, J. R., & Santos, E. A. (2003). Viewpoint Requirements Validation Based on Petri Nets. In *In Proceedings of the 17th Int. Conf. of Mechanical Engineering, Brazilian Mechanical Eng. Society*.

Simpson, R. M. (2005). GIPO Graphical Interface for Planning with Objects. In *Proceedings of the First International Competition on Knowledge Engineering for AI Planning, Monterey, Califormia, USA*.

Simpson, R. M. (2007). Structural Domain Definition using GIPO IV. In *Proceedings of the Second International Competition on Knowledge Engineering. Providence, Rhode Island, USA*.

Simpson, R. M., Mccluskey, T. L., Zhao, W., Aylett, R. S., & Doniat, C. (2001). An Integrated Graphical Tool to support Knowledge Engineering in AI Planning. In *In Proceedings of the 6th European Conference on Planning*.

Simpson, R. M., McCluskey, T. L., Long, D., & Fox, M. (2002). Generic Types as Design Patterns for Planning Domain Specification. In *Proceedings of AIPS'02 Workshop on Knowledge Engineering Tools and Techniques for AI Planning*, Toulouse, France.

Smith, D. E., Frank, J., & Cushing, W. (2008). The ANML Language. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008) Workshop on Knowledge Engineering for Planning and Scheduling. Sydney, Australia.*

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

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

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

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*, *25*(1-2), 161–197.

Tate, A., Drabble, B., & Dalton, J. (1996). O-Plan: a Knowledged-Based Planner and its Application to Logistics. In *Advanced Planning Technology ARPI*, pp. 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.

Udo, M., Vaquero, T. S., Silva, J. R., & Tonidandel, F. (2008). Lean Software Development Domain. In *Proceedings of ICAPS 2008 Scheduling and Planning Application workshop. Sydney, Australia.*

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, *11*(2), 93–155.

van Moll, J. H., Jacobs, J. C., Freimut, B., & Trienekens, J. J. M. (2002). The Importance of Life Cycle Modeling to Defect Detection and Prevention. In *Proceedings of the 10th International Workshop on Software Technology and Engineering Practice*, STEP '02, pp. 144–155, Washington, DC, USA. IEEE Computer Society.

Van Vliet, H. (2002). *Software Engineering: Principles and Practice* (2nd edition). John Wiley & Sons.

Vaquero, T. S., Silva, J. R., Tonidandel, F., & Beck, J. C. (2011). itSIMPLE: Towards an Integrated Design System for Real Planning Applications. In *To appear in: The Knowledge Engineering Review Journal, special issue on International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS): Cambridge University Press.*

Vaquero, T. S., Costa, G., Tonidandel, F., Igreja, H., Silva, J. R., & Beck, J. C. (2012). Planning and Scheduling Ship Operations on Petroleum Ports and Platforms. In *Proceedings of ICAPS 2012 Workshop on Scheduling and Planning Applications. Atibaia, Sao Pualo, Brazil*, pp. 8–16.

Vaquero, T. S., Romero, V., Tonidandel, F., & 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.*

Vaquero, T. S., Silva, J. R., & Beck, J. C. (2012). Post-Design Analysis for Building and Refining AI Planning Systems. In *Submitted to Journal of Engineering Applications of Artificial Intelligence.*

Vaquero, T. S., Silva, J. R., Beck, J. C., et al. (2010). Improving Planning Performance Through Post-Design Analysis. In *Proceedings of the ICAPS'10 Workshop on Knowledge Engineering for Planning and Scheduling. Toronto, Canada*, pp. 45–52.

Vaquero, T. S., Silva, J. R., Beck, J. C., et al. (2011). Acquisition and Reuse of Plan Evaluation Rationales on Post Design. In *Proceedings of the ICAPS'11 Workshop on Knowledge Engineering for Planning and Scheduling. Freiburg, Germany*, pp. 15–22.

Vaquero, T. S., Silva, J. R., Ferreira, M., Tonidandel, F., & Beck, J. C. (2009). From Requirements and Analysis to PDDL in itSIMPLE3.0. In *Proceedings of the Third ICKEPS, ICAPS 2009, Thessaloniki, Greece.*

Vaquero, T. S., Tonaco, R., Costa, G., Tonidandel, F., Silva, J. R., & Beck, J. C. (2012). itSIMPLE4.0: Enhancing the Modeling Experience of Planning Problems. In *Proceedings of ICAPS 2012 System Demonstration. Atibaia, Sao Pualo, Brazil*, pp. 11–14.

Vaquero, T. S., Tonidandel, F., Barros, L. N., & Silva, J. R. (2006). On the use of UML.P for modeling a real application as a planning problem. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 434–437.

Vodrazka, J., & Chrpa, L. (2010). Visual Design of Planning Domains. In *Proceedings of ICAPS 2010 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, pp. 68–69.

Wang, X. (1996). Planning While Learning Operators. In Drabble, B. (Ed.), *Proceedings of third International Conference on Artificial Intelligence Planning System*, pp. 229–236. AAAI.

Wickler, G. (2011). Using Planning Domain Features to Facilitate Knowledge Engineering. In *Proceedings of ICAPS 2011 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, pp. 39–46.

Wilkins, D. E. (1988). *Practical Planning*. Morgan Kaufman, Palo Alto, Ca., USA.

Wilkins, D. E., & Myers, K. L. (1995). A Common Knowledge Representation for Plan Generation and Reactive Execution. *Journal of Logic and Computation*, *5*(6), 731–761.

Xing, Z., Chen, Y., & Zhang, W. (2006). MaxPlan: Optimal Planning by Decomposed Satisfiability and Backward Reduction. In *Proceedings of the Fifth International Planning Competition, Int'l Conf. on Automated Planning and Scheduling (ICAPS'06)*, pp. 53–56.

Yoon, S., & Kambhampati, S. (2007). Towards model-lite planning: A proposal for learning & planning with incomplete domain models. In *Proceedings of ICAPS Workshop on AI Planning and Learning, ICAPS 2007*.

Zhuo, H. H., Yang, Q., Hu, D. H., & Li, L. (2010). Learning complex action models with quantifiers and logical implications. *Artificial Intelligence*, *174*(18), 1540–1569.