ANALYSIS AND MANAGEMENT OF PLANS PROVIDED BY AUTOMATED PLANNING SYSTEMS

VICTOR M. C. ROMERO[1], TIAGO S. VAQUERO[1], FLÁVIO TONIDANDEL[2], JOSÉ R. SILVA[1]

[1]*DesignLab, PMR -Departamento de Engenharia Mecatrônica e Sistemas Mecânicos, Escola Politécnica da Universidade São Paulo*
Av. Prof. Mello Moraes, 2231 – Cidade Universitária – CEP 005508-900 - São Paulo - SP
[2]*IAAA – Artificial Intelligence Applied in Automation Lab, Centro Universitário da FEI*
São Bernardo do Campo - SP
E-mails: {victor.romero, tiago.vaquero}@poli.usp.br, flaviot@fei.edu.br, reinaldo@usp.br

Abstract— The entire life cycle of a real planning problem design involves several procedures, each one with its importance and capacity of extracting knowledge from the given problem. One of these procedures is the analysis and management of plans, which is vital for debugging the problem model, as well as for the verification of the consistence of the generated plan by an automated planning system. In this way, this paper aims to show some alternatives of how representing and analyzing, in a visual and friendly interface, the effects of the plan actions execution, like the evolution of variables values and the snapshots of the generated plan. We also present the itSIMPLE environment as tool for developing all the specification, modeling and analysis of the planning problem.

Keywords— Automated Planning, Plan Visualization, Analysis and Management of Plans, Artificial Intelligence

Resumo— O ciclo de vida completo do projeto de um problema real de planejamento envolve inúmeros procedimentos, cada um com sua importância e capacidade de extrair conhecimento do problema dado. Um desses procedimentos é a análise e o gerenciamento de planos, o que é vital para depurar o modelo do problema, bem como para a verificação da consistência do plano gerado por um sistema de planejamento automático. Nesse sentido, este trabalho pretende mostrar algumas alternativas de como representar e analisar, em uma interface visual e amigável, os efeitos da execução do plano de ações, como a evolução de valores de variáveis e cenas do plano gerado. Também é apresentado o ambiente itSIMPLE como ferramenta de desenvolvimento de toda a especificação, modelagem e análise do problema de planejamento.

Palavras-chave— Planejamento Automático, Visualização de Planos, Análise e Gerenciamento de Planos, Inteligência Artificial

1 Introduction

The continuous development of the Automated Planning (AP) area, combined with its significant increase of importance, has brought us many tools and procedures for modeling systems and solving planning problems. Besides that, there has been in this area a great effort in modeling and analyzing not only toy problems, but also real and complex ones, which provides the utilization of the Automated Planning developments in real situations.

However, the use of AP to solve real problems brings up new requirements to model and analyze these problems efficiently. Therefore, it is vital that the modeling process applies all the analysis steps present in the life cycle of a planning system design. This life cycle, mainly in the initial phases, involves since the requirements analysis until the plan management and verification. Only with all those steps completely done that the generated plan will satisfactorily solve the problem and hence will be able to be implemented.

One important phase in that design process is the plan verification and refinement. During the problem modeling is very common to the designers to make mistakes, especially in such complex problems as the ones based in real planning systems. Besides that, usually the final model doesn't truly represent what the designer intended to, due often to language limitations and lack of requirements, even if the model is free of errors. Thus, not always the generated plan will achieve the solution expected by the designer. However, it is often impossible to tell, only by looking at a set of actions in text format (which is the response given by the majority of the planners) if the plan really represents the solution for the real problem as well as if the agents and resources of the domain are being well used. In this way, this article doesn't concern about the validation of plans, which is a formal procedure, but with the plan visualization, verification and analysis, which is an informal process made by the developers, designers and all participants themselves. This kind of process enables the designer to check issues like resources allocation, costs reduction, investing needs, among many others. In addition, this paper also aims to present an environment that makes possible to these participants to entirely perform this process, in an easy and intuitive way.

All the examples given in this paper, as well as the tool implementations, were made using the itSIMPLE environment (Vaquero et al. 2005; Vaquero et al. 2006), whose details will be given in the next sections.

2 Plan Analysis and Management

Plans are the result of the planning process. Usually, a plan is defined as a set of actions organized with some structure. The actions may occur with parallelism, i.e., more than one action may happen at the same time, or they may appear in sequence, which is the simplest case.

It is very important that the designer be able to visualize the plan in a properly way, so he/she can understand and estimate what would happen with the real system in an eventual implementation of the plan. This enables the designer to manage the plan generation and execution by testing all of the possible ways to achieve the same goal as well as to reason about the generated plan and, based on it, take decisions that will affect the real system.

However, the visualization of the plan is not a simple task, since it is usually outputted by the planner as a list of actions, normally in raw text format. This makes it impossible to the designer to visualize the plan execution or taking any other useful information of it, especially in large plans. Besides that, the planners don't have a standardized structure to present the plan, and each one does it as its own way. Therefore, it would be better for designers if the plan were presented in a visual and friendly interface.

The concept of plan analysis and management is rather new, but some works have already been done about it. The most recent ones have showed that the plan can be more than a simple list of actions. It could be used, for example, to save information about its own rationale, allowing the designers to understand the heuristic behind the plan generation process (Wickler et al. 2006). In addition, some planning tools offer plan visualization mechanisms. The software GIPO III (McCluskey and Simpson 2006), for instance, has a plan analyzer called OLHE (Object Life History Editor), which allows the designer to visualize the plan execution. The tool PlanWorks (Daley et al. 2005) also presents a modeling interface with plan visualization features. This work also emphasizes the importance of the plan visualization for refining and debugging the model, and the great benefits of this kind of analysis. The itSIMPLE environment, described in this paper, has its own plan analysis interface, but it uses a different approach than these other tools, which will be described through the next sections. It is important to mention that in this work we will not deal with time based domains, although the plan steps are used to illustrate the system evolution.

3 The itSIMPLE Environment

3.1 Presentation

The itSIMPLE (*Integrated Tool Software Interface for Modeling PLanning Environments*) is a Knowledge Engineering environment that aims to offer all the necessary tools for specification, modeling, analyzing and visualizing a planning problem.

In itSIMPLE, all the modeling process is made using UML (*Unified Modeling Language*) (OMG 2001), a well-known and widely used language, which facilitates the use of the environment itself, since the designer doesn't need to learn new languages to build his model. Many UML diagrams are used by itSIMPLE: the Use Case Diagram, to the initial specification and requirement analysis; the Class Diagram to create the domain static structure with agents, types, associations, attributes and operators present in the system; the State Machine Diagram to specify the dynamic aspects of the model showing the objects states during the plan execution; and finally the Object Diagram to build the snapshots representations (i.e. the planning problem itself with the initial and the goal states), using the abstract specification of the Class Diagram. Specifically in the State Machine Diagrams, the OCL (*Object Constraint Language*) (OMG 2003) is used to specify the objects states as well as the actions preconditions and effects.

All the information given by the designer in the modeling procedure is stored in files using an XML (*eXtensible Markup Language*) (Bray et al. 2004), so they can be easily represented afterwards. Besides that, the XML representation eases the translations processes, working as an intermediate language between the UML and the other representations, as PDDL (*Planning Domain Definition Language* - the standard language for representing planning domains and problems for planners) (McDermott, 1998) and Petri Nets (Murata, 1989). From the model, itSIMPLE is able to generate Petri Nets (simulating them for dynamic verification) and PDDL representations of the domain and problems, which can be used as input for the planners. More details on how itSIMPLE saves its database and creates its representations can be found at the works of Vaquero et al. (2005) and Vaquero et al. (2006). An example of itSIMPLE's UML interface is shown in Figure 1.

3.2 The design tools

The itSIMPLE environment encompasses the main initial phases of the problem design. The initial requirement analysis is built with the Use Case Diagram, and the problem modeling and model analysis is made with the others UML diagrams. The dynamic analysis can be done with the use of Petri Nets since UML diagrams do not provide all necessary features for such analysis (Watanabe et al. 1997) (Cheung et al. 1998).

Finally, the plan management and visualization can be done with the aid of variables charts, like XY graphs and step charts or charts that represent the plan, like Gantt Charts. Besides that the visualization

of the snapshots created during the plan is made, like in the other snapshots, with UML Object Diagrams.

In this way, after creating a plan with a planner, the designer can import it into itSIMPLE and simulate it. This enables him/her to have a complete view of the generated plan. All this features create an enriched environment for the analysis and management of the plans made by the user.

4 Plan Analysis Using itSIMPLE

As said before, the itSIMPLE environment provides many tools so the designer can more easily analyze and manage its model and its plans. Figure 2 shows the plan analysis interface. This analysis can be made in two different ways.

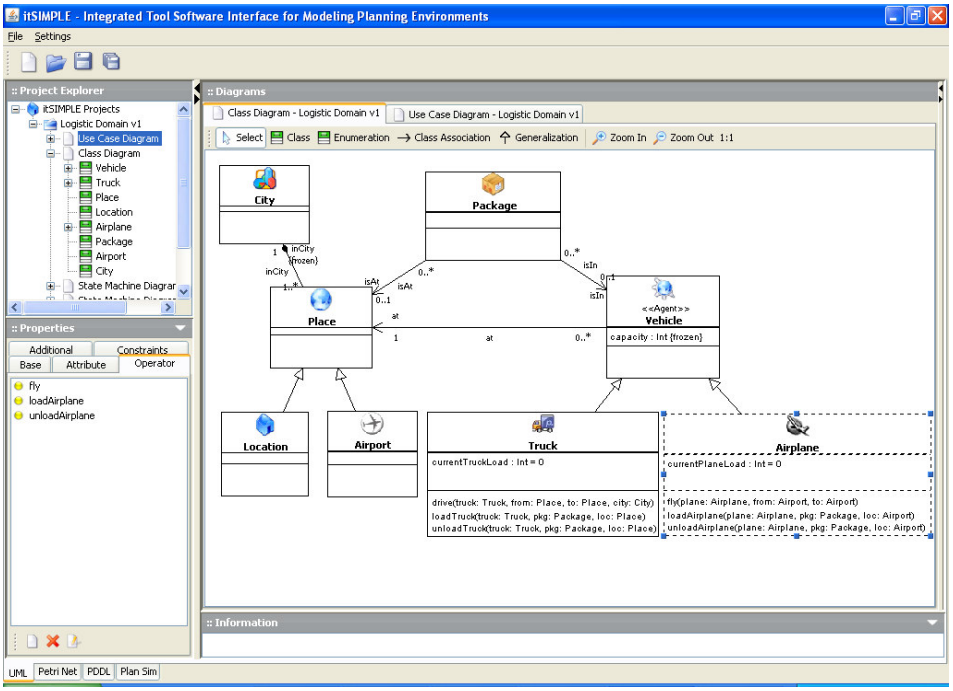

Figure 1. itSIMPLE's UML interface

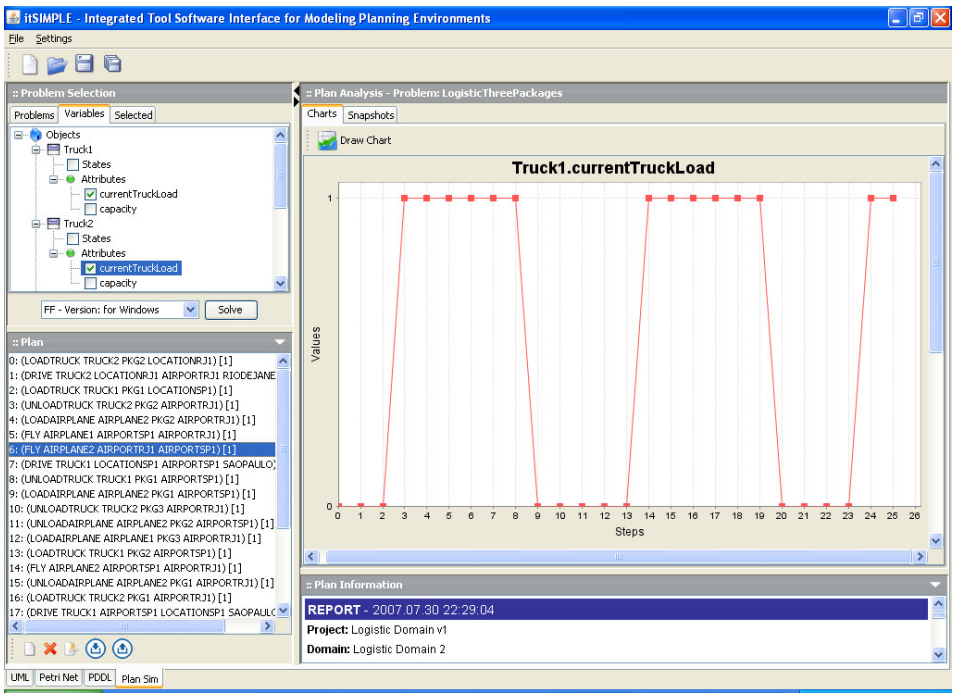

Figure 2. itSIMPLE's Plan Analysis interface

The first one, called *Analysis and Tracking of Variables*, consists in the visualization of the object attributes or states using XY and Gantt charts. The other, which we call *Movie Maker*, is done by observing the plan snapshots step by step, as they were scenes of a movie, starting in the initial state and going to the goal state.

This is very important in modeling, since it's very common to make mistakes in its specification. Since the plans are, *a priori*, supposed to solve the given problem, if any unwanted behavior is observed, than it's assumed that the inconsistence must have been originated by the domain modeling itself or, more specifically, by the problem formulation. This is very important for the design process, since the designer can start a refining cycle, based on the plans that are generated. That gives him/her the ability of not only taking a whole view of how the real system would response to the changes triggered by the plan execution, but also to debug and improve the domain model. Besides that, the designer can take useful information of the plan representation, which enables him/her to take strongly based decisions in areas like resources allocation and depot logistic, for example.

4.1 Analysis and Tracking of Variables

This kind of approach is based on the observation of the attributes of the objects that compound the modeled system. The itSIMPLE environment can generate simple XY charts, which shows how variable changes all along the plan execution. Combined to that, itSIMPLE also provides a Gantt Chart view of the plan, as shown in Figure 3. This kind of chart is commonly used in scheduling softwares, and is a very efficient way to observe a plan. Using those charts, the user can know exactly what is happening with the variables and which action is being executed at any step of the plan. An example of XY charts is shown in Figure 4.

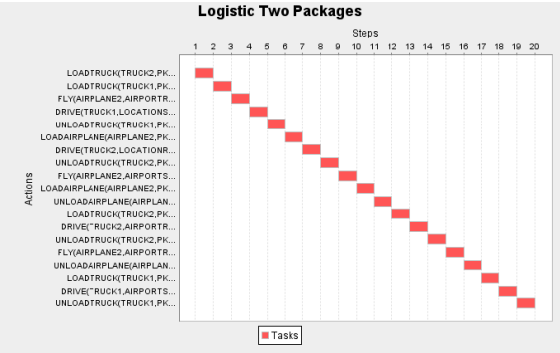

Figure 3. Example of Gantt Chart

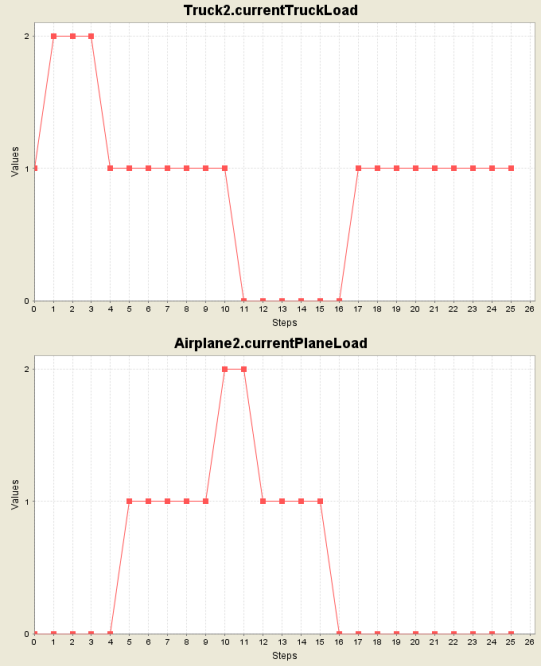

Figure 4. Example of XY chart

By checking the attributes evolution during the execution of the plan, designers can have a consistent estimative of the real behavior of the system. After this estimative, it's possible to reason about the generated plan, and realize whether the real system has either an insufficient infrastructure to implement the plan or, in opposition, a waste of resources that could be being used in other tasks.

The designer can also check whether the constraints defined in the model are being respected. Besides that, he/she can define critical lines for the attribute values, and be informed by the tool if these lines are being crossed. This represents situations when, although the constraints are being respected, a bigger attention should be given to the problem, since it's getting closer to an unsupported arrangement of the resources.

This is very useful in areas like manufacture and production chains, depot logistic, resources allocation, etc. In a petrochemical industry, for instance, the designer could check if the available fluid tanks are enough to store the production. In a machining industry, he/she could observe the machines use allocation and check if all the machines available are being used, and in what efficiency ratio. In this way, the analysis and management of the plans are very useful to realize whether the industry needs an increase in its investments, as well as to observe if the infrastructure would be able to hold an increase in the production, by checking the current usage of equipments. This has a tremendous impact in the industry, since the designer would be able to know, before even implementing the plans, what would be the real consequences and needs of the plans execution.

4.2 *Movie Maker*

The plan simulation generates snapshots, which represent the states of the system during the execution of each action of a plan. Each snapshot can be analyzed as part of a movie, whose script is represented by the plan itself. In this way, this kind of approach enables the user to have a general view of the system objects arrangement during the execution of the plan and, as in the variables tracking and analysis, check any kind of inconsistence or unexpected behavior (defined at the UML diagrams). This concept is illustrated below, in Figure 5.

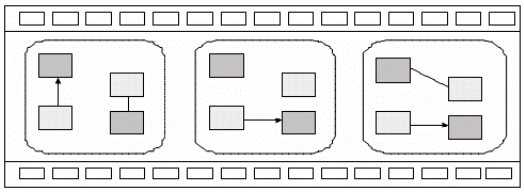

Figure 5. Plan snapshots seen as movie scenes

The itSIMPLE environment is intended to be critical about these scenes, and inform the user about any kind of situation that doesn't respect the given model constraints. The tool is able to make a diagnostic of each scene, and show to the designer whether any kind of problem happens in the snapshot. Besides that, it also informs whether the critical lines defined by the designer are reached, and in what level. This could be used, for example, to check the temperature of a nuclear reactor in a plant. The reactor can have a maximum temperature, which could not be reached at any state of a process, and a critical temperature, after which some kind of additional cooling procedure should be done. The tool would inform whether any of those temperatures, or both, would be reached in the real implementation of the plan. An example of snapshot is presented in Figure 6.

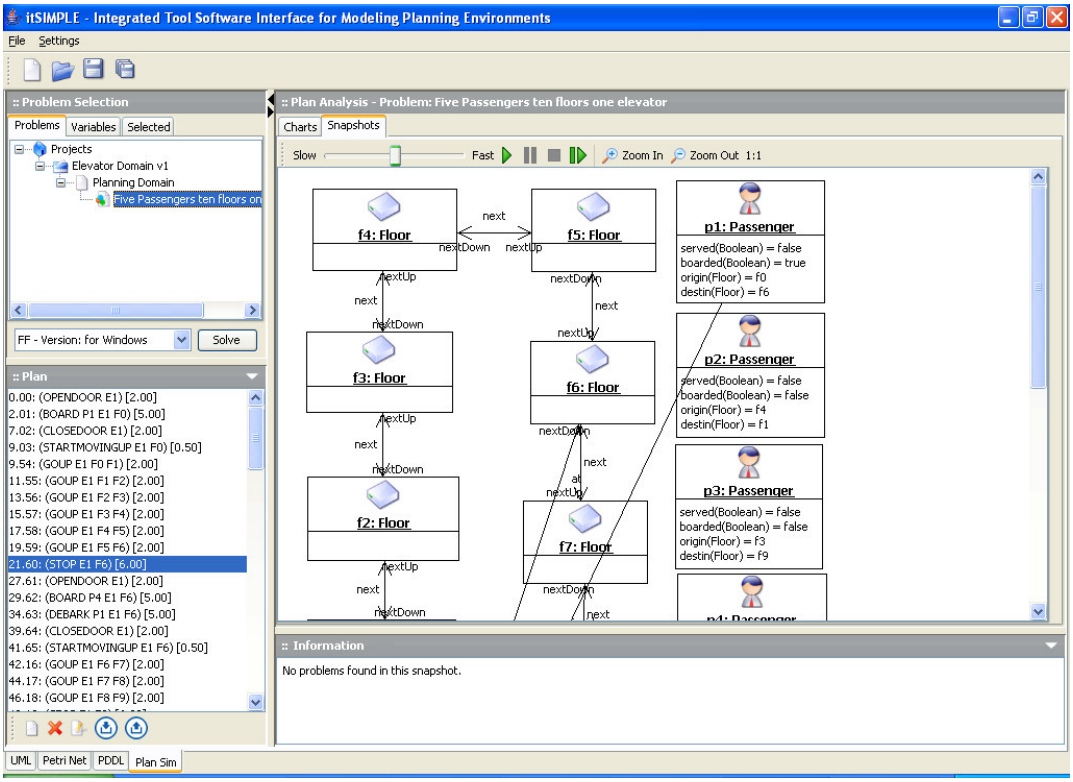

Figure 6. Example of snapshot in the plan analysis interface

5 Conclusion and Future Works

In this work we showed that the visualization and analysis of a plan is also one of the most important phases of the planning problems modeling, since it can be used to do the refinement and verification of the plan, as well as of the entire model. This kind of analysis also enables designers to use this information to take decisions that can affect the real system, as well as reason about the plan itself and the available infrastructure to implement the plan.

Therefore, the designer should be able to visualize the plan in an efficient way, which can't be done with a set of actions in text or list format, as given by most of the planners. In this way, we

proposed that the plan should be outputted in a structured way (for example, in a XML format), which would facilitate the conversion of the plan to visual interfaces and, in addition, the storage of information about the plan. Besides that, we showed how the itSIMPLE environment offers a friendly and comfortable interface to visualize the plan and variables evolution, which helps designers in analyzing and managing the sequence of actions, as well as checking the model consistence.

As future works, we plan to extend the current versions of the plan management and analysis tools. We intend to integrate planners with itSIMPLE, so the creation of the plan could be entirely done in the same environment. There will be a default library of planners and the designer will be able to load a different planner and use it inside itSIMPLE, without the need of manually loading the PDDL files into planners. This would allow the designer to investigate how different planners solve the same problem, and hence finding the better solution and the better planning technique for his/her problem. We also intend to improve the existing plan visualization tools to support time-based systems, which is a very important kind of modeling feature, since it brings the model to a more realistic stage.

References

Bray, T., Paoli, J., McQueen, C.M., Maler, E., Yergeau, F. (2004). "Extensible Markup Language (XML) 1.0 – Third Edition", http://www.w3.org/TR/REC-xml/.

Cheung, K.S., Chow, K.O. and Cheung, T.Y. 1998. Deriving scenarios of object interaction through Petri net. In: Proceedings of Technology of Object-Oriented Language and System.

Daley, P., Frank, J., Iatauro M., McGann, C. and Taylor, W. (2005). PlanWorks: A Debugging Environment for Constraint-Based Planning Systems. Entry in the 1st International Knowledge Engineering Competition.

McCluskey, T. L. and Simpson, R. M. (2006). Tool Support for Planning and Plan Analysis within Domains embodying Continuous Change. Workshop on Plan Analysis and Management, ICAPS 2006, Cumbria, UK.

McDermott, D. (1998). "The PDDL Planning Domain Definition Language" - The AIPS-98 Planning Competition Committee.

Murata, T. (1989). "Petri Nets: Properties, Analysis and Applications", In Proceedings of IEEE, v. 77, n. 4, pp. 541-580.

OMG (Object Management Group), "Unified modeling language specification: version 1.4" (2001). http://www.omg.org/uml.

OMG, OCL 2.0 – Object Constraint Language (2003). http://www.omg.org/cgi-bin/doc?ptc/2003-10-14.

Perez, O. J. G., Reines, F. C. P., Olivares, J. F., Vidal, L. C. and Hervas, T. G. (2006). Planning procces from a user perspective. Workshop on Plan Analysis and Management, ICAPS 2006, Cumbria, UK.

Vaquero, T. S., Tonidandel, F., Silva, J.R. The itSIMPLE tool for Modeling Planning Domains (2005). ICAPS 2005 Competition on Knowledge Engineering for Planning and Scheduling, Monterey, California, USA.

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. Short paper in Proceedings of ICAPS 2006 International Conference on Automated Planning and Scheduling, Cumbria, UK.

Watanabe, H., Tokuoka, H., Wu, W. and Saeki, M. 1997. A Technique for Analysing and Testing Object-oriented Software Using Coloured Petri Nets, IPSJ SIGNotes Software Engineering 117.

Wickler, G., Potter, S. and Tate, A. (2006). Recording Rationale in <I-N-C-A> for Plan Analysis. Workshop on Plan Analysis and Management, ICAPS 2006, Cumbria, UK.