Abstract— In the field of intelligent design, the early phase, dedicated to requirements modeling and analysis, plays a fundamental role, especially when analytic formal solutions are not suitable. Automated planning falls in that category - particularly when the target are "real world" systems. In requirement analysis Knowledge Engineering is explored to provide clues that can facilitate a convergence for a good planning solution. Therefore, a great effort has been made today in the area of Artificial Intelligence to define a reliable design process for automated planning that includes Knowledge Engineering treatment in the early phase, coupled to requirements modeling and analysis. This paper presents an integrated approach to requirements analysis based on GORE (Goal Oriented Requirements Engineering) that starts by modeling a knowledge architecture based on a domain and planning requirements represented in KAOS and converted in Petri Nets (PN) to analysis. A software tool called rekPlan is proposed to generate the PN graph. The analysis is made in another software tool proposed in our Lab that, GHENeSys (General Hierarchical Enhanced Net System), that support unified Petri Nets following ISO/IEC 15.909. A real case study is presented, based on classic problems of pre-salt petroleum industries.

Keywords— Requirement Engineering, GORE, Petri Net, Automated Planning

1 Introduction

Planning characterizes a specific type of design problem where the purpose is to find an admissible sequence of actions to bring the system from a given initial state to a target final state. Current approaches in the literature aim to improve the performance of automated planners by trying to optimize the search algorithms and the general solution (Lipovetzky and Geffner, 2017). In addition, most existing work on this direction is conceptually tested in synthesized artificial problems (closed problems that have limited set of actions) as. On the other hand, due to the extensive development in this area, some authors started to apply planning techniques to real world problems as well - like logistic problems - with a considerable higher number of variables, where the classic domain independent approaches are computationally prohibitive (Vaquero et al., 2012). Such alternative approach could bring some light and/or good results to challenge problems and could also gave some feedback to solve a fully automated, domain-independent problem.

Indeed, complex domains are hard to deal with when no abstraction is present. In these domains, a hierarchical decomposition based on the topological structure can lead to significantly better performance. Our preliminary case study, (ROADEF 2005), is based on a synthesized domains proposed by the automotive industry but shows an impressive gain of performance when hierarchical modeling are introduced in the design process.

In a standard planning domain such as Logistics, topological abstraction of the real world is key issue. In the classic Logistic problem several packages have to be transported from an initial location to a target destination. The Logistics problem proposed in ROADEF has a map of cities connected by airline routes. Transportation inside cities uses a truck (there is one truck in each city). Cities are abstracted, being treated as destination points. Inside a city, a truck can go from any point to any destination at no cost (Lin et al., 2016). However, in the real world, transportation within a city is a sub-problem that can involve considerable costs.

This paper intends to propose a requirement analysis formal procedure, based on hierarchical models, that starts by taking requirements of planning problems represented in KAOS (Keep All Object Satisfied) and proceed to analysis based on classic Petri Nets. This approach were inserted in a knowledge based tool called rekPlan (Requirement Engineering based on KAOS for Planning Problems) framework, that performs the KAOS/GHENeSys net translation (and eventually translates from GHENeSys to Linear Temporal Logic, LTL). Thus, the KAOS diagram is translated to a Petri net through a transference algorithm, proposed by (MARTINEZ SILVA, 2016)

In section 2 we focus on the use of semi-formal and formal methods in requirement analysis and in the challenge of jumping from a potentially inconsistent set of requirements to a stable and consistent new set. Section 3 will show some aspects about the design process to feed automated planners. In the next sections, we introduced KAOS and Petri Nets as candidate to treat a case study of a real problem of logistic in offshore petroleum exploitation, illustrating the importance of including specific domain knowledge to improve design performance. Finally, we analyze the process as a whole and add some concluding remarks and contributions to further works.
2 Requirement Analysis: Semi-formal and Formal Methods

Regardless of how the final specification is made, the success of any project depends on a correct and complete requirements definition. Requirements specification is used as a reference to test and validate what comes out from design and goes to implementation. The essential role of requirements engineering process is to capture a complete representation of system behavior, considering functional and non-functional aspects. Therefore, an inadequate identification of requirements turn out the major cause of system failures.

Many researchers and practitioners have used semi formal techniques to capture system requirements. In that case the analysis and verification phases inherit excessive flexibility and generality, derived from semi-formal approach. On the other hand, such methods are generally intuitive and very easier to use compared to formal methods, where a formal representation gives the support for a sound analysis and verification.

The literature proposes some techniques and methods that can be used in analysis and specification of requirements, but none of them guarantees a complete and consistent representation of these requirements. There are three fundamental problems in the analysis of requirements:

- The detection and analysis of inaccurate and incomplete requirements (Liu and Yen, 1996);
- The detection of inconsistencies and proposition of methods to manage it;
- Creation of a systematic process that takes the requirements informally specified, but already proved consistent, and transform it in formal specification.

In addition to these challenges we can also add the requirements volatility. Important requirements may lose importance during the design process, and may disappear or merge with other requirements. Besides that, the early detection of emerging requirements during the analysis phase is a hard task. Another important challenge is making sure there is a sound mapping between requirements and attributes of the system improving traceability and maintenance (Vaquero, 2011).

Many researchers propose to convert semi formal requirements into a formal representation (Baresi and Pezze, 2001). Among the most widespread formal methods appear Petri nets, as a formal approach to represent and validate requirements based on state space enumeration and property analysis. An alternative and feasible design discipline involves UML to capture semi formal systems requirements which are translated into a hierarchical Petri net to perform the analysis and verification phases. Possible inconsistencies will be detected in this translation and must be fixed.

Currently there are several approaches combining UML and Petri nets or some extensions. Zhao (Zhao et al., 2007) discusses some technical transformation of graphs, which can be used to convert UML diagrams in Petri nets. There are other proposals offering methods to build Petri nets representing systems functional behavior considering sequence, activity, state, or even use case diagrams (Guerra and Lara, 2003), (Denaro and Pezze, 2004), or another more recent which use a Graph Transformation for the analysis of UML activity diagram (Rahmoune et al., 2015).

However the modeling process resulted from this approach is very sensible to requirements analysis. Most difficulties were related to the use of UML redundant diagrams. Therefore, in this paper we will introduce a new method to model and analyze requirements based on a goal-oriented method: KAOS, (Lamsweerde, 2009) (Lapouchian, 2005), still using Petri Nets as a framework for formal requirements analysis avoiding the need to choose a base set of diagrams.

3 Design Process in Automated Planning

The interest to solve real world problems using Automated Planning techniques has been growing significantly in the last few years. In general, the major focus of the planning community is the pursuit planners efficiency neglecting the analysis aspects (Zimmerman and Kambhampati, 2003) (Upal, 2005), and the need to deal with complex real world problems.

To conduct the planning of an activity it is necessary to determine all features of the world in which it is embedded. Also, some restricting factors must be considered, concerning sub systems evolved, internal variables, correlations with other systems, and other constraints. In our approach OCL (Object Constraint Language) is used to express such constraints (del Foyo et al., 2012).

Formal specifications are the target of the whole process and they are expressed in planning by a language specifically created to this kind of problem called Planning Domain Definition Language or PDDL (McDermott, 2003). If the alternative approach is followed, based on UML, the specification language could be SysML(OMG, 2009). To help in the design and the requirement analysis phases, there are frameworks available such as iTSIMPLE (Integrated Tool and Knowledge Interface to the Modeling of Planning Environments) (Vaquero et al., 2005), that focus on the initial design process phases, such as specification and modeling. After design is concluded, it is necessary validate the model and to execute this task it is possible to use a formal representation like Petri nets.
The general purpose of this paper is to present a design process for automated planning systems, that is composed of two layers: 1) one where independent domain methods are applied; and 2) and the other that uses specific knowledge and requirements analysis and apply hierarchical Petri nets to increase the quality of AI planning applications. The challenge here is to discover the best way to insert the specific knowledge in the design process since we are working with three classes of problems where specific knowledge grows fast from benchmark to real world problems.

In the first level the purpose is to obtain a perspective of the model using KAOS (from Goal and Object Diagrams) to model the hierarchical aspects of real world problems. From that model it is possible to suggest a formalization of this structure based on Hierarchical Petri nets using techniques proposed by Silva and Silva (Silva and Silva, 2015). Those properties will serve to analyze similarities, repetitive cycles, invariants and other model properties.

4 KAOS method for the early design

KAOS approach is an goal-oriented implementation of GORE method which involves a rich set of formal analysis techniques based on Linear Temporal Logic (LTL). Goal-Oriented Requirement Engineering (GORE) is a sub-area of Requirement Engineering (RE), which addresses using of goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements (Van Lamsweerde, 2000). GORE methods mission is to justify the “why”, “what” and “who” dimensions, and to ensure that requirements meet the system objectives - including those of safety and security (Lamsweerde, 2009).

For a direct definition of KAOS method, we should first emphasize the notion of goal-driven (Lamsweerde, 2009). A goal is a prescriptive state declaring the purpose of some (existing or to-be) systems whose satisfaction generally arise from the collaboration between agents with some responsibility over the system. Therefore, goals should drive the requirements elicitation process resulting in domain-specific requirements, avoiding a difficult balance between functional and non-functional requirements.

KAOS as GORE method providing mechanisms with basis on formal logic LTL and graphical modeling representations for requirements of a problem in terms of goals. These goals express stakeholders needs and viewpoints of, in conflict or not, but eventually manageable (similar to other representation of requirements). Thus, KAOS is described as multi-paradigm framework combining different levels of reasoning: semi-formal, for modeling and structuring goals; and formal, based in the linear time logic formalism (Lamsweerde, 2009).

Graphically, goals are represented in KAOS diagram by parallelograms, while requirement borders are drawn in bold line and agents are represented by hexagons.

Leafs are defined as requirements, assumptions or domain properties. Depending if they are assigned to the software-to-be or to an environment agent are defined as requirements or assumptions respectively. Descriptive statements on environment, like physical laws, organizational policies are modeled as Domain Properties. Fig.1 shows how graphically are the main elements in KAOS. Goal, requirements and assumptions use parallelograms, but requirement borders are drawn in bold line; assumptions are filled yellow.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Goal to be achievement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>Goal assigned to single agent in software-to-be.</td>
</tr>
<tr>
<td>Expectation</td>
<td>Goal assigned to single agent in environment.</td>
</tr>
<tr>
<td>Domain Properties</td>
<td>Descriptive statements on environment (physical laws, organizational policies).</td>
</tr>
<tr>
<td>Operation</td>
<td>Operation to be realized to achieve a goal, requirement or expectation.</td>
</tr>
<tr>
<td>Agent</td>
<td>Agent responsible for the achievement of any goal, expectation or requirement</td>
</tr>
</tbody>
</table>

Figure 1: Main elements of KAOS diagrams

Goals can be classified according to its behavior, covering a maximal set of intended behaviors in a declarative and implicit policy.

5 Petrobras- Ship Operations on Petroleum Ports and Platforms

The general problem to be solved is based on the transportation and delivery of a list of requested cargo to different locations considering a number of constraints and elements such as available ports, platforms, vessel capacity, weights of cargo items, fuel consumption, available refueling stations in the ocean, different duration of operations, and costs (Vaquero et al., 2012). Given a set of cargo items, the problem is to find a feasible plan that guarantees their delivery while respecting the constraints and requirements of the ship capacities. The objective is to minimize the total amount of fuel used, the size of waiting queues in ports, the number of ships used, the make span of
the schedule and the docking cost. Fig.2 show the KAOS diagram to represent this problem modeled using reKPlan.

Now is possible to perceive a boundary between a planning system and its environment more clearly, through the representation of requirements and assumption. The planning system must be validate and to solve requirements, however the rest of domain information (assumptions and domain properties) don’t be discarded and can bee used as input of any planners.

Another KAOS diagram is the Object model, a mechanism that ensures the completeness of the object model: derives a minimum set of objects for each goal in the knowledge diagram.

A first representation of object model obtained is showed in Fig.3.

One of the contribution of this work is to provide an alternative approach, transferring requirements specifications modeled in KAOS to a more flexible representation which could accept some discrepancies: Petri Nets (High Level net specifically). We use the algorithm introduced by (Silva and Silva, 2015) to make a transference from KAOS diagram to an Unified Petri Net, that is, a net that follows the standard ISO/IEC 15.909 and can also present some extended elements. Such net is implemented in a tool called GHENeSys (General Hierarchical Enhanced Net System)(Silva et al., 2009).

---

**Figure 2:** Goal Diagram of Ship Operations on Petroleum Ports and Platforms Problem modeled in reKPlan tool

**Figure 3:** Objects derived from *Total amount of fuel used is minimized when a order was received* goal formal specification

---

6 **The Extended Unified Petri Nets GHENeSys**

First of all, we will defined a Unified Petri Nets used by the GHENeSys environment as the nets defined on the standard ISO/IEC 15.909-1 (Hillah et al., 2009) to Place/Transition P/T and High Level Nets (HLPN). By construction it is always
possible to build a P/T where property analysis could be done and synthesize an HLPN net that fold symmetries and reduce the graph by adding declarations derived from the ML functional language.

GHENeSys environment uses unified nets (Silva et al., 2009) (Silva and del Foyo, 2012) to formalize the design process of automated systems. As extended elements GHENeSys use hierarchic representation, the concept of gates\(^1\) and a special element, the pseudobox, very interesting to control and represent observable events not controlled by the system being modeled.

The GHENeSys net for Ship Operations on Petroleum Ports and Platforms Problem is showed in Fig.4

![Ship Operations on Petroleum Ports and Platforms Problem](image)

Figure 4: GENHeSys net of Ship Operations on Petroleum Ports and Platforms Problem

Applying analysis techniques over the generated reachability tree of this net, we concluded that:

- All markings are reachable from the initial marking (lifeness property).
- The GHENeSys net is free of deadlock.
- The net is finite because the reachability tree in finite too.

\(^1\) Gates are state elements that signs transitions enabling without moving the marks

7 Conclusion

The first conclusion we can derive from this work is about the efficiency of the combined use of KAOS and Petri nets to capture and analyze requirements in challenge real problems related to planning. Such combination could be used successfully in the design of real problems that demand the use of AI Planning techniques. However, such approach turns to be prohibitive to real problems, which make it attractive to use abstraction and hierarchy both in KAOS and Petri nets. That means more than a representation enhancement but a significant change in the design discipline.

Another important contribution is the insertion of timed constraints from earliest phase of the process through the quantifying the "eventually" and "always" temporal operators over logic formal representation.

We conclude that knowledge Engineering on the large real planning problems would benefit from a goal oriented approach. In practice the process using KAOS, is a lot more direct and agile, compared with the normal process using UML or a similar object-oriented approach. Besides, the formalization of requirements could be anticipated and improve the traceability by using the algorithm of translation presented in this paper.

Further work is addressed to improve hierarchical approach, translating automatically the analyzed requirements to HTN (Hierarchical Task Network) (Nau et al., 2003). Once we deal with extended hierarchical Petri Nets the translations would be direct. Relations between actions will be mapped in relations among proper structured elements making on going validation straight compared with other approaches (Magnaguagno and Meneguzzi, 2017). A step further in knowledge acquisition applied to planning is to investigate the introduction of heuristics and slice temporal dependence between action, and compare our approach (which is based on a Time Petri Net) with similar recent work (Qi et al., 2017) (Lotinac and Jonsson, 2016).

References


