Using GORE method for Requirement Engineering of Planning & Scheduling

Javier Martnez Silva
Department of Mechatronics Engineering
University of São Paulo, São Paulo, Brazil,
Professor Morais, 2231

Abstract

The growing interest in the automated planning community seeks for new and better results for real applications. Requirements' analysis is a key issue over design and needs to be enhanced to fit users and stakeholder expectations. Such scenario make the researchers to focus on Knowledge Engineering (KE) applied in elicitation of planning problems and domains - mainly the early stage of the process. This proposal introduces the applying of GORE method for Engineering Requirement of planing domain modeling.

Introduction

The Knowledge Engineering for planning automated collects contributions of crucial works of researchers such as Prof. Thomas Lee McCluskey, who since the early 90s began to publish papers in this field (McCluskey and Porteous 1993). Researching in design process for building knowledge models in real fields with high quality and reliability (McCluskey 2002), and the study of requirement engineering methods applied with existing automated planning techniques are key objectives of this field (Vaquero et al. 2013).

In this scope, O-Plan was one of precursor tool in acquisition and modeling knowledge of planning on an tasks-driven approach. Most recent distribution is a web service, which is used in a wide range of dependent-domain applications (Tate and Dalton 2003).

SIPE involves an ACT (Myers and Wilkins 1997) approach, in which a system is able to give response to events in real time by performing a best possible action. Modeling of all knowledge required to generate plan is possible while external events are occurring.

Both, O-Plan and SIPE planners are predecessors for GIPO, one of the planners registered in literature with mechanisms for acquiring and modeling knowledge of independent-domain applications (Simpson et al. 2001). It address to the syntactic and semantic verification of models, improving the performance of planners; the import and export from domain definitions to PDDL format; integration of planning algorithms jointly with its execution and simulation provides a friendly environment for users. Graphical representation of dynamic objects through state machines (Simpson 2005) (Simpson 2007) is allowed since version III.

ItSIMPLE is another tool based an object-oriented approach helping to designers achieve a detailed model of the domain (Vaquero et al. 2007). The must relevant contribution is the use of Unified Modeling Language (UML) providing diagrams such as use cases, classes, state machines, time and objects. Classes, properties, relationships and constraints are defined in the class diagram, thus are modeling static characteristics of the domain.

Table 1 shows PDDL as domain modeling language by the most of the tools.

Tools	Domain model
DISCOPLAN	PDDL
EUROPA	Action Notation Modeling Language
GIPO	PDDL
FlowOpt	Work-Flow Modeling
itSIMPLE	UML
JABBAH	Business Process Management Notation
ModPlan	PDDL
VIZ	Non-Standard graphical diagrams

Figure 1: Main tools and its approach for modeling problem domain

EUROPA (Extensible Universal Remote Operations Planning Architecture) (Barreiro et al. 2012) was designed to support planning for complex systems, such as spacecraft and rovers, combining two abstractions: Constraints Satisfaction Problems and networks for modeling Simple Temporal Problems. Domain of real problems is modeled by structures called Objects with ANML (Action Notation Modeling Language) as input language on a strategy state / activity (Smith, Frank, and Cushing 2008).

FlowOpt model processes over a workflow approach simplified guiding users to create correct models (Bartak and Cepek 2008); JABBAH combines modeling using BPMN language (Business Process Management Notation) with a workflow approach (Gonzalez 2009); and finally VIZ with a graphic language simple non-standard for describing the planning areas. All these tools enable translation of domain models to standard PDDL language (Vodrazka and Chrpa 2010).

Even PDDL being the must used language, PLEXIL (Plan Execution Interchange Language) is another language which is originally developed as a collaborative effort between NASA researchers and Carnegie Mellon University, for plans representation on real or simulated systems, in robotics, automation of operations in human habitats and systems involving intelligent software agents.(Biatek et al. 2014).

Of all these tools, itSIMPLE was the first to introduce requirement engineering techniques applied to planning problems (Vaquero and Silva 2009): requirements and relevant knowledge of the different viewpoints involved are represented by the diagram of use cases, a semi-formal representation of the UML language (OMG 2005).

GORE methods

Goal-Oriented Requirement Engineering (GORE) is a subarea of Requirement Engineering (RE), which addresses using of goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements (Van Lamsweerde 2000).

In the literature are registers a wide number of *goal* definitions: Goals as high level objectives of business, organization or system; they capture the reasons why a system is needed and guide decisions in various levels within the enterprise (Potts, Takahashi, and Antón 1994). A goal is a condition or state which engaging issues of the world achieved by an agent (Van Lamsweerde 2000)(Yu, Dubois, and Mylopoulos 1995). According to (Lamsweerde 2004) a goal is a prescriptive statement declaring the purpose of some (existing or to-be) systems whose satisfaction generally emerged from collaboration between agents with some responsibility over the system. These goals guide requirement elaboration process resulting in the definition of domain-specific requirements.

Goals cover different kinds of concerns: functional concerns associated with services to be requested, and non-functional concerns with quality of service (safety, security, accuracy, performance, and so forth) (Chung et al. 2012). Also goal issues are defined in Artificial Intelligence field - specifically in classical planning & scheduling problem in which solution is a sequence of actions (plan) that end in a state entailing some goal previously defined (Russell and Norvig 2010).

GBRAM [Bib19], I* framework (Potts, Takahashi, and Antón 1994), NFR (Chung et al. 2000), KAOS (Van Lamsweerde 2001), Goal/Strategy Map (Bider et al. 2005), GLR (Grigorev and Kirilenko 2013) are some methods based on GORE. Of these, KAOS and I* are the most cited.

KAOS method

KAOS approach is an goal-oriented implementation of GORE method which involves a rich set of formal analysis techniques based on Linear Time Logic (LTL). Indeed, KAOS stands for *Keep All Objectives Satisfied*(Lamsweerde 2009), describing a multi-paradigm framework that combines different levels of reasoning: semi-formal, for modeling and structuring goals; and formal, based in the linear time logic formalism. Therefore, KAOS combines a semantic net of basic concepts such as assumptions, operations, objects and agents, with linear time logic(Lamsweerde 2009).

Basically, KAOS is a goal-driven elaboration method that provides a specification language for capturing *WHY*, *WHO* and *WHEN* aspects in addition to the usual *WHAT* requirements.

Graphically, goals are represented in KAOS diagram by parallelograms, while requirement borders are drawn in bold line and agents are represented by hexagons as in Fig.2, goal diagram for block world problem (Russell and Norvig 2010).

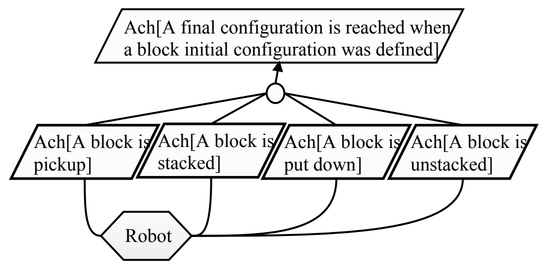

Figure 2: Goal Diagram of simple block world

I* method

I* method is a conceptual modeling technique for modeling and reasoning organizational environments and their information systems introduced by (Yu 1994). Strategic Dependency and Strategic Rationale models are key models of I* approach. Strategic Dependency model describes relationships of dependency among various agents over organizational environment. Strategic Rationale model describes stakeholders' concerns and viewpoints over the system and environment.

Fig.3 shows Strategic Dependency diagram of the same block problem. Analyzing of this model, shows how actors are key strategic into I* for representing motivations, intents and rationale behind actions to achieve goals. Next, we comparing both approaches with focus on modeling non functional requirements (NFR); understanding a boundary between system and it environment; modeling of objects involved in knowledge domain and actor's concerns.

KAOS versus I*

NFR in KAOS are mainly treated as goals and I* models as soft-goals which allowing for qualitative reasoning. Same mission of goal model in KAOS, help to clarify a boundary between a software-to-be and its environment. This model is organized in a tree where leaves are assignable to single agents (software or human). If leaves are assigned to the software-to-be, model as a requirement and if is assigned to environment agent is an assumption, respectively.

Objects related with knowledge domain can be modeled through of one of KAOS diagrams: Object Diagram. In I*

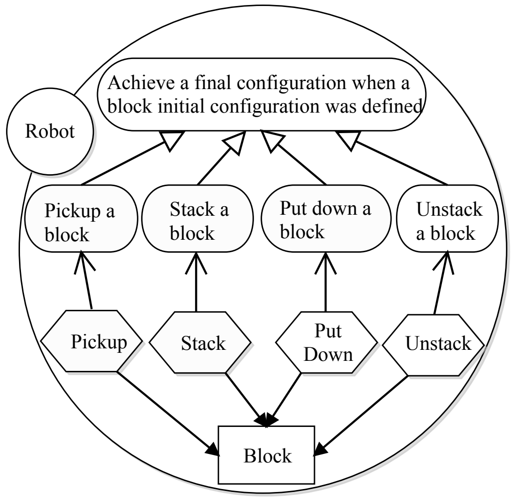

Figure 3: Strategic Dependency Diagram of simple block World

are represented by resources as dependence between agents.

Fig.4 shows Object Diagram from goals modeled in Goal Diagram.

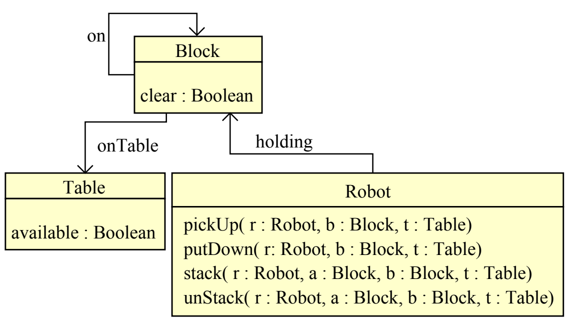

Figure 4: Object Diagram of simple block world

Another advantage of KAOS method is its potentiality of express to behavioral goals a formal representation in Linear Time Logic (LTL) formalism:

Goal Achieve A block is stacked
Def. Formal $\forall(a, b : Block)$
$[stacked(a, b) \rightarrow \Diamond(On(a, b),$
$b.clear = false)]$

Where goals are expressed formally as:

$$C \Rightarrow \Theta T$$

C is the current condition, T is the target condition and Θ is one of the LTL operators represented in Table1.

These operators can be quantified by a time stamp d , so that $\Diamond_d$ means *eventually* in the future before deadline d, and $\Box_d$ means *always* in the future up to deadline d.

Table 1: Temporal Logic Operators

Operator	Description
$\bigcirc$	In the next state
$\Diamond$	Eventually in the future
$\Box$	Always in the future

GORE for planning & scheduling

A key challenge aims to modeling the planning & scheduling problems features through requirements using formal methods - even for medium and large problems - with a schematic language, a first stage of eventual verification and validation from initial models to a consistent model transferable to automatic planners in a later stage.

Our proposal is to provide a clear process for design early stage in which modeling of requirements is a systematic process using GORE method, considering time constraints (such as the duration of sub-processes) and methods for requirement analysis using Petri nets, guaranteeing a consistent input to the automated planners.

Fig.5 provides an overview of steps to follow on the design of planning problem proposed by (Vaquero et al. 2013).

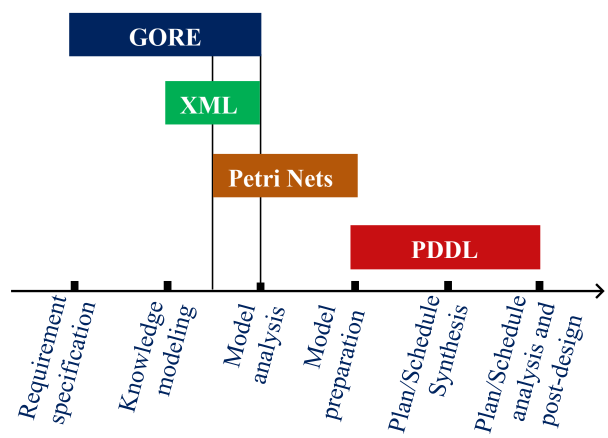

Figure 5: How is added GORE methods over early design phases of planning problems proposed by (Vaquero et al. 2013)

Over these phases, we propose GORE for requirement specification and knowledge modeling stages, and Petri Net formalism in the later stage to modeling and analyzing operations associated to certain goals. XML is proposed for parsing and as starting point in obtaining of PNML(XML alternative for Petri nets).

Improving the design of problems in planning & scheduling- specifically on the early steps of elicitation and analysis of requirements- using GORE (Goal Oriented Requirement Engineering) will provide a comparison study - in terms of formalization - with other approaches as UML.

References

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; et al. 2012. Europa: A platform for ai planning, scheduling, constraint programming, and optimization. In *Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12)–The 4th International Competition on Knowledge Engineering for Planning and Scheduling*.

Bartak, R., and Cepek, O. 2008. Nested precedence networks with alternatives: Recognition, tractability, and models. *Artificial Intelligence: Methodology, Systems, and Applications* 235–246.

Biatek, J.; Whalen, M. W.; Heimdahl, M. P.; Rayadurgam, S.; and Lowry, M. R. 2014. Analysis and testing of plexil plans. In *Proceedings of the 2nd FME Workshop on Formal Methods in Software Engineering*, 52–58. ACM.

Bider, I.; Johannesson, P.; Nurcan, S.; Etien, A.; Kaabi, R.; Zoukar, I.; and Rolland, C. 2005. A strategy driven business process modelling approach. *Business Process Management Journal* 11(6):628–649.

Chung, L.; Nixon, B.; Yu, E.; and Mylopoulos, J. 2000. Nfr in software engineering.

Chung, L.; Nixon, B. A.; Yu, E.; and Mylopoulos, J. 2012. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media.

Gonzalez, A. 2009. JABBAH: A Java application framework for the translation between business process models and HTN. *Association for the Advancement of Artificial Intelligence* 28–37.

Grigorev, S., and Kirilenko, I. 2013. Glr-based abstract parsing. In *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, 5. ACM.

Lamsweerde, A. V. 2004. Elaborating security requirements by construction of intentional anti-models. *Proceedings. 26th International Conference on Software Engineering* (May):148–157.

Lamsweerde, A. v. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*, volume I. Wiley.

McCluskey, L., and Porteous, J. M. 1993. Two Complementary Techniques in Knowledge Compilation for Planning. *In Proceedings of the 3rd International Workshop on Knowledge Compilation and Speedup Learning*.

McCluskey, L. 2002. Knowledge Engineering: Issues for the AI Planning Community. *Proceedings of the AIPS-2002 Workshop on Knowledge Engineering Tools and Techniques for AI Planning, Toulouse, France*.

Myers, K. L., and Wilkins, D. E. 1997. The Act Formalism. *SRI International Artificial Intelligence Center*.

OMG. 2005. UML 2.0 OCL Specification.

Potts, C.; Takahashi, K.; and Antón, A. I. 1994. Inquiry-based requirements analysis. *IEEE software* (2):21–32.

Russell, S., and Norvig, P. 2010. *Artificial Intelligence. A Modern Approach*. Pearson Education, Inc, third edit edition.

Simpson, R.; McCluskey, L.; Zhao, W.; S, A. R.; and Doniat, C. 2001. GIPO: an integrated graphical tool to support knowledge engineering in AI planning. *Proceedings in European Conference on Planning (ECP-2001).*

Simpson, R. M. 2005. Gipo graphical interface for planning with objects. *International Competition on Knowledge Engineering for Planning and Scheduling* 34–41.

Simpson, R. 2007. Structural domain definition using GIPO IV. *In Proc. 2nd Int. Competition on Knowledge Engineering for Planning and Scheduling* 3(Hoffmann).

Smith, D.; Frank, J.; and Cushing, W. 2008. The anml language. *Proceedings of ICAPS-08.*

Tate, A., and Dalton, J. 2003. O-Plan: a Common Lisp planning web service. *Proceedings of the International Lisp Conference*.

Van Lamsweerde, A. 2000. Requirements engineering in the year 00: a research perspective. In *Proceedings of the 22nd international conference on Software engineering*, 5–19. ACM.

Van Lamsweerde, A. 2001. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, 249–262. IEEE.

Vaquero, T. S., and Silva, R. 2009. From Requirements and Analysis to PDDL in itSIMPLE3.0. *Proceedings of the* ....

Vaquero, T.; Romero, V.; Tonidandel, F.; and Silva, R. 2007. itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains.

Vaquero, T. S.; Beck, J. C.; McCluskey, L.; and Silva, R. 2013. Knowledge Engineering for Planning & Scheduling: Tools and Methods. 1:1–15.

Vodrazka, J., and Chrpa, L. 2010. Visual design of planning domains. *In: Proceedings of ICAPS 2010 workshop on Scheduling and Knowledge Engineering for Planning and Scheduling (KEPS)* 2–3.

Yu, E.; Dubois, E.; and Mylopoulos, J. 1995. From organization models to system requirements. a cooperating agents approach. In *in: 3rd Intl. Conf. on Cooperative Inf. Sys.(CoopIS-95.* Citeseer.

Yu, E. 1994. *Modeling strategic actor relationships for process reengineering*. Ph.D. Dissertation, PhD Thesis, University of Toronto.