INTEGRATED TOOLS ENVIRONMENT FOR MODELING AND ANALYSIS OF AUTOMATED PLANNING SYSTEMS

Tiago Stegun Vaquero, tiago.vaquero@poli.usp.br
Fernando Sette, fernando.sette@poli.usp.br
Eston Almança dos Santos, eston.santos@poli.usp.br
José Reinaldo Silva, reinaldo@.usp.br
Escola Politécnica da USP, Av. Prof. Mello Morais, 2231

Abstract. *A great effort has been made nowadays in the area of Artificial Intelligence for defining reliable automated planning systems that can be flexible and integrated. That leads to the need of a systematic design process, in which the initial phases are not neglected. In this work we show the importance of these initial design phases applied to planning systems taking advantage of a parallel characterization of domain and problem. Thus, we propose a modeling and analyzing environment, called itSIMPLE$_{2.0}$, which specially contemplates viewpoint concepts from the Requirements Engineering, as well as general concepts from Knowledge Engineering and the Engineering Design. In this environment, it's possible to integrate different system representations and analysis like UML, XML, Petri Nets, and PDDL, in order to obtain a well-formed model which represents the viewpoints of important classes of participants. This integration, which is totally XML based, aims to provide the potential of each language and analysis engine during the design process of an automated planning system for real applications. A case study is included to the management of transportation of oil in the port of São Sebastião in São Paulo province. Thus, this well-known and complex example is used to compare the facility of announcing and structuring the problem in the early phase of the design process.*

Keywords: *Analysis, Specifications, Modeling, Integrated and Flexible Systems, Automated Planning*

1. INTRODUCTION

A great effort has been made nowadays in the area of Artificial Intelligence for defining reliable automated planning systems that can be flexible and integrated. In fact, there is a rising motivation to apply all developments already achieved in the Automated Planning field in real and complex applications. However, this scenario leads to the need of a systematic and disciplined design process, in which the initial phases are not neglected, and also Knowledge and Requirement Engineering tools and methodologies that have a fundamental role for supporting designers.
In the Automated Planning there are few tools that assist designers to better understand, specify, visualize, verify and validate their planning domain models. Some of them was presented at the First International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS 2005) such as itSIMPLE (Vaquero et al., 2005), ModPlan (Edelkamp and Mehler, 2005) and GIPO (Simpson, 2005).

This paper describes the itSIMPLE$_{2.0}$ tool that was designed to give support to users during the construction of real and complex planning domain applications mainly in the initial stages of the design life cycle. These initial stages encompass processes such as domain specification, modeling, model analysis and model testing, all of them crucial for the success of the application. itSIMPLE$_{2.0}$ is an open source project implemented in Java that presents a enhanced integrated environment with well-known representation languages such as UML (OMG, 2001), XML (Bray et al., 2004), Petri Nets (Murata, 1989) and PDDL (Fox and Long, 2003), each one of them with its best contribution to the whole design process. The main purpose of these languages is to lead designers from the informality of real world requirements to formal domain models.

Starting with requirements elicitation, specification and modeling, itSIMPLE$_{2.0}$ proposes a special use of UML - Unified Modeling Language - in a planning approach (named UML.P) which we believe can contribute to the knowledge acquisition process as well as to the domain model visualization and verification. itSIMPLE$_{2.0}$ focuses also on the use of Petri Nets for requirements and dynamic domain model analysis since it is a formalism with great potential for model checking and simulation. The tool also integrates PDDL representation to test models with general planners (automated planning techniques). XML (Extended Markup Language) is used as an intermediate language that can translate the model from UML to other representations such as PDDL or Petri Nets. Besides all the appropriated interfaces for dealing with all these languages, the environment also gives to end-users an interface for analysis and management of plans where designers can observe the behavior of the model during the execution of sequences of actions provided either by users or by planners. This is done by using variable observation in XY and Gantt charts.

In order to show the potential of itSIMPLE$_{2.0}$ during the design process, a case study is presented while the tool features are depict. This case study represents a real and complex planning problem such as the management of transportation of crude oil in the port of São Sebastião in São Paulo province. This real problem deals with the planning of the daily activities of a petroleum plant for docking, storing and distributing oil. The planning of these operations is very important to the functioning of refineries and constitutes a complex problem of difficult mathematical modeling (Dahal

et al., 2003). When planning over this problem engineering must deal with tankers allocation, docking scheduling, tank volume control, crude oil storage with price maximization (avoiding mixing certain types of crude oils) and minimization of costs. In fact, this problem presents many challenges, such as resource allocation, sequencing, scheduling and optimization, among others.

This paper is organized as follows: First, we describe the itSIMPLE$_{2.0}$ Environment. Then we describe the requirements of the case study. Next, we present each design process stage (such as requirements, modeling, model analysis, model testing and plan analysis) encompassed by the itSIMPLE$_{2.0}$ environment through the case study. The paper ends with a discussion and with a conclusions for this work.

2. THE itSIMPLE$_{2.0}$ INTEGRATED ENVIRONMENT

The itSIMPLE$_{2.0}$ environment aims to help designers to overcome the problems encountered during life cycle of planning application projects, mainly at the requirements specification, modeling and analysis phases. The itSIMPLE$_{2.0}$ is designed to permit users to have a disciplined and structured design process to create knowledge intensive models for several planning domains. The suggested process for designing planning domains follows a cyclic sequence of phases like the one showed in Figure 1(a).

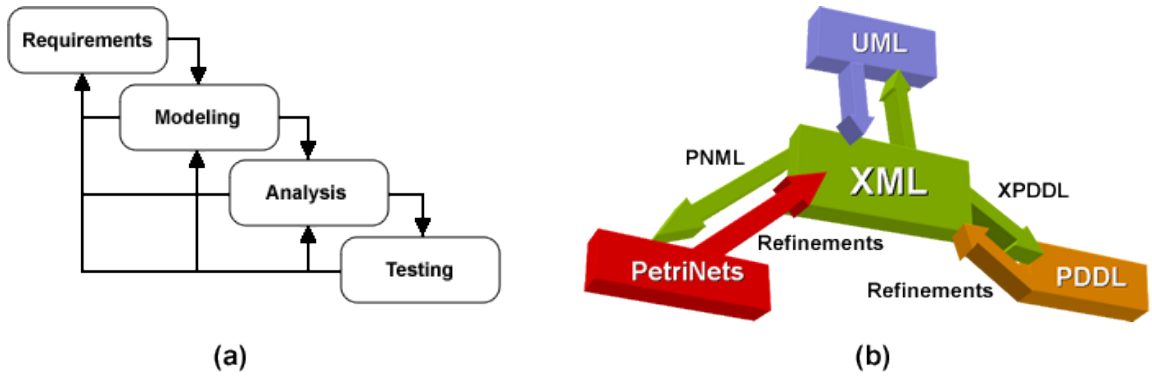


Figure 1. The Design Process and The architecture of the integrated languages in itSIMPLE$_{2.0}$

The itSIMPLE$_{2.0}$ environment is independent of any existing planner, i.e., the processes of modeling, verification and dynamic domain validation are independent of any particular AI planning techniques. However, these techniques would be attached to the process analysis as a name list in order to be chosen (automatically or not) for a planning application or even to work as a testbed for models.

The environment was designed to incorporate a toolset (representation languages and theories) capable of dealing with requirements and knowledge engineering as shown in Figure 1(a). Among many available specification languages, itSIMPLE$_{2.0}$ started by using the semi-formal language UML (which is a well-known diagrammatic language commonly used in the Requirement Engineering) for the requirement process and modeling. It is one of the most used language that models a great variety of applications and we believe that most engineers, working in several application areas, are somehow familiar with this representation. This fact makes UML a good choice for a modeling language in the planning domains context, mainly because of its suitability to make a first model (tracking requirement specifications). Since dynamic aspects are fundamental in planning approach, formal languages and theories that provide good representation of state space evolution may become an alternative to a sound planning domain specification. Following this principle, the environment provides the use of Petri Nets (Murata, 1989), which is a formal representation, which can help designers to perform dynamic domain validations deploying visual and animated information to the entire dynamic system. By using Petri Nets it is possible to utilize all available techniques based on graphs in order to formally analyze the dynamic features of a planning domain (such as deadlocks, invariants, parallelism, concurrency and others). Also, since the AI Planning community has accepted the PDDL as the standard specification language for planner inputs, itSIMPLE$_{2.0}$ integrates the capabilities of dealing with such language mainly in the model testing stage.

In order to hold all information available in several representation languages (UML, Petri Nets and PDDL) itSIMPLE$_{2.0}$ uses the well-known language XML (Bray et al., 2004) which is commonly used in data transactions systems, web applications and data sharing systems. The important point on using XML is that some proposed integrated languages have direct representation in XML such as PNML - which stands for Petri Nets Markup Language (Billington et al., 2003) - for Petri Net representation and XPDDL - eXtensible Planning Domain Definition language (Gough, 2004) - for PDDL. In itSIMPLE$_{2.0}$ all internal verifications and translations are performed in the structure and data available in the XML model which came from UML diagrams. Actually, in order to create a Petri Net representation, a model is first represented in PNML form and then showed to the user as a Petri Net graph for simulation and analysis. In the case of PDDL, all

necessary data is extracted from XML specification in order to first translate the model to a XPDDL representation and then to PDDL. In fact, we believe that XML can really help designers to share and maintain their domains models and knowledge. Considering all these issues, the integrated framework architecture of itSIMPLE$_{2.0}$ is shown in Figure 1(b).

By using itSIMPLE$_{2.0}$ translators, designers are able to change from one language to another any time they want. Users can also deal with many projects and domains at the same time, which allows the reusability of models. In the following sections we describe the Crude Oil Supply problem as a planning problem and then we present each main phase of the domain design process illustrated in Figure 1(a) for dealing with such problem using the integrated environment provided by itSIMPLE$_{2.0}$.

3. THE CASE STUDY - CRUDE OIL SUPPLY PROBLEM

Operations with crude oil involve the unloading of tankers in docking stations into distribution tanks, and the supply of refineries. Since the refineries are constantly consuming oil, the plan must guarantee that, at all moments, the amount of oil in the refineries remains above a minimum level, while minimizing the cost of distribution.

Nowadays, most research work done in this area has utilized mathematical programming where the models are adapted to mixed-integer linear programming (MILP) or mixed-integer non-linear programming (MINLP) to find solutions to this problem. However, current methods have failed to show feasible solutions or require a great amount of time to solve these problems. Furthermore, MILP methods require the use of linearization, which leads to flaws in the final solutions, while the discretization necessary in MINLP methods greatly increases the size of the problem (Li et al. 2005). Therefore, there is no reliable efficient and robust algorithm for this real, and very important, problem in current literature (Li et al. 2005).

In this work, a real planning problem encountered in one of the main oil supply distribution complexes of Brazil will be used to illustrate the tool potential under the automated planning perspective . The domain description and requirements was based on the work of Más and Pinto (2003) and the information provided by Petroleo Brasileiro S.A. (Petrobrás), the main petroleum producer and distributor in Brazil.

In this case study the crude oil is processed in four refineries in the State of São Paulo (Brazil): Paulinia (REPLAN), Sao Jose dos Campos (REVAP), Cubatao (RPBC) and Capuava (RECAP), which are supplied through a pipeline network that leaves the Sao Sebastiao terminal (GEBAST). The system also contains two intermediate substations (SEBAT, in Cubatao, and SEGUA, in Guararema), as well as pumping stations in Rio Prado and Guaratuba. All the crude oil that is consumed by the State of São Paulo comes through GEBAST and is distributed by two pipelines: OSVAT and OSBAT. This system is detailed in Figure 2.

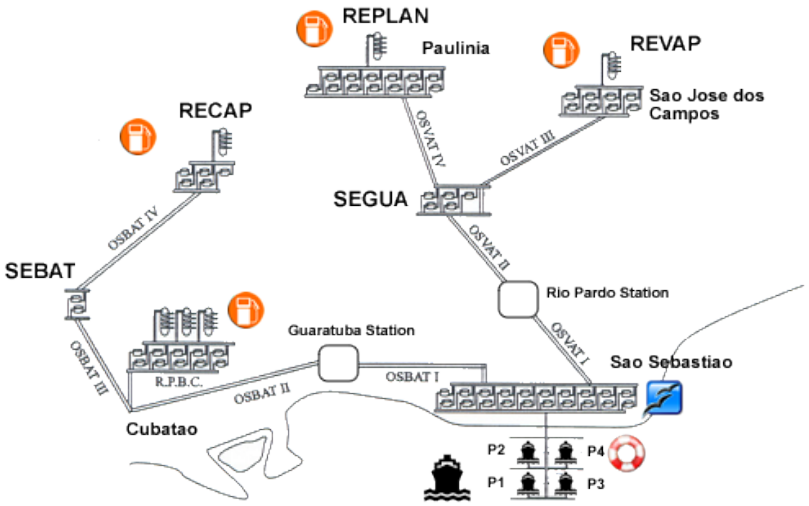


Figure 2. Crude oil distribution infrastructure of Petrobrás in São Sebastião - São Paulo, Brazil

This work considers only the planning of the São Sebastião terminal operations: docking of oil tankers and storage and distribution of crude oil to refineries. In this case study, time constraints are not considered and some simplification are also made as will be presented during domain and problem requirements depiction.

3.1 Domain and Problem Requirements

The distribution plant considered consists of a port, refineries and pipelines that carry the oil to the refineries where it will be processed. The port contains piers, tanks, and an internal pipeline that connects the two structures. This last item has already been subject of study in the planning community, having appeared in ICAPS'04 as a domain in the fourth

International Planning Competition IPC-4 (Hoffmann et al. 2004). However, while this problem is very operational in nature, this paper is concerned with a more strategic issue that is the planning of crude oil distribution in order to maximize profit leaving the pipeline apart.

The planning of port activities involves several activities such as: assignment of tanks to piers, unloading of the tankers to the tanks in the terminal and unloading of the terminal tanks to the pipelines (Más and Pinto, 2003).

Tanker requirements. The crude oil arrives at GEBAST through oil tankers, which are unloaded at the docking stations and stored in the tanks of GEBAST. Each docking station has a limitation regarding the size of the tankers it can receive. In order to unload oil from tankers it is necessary to connect them by pipes and pumps.

Tank requirements. Petrobrás processes several different types of oil in its refineries. Since reserving a tank for each oil type is not practical, these are grouped into classes. The crude oil types that belong to a class can be mixed together without losing value (Más and Pinto, 2003).

At a given moment, a tank can be in either one of three states: inoperative, loading or unloading. Under no circumstance can a tank be unloading and loading simultaneously (Más and Pinto, 2003). Furthermore, there are some restrictions concerning the presence of brine in the tankers inventories. Since every oil type unloaded at Sao Sebastião contains brine (even after separation in petroleum production platforms), the tanks must undergo a settling period (during which the tank remains inoperative), having received crude oil from a tanker, before it can send oil to the refineries. During this period, the brine settles in the bottom of the tank. This is done in the tanks of GEBAST because it is not desirable to transport brine through the pipelines or send it the refineries (Más and Pinto, 2003).

In order to prevent the accumulation of volatile components, the tanks operate using a floating roof system. Since a minimum safety level is required in order o avoid damage to these structures, the tanks cannot ever be fully unloaded (Más and Pinto, 2003). This restriction is, usually, about two meters, which represent about 15% of the total capacity. Therefore, each tank has a maximum and minimum capacity that must be respected in the planning of their operation.

Pipeline requirements. The pipelines are used to send oil from the terminal to the refineries that will process it and are able to transport, simultaneously, more than one crude oil type. During this operation, an interface forms between two different oil types that results in a loss of their properties (Más and Pinto, 2003). Furthermore, a pipeline cannot unload tanks simultaneously.

In this case study, the focus is the planning of port activities in order to minimize the oil interface cost in the pipelines during a day job.

4. DESIGN PROCESS USING itSIMPLE$_{2.0}$

In this section we present the design process of the Crude Oil supply problem made using the itSIMPLE$_{2.0}$ tool. The main phases of this design process are show in the following order: Requirements Acquisition, Domain Modeling, Dynamic Model Analysis, Model Testing and Plan Analysis.

4.1 Acquiring Requirements and Viewpoints

In fact, requirements reflect the needs of customers and users of a system. They should include a justification for this system, what the system is intended to accomplish, and what design constraints are to be observed. Truly, when dealing with real life planning application such as the Crude Oil Supply the requirements elicitation is one of the most important stage in the whole design process. During elicitation, knowledge engineers need to gather all the viewpoints from domain experts, users, stakeholders, sponsors and also planning experts.

All the acquired requirements need to be documented and discussed exhaustively for saving time and resources in further stages. One of the well-known approaches for dealing and documenting requirements is the use case driven approach. By using use case diagrams from UML, it is possible to express requirements in a high level of abstraction. Following this approach, itSIMPLE$_{2.0}$ allows designers to build use case diagrams trying to specify the planning domain in a structured way. In these diagrams, each use case holds information such as descriptions, pre and postconditions, flow events, invariants, constraints, issues and others relevant information that represents the domain requirements. In order to help the documentation process, itSIMPLE$_{2.0}$ automatically generates an structured documentation which turns to be the principal reference for the early design phase. Figure 3 the resulting use case diagram for the case study classical planning domain Logistic.

4.2 Modeling with UML.P

As highlighted before, in the itSIMPLE$_{2.0}$ environment, designers model their planning domains by using UML diagrams. The UML was first defined by OMG Unified Modeling Language Specification between 1996 and 1997 (D'Souza and Wills, 1999), and nowadays is one of the most used languages to model a great variety of applications. Besides, UML has flexibility to attend many kinds of models in an object-oriented fashion since it is widely accepted as the standard for object-oriented analysis and design. This semi-formal modeling language is largely used for the modeling and visualiza-

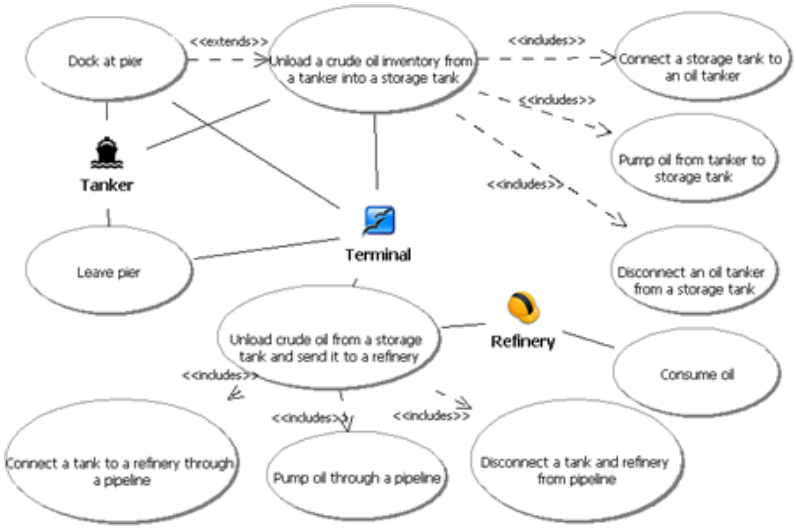


Figure 3. Use case driven approach for domain specification in itSIMPLE$_{2.0}$

tion of system models. itSIMPLE$_{2.0}$ allows designers to use, in the planning domain context, all the collection of best engineering practices that are embedded in UML (OMG, 2001).

Since UML is a general purpose modeling language it turns to be necessary to propose an extended representation that could deal with specification features that are intrinsically related to planning domains. It was called UML.P (UML in a Planning Approach) and was firstly introduced in (Vaquero et al., 2005). For instance, some of the UML diagrams can be directly applied for planning domains such as class diagram, state chart diagram (also known as state machine diagram) and object diagram. In itSIMPLE$_{2.0}$ the designer can model a planning domain by using gradually these diagrams in order to model both domain and problem features as we depict in the following sections following the case study. Figure 4 shows a screenshot of itSIMPLE$_{2.0}$ interface for modeling activities.

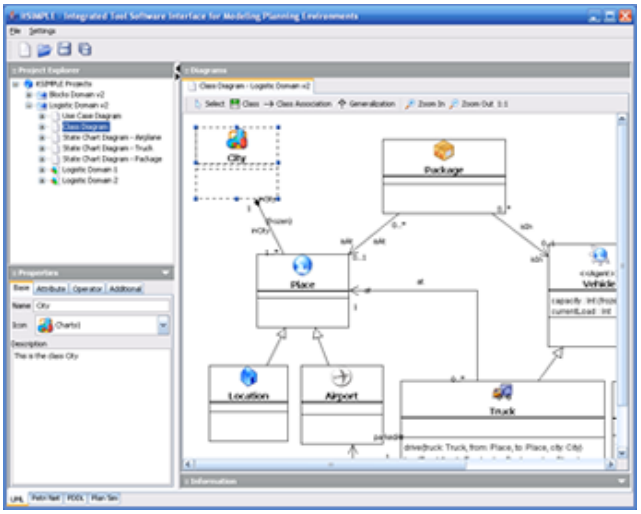


Figure 4. itSIMPLE$_{2.0}$ interface for modeling with UML.P

4.2.1 Domain Modeling

By domain features modeling we mean the process of modeling not only the static structure of a planning domain - with the definition of classes, attributes, associations and structural constraints - but also the dynamic features such as states, actions an their pre and post conditions. In order to model these features in UML.P, two main diagrams are needed: the class diagram (for static features) and the state chart diagram (for dynamic characteristics).

The class diagram is the commonly used in object-oriented modeling process. itSIMPLE$_{2.0}$ provides a clean and very intuitive interface for modeling the main static structure of a planning domain, either a classical domain or a real life one, as depicted in (Vaquero et al., 2006). Beyond all the classes, attributes and association definitions, the designer can also

specify which classes are capable of performing actions by declaring operators (each operator with its set of parameters). Assigning classes with their operators means that users are deciding which classes perform each action. Classes capable of performing actions are what we call the agent, while others are considered only resources in the model.

Figure 5(a) shows the class diagram designed for the oil distribution problem at Sao Sebastião terminal. The diagram consists of eight classes that model all the entities relevant to the real problem being modeled. The Economic_Results class is a utility class that stores variables that are relevant to all other classes in the model such as interface costs. In this particular case, it corresponds to cost and revenue variables which are used as metric for the optimization of profit. As mentioned earlier, the refineries are not modeled in the class diagram, since time has not been included in the modeling of the problem.

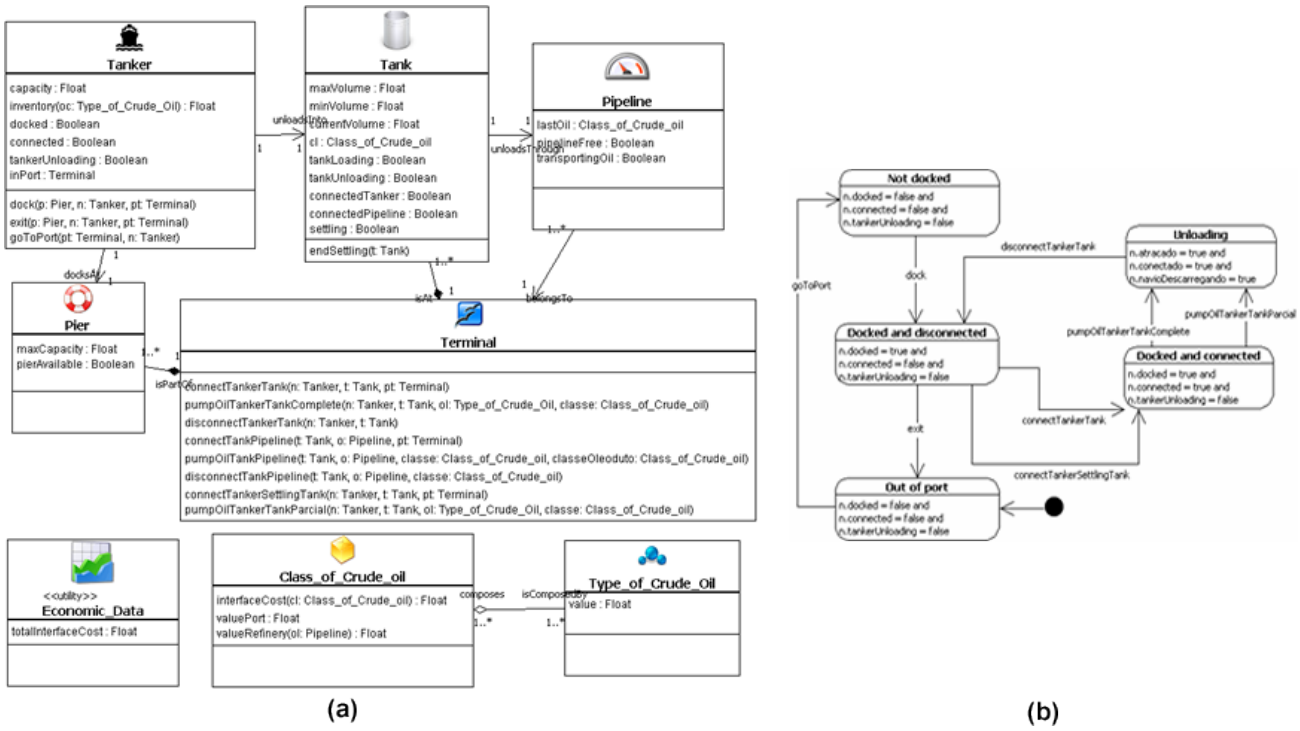


Figure 5. Modeling static features with class diagram (a) and dynamic features with state chart diagram (b)

Another diagram used for modeling the domain features is the state chart diagram. In UML.P the state chart diagram is responsible for representing dynamic features of the domain model. Such dynamic representation is actually one of the main bottleneck in the planning domain modeling process.

In UML.P the designer build a state chart diagram for each class that has a dynamic behavior in the model. By defining one diagram for each dynamic class, users can view their model as a set of regions in the state space.

User specifies in the state chart diagram all the pre and post conditions of actions using the language OCL (Object Constraint Language) (OMG, 2003) following the same approach presented in (D'Souza an Wills, 1999). OCL is a predefined formal language of the UML to represent constraints, invariants and pre and post conditions of actions. In itSIMPLE$_{2.0}$, states are defined by using OCL expressions representing the possible values of attributes. This expressions are also used in transition arcs which represents an action in the chart. A complete action representation is the result of the union of all pre and postcondition expressions declared in all state chart diagram where such action appears. itSIMPLE$_{2.0}$ helps the user with an OCL editor to avoid mistakes while writing such expression. Figure 5(b) shows the State Chart diagram for the class Tanker.

In order to complete the domain features modeling process using itSIMPLE$_{2.0}$, designer can represent the objects that compose the domain, i.e., the agents and the resources that will be used to model the planning problems. For that, users can utilize object diagrams as a repository of objects to be used in all problems.

4.2.2 Problem Modeling

A problem statement in a planning domain is usually characterized by a situation where only two points are known: the initial and goal state. The diagram used to describe these states is called Object Diagram or Snapshots. A good definition of Snapshot is: The Snapshot is a depiction of a set of objects and the values of some or all of their attributes at a particular point of time (D'Souza and Wills, 1999).

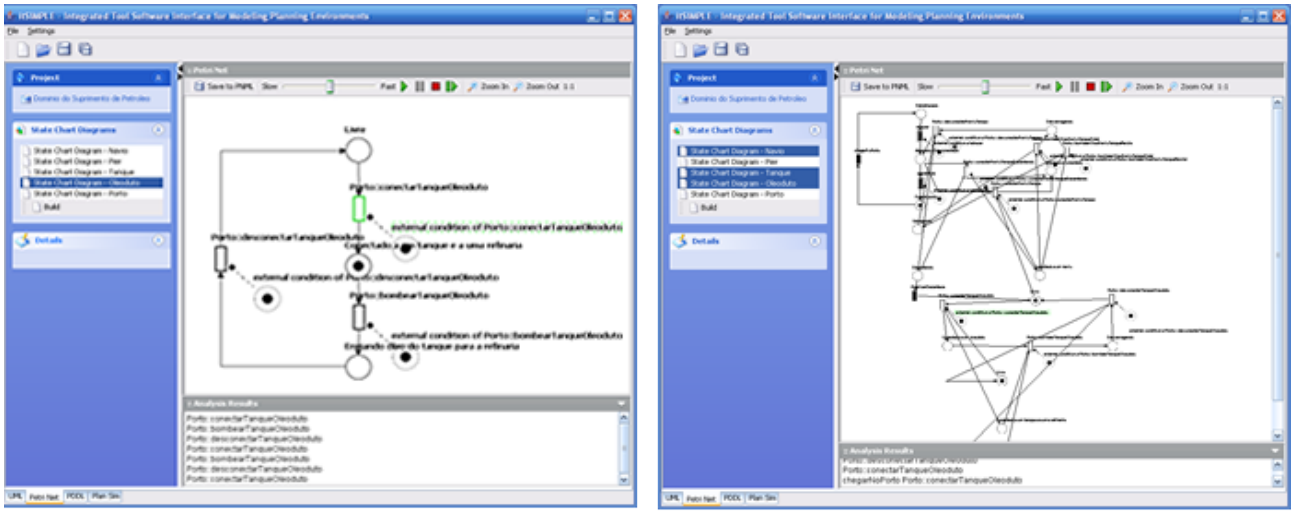
In other words, a snapshot is a picture of a specific time and an instantiation of the domain structure. Such instantiation represents features such as: how many objects are in the problem; what their classes are; how they are arrange; what the values of each object attribute are and how they are related with each other. The itSIMPLE$_{2.0}$ tool helps modeler to specify consistent states, i.e., it helps users to create states by promoting verification on all the constraints and features defined in Class Diagram. For example, when two objects are associated, itSIMPLE$_{2.0}$ provide: verification on all possible associations between them in the Class Diagram; verification of the multiplicities aspects avoiding inconsistent associations.

4.3 Model Analysis

Domain analysis process is becoming one of the main studied topic in the KE for AI Planning, and, indeed, this process has a great impact on the quality of the domain model being built. As mentioned before, itSIMPLE$_{2.0}$ also provides mechanisms for helping designers to analyze and verify their model focusing on dynamic aspects. It is done by representing state chart diagrams into Petri Nets. As highlighted in this paper and in (Vaquero et al., 2005), each state chart diagram shows the dynamic characteristics of objects (of a specific class) being affected by actions that can appear in many state chart diagrams. Therefore, there are usually many state chart diagrams in a single domain model.

In this framework, each state chart diagram can be viewed as a module of the entire domain. Considering this modularity introduced by the state chart diagrams in UML, it is possible to represent these modules through the concept of *modular PNML* (Kindler and Weber, 2001). This concept is used by itSIMPLE$_{2.0}$ to represent Petri Nets. It is important to highlight that OCL expressions are not considered in the PNML representation in itSIMPLE$_{2.0}$.

The dynamic domain analysis process using the state charts and Petri Nets is divided in two main analyses: *Modular Analysis* and *Interface Analysis*. In the *Modular Analysis* users can analyze each module in PN individually checking locally for deadlocks and undesirable features. This analysis is very useful for large applications. In the *Interface Analysis* users can analyze two or more PN modules all together observing the their interaction and how they compete for resources. This last analysis is also useful for seeing the model as a whole while observing the interaction between classes. The itSIMPLE$_{2.0}$'s interfaces for dealing with such analysis are shown in Figure 6 where (a) is the interface for *Modular Analysis* and (b) for *Interface Analysis*.



(a) (b)

Figure 6. Model Analysis with Petri Nets - Modular and Interface Analysis

4.4 Model Testing with General Planners

In order to perform model testing for verification and validation of the planning domain model, itSIMPLE$_{2.0}$ enables the user to represent UML models (holded in XML) in a PDDL format. As highlighted before, the tool transforms the model into a XPDDL representation and then into a PDDL. itSIMPLE$_{2.0}$ allows designers to deal with features from PDDL2.1 and PDDL3 (Gerevini and Long, 2005) such as general constraints and state trajectory constraints. However, itSIMPLE$_{2.0}$ do not deal with time constraints. itSIMPLE$_{2.0}$ builds the domain and the problem specification in PDDL separately. In order to build the PDDL specification to the domain, the tool extract types, predicates (attributes and associations) and functions (attributes) from the class diagram and the actions, as well as pre and postconditions from the OCL expression available at all state chart diagrams (some domain constraints are extract from associations rules). To build a

problem specification in PDDL, itSIMPLE$_{2.0}$ extracts objects and the instantiated situation from the snapshots, including the state trajectory constraints. It also provides a PDDL editor for additional modeling features that the designer wants to include. itSIMPLE$_{2.0}$'s interface for dealing with PDDL is shown in Figure 7.

For the current case study, three planners were chosen and testes using the PDDL representation generated by itSIMPLE: Metric-FF (Hoffman, 2003), SGPlan (Hsu et al., 2006) and MIPS-XXL (Edelkamp et al., 2006). In fact, only these three were able to deal with such model. They were run on a Intel Pentium IV computer with 1.0 gigabyte of memory, running Fedora Linux. Table 1 shows the results for realist problems encountered during a daily or a weekly job in the port. Table 1 shows the number of tanks and oil tankers in the problem, the number of actions in the plan and the computational time used by the planner to find a solution (in seconds). All experiments were terminated once ten minutes had elapsed and no solution been found: a "TIME" symbol in the tables indicates when this happened. The cases where the planner terminated without finding a solution to the problem are designated by an "X" symbol. Indeed, from the Table 1 it is possible to observe that the available planners have difficulty to solve such a real problem which indicates that dedicated planners must be implemented for this case. This point will be discussed in the final section.

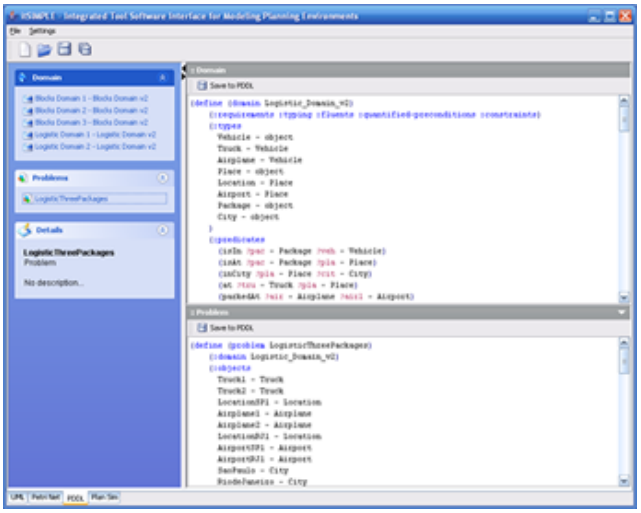


Figure 7. Interface for dealing with PDDL representation

Table 1. Testing the model using realistic problems

# of Tankers	# of Tanks	Metric FF		SGPlan		MIPS-XXL	
		Actions	Time	Actions	Time	Actions	Time
1	18	-	-	7	0.34	X	-
2	18	-	-	17	1.86	X	-
3	18	-	-	31	4.70	X	-
4	18	-	-	62	55.13	X	-
5	18	-	-	75	121.00	X	-
6	18	-	-	78	42.88	X	-
7	18	-	-	TIME	TIME	X	-
8	18	-	-	97	74.43	X	-
9	18	-	-	TIME	TIME	X	-
10	18	-	-	TIME	TIME	X	-
11	18	-	-	115	303.78	X	-
12	18	-	-	126	386.72	X	-
13	18	-	-	132	478.73	X	-

4.5 Plans Analysis

One important phase in the design process that can be performed once the PDDL representation is available is the Plan Analysis. Designer's mistakes are common during the domain modeling, especially when modeling real systems. Besides, sometimes the final model doesn't truly represent what the designer intended to, or, it does not fit what is defined in the

requirements. Thus, not always the generated plan will achieve the solution expected by the designer. However, even using general planners as the only interface for analysis, it is often difficult to tell, only by looking at a set of actions in text format, whether the plan really represents the solution for a real problem or whether the domain agents and resources are being well used, or either if a constraint is being violated. In this way, itSIMPLE$_{2.0}$ intends to make the verification of plans easier and more productive.

The tool provides two ways of performing plan analysis. The first one is made through the use of XY and Gantt charts in what we call the *Analysis and Tracking of Variables*. The second one is made by observing screenshots in the same way as seeing a movie which starts from the problem initial states and goes to the goal state, shot by shot. This second process we call *Movie Maker*. These analysis are very useful in areas like manufacture and production chains, depot logistic, resources allocation, etc. In our study case, for example, the designer could check if the available fluid tanks are enough to storage the production. In a machining industry, the user 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 resources. 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. A partial view of the itSIMPLE$_{2.0}$'s interface for dealing with Plan Analysis is shown in Figure 8. In this figure the loads of the tanks are observed during a given plan execution.

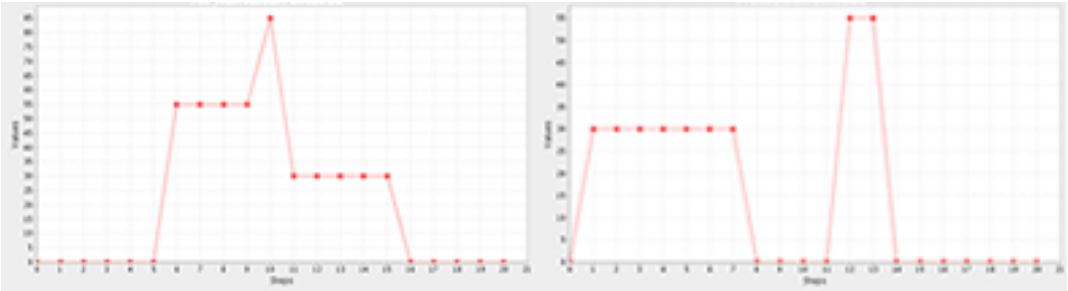


Figure 8. Plan Analysis with XYChart - tanks load observation

5. DISCUSSIONS AND CONCLUSION

The idea of a planning environment for the design of real world planning problem is promising and appropriated to the current scenario, where new domain-independent planners are evolving and being tested to encompass situations more and more complex. The itSIMPLE$_{2.0}$ considers a life cycle for real planning problems based on the use of well-know tools used in real domain specification in an object oriented modeling process for planning domains. Such flexibility of itSIMPLE$_{2.0}$ turns it an interesting environment to test new formal presentations for planning, seeking for a complete specification language that covers all real world problems and also requirements and model problems to test the capacity of new planners.

In this work, a real complex planning problem, such as the planning of the daily activities of a crude oil distribution plant, was used to illustrate the design process using itSIMPLE$_{2.0}$ and also to show the ability of current available planners to solve such problems. Since this is a large and very complex problem, the requirement, specification and modeling processes were done with the support of the KE tool, itSIMPLE$_{2.0}$. Due to the domain model properties, few planners were able to deal with all the characteristics involved. For this reason only three planners were used: Metric-FF, SGPlan and MIPS-XXL. Generally, these planners did not show a successful performance when solving realistic problems in the studied domain. They did not have satisfactory results on either time of response or plan quality or both. In fact, it seems that domain models that use numeric variables to a great extent cause a great impact in the planning process when compared with STRIPS-like domains. In fact, it is common to find real planning problems that possess important and essential numeric variables, especially those which requires optimization of certain parameters.

In fact, it is important to notice that itSIMPLE$_{2.0}$ is able to deal with many of the features encountered in real planning application concerning the mainly the requirements and modeling processes, but the current available general planners are still in constant improvement and all development effort is focused on real application. Indeed, the itSIMPLE project aims to investigates requirements and knowledge engineering aspect for bridging the gap between real planning problems and available planning techniques where it is not possible to neglect parts of the life cycle, which is driven to design. The presented environment still aims to fulfill as much as possible the knowledge engineering requirements in order to send valuable information extracted from the domain experts, planning experts and from the model itself to the planning systems, permitting these system to operate with more complex domain and to find plans more efficiently.

6. ACKNOWLEDGEMENTS

The authors are very grateful to all Design Lab and IAAA members for the enormous help in the itSIMPLE project, specially to the students Victor Romero for all support and implemented features.

7. REFERENCES

Billington J., Christensen S., van Hee KE, Kindler E., Kummer O., Petrucci L., Post R., Stehno .C and Weber M., 2003, "The Petri Net Markup Language: Concepts, Technology, and Tools". In van der Aalst WMP, Best E, eds., Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, Vol. 2679 of Lecture Notes in Computer Science, pp. 483-505.

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

D'Souza, F. D. and Wills, A.C., 1999, "Object, Components, and Frameworks with UML : The Catalysis Approach". Addison-Wesley. United States of America and Canada.

Dahal, K., Galloway, S., Burt, G., McDonald, J., and Hopkins, I. 2003, "Port system simulation facility with an optimization capability", International Journal of Computational Intelligence and Applications, vol. 3, no. 4, pp. 395-410.

D'Souza, F. D. and Wills, A.C., 1999, "Object, Components, and Frameworks with UML Ű The Catalysis Approach". Addison-Wesley. United States of America and Canada.

Edelkamp, S., Jabbar, S., and Naizih, M. 2006. Large-scale optimal PDDL3 planning with MIPS-XXL. In 5th International Planning Competition Booklet (IPC-2006), pp. 28-30, Lake District, England, July 2006.

Edelkamp, s. and Mehler, T., 2005, "Knowledge acquisition and knowledge engineering in the ModPlan workbench", ICAPS 2005 Competition on Knowledge Engineering for Planning and Scheduling, Monterey, California, USA.

Fox, M. and Long, D., 2003, "PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains", Journal of Artificial Intelligence Research 20:61-124.

Gerevini, A. and Long, D., 2005. "Plan Constraints and Preferences in PDDL3: The Language of Fifth International Planning Competition", Technical Report, Department of Electronics for Automation, University of Brescia, Italy, August 2005.

Gough, J., 2005, "XPDDL: A XML version of PDDL". Available in http://www.cis.strath.ac.uk/ jg/XPDDL/, (accessed in April 2007).

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. Journal of Artificial Intelligence Research. Accepted for special issue on the 3rd International Planning Competition.

Hoffmann, J., Edelkamp, S., Englert, R., Liporace, F. S. Thiébaux, and S. Trüg. 2004. Towards Realistic Benchmarks for Planning: the Domains Used in the Classical Part of IPC-4. In Booklet of the Fourth International Planning Competition, pp. 7-14, June 2004.

Hsu, C.W., Wah, B.W., Huang, R., and Chen, Y. X. 2006. New Features in SGPlan for Handling Soft Constraints and Goal Preferences in PDDL3.0. Proc. Fifth International Planning Competition, International Conf. on Automated Planning and Scheduling ICAPS'06, June 2006.

Kindler, E., and Weber, M., 2001, "A universal module concept for petri nets", In Proceedings of the 8th Workshops Algorithmen und Werkzeuge fr Petrinetze, 7 12.

Li, J., Wenkai Li, Karimi, I.A., and Srinivasan, R. 2005. Robust and Efficient Algorithm for Optimizing Crude Oil Operations, American Institute of Chemical Engineers Annual Meeting.

Más, R., and Pinto. 2003. A Mixed-Integer Optimization Strategy for Oil Supply in Distribution Complexes, Optimization and Engineering, vol. 4, no. 1-2, pp. 23 64, Jun. 2003.

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

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

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

Simpson, R. M., 2005. "GIPO Graphical Interface for Planning with Objects", ICAPS 2005 Competition on Knowledge Engineering for Planning and Scheduling, Monterey, California, USA.

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

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

8. Responsibility notice

The authors are the only responsible for the printed material included in this paper