Improving Planning Performance Through Post-Design Analysis

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

Abstract

In this paper, we investigate how knowledge acquired during a plan analysis phase that follows model design affects planning performance. We describe a post-design framework that combines a knowledge engineering tool and a virtual prototyping environment for the analysis and simulation of plans. Our framework demonstrates that post-design analysis supports the discovery of missing requirements and guides the model refinement cycle. We present two case studies using benchmark domains and eight state-of-the-art planners. Our results demonstrate that significant improvements in plan quality and an increase in planning speed of up to three orders of magnitude can be achieved through a careful post-design process. We argue that such a process is critical for deployment of planning technology in real-world applications.

Introduction

Over the last decade, both research effort and industry interest have been directed towards the application of AI Planning techniques to solve real-life problems. As a result, it has become clear that the process of developing algorithms for synthesizing plans forms only one part of the complex design life cycle of a real-world planning application. Most of the problems identified as suitable to being solved with a planning approach are characterized by a need for substantial knowledge management, reasoning about actions and a careful consideration of quality metrics and criteria. The design process of real applications must have a strong commitment to these prerequisites in order to result in reliable, deployed planning systems.

Design decisions about knowledge modeling and planning algorithm development drastically affect the quality of plans. From a planning technology perspective, in a *ceteris paribus* scenario, factors such as the improper choice of planning techniques and heuristics may lead to the generation of poor quality solutions. From a knowledge engineering perspective, lack of knowledge, ill-defined requirements and inappropriate definition of quality metrics and preferences can contribute directly to malformed models and, consequently, to unsatisfactory plans, independent of the planning algorithm. Traditionally, much of planning research has focused on the former perspective, in which new algorithms are developed and tuned to obtain high performance and better plans. Not much investigation has been done on the knowledge engineering (KE) perspective, especially re-modeling the planning problem based on observations and information that emerge during the design process itself.

In plan analysis, hidden knowledge and requirements captured from human feedback raise the need for a continuous re-modeling process. The capture and use of such human-centered feedback is still an unexplored area in the knowledge engineering for AI planning. Moreover, the extent of impact of such feedback and re-modeling on the planning performance is unknown. In order to deal with such post-design analysis, techniques such as simulation, visualization and virtual prototyping, commonly used in other disciplines (Cecil and Kanchanapiboon 2007), can help design teams identify new requirements and inconsistencies in the model.

In this paper, we present a post-design tool for AI planning that combines the open-source KE tool itSIMPLE (Vaquero et al. 2007) and a virtual prototyping environment to support identification of inconsistencies and hidden requirements. We describe two case studies showing that post-design not only improves plan quality, but also improves planning performance even in benchmark problems. The main contributions of this work are:

1. The creation of a framework to support post-design analysis for planning;
2. Two case studies that demonstrate that improvements in plan quality, an increase in solvability and a reduction of planning time of up to three orders of magnitude can be achieved through a careful post-design process.

This paper is organized as follows. First, we discuss concepts in knowledge engineering for planning and their role in plan analysis and post-design. Then, we present the post-DAM project describing the integration of itSIMPLE and a virtual prototyping tool. Next, we present two case studies and the results. We conclude with a discussion of our results.

Knowledge Engineering and Post-Design

Requirements engineering (RE) and knowledge engineering (KE) principles have become important to the success of the design and maintenance of real world planning applications.

While pure AI planning research focuses on developing reliable planners, KE for planning research focuses on the design process for creating reliable models of real domains (McCluskey 2002; Vaquero et al. 2007). A well-structured life cycle to guide design increases the chances of building an appropriate planning application while reducing possible costs of fixing errors in the future. A simple design life cycle is feasible for the development of small prototype systems, but fails to produce large, knowledge-intense applications that are reliable and maintainable (Studer, Benjamins, and Fensel 1998).

Research on KE for planning and scheduling has created tools and techniques to support the design process of planning domain models (Vaquero et al. 2009b; Simpson 2007). However, given the natural incompleteness of the knowledge, practical experience in real applications such as space exploration (Jónsson 2009) has shown that, even with a disciplined process of design, requirements from different viewpoints (e.g. stakeholders, experts, users) still emerge after plan generation, analysis and execution. For example, the identification of unsatisfactory solutions and unbalanced trade-offs among different quality metrics and criteria (Jónsson 2009; Rabideau, Engelhardt, and Chien 2000; Cesta et al. 2008) indicates a lack of understanding of requirements and preferences in the model. These hidden requirements raise the need for iterative re-modeling and tuning process. In some applications, finding an agreement or a pattern among emerging requirements is an arduous task (Jónsson 2009), making re-modeling a non-trivial process.

A fundamental step in the modeling cycle is the analysis of generated plans with respect to the requirements and quality metrics. Plan analysis naturally leads to feedback and the discovery of hidden requirements for refining the model. We call '*post-design analysis*' the process performed after plan generation, in which we have a base model and a set of planners and investigate the solutions they generate. Some of the AI planning research on plan analysis has developed tools and techniques for plan animation (McCluskey and Simpson 2006; Vaquero et al. 2007), visualization (e.g. Gantt charts), and plan querying and summarization (Myers 2006). However, such work does not explore the effects of the missing knowledge and the re-modeling loop in the planning process. The investigation of modern analysis techniques such as simulation for planning it is still an emerging field.

The postDAM Project

The postDAM project aims to investigate post-design techniques to enhance the modeling cycle and increase the quality of plans. The project focuses on combining some of the recently developed tools in KE for planning with virtual prototyping. Virtual prototyping is commonly used in other engineering fields (e.g. mechanical engineering) to validate models and identify missing requirements (Cecil and Kanchanapiboon 2007) where producing a real prototype is impractical and costly.

The project proposes a framework that integrates the KE tool, itSIMPLE (Vaquero et al. 2009b), and the 3D content creation environment, Blender[1], for virtual prototyping. The former is a robust design system dedicated to AI planning in which a set of languages and validation engines are used to create domain models in a disciplined design process. The latter is an open source tool, widely used for creating games and animations. Blender provides several mechanisms for the definition and simulation of 3D elements, including their physical properties (such as mass, collision, gravity, inertia, velocity, strength, and sound effects), to mimic some real world characteristics.

The integration of these tools aims to close the design loop, from requirements acquisition to plan analysis in the post-design. In this loop, itSIMPLE is responsible for supporting users during design and re-design of the models while Blender, properly integrated with the KE tool, encompasses the simulation of plans provided by planners.

During model construction in itSIMPLE, designers perform the initial design phases to develop domain models in the *Unified Modeling Language*. An important step in this design process is the identification and specification of quality metrics, along with their respective importance. These quality metrics are characterized in the form of weighted domain or plan variables, i.e., numeric variables that are directly (or indirectly) related to the quality of plans. Examples of domain and plan variables are: the number of occurrences of a specific action in a plan; the total consumption of fuel; the number of robots used for a particular purpose; and the energy remaining in a battery. The definition of quality metrics in itSIMPLE uses the approach described in (Rabideau, Engelhardt, and Chien 2000). During simulation, the designer can analyze and evaluate different solutions while contrasting them based on the quality metrics. Different viewpoints can communicate during 3D visualization in order to validate and adjust the model based on their impressions. Figure 1 illustrates this iterative refinement process.

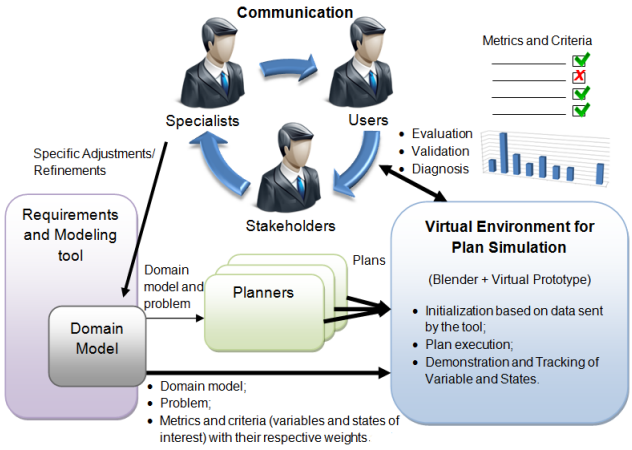

Figure 1: The post-design framework

In order to provide an integrated design iteration, a communication channel between itSIMPLE and Blender was developed in which data is sent from the KE tool to the 3D

[1]Blender, available at www.blender.org

environment. The data sent by itSIMPLE consist of the domain model, the problem instance and the quality metrics to be considered (all in an XML representation (Vaquero et al. 2009b)). Since users can run several state-of-the-art planner from the itSIMPLE's GUI, the generated plans are sent directly to the 3D simulator.

Blender reads the data from itSIMPLE and generates a *virtual prototype* of the model based on a predefined library of graphical objects and their physics. These objects are designed in such a way that they can perform and react to the actions defined in the model from itSIMPLE. The Blender application reads the main elements of the domain and problem instance such as classes of objects, the objects and their properties, and additional information regarding the graphical position of the elements that has been stated by the user. Classes are used to identify the necessary graphical elements from the predefined library, while objects, properties and location information are used to instantiate and initiate the graphical elements in the initial scene. All the elements in the problem instance definition are found in the 3D representation. Having the initial state established in the 3D scene, the plan provided by a planner is then simulated (we assume that plan actions are deterministic). In the simulation, the actions are sent to each involved object, step-by-step. Each object is implemented to act based on the instructions that it is given. In each step of the simulation, the values of the metrics are stored to provide a clear view of their changes over the plan.

At the end of simulation, Blender 3D produces a plan report that can be analyzed by users. The report contains the evolution of the chosen quality metrics along with the cost of the plan.

Case Studies

In this section, we present two case studies using benchmark domains from the *International Planning Competitions* (IPC). The domains are the Gold Miner domain from IPC-6 and the Storage domain from IPC-5. Both were chosen from recent competitions based on the clear correspondence between objects in the real and virtual world.

The procedure used for each case study is as follows:

1. We created an initial model in itSIMPLE guided by the original PDDL representation to simulate the design process. Since itSIMPLE generates PDDL output as a communication language to planners, we verify that such output is exactly the same as the original PDDL version of the benchmark domain. This model is called $Original$.
2. We selected three problem instances from the 30 IPC instances to be analyzed in-depth. The selected set of problem instances is called the *design set*. Eight planners were chosen to be run (using default arguments) with a 20-minute time-out for each problem instance.
3. In addition to the PDDL reproduction process, we used itSIMPLE to define quality metrics for the domain.
4. Using the virtual prototype in Blender, we studied every generated plan and its execution. With the analysis and plan reports, we manually introduced modifications to the model in itSIMPLE. We repeated the plan simulation and model refinement, going back and forth with new ideas and results, until we had two new models (A and B) each representing one major change and a third new model (AB) incorporating them both.
5. We then took the remaining 27 problem instances and tested all four models ($Original$, A, B, and AB) with the eight planners. We call this set of instances the *testing set*.

In order to analyze the impact of the refinement cycle, we compare the models A, B and AB to the $Original$ over all 27 problem instances from the testing set. This analysis considers the changes on plan quality, plan length, speed, and solvability. PDDL terms and elements are used to describe the adjustments made to the original model in the remodeling process to facilitate the explanation.

The Gold Miner Domain

The Gold Miner is a benchmark domain from the learning track of IPC-6 (2008). In this domain, a robot is in a mine and has the objective of reaching a location that contains gold. The mine is represented as a grid in which each cell contains either hard or soft rock. There is a special location where the robot can either pickup an unlimited supply of bombs or pickup a single laser cannon. The laser cannon can be used to destroy both hard and soft rock, whereas the bomb can only penetrate soft rock. If the laser is used to destroy a rock that is covering the gold, the gold will also be destroyed. However, a bomb will not destroy the gold, just the rock. This particular domain has a simple optimal strategy[2] in which the robot must (1) get the laser, (2) shoot through the rocks (either soft of hard) until it reaches a cell neighboring the gold, (3) go back to get a bomb, (4) explode the rock at the gold location, and (5) pickup the gold. In this case study we used the propositional typed PDDL model from the testing phase of IPC-6.

For our design set, we chose three problem instances considering the variety of the number of objects and difficulty. The first (gold-miner-target-5x5-02) and the second (gold-miner-target-5x5-01) instances have a 5x5 mine with distinct positions of gold, bombs, cannon, and soft and hard rocks. The third instance (gold-miner-target-6x6-05) has a 6x6 mine also with a particular position of the domain elements.

The quality metrics chosen for this study are (1) the travel distance of the robot (weight 2), (2) the bomb usage (weight 1), and (3) the laser cannon usage (weight 1). In itSIMPLE, we specified these metrics as counters of the actions *move*, *detonatebomb*, and *firelaser*, respectively. For this domain we selected SGPlan5, MIPS-xxl 2006, LPG-td, MIPS-xxl 2008, SGPlan6, Metric-FF, LPG 1.2, and hspsp to solve the problem instances.

During the first post-design analysis with the original model and the design set, we carefully investigated all 24 generated plans through the 3D simulation using Blender. Figure 2 shows an example of the simulation.

[2]IPC-6 2008. http://eecs.oregonstate.edu/ipc-learn/

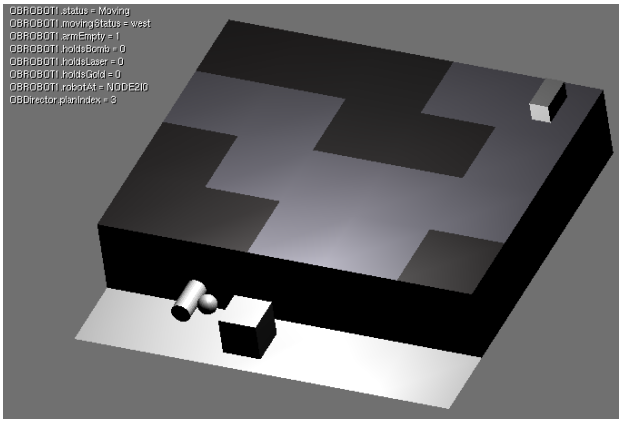

Figure 2: Virtual prototype and simulation of the Gold Miner domain. The robot is represented as a cube in the bottom. Soft and hard rocks are light and dark gray areas respectively. We used basic shapes to represent objects, however, there is no restriction on using complex 3D shapes and skins.

A number of observations were made in the first analysis:

- One planner generated invalid solutions in which the robot used the laser at the gold location, destroying the gold.[3]
- Some planners provided (valid) plans in which the laser cannon was fired at an already clear location.
- Unnecessary move actions were present in some plans.

In order to fix these non-optimal and flawed behaviors, we refined the original model. Concerning the planner assigning the robot to fire at the gold, the original model does not prevent such situation: there was no precondition on the *firelaser* operator that explicitly constrains this behavior. Therefore, a precondition to the operator was added: *(not (gold-at ?loc))*. Regarding the unnecessary *firelaser* occurrences, a second precondition was added to the same *firelaser* operator, in this case *(not (clear ?loc))*. We call this set of modifications A. The resulting model was sent to the planners to solve the same problem instances, resulting in a new post-design iteration.

During the second analysis process, additional observation were collected:

- Invalid plans were no longer being generated.
- The undesirable firing behavior from the initial observation was eliminated.
- In most of the plans, at the goal state, the laser cannon was left in a different position from the initial one. As a new requirement, the robot could leave the laser only at the same spot as the bomb source.
- Unnecessary move actions were still being found in some solutions.

These new observations guided the second re-modeling loop. A precondition to the *putdownlaser* operator was added, forcing the robot to always drop the cannon at the location of the bombs. The precondition *(bomb-at ?loc)* was used for this purpose. We call this modification B. The plans generated with model B properly controlled the location where the cannon was left. The combination of the previous and the current set of adjustments is called AB.

As a result of the AB modification, most of the generated plans to the design set converged to the same solution. The main issues raised during post-design analysis were eliminated, except some unneeded move actions.

To analyze and compare the effects of the post-design analysis on planning performance, the $Original$ model was compared to the models A, B, and AB. The selected planners were run on the testing set for each of the four models. Table 1 illustrates the comparison concerning run-time (speed) and solvability on the testing set. The table shows the total time (including time-outs) for each planner to solve all 27 problem instances in each of the four models. In order to determine the speed-up values, we first define $t^M_{p,k}$ as the time planner p takes to solve problem instance k using model M. We then define the speed-up ratio for each of the new models (A, B, AB) compared to the $Original$ model as follows:

$$r^M_{p,k} = \frac{t^{Original}_{p,k}}{t^M_{p,k}}. \quad (1)$$

For a particular planner and model, we calculate the mean and the median of the speed-up ratios. The mean speed-up value presented in Table 1 for each model is the mean of means of speed-up ratios over all planners. Similarly, the median speed-up is the median of the medians of speed-up ratios over all problem instances and planners for each model. Model AB shows a significant speed-up compared to the $Original$. Even A and B individually provide a significant speed-up. The maximum speed-up ratios observed on an individual problem instance were 7,547 with model A, 5,068 with model B and 5,185 with model AB. However, we also observed that in some cases the new models were slower than the original. The lowest ratio observed was 0.26 with model B. Considering the total time to solve the testing set, all planners perform better in the AB model. Because MIPS-xxl 2008 was run improperly it is not considered in the analysis showed in Table 1.

Table 1 also illustrates the number of instances solved by the planners in each model, as well as the percentage improvement of each new model compared to the original. All problem instances were solved in the AB model, a 22.7% improvement compared to the original model.

Improvements are not only found on speed and solvability, but also on plan length and quality. By looking at each problem instance and the four plans generated by a particular planner, we can determine the model that gives the best performance on three criteria: run time, plan length and plan quality (cost). Table 2 illustrates the number of times each

[3]This was observed with MIPS-xxl 2008. We contacted the authors and the bug appears to arise from the way we called the planner (e.g. a special script is necessary) but due to time limitations, we were unable to re-run these instances with the correct script.

Time (s)					Problems Solved				
Planners	Original	A	B	AB	Planners	Original	A	B	AB
SGPlan5	3,303.42	1.93	3.65	1.99	SGPlan5	14	27	27	27
MIPS-xxl 06	3,995.34	983.79	2,015.28	22.13	MIPS-xxl 06	15	25	21	27
LPG-td	41.04	29.69	39.28	27.83	LPG-td	27	27	27	27
SGPlan6	3,925.85	3.13	4.65	3.51	SGPlan6	18	27	27	27
Metric-FF	1,707.00	3.79	4.35	3.78	Metric-FF	26	27	27	27
LPG 1.2	10.58	5.58	4.47	4.70	LPG 1.2	27	27	27	27
hspsp	37.53	16.77	8.22	8.19	hspsp	27	27	27	27
Mean Speed-Up		469.70	267.97	479.67	Total	154	187	183	189
Median Speed-Up		7.33	6.82	10.54	Improvement		21.4%	18.8%	22.7%

Table 1: Total time (including time-outs) required to solve all problem instances and the solvability comparison for the Gold Miner domain models.

	Best Time Occurrence				Best Plan Length Occurrence				Best Plan Quality Occurrence			
Planners	Original	A	B	AB	Original	A	B	AB	Original	A	B	AB
SGPlan5	0	10	5	13	14	13	24	13	14	17	24	17
MIPS-xxl 06	0	4	0	23	11	19	21	25	11	19	21	25
LPG-td	2	10	2	13	5	24	10	27	5	24	10	27
SGPlan6	0	19	2	7	17	19	24	19	17	20	24	20
Metric-FF	0	9	9	10	26	13	23	13	26	17	23	17
LPG 1.2	3	3	10	12	12	15	18	12	15	17	19	12
hspsp	0	0	19	8	27	27	27	27	27	27	27	27
Total	5	55	47	86	112	130	147	136	115	141	148	145

Table 2: Best time, plan length and plan quality comparison over the Gold Miner models.

model results in the best solution with respect to each of the three criteria for a given planner. For example, LPG-td generated the best plan lengths compared to the other models using LPG-td in 5 cases with the $Original$ model, 24 with model A, 10 with model B, and 27 with model AB. The numbers sum to greater than 27 due to ties. Better plans are usually found with refined models.

The Storage Domain

The Storage domain is one of the benchmark domains from the deterministic track of IPC-5 (2006). This domain involves moving a certain number of crates from containers to depots using hoists. Inside a depot, each hoist can move according to a specified spatial map connecting different areas of the depot, represented by a grid. Transit areas are used to connect depot areas to containers and also depots to depots. The domain has five actions: (1) lifting a crate with a hoist; (2) dropping a crate from a hoist; (3) moving a hoist into a depot; (4) moving a hoist from one area of a depot to another; and (5) moving a hoist outside a depot. At the beginning of each problem, all crates are inside the containers waiting to be transported to the necessary depot. For this case study we used the propositional version of the PDDL domain model.

The design set for this domain is composed of three problem instances with different numbers of elements and difficulty. In the first instance (p10), four crates must be allocated in one depot using a single hoist. In the second (p16), six crates must be carried from two containers into two depots by three available hoists. The third instance (p20) has ten crates stored in three containers, three depots, and three hoists.

The quality metrics specified for this domain are the numbers of occurrence of each operator in the plan: move (weight 2), lift (weight 1), drop (weight 1), go-out (weight 3); and go-in (weight 3). The weights used in this experiment were inspired by the PDDL numeric version of the domain. We selected the same planners used in the previous case study, except LPG 1.2 which was removed due to a bug identified while running the design set. Instead we used FF 2.3.

During the first post-design iteration, the plans for the design set were analyzed in the virtual prototype platform. Figure 3 shows the simulation of the first problem instance from the design set.

The main observation raised while analyzing the plans provided by planners were:

- In some solutions, the hoists were putting the crates back into the containers. This scenario happens mainly when hoists left other crates on the main access to the depots, blocking access. Such situation forced the hoist into a lift-drop loop in and out of the containers.
- The fact that hoists drop crates on the doorway of depots forced them to rearrange the crates, which culminated in unnecessary actions to correct the previous decisions.
- Some of the plans included unnecessary lifts and drops of the same crate. This lift-drop loop happened usually in

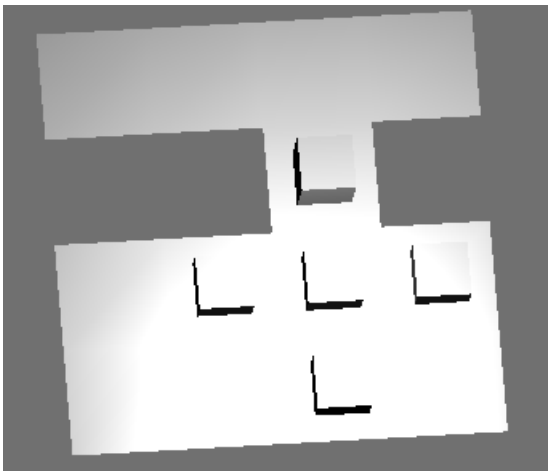

Figure 3: Virtual prototype and simulation of the Storage domain. Three crates at the depot and a hoist on the transit area.

transit and depot areas.

- Some solutions contained unnecessary move actions.

We focused on the fact that crates were being placed back into the containers. In order to constrain that situation, a simple modification of the parameters of operator *drop* was made. The original operator has the following PDDL representation *(?h - hoist ?c - crate ?a1 - storearea ?a2 - area **?p - place**)*. The adjusted version of the *drop* operator differs from the original in the last parameter: ***?p - depot***. The parameter *p* constrains the location where crates can be dropped. We call this modification A.

During the second iteration of simulation with adjustment A, the following observation were acquired:

- Crates were no longer being put back into the containers.
- Some of the plans still contained unnecessary lifts and drops of the same crate, but fewer than the original model.
- The unneeded move actions still occurs in some plans.

In the second re-modeling iteration, we focused on the unnecessary lift and drop cycles of the same crate. The approach taken for this problem was to make the hoist memorize the last lifted crate. For that, a new predicate was specified *(lastLifted ?h - hoist ?c - crate)*. This predicate was added to the precondition of the *lift* operator in the following form *(not (lastLifted ?h - hoist ?c - crate))*, as well as in the post-condition as *(lastLifted ?h - hoist ?c - crate)* to record the current crate. The precondition constrained the planner not to assign a hoist to lift the same crate again. The added post-condition defines the previously lifted crate. It is important to note that this is just one approach to attempt to tackle the issue. More elaborate re-modeling could be performed. We call this individual modification B. The combination of the refinements A and B generated the model AB. Note that A, B and AB do not require modification of the problem instances, even with the new predicate in B, since the hoists start with no previous lifted crates.

By analyzing the model AB in the post-design framework, the unnecessary lift-drop loops of the same crate were reduced drastically. However, some plans exhibited the problem in a different form: two hoists alternatively lift and drop the same crate. The occurrence of such situations was rare. The unnecessary move actions were reduced but not eliminated.

Table 3 shows the impact on run-time and solvability when running the selected planners over the testing set with the original and the new models. A and B are again analyzed separately for a better view of individual effects. In this table, the speed-up is not as impressive as the first case study. The maximum speed-up ratios observed for a single problem instance were 4,411 with model A, 1,597 with model B and 5,194 with model AB. The minimal ratio observed was 0.01 with model B. We do not have a significant improvement in the solvability with the new models. The highest one is the AB with a 9.4% improvement on the total number of problems solved. As above, the results of MIPS-xxl 2008 are not included due to the use of an incorrect script.

Table 4 illustrates in which model each planner provides it best run time, plan length and plan quality. For example, SGPlan6 provided the best plan lengths on 4 problem instances with the original model; 10 with model A, 17 with model B, and 18 with model AB. Similarly to the previous case study, the table shows that better plans are found more often using the refined models.

Discussion

In this section we present some of the main discussions raised by the case studies.

Knowledge Acquisition and Extraction

The case studies presented above demonstrate that even in benchmark domains missing requirements and modeling issues emerge in the post-design analysis. In real planning applications, we expect such gaps to be very common due to the difficulties of obtaining the necessary knowledge and requirements. The knowledge acquisition process in real-world applications is not the pure collection of already existing requirements during the beginning an application design (Studer, Benjamins, and Fensel 1998). Tacit knowledge and hidden and unknown requirements must be discovered and considered. Therefore, knowledge must be built up and structured during an iterative design process, especially during the initial phases and after design. Domain modeling is an iterative process in which new observations may lead to a refinement of the already built-up model (Studer, Benjamins, and Fensel 1998), even over time; moreover, the model itself may guide the further acquisition of knowledge.

In this work, both the KE tool and the 3D simulation environment have an important role in the discovery of missing requirements and the refinement cycle. The use of virtual prototyping, in particular, has shown to be a powerful technique on plan validation and new requirements identification as opposed to looking at plan traces. Visual and sound effects can give experts and non-experts a clear view of the domain model as well as the planning strategy. The

Time (s)					Problems Solved				
Planners	Original	A	B	AB	Planners	Original	A	B	AB
SGPlan5	11,562.25	10,137.26	7,686.23	7,470.18	SGPlan5	18	19	21	21
MIPS-xxl 06	3,433.82	3,334.95	3,913.46	3,347.97	MIPS-xxl 06	16	16	14	16
LPG-td	5,980.77	7,124.50	5,796.66	4,775.16	LPG-td	25	26	25	26
SGPlan6	8,230.74	10,177.03	7,754.42	7,508.96	SGPlan6	18	19	21	21
Metric-FF	13,441.49	9,767.35	13,205.63	12,003.99	Metric-FF	16	19	16	17
FF 2.3	14,416.93	9,935.53	15,603.44	12,117.64	FF 2.3	15	19	16	17
hspsp	21,782.20	21,744.39	21,117.63	20,989.91	hspsp	9	9	10	10
Speed-Up mean		37.22	13.03	43.49	Total	117	127	123	128
Speed-Up median		1.15	1.00	1.04	Improvement		8.5%	5.1%	9.4%

Table 3: Total time (including time-outs) required to solve all problem instances and the solvability comparison for the Storage domain models.

	Best Time Occurrence				Best Plan Length Occurrence				Best Plan Quality Occurrence			
Planners	Original	A	B	AB	Original	A	B	AB	Original	A	B	AB
SGPlan5	2	8	3	8	13	13	16	16	13	13	16	16
MIPS-xxl 06	2	7	1	6	13	14	12	13	10	13	13	13
LPG-td	5	6	4	11	9	9	17	18	9	9	15	18
SGPlan6	0	11	2	9	4	10	17	18	4	10	17	18
Metric-FF	1	10	3	9	15	16	10	12	14	17	10	12
LPG 1.2	2	12	1	5	12	15	14	17	13	15	14	16
hspsp	1	4	1	4	9	9	10	10	9	9	10	10
Total	13	58	15	52	75	86	96	104	72	86	95	103

Table 4: Best time, plan length and plan quality comparison on the Storage domain.

KE tool was also essential in the process, especially in the re-modeling phases. A metric-focused analysis, using for example the plan report, helps the designer to determine the subset of high quality solutions as well as the proper set of quality criteria.

Another important factor on discovering a lack of knowledge in the model and hidden requirements is the presence of different levels of quality over the analyzed plans. The identification of bad plans, for example, proved to be a powerful guidance on the re-modeling process. Bad plans not only raise the need for new constraints on the model, but also help designers to capture user's feedback and preferences. In our case studies, the generation of distinct plan qualities was enhanced by using a variety of planners. Since the lack of knowledge in the model can impact differently on the planners, their different responses also contribute to the identification of model issues. After the adjustment process these different responses are narrowed as many of the plans converge to the same solution over different planners.

We observed that planners can be very sensitive to the presence or absence of specific knowledge in the model. As an example, in the Gold Miner domain, the adjustment cycle made some of the planners perform impressively better; however, in the Storage domain, the addition of knowledge negatively affected the planners' internal heuristics. In fact, adding missing constraints not necessarily implies in faster responses from the planners; however even with a higher run-time we are moving toward better plans. These facts suggest that IPC results could be different if such issues were considered.

Modeling and Planning

The case studies showed that the planning performance indeed improved with a post-design analysis. We achieved speed-ups through a careful plan analysis and re-modeling process, without changing or adjusting planners. In some cases we have added obvious knowledge, from a human perspective, to the model; however, its explicit representation facilitates the search process of the planners. This evidence reinforces that both aspects, model and planner, must be carefully designed and refined. A sole emphasis on improving planners, neglecting observations and feedback from the design process itself, can *de facto* prevent or constrain planning use in real applications.

We believe that our results represent a challenge for practical planning research. The central justification for building general-purpose planners is that domain experts cannot be expected to also be planning experts. Domain experts should be able to concentrate on modeling the domain, treating the solver as a black-box. In practice, therefore, the *only* options available to a domain expert are domain and problem re-modeling. The use of planning technology by someone who is not a planning expert therefore depends entirely on the extent to which domains can be modeled (and remodeled) to allow planning algorithms to achieve satisfactory performance. Yet, we know very little about how domain modifications affect planning algorithms and we can provide little advice to domain experts (without becoming domain

experts ourselves) on what changes are likely to be positive.[4] Tools, such as the one presented here, to allow domain experts to investigate changes and planning experts to begin to develop an understanding of the impact of changes on their algorithms are therefore critical.

There is a fundamental mismatch between the target domains of the postDAM framework (i.e., real-world planning applications) and the case studies using IPC domains. A true test of our tool should be in the form of a case study with a real problem (e.g., (Cesta et al. 2008; Jónsson 2009; Vaquero et al. 2009a)). However, such case studies are not reproducible and often rely for success on external factors: significant interest from the client, success of other parts of the mission (e.g., landing on Mars), and larger economic forces. System building research is extremely valuable but sometimes inaccessible and difficult to generalize. Research in AI planning has shifted toward a more empirical style since the beginning of the IPC, where research innovations can be reproduced and directly compared. This style, too, has substantial benefits as can be observed from the gains in solver performance. The design of our case studies was an attempt to bridge the gap between "real" applications and "academic" benchmarks and to encourage further research on modeling in planning. We have shown that even in benchmark domains that, by definition, do not include a wealth of unrepresented knowledge, it is still possible to substantially increase solver performance by domain re-modeling.

Conclusion

In this paper, we have described a post-design framework to assist the discovery of missing requirements and to guide the model refinement cycle. We have demonstrated that following a careful post-design analysis, we can improve not only plan quality but also solvability and planner speed. The modifications made through the observations acquired during post-design resulted in impressive speed-up of state-of-the-art planners. In a real planning application, the analysis process that follows design becomes essential for having the necessary knowledge represented in the model. Post-design analysis is critical for deployment of planning technology in real-world applications.

References

Cecil, J., and Kanchanapiboon, A. 2007. Virtual engineering approaches in product and process design. *The International Journal of Advanced Manufacturing Technology* 31(9-10):846–856.

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

Jónsson, A. K. 2009. Practical planning. In *ICAPS 2009 Practical Planning & Scheduling Tutorial. Greece, Thessaloniki.* Available at: http://icaps09.uom.gr/tutorials/tutorials.htm.

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

McCluskey, T. L. 2002. Knowledge engineering: Issues for the AI planning community. *Proceedings of the AIPS-2002 Workshop on Knowledge Engineering Tools and Techniques for AI Planning, Toulouse, France*.

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.

Rabideau, G.; Engelhardt, B.; and Chien, S. 2000. Using generic preferences to incrementally improve plan quality. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Breckenridge, CO.*

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

Studer, R.; Benjamins, V. R.; and Fensel, D. 1998. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering* 25:161–197.

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

Vaquero, T. S.; Sette, F.; Silva, J. R.; and Beck, J. C. 2009a. Planning and scheduling of crude oil distribution in a petroleum plant. In *Proceedings of ICAPS 2009 Scheduling and Planning Application woRKshop. Thessaloniki, Greece.*

Vaquero, T. S.; Silva, J. R.; Ferreira, M.; Tonidandel, F.; and Beck, J. C. 2009b. From requirements and analysis to PDDL in itSIMPLE3.0. In *Proceedings of the Third ICKEPS, ICAPS 2009, Thessaloniki, Greece*.

[4] We believe that the fact that the planners used in our experiments all tended to improve their performance is an indication that such advice may exist.