itSIMPLE$_{2.0}$: An Integrated Tool for Designing Planning Domains

Tiago S. Vaquero[1] and **Victor Romero**[1] and **Fernando M. Sette**[1]
Flavio Tonidandel[2] and **José Reinaldo Silva**[1]
[1]Escola Politécnica - Universidade de São Paulo
Design Lab, Mechatronic and Mechanical Systems Department - São Paulo, Brazil
[2]Centro Universitário da FEI
IAAA - Artificial Intelligence Applied in Automation Lab - São Bernardo do Campo, Brazil
{tiago.vaquero, victor.romero, fernando.sette}@poli.usp.br, flaviot@fei.edu.br, reinaldo@usp.br

Abstract

A great effort 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 the evolution of the tool itSIMPLE which implements a KE integrated environment where designers can perform knowledge acquisition, domain modeling, domain model analysis, model testing, maintenance and plan analysis processes by using different well-known languages such as UML, Petri Nets, PDDL and XML, each one of them with its best contribution. The tool supports users in an organized object-oriented domain design process with a friendly and easy-to-use interface.

Introduction

The rising demand for reliable planning systems has become a great motivation to apply all developments already achieved in real life applications. This scenario leads to the need of a systematic design process, in which the initial phases are not neglected and where Knowledge and Requirement Engineering tools and methodologies have a fundamental role for supporting designers. These concepts have been addressed in the planning community in several initiatives, such as the first International Competition on Knowledge Engineering for Planning and Scheduling - ICKEPS 2005. This competition has brought extremely important knowledge and design engineering issues and showed powerful tools, such as itSIMPLE (Vaquero, Tonidandel, & Silva 2005), ModPlan (Edelkamp & Mehler 2005) and GIPO (Simpson 2005), all of them assisting designers to better understand, specify, visualize, verify and validate their planning domain models.

The itSIMPLE tool was designed to give support to users during the construction of a planning domain application mainly in the initial stages of the design life cycle. These initial stages encompass processes such as domain specification, modeling, analysis, model testing and maintenance, all of them crucial for the success of the application. The evolution of the itSIMPLE, called itSIMPLE$_{2.0}$, 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 & Long 2003), each one of them with its best contribution to the whole design process, leading designers from the informality of real world requirements to formal domain models. Starting with requirements elicitation, specification and modeling, itSIMPLE 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 (from different viewpoints) as well as to the domain model visualization and verification. itSIMPLE$_{2.0}$ focuses also on the use of Petri Nets for dynamic domain analysis since it is a formalism with great potential for model checking and simulation.

itSIMPLE$_{2.0}$ is an open source project implemented in Java that provides a user-friendly GUI to model and analyze many planning domains at the same time. This fact usually contributes to domain model reusability and maintenance.

This paper is organized as follows: First, we describe the itSIMPLE$_{2.0}$ Environment and then we give a brief explanation about the integrated languages available. Next, we present each design process stage encompassed by the new itSIMPLE$_{2.0}$ environment such as requirements, modeling, model analysis, model testing and plan analysis. The paper ends with a conclusions for this work.

The itSIMPLE$_{2.0}$ 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 design process to create knowledge intensive models for several planning domains. The suggested process for designing planning domains follows a cyclic sequence of phases inherited from Software Engineering applications combined with modeling experiences acquired in the design of planning domain. Such process is shown in Figure 1.

The environment was designed to incorporate a toolset (representation languages and theories) capable of dealing with requirements and knowledge engineering as shown in

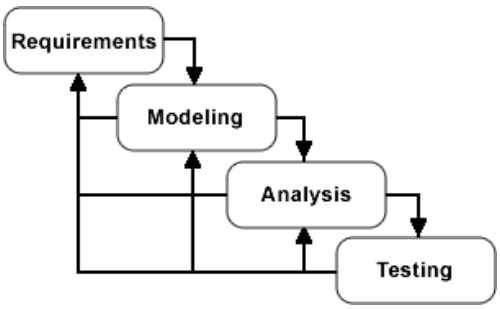

Figure 1: Planning domain design processes in itSIMPLE$_{2.0}$

Figure 1. 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. The environment also provides the use of Petri Nets (Murata 1989), a formal representation that can help designers to perform dynamic domain validations deploying visual and animated information to the entire dynamic system. Also, since the AI Planning community has accepted the PDDL as the standard specification language for planner inputs, itSIMPLE 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 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 2002) - for PDDL. In fact, we believe 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 2.

Figure 2: The architecture of the integrated languages

By using itSIMPLE$_{2.0}$ translators, designers are able to change from one language to another at 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 present each main phase of the domain design process illustrated in Figure 1 using the integrated environment provided by itSIMPLE$_{2.0}$.

Domain 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 & 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 visualization 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, Tonidandel, & Silva 2005). For instance, some of the UML diagrams can be directly applied for planning domains such as use case diagram, 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. Figure 3 shows an example of the use case diagram for the classical planning domain Logistic and Figure 4 shows a screenshot of itSIMPLE$_{2.0}$ of the class diagram from the same domain.

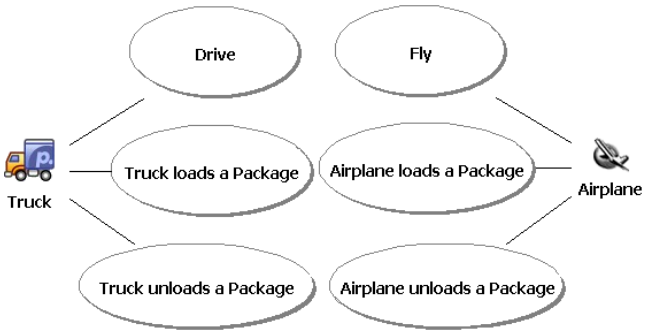

Figure 3: Use case driven approach for domain specification

A problem statement in a planning domain is usually characterized by a situation where only two states are known: the initial and goal state. The diagram used to describe these states is the object diagram or Snapshots. A good definition

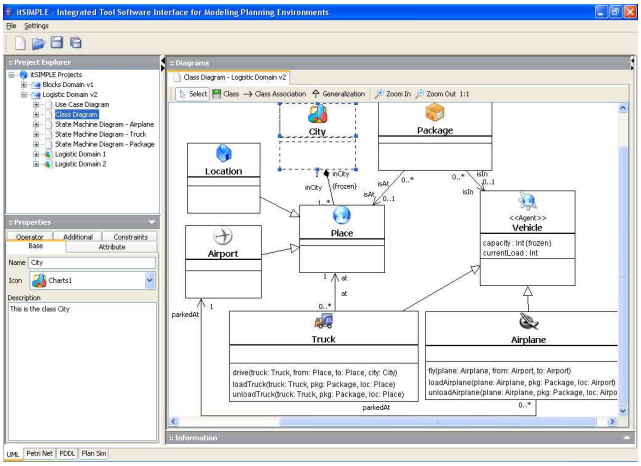

Figure 4: itSIMPLE$_{2.0}$ interface for modeling with UML.P

of Snapshot is given as 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 & Wills 1999). The itSIMPLE$_{2.0}$ tool assists designer to specify consistent snapshots by promoting automatic verification on all the constraints and features defined in the class diagram. Figure 5 shows a screenshot with an example of an initial snapshot of a Logistic problem.

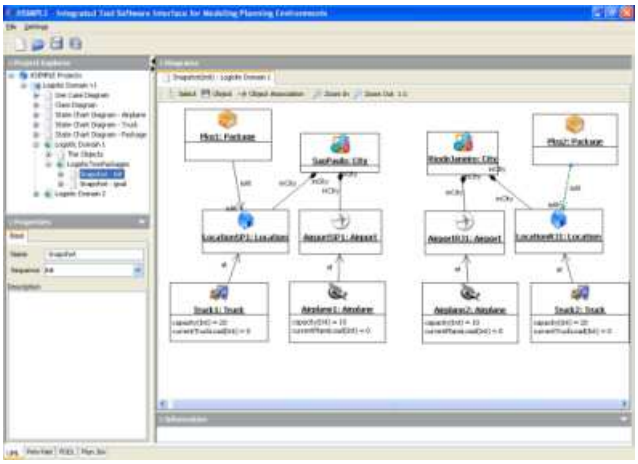

Figure 5: The initial state of a Logistic problem

Another important feature available in itSIMPLE$_{2.0}$ is the definition of intermediate states (snapshots) in a problem. These intermediate states can represent situations that must be avoided or some kind of state trajectory constraints during the planning process. This capability follows the concepts introduced in the definition of PDDL3 (Gerevini & Long 2006). Thus, the use of itSIMPLE$_{2.0}$ make it possible to model situations that must happen always, sometimes, at-most-once or never during the evaluation of a planning problem. This particular tool capability can be a powerful modeling characteristic for many real planning problems. All these constrained situations are considered snapshots in the tool environment.

Domain Model Analysis with Petri Nets

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.* 2006), 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. Figure 6 shows a screenshot with the state chart diagram representing the dynamic features of the class package from the Logistic domain.

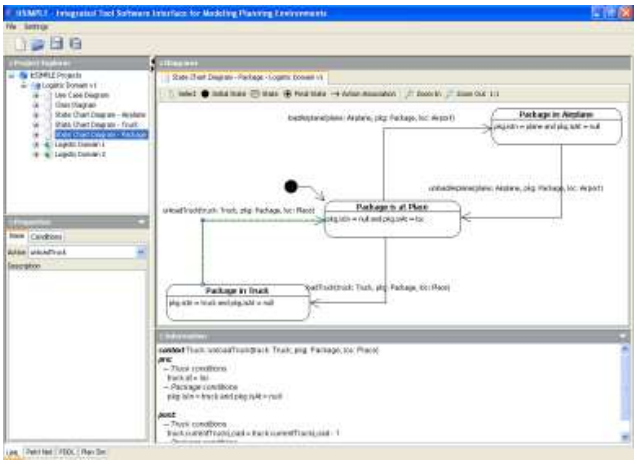

Figure 6: Interface for dealing with state chart diagrams

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 & 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*. The *Modular Analysis* tries to verify each module one-by-one, taking into account the dependencies with others modules (represented by what we call Gates). The modular analysis through Petri Nets allows the designer to detect features like parallelism, concurrencies and undesirable deadlocks throughout simulation processes.The *Interface Analysis*, on the other hand, has the purpose of investigating dynamic interactions among modules. During this analysis, designers can verify not just

one module but many of them, all together, visualizing the dependencies among classes during actions domain executions. itSIMPLE$_{2.0}$'s interface for dealing with dynamic domain analysis with Petri Nets is shown in Figure 7.

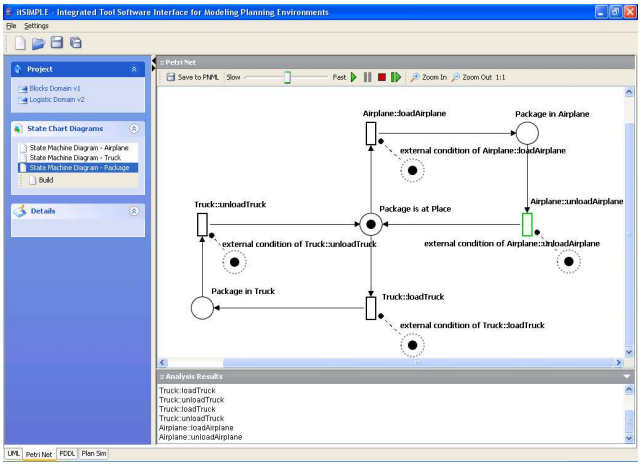

Figure 7: The dynamic analysis with Petri Nets interface

Model Testing and Plan Analysis

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 & Long 2006) 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 8.

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.

The tool provides two ways of performing plan analysis. The first one is made through the use of XY and Gantt charts

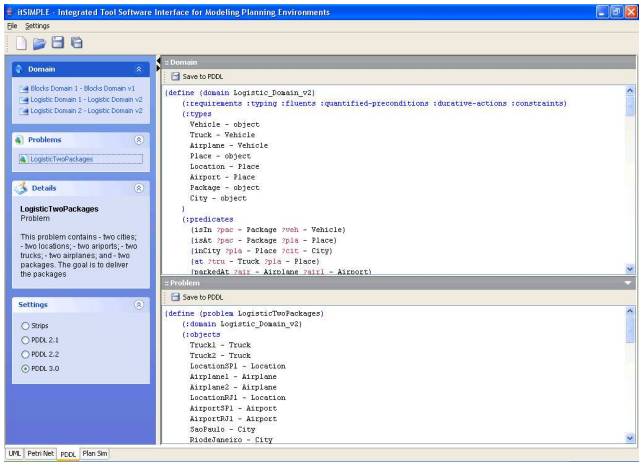

Figure 8: Interface for dealing with PDDL representation

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. We call this second process *Movie Maker*. itSIMPLE$_{2.0}$'s interface for dealing with Plan Analysis is shown in Figure 9.

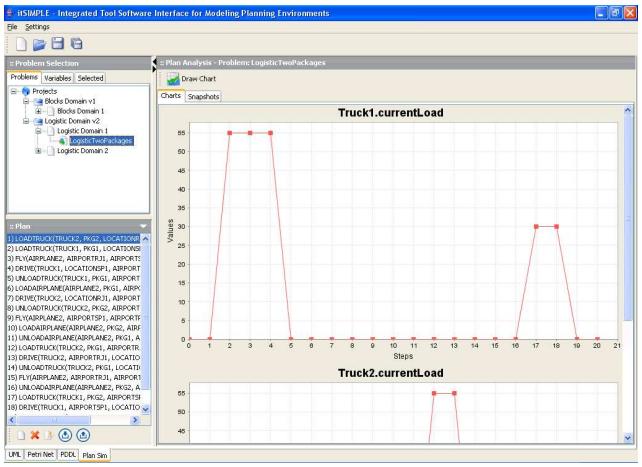

Figure 9: itSIMPLE's Plan Analysis interface

The user can also define its own plan for Plan Analysis by giving it to itSIMPLE$_{2.0}$, action by action. Some works also addressed the relevance of this flexibility (Simpson 2005). In fact, the designers usually refine the model while performing such analysis by including additional constraints into the model for avoiding, for example, undesirable states. In the *Movie Maker* analysis the user observe the screenshots as UML snapshot diagram going forward or backward when necessary.

Conclusion

The idea of a Knowledge Engineering environment for designing 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 design life cycle for real planning problems based on the use of well-known languages and tools utilized in real domain specification in an object-oriented approach. All remaining features could be composed and unified, once they could be expressed in a language that is compatible with XML.

Such flexibility of itSIMPLE$_{2.0}$ makes it an interesting environment for gathering requirements, domain modeling, model analysis, model testing and also as a testbed for new formal representations for planning.

itSIMPLE$_{2.0}$ is in constant improvement seeking for requirements and knowledge engineering aspects for bridging the gap between real planning problems and available planning techniques where it is not possible to neglect the initial phase of the design process. 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 and pass it to the planning systems.

The itSIMPLE tool is available in the projects link on the DesignLab web page http://designlab.grupoe.com.br.

References

Billington, J.; Christensen, S.; van Hee, K.; Kindler, E.; Kummer, O.; Petrucci, L.; Post, R.; Stehno, C.; and Weber, M. 2003. The petri net markup language: concepts, technology, and tools. In *Proceedings of the 24th Int Conf on Application and Theory of Petri Nets, LNCS 2679, Springer*, 483–505.

Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.; and Yergeau, F. 2004. Extensible Markup Language (XML) 1.0 (Third Edition). Technical report.

D'Souza, D. F., and Wills, A. C. 1999. *Objects, components, and frameworks with UML: the catalysis approach*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Edelkamp, S., and Mehler, T. 2005. Knowledge acquisition and knowledge engineering in the modplan workbench. In *Proceedings of the First International Competition on Knowledge Engineering for AI Planning, Monterey, California, USA*.

Fox, M., and Long, D. 2003. Pddl2.1: An extension of pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)* 20:61–124.

Gerevini, A., and Long, D. 2006. Preferences and soft constraints in pddl3. In Gerevini, A., and Long, D., eds., *Proceedings of ICAPS workshop on Planning with Preferences and Soft Constraints*, 46–53.

Gough, J. 2002. Xpddl 0.1b: A xml version of pddl.

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.

Murata, T. 1989. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, 541–580.

OMG. 2001. *OMG Unified Modeling Language Specification, m Version 1.4*.

Simpson, R. M. 2005. Gipo graphical interface for planning with objects. In *Proceedings of the First International Competition on Knowledge Engineering for AI Planning, Monterey, 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)*, 434–437.

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, Californiа, USA*.