Requirements and Work Domain Analysis in Automated Planning Systems

Rosimarci Tonaco Basbaum
Department of Mechatronic Engineering, University of São Paulo, Brazil
rosimarci@usp.br

On the intelligent design field the requirement analysis phase has a fundamental role - especially for real life systems - because it has the ability to identify or redesign variables which can potentially increase the model accuracy generated by the automated planner. A great efort has been made today in the area of Artificial Intelligence for defining reliable automated planning systems that can be applied in real life applications. That leads to the need of a systematic design process, in which the initial phases are not neglected and where Knowledge and Requirement Engineering tools have a fundamental role for supporting designers. Following this principle, this paper presents a method which implements a KE integrated environment where designers can perform knowledge acquisition, domain modelling, requirement analysis, model testing, maintenance by using diferent well-known languages such as UML, Petri Nets, PDDL and XML, each one of them with its best contribution. This paper aims to create a better and disciplined way to design planning applications using UML and Petri nets, using itSIMPLE framework, are quite promising.

Introduction

Planning characterizes a specific kind of design problem where the purpose is to find a set of admissible actions to solve a problem. The current approaches in the literature aim to improve the performance of the planner trying to optimize the search algorithms and the general solution. Besides that most of the proposals applied as a proof of concept to sintetized problems - closed problems that have a limited set of actions, like the Blocks World-Arm. Due to the extensive development in this area some authors started to apply planning techniques on real life problems - like localization, trajectory corrections and logistics - where the numer of variables and actions are high, and this increases the model's complexity .

It is important when dealing with real life problems to have a design life cycle - a defined sequence of processes that can address the designer to create a more adherent model. UML combined with Petri nets offering a disciplined design process, providing a visual tool - UML - and a validation tool - Petri nets - to analyse the models.

The design process has two major parts: the elicitation and documentation phase, using UML (OMG 2004) as the modeling language and the high lvel Petri nets to derive the UML diagrams. Throught the Petri net the requirement analysis will be performed using the available techniques, such as invariant analysis and equation state matrix (Murata 1989). The first step detects contradictions and conflicts between the requirements, or the diferent view points in the diagrams. The second step identifies deep inconsistencies, that are hidden in the dinamic perspective of the plans, as well as additional information about the model that could be interpreted for the planners. Such informaton includes, new constraints, invariants, partial solution strategies, characteristics that could help to improve the planner performance.

The tool we choosed to use is the itSIMPLE framework (Vaquero et al. 2007b), which currently is not a 100% ready to performe the UML/Petri net translation, but it will be during the devopment of this project. In the section 2 it will be discussed the advantages os UML in automated planning, followed by a brief description of Petri Nets, after that it will be presented a study case, the results and discussions and the conclusion.

The Problem

Applications that deal with real life problems became a must over the recent developments in the Automated Planning field. In general, the major focus of the planning community is, the pursuit the planners efficiency, and neglecting the analysis aspects. (Zimmerman and Kambhampati 2003) (Upal 2005).

To conduct the planning of an activity it is necessary to determine all features of the system in which it is embedded. Some factors must be considered, for exemple the sub systems envolved, internal variables, correlations other systems, constants and constraints. Such specification is called system modeling, and from it depends the success of the result obtained from the planning process. In this aspect several points becomes important, such as the proposed model complexity, and its accuracy from the original real life system.

In the design procces languages such as the traditional PDDL (McDermott 2003), or the UML (OMG 2009) are

used. To help in the design and the requirement analysis phases, there are frameworks available as itSIMPLE (Vaquero, Tonidandel, and Silva 2005) (Vaquero et al. 2007a), that focus on the initial design process phases, such as specification and modeling. After design is concluded, the output model is processed by tools called planners, that can proposes a set of actions (Ghallab, M.; Nau, D.; Traveso 2004).

To design real life systems,there are two kye challenges: 1) create a design discipline for modeling real life systems, using UML as the representation language; 2) translate and synthesis of the UML diagrams in a unique high level Petri net, which will be analyzed in order to obtain information that can anticipate problems in the model helping in the design phase the generation of suitable plans.

Objectives

The general purpose of this paper is to come up with a design process for automated planning systems, that is composed of two layers: one where independent domain methods are applied. The another layer uses the specific knowledge and requirements analysis applying high level Petri nets to increase the quality of planning problems solutions in Artificial Intelligence. The challenge here is to discover where to insert the specific knowledge and how to include this in the design process since we are working with three classes of problems where the specific knowledge level increases from benchmark to real life problems. The classes are: benchmarks, intermediates (like ROADEFs) and real life problems (like PetrobrÃ¡s problem presented in IKEPS 2012).

In the first case the purpose is to follow the original work of (Hoffmann 2003a), where is proposed the problem structure concept used in this paper, and from that to suggest a formalization of this structure based on high level Petri nets. The properties will serve to analyze the similarities, repetitive cycles, invariants and other properties between the models.

In the second level the specific knowledge can be included as dynamic relationships and actions properties, that will be inserted im UML diagrams and translated in a Petri net (therefore having a format compatible with the previous phase). The structure receives the dependent-domain knowledge, to apply the analysis techniques of high level Petri nets. These techniques are particularly sensitive whem applied in real life problems, requiring a different approach from the academic applications. Real life systems must follow a very disciplined design process, grounded in Knowledge Engineering, whose initial stage is composed by elicitation and requirement analysis. Such design process is the study focus of many researchers in Automated Planning area (McCluskey et al. 2003b).

Since our focus are on intermediate and real life problems, and need more specific knowledge to generate the model. We start from the independence hypothesis, that is to design the work domain separately from the planning problems. This approach allows to develop a more flexible model.

The objective is develop a framework, called Dynamic Analyzer (DyNa), in order to make requirements analysis using Petri Nets, especially the dynamic of actions. The design process has two major parts: the elicitation and documentation phase, using UML (OMG 2009) as the modeling language and the oriented-obeject Petri net (from GHENeSys, that is a class of high level Petri nets) (San and Miralles 2012) which is the synthesis of the UML diagrams. Throught the Petri net the requirement analysis will be performed using available techniques. The first step detects contradictions and conflicts between the requirements, or the diferent view points in the diagrams. The second step identifies deep inconsistencies, that are hidden in the dynamic perspective of the plans, as well as additional information about the model that could be interpreted for the planners. This step aim to improve the automated planning process performance. Such information includes, new constraints, invariants, partial solution strategies, characteristics that could help to improve the planner answers in the processing time and in the generated plans quality.

Motivations and Contributions

The automated planning area carries an indefinition problem:

- the study made until today is historically connected to search solution methods to planning problems in an automatically way and domain independent. Thus, the solutions could be inserted in intelligent automated mechanisms, especially robots and another autonomous systems.
- the fomal techniques developed domain independently led to techniques that today are very important in several research fields, like logistics, diagnostic systems, navigation, space robots, satellite systems, etc. This demand is independent of the adequacy of formal techniques to the systems based on specific knowledge to provide real solutions, while increases the discovery of new independent domain solutions.

The sub goal is the design process proposal, and justifies itself, since it contributes to an emerging area and very important to the Engineering Design, that is the inicial phase, where the elicitation and requirement analysis are made.

Requirement analysis in real life systems is a topic that has been currently discussed in important Automated Planning conferences and workshops (McCluskey et al. 2003a) (McCluskey 2002) (Hoffmann 2003a). However, there are not still many works in this research line, especially using Petri nets in the requirement analysis and validation. Recent studies show the need of requirement analysis of real life systems in automated planning, where the problems are modeled using UML (Vaquero 2011) (Simpson 2005) (Simpson, Kitchin, and McCluskey 2007). However, the literature is still very scarce when the subject is Petri net applied to the problem's design process, especially in real life problems.

The main contributions are:

- Propose a diciplined design process for real life problemas, including the independence hypothesis.
- Propose a formal representation (using high level Petri nets) to the problem structure discussed in (Hoffmann 2003b).

- Dtermine a minimal set of UML diagrmas to represent real life problems in Automated planning.
- Propose a closed process to translate minimal set of UML diagrams into a high level Petri net.

UML for Design of Real Life Automated Planning Problems

Accordingly to the literature (Booch, Rumbaugh, and Jacobson 2006) it is clear that the use of UML to model real life systems and planning problems is increasing because of its visual modeling representation (Vaquero and Silva 2005). One of the benefits of using UML is the viewpoints that this language can offer to the model, dismembering the system in different viewpoints is a good approach, and can help in the design process, separating the problem into smaller parts that can be addressed using distinct diagrams. But, representing it using UML can be a challenge because some conflict interpretations can be generated accross the different diagrams during the requirement analysis.

That said what is the minimal set of UML diagrams necessary to represent a planning problems? The goal is to use diagrams that have different and complementary viewpoints. The answer is a minimal set capable to represent a planning problem in an optimazed way. Therefore, it is necessary to consider that planning applications are divided in two parts: domain environment and planning problem.

The first step is identify which requirements are represented in each part. Consider that the domain environment represents the static attributes of the system, and the planning problem represents the dynamic attributes, including possible actions. Both, domain environment and planning problem, can be modeled in different Class Diagram, aiming to not overlap the viewpoints. The Class Diagram is a static representation form adequated to represent domain environment that will not have methods in the classes since they represent possible actions. As said before, the planning problem is dynamic, but can be modeled in a Class Diagram too, in this case the methods need to be declared in the classes to represent the actions. To complement the planning problem model, the user must use the State Machine, Activity and Sequence Diagrams that contribute to the system dynamic representation. Using this design method, the domain environment model remains static, and the planning problem can be changed according to the necessity, offering an independent and flexible model. The Object Diagram is used to create problem scenarios, such diagram can represent the initial and goal state and from these diagrams the planner can find the action plan.

The Use Case Diagram is unnecessary, because it brings redundant information that is similar to a textual description of the problem. Irwin and (Irwin and Turk 2005) argued that in the representation via Use Case there is no consistency and accuracy. Many authors believe that the Use Case Diagram is not useful in the requirement analysis, like (Siau and Lee 2004). The literature points that there are criticisms regarding the use of this diagram. (Dobing and Parsons 2006) questions the naturality involved on the Use Case Diagrams in the object modeling and the idea of these diagrams facilitating the communication and requirement analysis. Even with all those criticisms the Use Case Diagram can be useful to separate the problem into smaller parts, this is an interesting approach that can help the design of the Activity Diagram. However, for this first proposal the Use case Diagram will not be part of the minimal set.

The proposed minimal set consists of the following diagrams:

- Class Diagram;
- State Machine Diagrama;
- Activity Diagram;
- Sequence Diagram;
- Object Diagram.

With this proposal the set of UML diagrams used in itSIMPLE has increased. Using the minimal set it is possible to have a concise and syntactically correct representation. However, it is impossible to validate the model since UML is a semi-formal representation language, an alternative is to use Petri nets to execute the validation, but when the diagrams are grouped the viewpoints are merged and with that the error identification becomes complex. Nevertheless, in the UML phase, can be performed a syntactic analysis to find errors in the model. This can be done by analyzing the xml file of the itSIMPLE project.

Basics of Dynamic Analysis using Petri Nets

The use of Petri nets is quite promising and throught this technique the users can perform the verification and validation of the model. In the first running exercises and study cases it were used the place/transition nets that is a classical Petri net, Figure 1 shows the life cycle of the automatic planning.

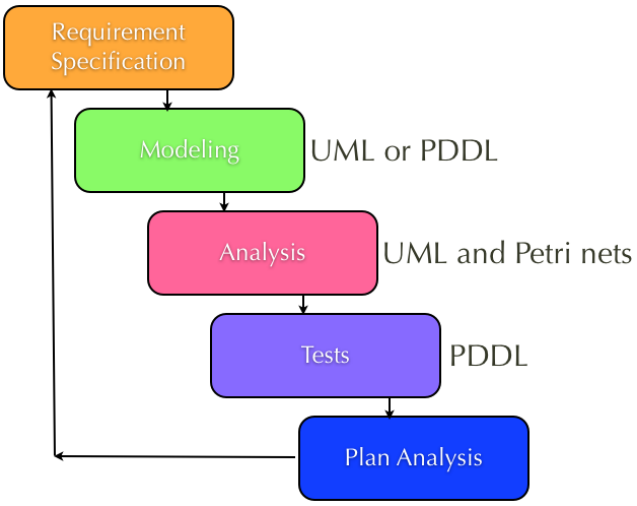

Figure 1: Life cycle of the automatic planning

Currently in the literature Petri nets its been used to improve the planner performance. Many authors, proposes the use of Petri nets to reduce the planner state space search.

However, little has been found about the true benefits of using this technique in the requirement analysis of planning problems. This is because the majority of work in planning are Artificial Intelligence applications, in which the planning problem is synthetised only to test the planners ability to offer an acceptable answer. Recently, has been arised an interest in real life problems, which significantly changes the scenario of research in planning. Therefore, it is necessary to perform requirement analysis for real life systems because they are complex and need a differentiated design process explained in the previous section.

Petri nets is a great tool to validate the requirements (Silva 2004) , it allows a general representation of the system and it is a good formalism for real life systems, because it represents the action dynamics. With Petri nets it is possible to group the UML diagrams in a single net. The idea is to find interactions between the objects represented in all diagrams and translate them into an unique Petri net, that can represent the intire system process.

Currently, the itSIMPLE translate the UML model into Petri nets using only the State Machine Diagrams, (Vaquero 2007). Figure 2 shows a simple example of how this translation occurs.

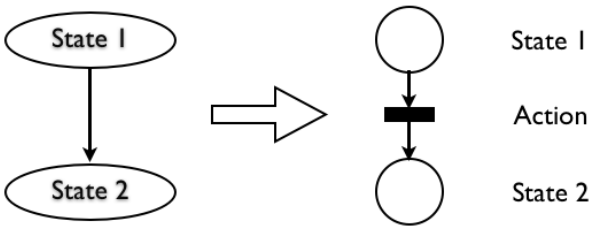

Figure 2: Translation of State Diagrams in Petri Nets.

The approach using only State Machine Diagrams in the translation process does not allow a correct analysis of the system. From each State Machine Diagram is derived a Petri net, therefore it is necessary to find the interaction points of the nets and link them into a single Petri net. One of the ideas of this work is to use not only the State Machine Diagram, but also the Activity and Sequence Diagrams to derive the Petri net.

Once the Petri net was defined it is necessary to search for straightforward and trivial conflicts and the presence of deadlocks. After that, a semantic analysis is performed aiming to verify whether the network in fact represents the planning problem. The semantic analysis can be accomplished using some of the Petri nets properties, like the behavioral and structural properties.

Behavioral properties of Petri nets are related to the behavior of the net and change during the simulation. They are essentially dependent on the execution stage in which the network is located and are, therefore, marking-dependent (Murata 1989).

Structural properties are those that depend on the topological structures os Petri nets. They are independent of the initial marking *M0* int the sense that these properties hold for any initial marking or are concerned with the existence os certain firing sequences from some initial marking. A formal definition of a Petri net is given in Table 1 (Murata 1989).

A Petri net is a 5-tuple, $PN = (P, T, F, W, M0)$ where:
$P = \{p1, p2, ...pn\}$ is a finite set of places,
$T = \{t1, t2, ..., tn\}$ is a finite set of transitions,
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
$W: F \rightarrow \{1, 2, 3, ...\}$ is a weight function,
$M0: P \rightarrow \{0, 1, 2, 3, ...\}$ is the initial marking,
$P \cap T = \oslash$ and $P \cup T \neq \oslash$.

Table 1: Formal definition of a Petri net.

A Petri net structure $N = (P, T, F, W)$ without any specific initial marking is denoted by N.A Petri net with the given initial marking is denoted by $(N, M0)$.

A Petri net, that is a good representation for a planning problem, should have present some properties, that are (Murata 1989):

- Reachability: The firing of an enable transition will change the token distribution (marking) in a net according to the transition rule. A sequence of firings will result in a sequence of markings. A marking *Mn* is said to be reachable from a marking *M0* if there exists a sequence of firings that transforms *M0* to *Mn*.

- Boundedness: A Petri net *(N, M0)* is said to be bounded if the number of tokens in each place exceed a finite number k for any marking reachable from *M0*, i.e. $M(p) \leq k$ for every place p and every marking $M \in R(M0)$.

- Liveness: The concept of liveness is closely related to the complete absence of deadlocks in operating systems. A Petri net *(N, M0)* is said to be live (or equivalently *M0* is said to be a live marking for *N*) if, no matter what markinghas been reached from *M0*, it is possible to ultimately fire any transition of the net by progressing through some further firing sequence. This means that a live Petri net guarantees deadlock-free operation, no matter what firing sequence is chosen.

- Reversibility: A Petri net *(N, M0)* is said to be reversible if, for each marking *M* in *R(M0)* is reachable from *M*.

Methods of analysis for Petri nets may be classified into the following groups: 1) the coverability graph method (including reachability tree); 2) the matrix-equation approach, and 3) reduction or decomposition techniques. The first method involves essentially the enumeration of all reachable markings or their coverable markings. It should be able to apply to all classes of nets, but is limited to small nets due to the complexity of the state-space explosion. On the other hand, matrix equations and reduction techniques are powerful but in many cases they are applicable only to special subclasses of Petri nets or special situations (Murata 1989).

Results and Discussion

Modeling the work domain separately (Independence Hypothesis) make sense because it is common to many planning problems. With this approach we believe that is possible to analyse how the work domain constraints influences the planning problem. Besides that, one can analyse easily the structure that represents the work domain. And this representation is very closer to what (Hoffmann 2003a) calls problem structure.

According to Hoffmann (2003), benchmark problems has a sort of structure which repeats in diferent planning aplicca-tions. That structure is what the author called problem structure. Hoffmann (2003) raises a number of questions about the structure of the problem, among them two questions are relevant here: 1) What are the common patterns in the domain structures? Is it possible to find a description of these domains? The author points out that patterns could be an answer in the space-state graph topology.

The approach via Petri nets is what Hoffmann (2003) calls problem topology. And with this approach we can identify phenomena like dead-ends and goal ordering that he disccuss in his lecture notes. The dead-ends is a kind of deadlocks in Petri nets, and this is easily solved. Another possibility is to observe if the planner can reach subgoals before reaching the goal state. With that is plausible to say that this characteristic can help to solve the ordering goals problem presented in (Hoffmann 2003a). But, in this case we have positive results just for small and closed problems as will be shown below.

Let us consider the benchmark problem Blocks-World, where the problem instance is:

- Initial state: clear(b), clear(c), ontable(a), ontable(b), on(c, a), handempty.
- Goal: on(b,c) on(a,b) ontable(c).

Figure 3 shows all possible states of Blocks-World Arm, from state 1 where the blocks are on table to state state 13. The initial state of our problem instance is state 11 and our goal state is state 9.

From Figure 3 was designed to Petri net of Fig 4. In this figure each place represents a possible state of figure 3. And there is an essential node, node 1 which divide the net in components. Each component represents the movement for each block. If we remove node 1, we will have three different Petri nets. The Sussman paradox occurs when the processing goes from essential node and comes back to the same component entering into a loop. It possible to solve that problem with just three actions: 1) it is mandatory pass through essential node; 2) since passed the essential node do not return to the same component (not repeat the initial action); 3) run the action ”move” to the respective component.

With this simple solution we can resolve the Sussman paradox and goal ordering phenomena presented in (Hoffmann 2003a). Tests has not been done already with large and complex problems. But, the results found in small problems is a good sign that this technique can be also applied for complex problems. However, for real life problems, we believe that it will be necesary to use high level Petri nets.

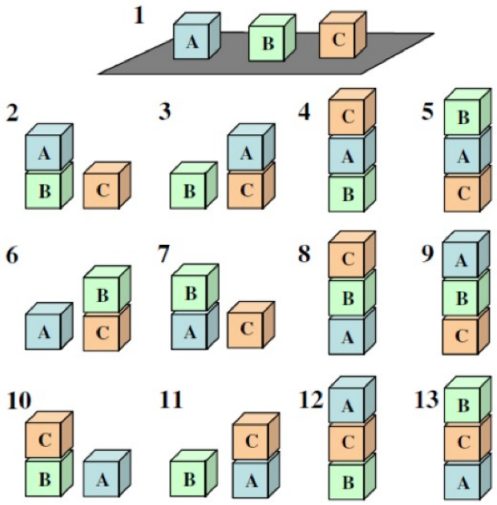

Figure 3: Possible States for Blocks-World Arm Problem.

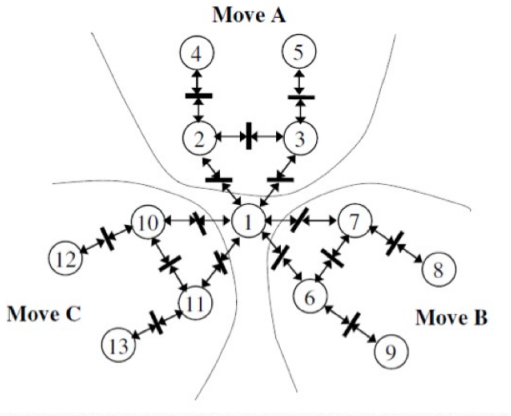

Figure 4: Blocks-World Arm Petri Net.

Conclusion

After some case studies, we found that the Petri net generated using he current version of itSIMPLE is not correct. Consider just the State Machine Diagrams is not enough to derivate the Petri net. But, currently the State Machine Diagram is the only way to represent the dynamic part of the system, this was the main motivation to add other diagrams that can represent the dynamic aspects in a better way. Another weakness of itSIMPLE is how the system is modeled. There is no project discipline and this can lead the user to wrongly model the problem, confusing the work domain with the planning problem. Our proposal is to separate them in different Class Diagrams, and from the Class Diagram - that represents the planning problem - design other diagrams, since they can offer different viewpoints that can complement the information needed to generate the Petri net.

In this paper it was presented a proposal that uses Petri net in Knowledge Engineering to improve the design pro-

cess. The purpose of the thesis aims to create a better and disciplined way to model planning aplications using UML and Petri nets, even though the method has not been implemented and validated yet, the results found in the case studies are quite promising. Using the method proposed in this paper it is expected to have an improvement of the plans, the planners performance and for the problems structures (Hoffmann 2003b).

References

Booch, G.; Rumbaugh, J.; and Jacobson, I. 2006. *UML - User Guide*. Elsevier.

Dobing, B., and Parsons, J. 2006. How UML is used. *Communications of the ACM* 49(5):109–114.

Ghallab, M.; Nau, D.; Traveso, P. 2004. *Automated Planning: Theory and Practice*. CA: Morgan Kaufman.

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

Hoffmann, J. 2003b. Utilizing Problem Structure in Planning A Local Search Approach. *LECTURE NOTES IN COMPUTER SCIENCE* Volume 285.

Irwin, G., and Turk, D. 2005. An Ontological Analysis of Use Case Modeling. *Information Systems* 6(1):1–36.

McCluskey, T.; Aler, R.; Borrajo, D.; Haslum, P.; Jarvis, P.; Refanidis, I.; and Scholz. 2003a. Knowledge Engineering for Planning Roadmap.

McCluskey, T. L.; Aler, R.; Borrajo, D.; Haslum, P.; Jarvis, P.; Refanidis, I.; and SCHOLZ. 2003b. Knowledge Engineering for Planning Roadmap.

McCluskey, T. 2002. Knowledge engineering: issues for the AI planning community. *The AIPS-2002 Workshop on Knowledge Engineering Tools and Techniques for AI Planning*.

McDermott, D. V. 2003. PDDL2.1 - The Art of the Possible? Commentary on Fox and Long. *Journal of Artificial Intelligence Research (JAIR)* 20:145–148.

Murata, T. 1989. Petri Nets: Properties, Analisys and Applications.

OMG. 2004. UML 2.0 OCL Specification.

OMG. 2009. OMG Unified Modeling Language TM (OMG UML), Superstructure.

San, A., and Miralles, P. 2012. *GHENeSys , uma rede unificada e de alto nivel*. Ph.D. Dissertation, Sao Paulo.

Siau, K., and Lee, L. 2004. Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML. *Requirements Engineering* 9(4):229–237.

Silva, J. R. 2004. Applying Petri nets to requirements validation. *IFAC Symposium on Information Control Problems in Manufacturing Salvador* 1:508–517.

Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. L. 2007. Planning domain definition using GIPO. *The Knowledge Engineering Review* 22(02):117.

Simpson, R. M. 2005. Gipo graphical interface for planning with objects. *Proceedings of the First International Competition on Knowledge Engineering for AI Planning*.

Upal, M. A. 2005. Learning to Improve Plan Quality. *Computational Intelligence* 21(4):440–461(22).

Vaquero, T. S., and Silva, J. R. 2005. The itSIMPLE tool for Modeling Planning Domains. *Artificial Intelligence*.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007a. 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.; Romero, V.; Tonidandel, F.; and Silva, R. 2007b. itSIMPLE 2 . 0 : An Integrated Tool for Designing Planning Domains. *Design* 336–343.

Vaquero, T. S.; Tonidandel, F.; and Silva, J. R. 2005. The itSIMPLE tool for Modelling Planning Domains. In *Proceedings of the First International Competition on Knowledge Engineering for AI Planning, Monterey, Califormia, USA*.

Vaquero, T. S. 2007. *ITSIMPLE : Iintegrated Tools Environment For Modeling and Domain Analysis*. Dissertation, Polytechnic - University of Sao Paulo.

Vaquero, T. S. 2011. *Post-design for Automated Planning Problems: an approach combining diagnosis, virtual reality and reuse of rationales*. Tese de doutorado, Polytechnic - University of Sao Paulo.

Zimmerman, T., and Kambhampati, S. 2003. Learning-assisted automated planning: looking back, taking stock, going forward. *AI Magazine* 24(2):73–96.