itSIMPLE4.0: Enhancing the Modeling Experience of Planning Problems

Tiago Vaquero[1] and **Rosimarci Tonaco**[2] and **Gustavo Costa**[2]
Flavio Tonidandel[3] and **José Reinaldo Silva**[2] and **J. Christopher Beck**[1]
[1]Department of Mechanical & Industrial Engineering, University of Toronto, Canada
[2]Department of Mechatronics Engineering, University of São Paulo, Brazil
[3]IAAA Lab, University Center of FEI - São Bernardo do Campo, Brazil
{tvaquero,jcb}@mie.utoronto.ca, {rosimarci, gustavorochacosta, reinaldo}@usp.br, flaviot@fei.edu.br

Introduction

The itSIMPLE project (Vaquero et al. 2007; 2009) is a research effort to develop a reliable knowledge engineering (KE) environment to support the design of AI planning applications. Unlike other KE tools for AI planning, itSIMPLE focuses on the initial phases of a disciplined design cycle, facilitating the transition of requirements to formal specifications. Requirements are gathered and modeled using *Unified Modeling Language* (UML) (OMG 2005) to specify, visualize, modify, construct and document domains or artifacts, in an object-oriented approach. A second representation, Petri Nets (PN) (Rozemberg and Engelfriet 1998; Murata 1989), is automatically generated from the UML model and used to analyze dynamic aspects of the requirements such as deadlocks and invariants. A third representation, Planning Domain Description Language (PDDL) (Gerevini and Long 2006), is also automatically generated in order to input the planning domain and instance into an automated planner.

itSIMPLE's framework and translators reduce the gap between real planning applications which are seldom represented directly in PDDL and state-of-the-art AI planners. itSIMPLE is an open-source, Java-based system that has been applied to several real planning applications since 2005, including petroleum supply port management (Sette et al. 2008), project management (Udo et al. 2008), advanced manufacturing (Vaquero et al. 2006), information systems, and intelligent logistics systems.

In this paper we describe the new features implemented in version 4.0 of the itSIMPLE system. These new features aim to enhance the modeling experience and provide designers with extra tools to facilitate the model creation process, from knowledge acquisition and modeling to plan generation and analysis.

In what follows, we give a brief overview of the design environment of itSIMPLE, sketching each available phase of a design process, and the framework that integrates a set of languages and formalisms used during the design process. Next, we describe the new features of the tool (version 4.0) and some future directions.

The Design Environment: Towards a Disciplined Modeling Process for Planning

A completely formal design process is not possible since it starts, by definition, with non-formalized, and perhaps tacit, knowledge. itSIMPLE is divided into four primary phases, which may be re-entered multiple times in an iterative fashion and that define a design process which aims to capture the essence of non-formalized knowledge and transform it into a formal description of the planning domain. Each phase is described below.

Requirements Elicitation and Modeling

Any planning system is embedded in a real environment, that is, a myriad of "non-system" tools, objects, people, and processes with which it must interact. It is necessary to have a detailed model of this domain environment and it is important that this model be developed independently of the planning system (McDermott 1981) based on the concept of a *work domain* from Cognitive Design (Naikar, Hopcroft, and Moylan 2005).

In itSIMPLE, requirements and knowledge are gathered and modeled in an object-oriented fashion using a suite of UML diagrams: class, state-machine, timing, and object diagrams. The UML diagrams are used to represent the main aspects of the domain objects in the work domain and planning problem such as static information (objects and agents, defined by classes and relations), dynamic information (state transitions and features changed by actions) and problem instance description (snapshots of objects and relationships describing the initial state, goal state and desirable intermediate states for instance).

Domain Analysis

The Domain Analysis phase is based on static information analysis and validation of dynamics using a state-transition approach. Static analysis is performed by creating snapshots and possible scenarios (using object diagrams) based on the class diagrams and all constraints defined on them (Vaquero et al. 2007). Dynamic analysis is performed by simulation of Petri Nets created from UML models.

Plan Development

After modeling and analyzing the domain, itSIMPLE uses PDDL to communicate the model to solvers, including Metric-FF, FF, SGPlan, MIPS-xxl, LPG-TD, LPG, hspsp, SATPlan, Plan-A, Blackbox, MaxPlan, LPRPG, POPF, and Marvin. Designers can run these planners and obtain the resulting plans to be analyzed, all in the same environment. This feature gives itSIMPLE a significant flexibility to exploit recent advances in solver technology.

Plan Analysis

itSIMPLE provides some functionality for plan analysis, including plan visualization and plan simulation. A plan visualization and simulation is provided by a functionality called "Movie Maker" (Vaquero et al. 2007). This functionality captures the model of a domain and the plan specification in PDDL and shows the simulation of interactions between the plan and the domain through a sequence of object diagrams. Another important functionality is to be able to analyze plans according to domain variables, called "Variable Tracking".

Language Framework

The language framework of itSIMPLE was designed to be flexible and open, allowing different languages to be added. In order to integrate the languages and phases noted above, the environment uses extensible markup language (XML) (Bray et al. 2004) as a core meta-language. More details about the translation processes can be found in (Vaquero et al. 2009).

The New Features and Improvements

In this section we describe the new developed features in itSIMPLE4.0.

PDDL from Start to End, If Needed

In order to support planning experts, we have expanded itSIMPLE's framework to allow the creation of PDDL models from scratch. We have designed an initial PDDL editor so that user can load, edit and save PDDL files with domain and problems descriptions.

PDDL files from IPC or others sources can be easily loaded in itSIMPLE. The tool imports and automatically separates the elements of domain and problems contained in the file. The user can directly edit the domain model and problem description in the itSIMPLE interface. The PDDL editor in itSIMPLE also allows the user incorporate new actions, predicates, goals, initial states and many others PDDL components by using templates. The editor has a syntax highlighting that differentiates language elements by color. After editing, users can use any planner integrated with the itSIMPLE tool and analyze the generated plan.

Using Modeling Patterns

In domain-independent planning, some researchers have investigated the identification of common structures in the domain model in order to trigger a more appropriated planning strategy (Long and Fox 2000; Clark 2001). Long & Fox (2000) studied the detection of patterns (model symmetry and generic types) in the knowledge model, focusing on enhancing the performance of automated planners (e.g., STAN) by exploiting specialized techniques when particular structures are identified. Patterns have been found in several planning problems that have *transportation* (Long and Fox 2000) or *construction* (Clark 2001) characteristics. In (Long and Fox 2000), the authors characterize the main types existing in transportation problems, including, for instance, *mobile* (types of objects that move on a map), *location*, *portable* (type of object that are carried by mobiles on a map) and a hierarchy of other mobile-related generic types (Long and Fox 2000). Simpson *et al.* (2002) have made these patterns (generic types) available in the modeling tool GIPO for domain knowledge designers. Their work has provided an initial pattern language for AI planning problems for the mobile-related patterns. To our knowledge, however, none of this work has provided a definition, description format or catalog of design patterns for AI planning. Our work in this direction can be seen as a continuation of the work done by Simpson *et al.* (2002). However, our long-term goal is to (1) provide such a definition and description format of design patterns for AI Planning in a object-oriented fashion and (2) propose an initial design patterns catalog.

As a step toward our goal to provide modeling patterns to users, we have designed an initial (small) set of patterns and made them available for the user in UML. We provide a description of the intention of the pattern, scenarios, applicability, and the model representation using UML. We followed a simple approach on this version of the system in which users can import a pattern as a predefined set of UML diagrams (classes, constraints and action definitions). Based on previous work and analysis of benchmark and real planning problems we provide the following initial set of patterns:

- *Move & Reach*: a basic pattern that can be used to represent objects that can move and must reach certain positions or locations on a map. The pattern encapsulates the behavior of two classes of objects (roles): Mobile (agent) and Location. The common behavior of mobile types has already been observed and defined in (Long and Fox 2000). Here we are representing them in a object-oriented fashion with UML.
- *Transportation*: a pattern that represents agent objects (carriers) that can carry cargo items (portables) between locations on a map. The pattern encapsulates the behavior of the classes Carrier (agent), Cargo and Location. The common behavior of carriers and portable types has also been observed and defined in (Long and Fox 2000).
- *Stack & Place*: a pattern that can be used to represent an agent object that can pile up, fit and organize item objects on a surface or in a container. The pattern encapsulates the behavior of the classes Stacker (agent), Item and Surface (e.g., grid).
- *Assembling*: a pattern representing agent objects that must compose or decompose parts to create structures or other composed parts. Such (dis)assembling process has to fol-

low an order. The pattern encapsulates the behavior of the classes Assembler (agent) and Part.

Time-based Models

We have recently (since version 3.5) added features to itSIMPLE to allow the representation of some time constraints. The tool provides timing diagrams to describe how (boolean) properties of objects change during the execution of an action, i.e., defining if a property becomes true at the beginning or at the end of a durative action. These diagrams are used to translate the conditions and effects of the actions to PDDL while assigning the right temporal operator to them (e.g., *at start*, *at end*, *over all*) (Fox and Long 2003). However, the tool did not cover the spectrum of timing constraints expressible in PDDL. In the new version of itSIMPLE, we have refined the way users input time constraints. When using a timing diagram, users are now able to specify whether the diagram represents the effect or the precondition of the action (the system then translates the conditions to PDDL with the right temporal operator). The use of timing diagrams in the tool is still restricted to an action horizon (as opposed to multiple actions or a series of actions) and to boolean properties. However, we have made another extension to overcome the latter problem and to open ways to a more elaborated definition of time constraints in the actions' conditions and effects. Since users define pre- and post-conditions with the Object Constraint Language (OCL) (OMG 2003) in itSIMPLE, we let them index, or annotate, their OCL sentences with the desired temporal interval (e.g., [t1,t2] or (t1,t2]). For example, users can annotate a precondition sentence such as '*truck.at = loc and pkg.in = truck*' with the temporal interval [0,10], meaning that the condition must be true from time point 0 to 10 after an action has started. It is also possible to use reserved words such as *start* and *end* in the interval (e.g., [*start*,*start*] or [*start*,*end*] or [*end*,*end*]) to annotate any sentence in the precondition and effect (note that a precondition annotated with (start,end) or [start,end] would be translated to PDDL using the *over all* temporal operator). This extension is just one more step to translating timing constraints to PDDL. We are working on other approaches, such as timelines, to provide more modeling capabilities for time-based models.

Wizards for State Creation

We have been designing features to facilitate the task of creating large scale problem instances using UML object diagrams. We have developed wizards that can reduce the time spent creating the initial state, goal state, or any intermediary preferable state as snapshots. In many cases, users have to specify and input several pieces of data, for example about the distance between locations. Depending on the number of locations the model contains, it can be time consuming to put this information in the model. itSIMPLE allows the user to import this information from a file or even link the file in the model so the tool can use it when translating to PDDL and sending the model to a planner. The file format is currently limited to text files with each record in each line. However, we plan to expand this idea to cover input data from other types of files and chiefly from databases: users provide the queries and the database access so the tool can take care of retrieving the information from the database.

Creating associations among several objects in a diagram can also be tedious. We have designed a wizard that allows users to select a group of objects and associate them in different ways. For example, users can associate all of them, associate just the neighbors, just the object in the right, just in the left, and so forth. We have also designed wizards that can create a predefined network of objects. For example, users can create grids of locations or places by providing the size of the grid, the class that represents the nodes in the grid, the association that defines the edges of the network, and the list of names of the generated objects (if necessary). In addition, when selecting more the one object of the same type it is possible to set their attribute values once, without having to define them individually.

Exchanging and Sharing Experimental Setups

As described in previous sections, itSIMPLE is integrated with several automated planners. Users can set up experiments with the available planners, simulating what is done in IPC. In previous versions of the system, designers were able to select their favorite planners and run them over a set of planning problem instances or even over several instances of different domains. In the current version, users are able to store the setup of an experiment in a XML file and share it with others. A stored experimental setup can be used to run the same experiment in different machines. For example, one can design his/her experiment on a laptop, export the setup and run it on a server later. It can also be exchanged and reused among researches and designers. Such a setup stores the information about what planners must be executed, in which order, what needs to be recorded (e.g., runtime, metrics), and the timeouts. Initially, users need to have itSIMPLE installed in the machines that the experiment will run.

Future Directions

In the itSIMPLE project, we aim to provide the ultimate tool for modeling, testing, analyzing and deploying AI planning domain models. itSIMPLE will be extended to deal with semantic analysis of UML and PDDL models, cross-validation among model components and model dynamic simulation through Petri Nets in the future. Many items that lead the tool to encompass all of these features have been implemented, but there are many more to develop. itSIMPLE must be improved with a more complete and accurate time-based modeling, as well as allowing the user to model HTN domains and domains with probabilistic features. HTN domains can be easily modeled by using activity diagrams which are native to UML. Probabilistic features demand an extension of UML: a more detailed analysis of the viability of this direction is needed.

Since itSIMPLE works with UML, it is natural to think that it can deal with design patterns to help the user to model domains by capturing some past solutions to improve the current model. Initial design patterns have been already incorporated in itSIMPLE tool. However, much work is still

needed to reach a full use of design patterns in UML planning models.

itSIMPLE has been remodeled to become more feasible and useful. Some new ways to model and manipulate the planning environment and applications in UML, PDDL or Petri Nets have changed and will be changing more in the near future. All of these modifications aim to provide a tool with more usability and user-friendly features.

Conclusion

itSIMPLE has been developed since 2004 and presented since the first ICKEPS competition in 2005. Since then, we have been working on eliminating remaining gaps between practical applications and planning systems. Many features have been implemented over the last eight years and itSIMPLE is finally becoming a tool that can allow researchers, industrial users, students and other stakeholders to take advantage of recent planning development. The new features described in this paper show that itSIMPLE has reached a mature level with a stable version of UML modeling, domain analysis and planning simulations. It is a user-friendly tool not only for beginners, students and non-planning experts, but also for planning experts.

References

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

Clark, M. 2001. Construction domains: A generic type solved. In *Proceedings of the 20th U.K. Planning and Scheduling Workshop*.

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.

Long, D., and Fox, M. 2000. Automatic Synthesis and use of Generic Types in Planning. In *Artificial Intelligence Planning and Scheduling AIPS-00*, 196–205. Breckenridge, CO: AAAI Press.

McDermott, J. 1981. Domain knowledge and the design process. In *DAC '81: Proceedings of the 18th conference on Design automation*, 580–588. Piscataway, NJ, USA: IEEE Press.

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

Naikar, N.; Hopcroft, R.; and Moylan, A. 2005. Work domain analysis: Theoretical Concepts and Methodology. Technical report.

OMG. 2003. *UML 2.0 OCL Specification m Version 2.0*.

OMG. 2005. *OMG Unified Modeling Language Specification, m Version 2.0*.

Rozemberg, G., and Engelfriet, J. 1998. Elementary net systems. *Lecture Notes in Computer Science* 1491:12–121. Springer.

Sette, F. M.; Vaquero, T. S.; Park, S. W.; and Silva, J. R. 2008. Are automated planners up to solve real problems? In *Proceedings of the 17th World Congress The International Federation of Automatic Control (IFAC'08), Seoul, Korea*, 15817–15824.

Udo, M.; Vaquero, T. S.; Silva, J. R.; and Tonidandel, F. 2008. Lean software development domain. In *Proceedings of ICAPS 2008 Scheduling and Planning Application workshop. Sydney, Australia.*

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

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