A Quality Framework for Automated Planning Knowledge Models

Mauro Vallati[a] and Thomas Leo Mccluskey[b]
School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield, U.K.

Keywords: Automated Planning, Domain Models, Knowledge Engineering, Quality of Models.

Abstract: Automated planning is a prominent Artificial Intelligence challenge, as well as a requirement for intelligent autonomous agents. A crucial aspect of automated planning is the knowledge model, that includes the relevant aspects of the application domain and of a problem instance to be solved. Despite the fact that the quality of the model has a strong influence on the resulting planning application, the notion of quality for automated planning knowledge models is not well understood, and the engineering process in building such models is still mainly an ad-hoc process. In order to develop systematic processes that support a more comprehensive notion of quality, this paper, building on existing frameworks proposed for general conceptual models, introduces a quality framework specifically focused on automated planning knowledge models.

1 INTRODUCTION

Automated planning is an area within Artificial Intelligence that has matured to such a degree that there exists a wide range of applications utilising planning. Embedding automated planning engines within applications is a specialist activity, as the application requires a *knowledge model* to be built for operation with the planning engine. Planning knowledge models are conceptual models, in that they are explicit (and formal) representations of some proportions of reality as perceived by some actor (Wegner and Goldin, 1999). These models may contain representations of objects, relations, properties, functions, resources, actions, events and processes, in the application domain. There are significant differences between generic conceptual models and planning knowledge models, however, in that the latter is aimed more for its operational value than for its use in interactions and communications with domain experts and other stakeholders.

Up to now, despite the existence of quality frameworks for various conceptual models (see, e.g., (Lindland et al., 1994; Krogstie, 2012; Krogstie et al., 2006)), there has been no overall framework for considering the quality of the various components involved in the life cycle of the planning knowledge model. There has been research into the quality of planning applications in terms of verification and validation (Frank, 2013), and in terms of accuracy and completeness of the knowledge model (McCluskey et al., 2017; Vallati and Kitchin, 2020), but no overall conceptual model covering the many aspects of such models.

In this paper, building on existing frameworks proposed for general conceptual models, we introduce a quality framework specifically focused on automated planning knowledge models. The main benefit of a framework is to replace ad-hoc notions of quality, and ad-hoc knowledge engineering processes, with a connected, composite and over-arching notion, that can be used within all work relating to this endeavour. The proposed framework is exploited for introducing and describing specialised aspects of quality of automated planning knowledge models, which are based on semiotic layers and on planning-specific aspects.

As a first solid contribution to the field, we demonstrate the usefulness of the proposed framework as a means for assessing a range of tools environments for knowledge engineering for automated planning. The exercise highlights areas and processes that the existing tools do not support appropriately, and where future work of the planning community should be focused. Finally, we argue for the importance of a comprehensive quality framework for the planning area, by providing a list of benefits that the framework is likely to bring into the AI planning field. The paper is aimed at giving knowledge engineers a broader perspective on the issues involved in capturing and

[a] https://orcid.org/0000-0002-8429-3570
[b] https://orcid.org/0000-0001-8181-8127

Vallati, M. and Mccluskey, T.
A Quality Framework for Automated Planning Knowledge Models.
DOI: 10.5220/0010216806350644
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 635-644
ISBN: 978-989-758-484-8

maintaining planning knowledge, and to provide AI and planning experts with an over-arching notion of quality, and an example of its usefulness.

The remainder of this paper is organised as follows. First, the required background on automated planning is provided, and the distinctive characteristic of planning knowledge models are then presented. Then, the proposed quality framework is introduced, and the involved quality concepts are then detailed in the subsequent section. After that, it is shown how the quality framework can be concretely applied. Finally, a discussion on the benefits of the proposed framework is provided, and conclusions are given.

2 AUTOMATED PLANNING

Automated planning deals with finding a (partially or totally ordered) sequence of actions transforming the environment from some initial state to a desired goal state (Ghallab et al., 2004). As an example, in the classical STRIPS-inspired representation, *atoms* are predicates and *States* are defined as sets of ground predicates. A *planning operator* $o = (name(o), pre(o), eff^-(o), eff^+(o))$ is specified such that $name(o) = op\_name(x_1, \ldots, x_k)$ (*op_name* is a unique operator name and $x_1, \ldots x_k$ are variable symbols (arguments) appearing in the operator), $pre(o)$ is a set of predicates representing the operator's preconditions, $eff^-(o)$ and $eff^+(o)$ are sets of predicates representing the operator's negative and positive effects. *Actions* are ground instances of planning operators. An action $a = (pre(a), eff^-(a), eff^+(a))$ is *applicable* in a state s if and only if $pre(a) \subseteq s$. Application of a in s (if possible) results in a state $(s \setminus eff^-(a)) \cup eff^+(a)$.

This knowledge is made explicit in two components: a domain model and a problem instance, together forming the knowledge model. When using the dominant family of planning knowledge representation languages – the PDDL, the domain model and problem instance are provided to planners as two different files, and the same domain model is used for all the problems of the application. In the restricted world view of classical planning, a domain model is specified via sets of predicates and planning operators. A *problem instance* is specified via an initial state and set of goal atoms, that need to be reached. A *solution plan* is a sequence of actions such that a consecutive application of the actions in the plan (starting in the initial state) results in a state that satisfies the goal.

The classical planning model can be extended, in order to handle a wider range of constraints and increase expressiveness. On this matter, the interested reader is referred to (Ghallab et al., 2004).

2.1 Peculiarities of Planning Knowledge Models

A major part of the AI planning component is played by the knowledge model, which enables rational planning and reasoning. Therefore, to evaluate the planning component, one needs to be able to discuss and ultimately measure the quality of the constructed model.

In the last decades, a number of frameworks have been introduced for evaluating the quality of conceptual models, and for identifying quality notions and corresponding goals and metrics. The SEQUAL framework (Lindland et al., 1994) was initially introduced as a general framework for understanding quality of any conceptual model, and then extended by adding a larger number of quality aspects (Krogstie et al., 1995; Krogstie and Jørgensen, 2002), or by focusing on specific types of models, such as interactive models (Krogstie et al., 2006) or business process models (Krogstie, 2012), data (Krogstie, 2013), and enterprise models (Krogstie and de Flon Arnesen, 2004).

Essentially, planning knowledge models are conceptual models, in that they are explicit (and formal) representations of some proportions of reality as perceived by some actor (Wegner and Goldin, 1999). Nevertheless, there are significant differences between generic conceptual models and planning knowledge models. Planning knowledge models are usually formulated into a planner input language directly from a requirements specification, which summarises the current understanding and interpretation of the application domain. The planning component is usually a module of a larger framework, that has a very limited interaction with human experts and stakeholders (McCluskey et al., 2017). A good example is that of an autonomous agent, which contains a planning component within a module of the larger (and more complex) architecture, that is supposed to operate autonomously, with no (or limited) needs for humans in the loop. It is of course important that an expert can read and understand the models for validation and maintenance purposes, but the main focus is on effectiveness and efficiency. In fact, once a model has been thoroughly tested and validated, with the support of domain and planning experts, it is usually reformulated for the sake of performance. Automated reformulation of a model, of course, tends to make it more difficult to analyse and interpret the model by human experts.

The main differences between general conceptual models and planning knowledge models can be found in the modelling process. General conceptual modelling processes may have various, and sometimes unclear, goals. The modelling may be performed for representing a process, in order to analyse and improve it, or may aim at generating models that can be shared in order to transfer knowledge between different actors. Instead, in Planning, the modelling has a clear and well defined goal: enabling a planning engine to generate solution plans, given some constraints bounds, that show some desired characteristics. In other words, the quality of a planning model also depends on its ability to allow planning engines to generate solution plans that show some predefined characteristics. Characteristics may vary, according to the expressiveness of the exploited language and the type of planning that is performed. Solution plans may be required to be *optimal*, with regards to some aspects like the number of actions, the overall cost, or the predicted duration.

The aforementioned dependencies, and the a-priori known goal of performing modelling for planning, have to be reflected into quality frameworks that should incorporate both planning engine and solution plan. The planning engine plays an important role in the quality framework because, as routinely observed in International Planning Competitions, different planning approaches show very different behaviours even when provided with the same knowledge model. Furthermore, different planning engines tend to support different subsets, and different versions, of the PDDL language. The selection of the planning engine is therefore an important step of the planning modelling process, as it affects (and is affected) by the encoding language, and by the knowledge model itself.

3 THE QUALITY FRAMEWORK

The proposed quality framework for automated planning knowledge models aims at representing all the aspects that affect the quality of models, and highlighting how the different aspects interact. Figure 1 presents the basic ideas of the quality framework, which originate from the specialisation of the SEQUAL framework and the subsequent work of Krogstie et al (Krogstie et al., 2006). The framework considers different components (represented as boxes in the Figure), and processes (represented as arrows and their annotations in the Figure). The following components are introduced:

L represents the language that is used to encode the model. This can be, for instance, a version of PDDL. In this context, the language is expected to have a well-defined syntax, vocabulary, and operational semantics: this means that, independent of planning engine and application domain, there is a defined process for executing plans which correspond to sequences of actions in the application domain. In other words, there exists a single interpretation of the dynamics of a well-formed knowledge model (McCluskey, 2002).

D is the domain specification, a set of requirements for the application domain at hand. Such requirements can be described informally in diagrams and textual documents, or described (at least in part) in a formal language. The requirements specification would include information about the dynamics of the domain, the kind of problems the planning engine will have to solve, and the kind of plans (solutions) that need to be provided as output.

M represents the model externalisation in a language *L*, that is, a formal specification of the application domain part of the requirements specification. This represents entities invariant over every problem instance, such as object classes, functions, properties and relations; a specification of domain dynamics; and a specification of problem instances that has to be reasoned upon by the planning engine. In other words, this is what we refer to above as the *knowledge model*, around which the quality framework sits.

I is the interpretation of the domain requirements *D*, that is, the internal specification of either a human (expert) or of an automated technique, that results in the generation of the model *M*, in the selected language *L*. In other words, it can be described as the understanding that the interpreter has of the domain requirements. As it is apparent, the internal specification can vary significantly among experts and, as well, among different automated encoding techniques, which may rely on very different technologies in order to be able to provide a complete model externalisation.

K represents the relevant explicit current knowledge that is available to the human expert or the automated technique and that affects the interpretation *I*. Such knowledge should can encoded in a formal or semi-formal way, so that it can be exploited by any interpreter wishing to. This may include, for example, incomplete models previously generated, or textbook knowledge about the dynamics of the domain, or heuristics to be used in the planning engine. It is worth emphasising that this knowledge is not a model externalisation, but it should be understood as a source of knowledge on which an agent relies to create the externalisation.

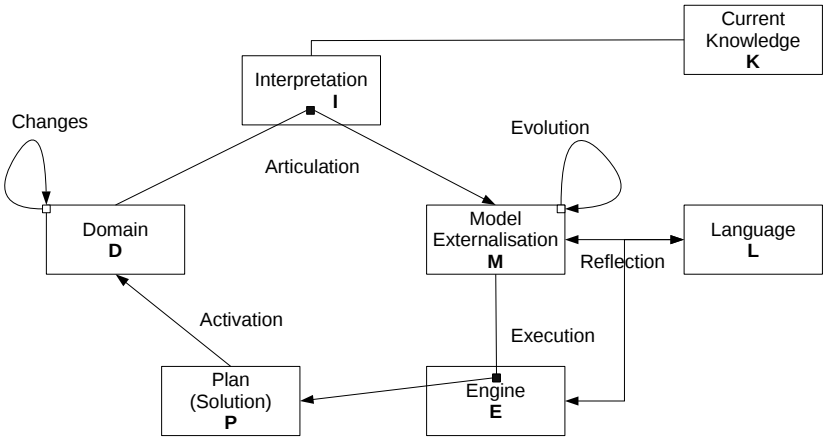

Figure 1: The proposed quality framework for planning models. Boxes are used to represent components, while processes are represented as arrows linking the involved components.

E is the set of rules and techniques exploited by a planning engine in order to generate, given the model externalisation of the domain and of a problem, a solution plan *P*. In fact, *E* is not to be considered as an actual planning engine, rather it should be the description and specification of the engine, and how it will operate on the provided models. It can be, for instance, under the form of pseudo-code, specifying the steps (pre-processing, search, optimisation) and their behaviour.

Finally, *P* stands for the solution plans that can be obtained, using the engine *E* on the model externalisation *M*. *P* has a twofold meaning in this context: it can represent a specific plan, in terms of set of actions to be performed in order to reach the goal of a given planning problem model, but it can also indicate –through a specifically-designed language or formalism– the characteristics of the plans that can be generated.

The framework depicted in Figure 1 includes also the notion of processes. Processes represent interactions between components, that lead to changes in one (or more) of the involved components. In our framework, processes have been named following, to some extent, the existing nomenclature proposed in (Krogstie et al., 2006), appropriately extended and modified for the sake of dealing with planning domain and problem models.

Articulation ($D \rightarrow I \rightarrow M$) is the process where the domain *D* is encoded as a model *M* by means of a specific language *L*. The articulation is performed by an interpreter, on the basis of her interpretation *I* of the domain, and of the available knowledge *K*. No assumptions are made on the nature of the interpreter: the role can be played by a human expert, by an automated technique, or by a mix of those. Similarly, no assumptions are made on the nature of the interpretation *I*; this can be fully formal (i.e., a model itself but in a different formalism) or natural language, or sketches, etc. Notably, this process can be repeated if the interpreter is changed, or the interpretation *I* is updated. This can be the result of improvements in the current knowledge *K* or, for instance, because of a better understanding of the specifications.

The *Reflection* process stands for the impact that a language *L* has on the model externalisation, as well as on the planning engine *E*. The impact on the model is extremely intuitive: different languages provide different expressive power, and different ways for formalising the relevant specification requirements of the domain. In some versions of PDDL, for instance, it is not possible to specify the duration of an action. Furthermore, the language has also a strong impact on the planning engine, as different engines support different languages, or different subsets of the language's features. Therefore, the selection of the language leads to the possibility (or not) of using some planning techniques. Furthermore, similar dynamics can be differently encoded in different languages, with a potentially different impact on the performance of planning engines.

Execution ($M \rightarrow E \rightarrow P$) is the process of generating solution plans, by providing as input to the planning engine *E* the model externalisation *M*. As aforementioned, *P* can therefore represent a single solution plan, or can be specified in terms of characteristics of plans that can be generated via the execution process.

Activation ($P \rightarrow D$) captures the changes that the use of the model may trigger in the domain specification *D*. Changes can include, for instance, the refinement of the domain specification on the grounds of some unexpected results. The work of Vaquero

et al (Vaquero et al., 2013a) on *post-design analysis* falls into this category, where using the model in planning may lead to the discovery of missing requirements and hence lead to changes in D. Differently from interactive and business process models, planning knowledge models are activated indirectly, via solutions P. Therefore the activation is usually less frequent in planning, and it tends to point to changes in the domain that are due to the exploitation of the automated planning technology in the application domain, that may allow to perform actions in a different way. In a given domain, for instance, the use of planning may allow to perform more complex tasks, that were previously impossible to perform. This can then be reflected in changes in the domain specification.

Finally, there are two processes that, respectively, affect the domain D and the model externalisation M, only. The *changes* process incorporates changes to the specification. This can be the case, for instance, of a logistic company that decides to extended its transport fleet by including different kind of vehicles. That would change the domain specification, and would then need to be reflected in the model. The *evolution* process focuses on the evolution of the model M. Evolution can be due to the use of reformulation techniques, that change the model externalisation in order to improve the performance of the engine E. In fact, this process is only aimed at capturing the modifications that are made on a model M to make it more amenable to engines, or easier to read for human experts.

4 QUALITY CONCEPTS

Previous work in the area of knowledge engineering, and particularly knowledge capture, for AI planning seems to point to the direction of a single general quality notion - that of *Semantic Quality* (McCluskey et al., 2017; McCluskey, 2002). However, as pointed out by the strand of research based on the SEQUAL framework (Lindland et al., 1994), that general quality notion cannot be directly evaluated nor measured. It is therefore pivotal to introduce different quality dimensions, that should include aspects and elements that can, in principle, be measured and analysed. An in-depth explanation of quality aspects for conceptual models is provided in (Krogstie et al., 1995). Remarkably, quality levels are defined following the levels of the semiotic ladder (Stamper, 1996). The semiotic ladder introduced six levels, corresponding to different dimensions (either related to the IT platform or to the human society) allowing to describe the properties of 'signs' –in a broader semiotic sense.

In the following we specialise the main quality types in order to fit the needs of planning knowledge models, and expected users and knowledge engineers. Noteworthy, due to the inner aims of planning knowledge models –that are not mainly focused on communicating knowledge, but on allowing the generation of solution plans–, we have to introduce quality aspects that are not covered in the semiotic ladder, and to drop some aspects that are not relevant for the purposes of planning. Quality aspects separate the goals, which represent what this aspect is trying to assess and maximise, from the means for achieving such goals. In other words, the quality aspects are related to the components (from the proposed framework) that are involved in the assessment. There is not a one-to-one connection between the processes presented in the previous section, and the quality aspects. The processes are designed to represent interactions between components, and have to be supported in order to make sure that the delivered changes are beneficial for the involved components; in that, they are indirectly affecting the different quality aspects. However, the notion of quality is inevitably entangled with the actual components, because components are the tangible outcome of the processes.

Physical Quality: this aspect focuses on the availability and accessibility of the model M. Following the existing literature, this quality has two main goals, the externalisation and the internalisability. The former refers to the fact that the model M is an artefact, resulting from the externalisation (in other words, of making explicit) of the interpretation knowledge I of an interpreter, and is based also on the available current knowledge K. Externalisation also covers the fact that the considered application domain can be represented under the form of some symbolic model, specifically using available planning-oriented languages. The internalisability stands for the fact that the model M is persistent and available to interpreters that can understand it and interpret it, and can be used by an appropriate engine E to generate solutions. In other words, the internalisability focuses on the fact that the model is available for the planning engine, as well as to experts that may need to check or revise it. At a first glance this may seem trivial in the typical planning settings, particularly for the domain model part of the knowledge model. However, in terms of problem instances, it may be the case that such problems are automatically generated by combining information gathered from different sources (sensors, data bases, etc.), and the process may be hard to reproduce. This is the case for many planning applications (see, e.g. (Venturelli et al., 2017)). In such scenarios, Internalisability aims also at max-

imising the availability of planning knowledge models, so that future validations and evaluations can be performed.

Syntactic Quality: is probably the easiest quality aspect that can be measured and assessed, as it is aimed at the syntactic correctness of the model M with regards to the selected modelling language L. It is important to remark that planning engines E may add additional constraints on the syntax of the language, due to partial support of some language features, for instance. The planning engine cannot be selected in isolation: the language and the planning engine are affecting each other and, of course, decisions taken with regards to E and L have repercussions on the rest of the modelling process. For this reason, it is necessary to include the engine in the analysis of the syntactic quality.

Semantic Quality: aims at the goals of accuracy and completeness. In this context, we rely on the definitions provided in (McCluskey et al., 2017). In a nutshell, accuracy is focused on relating the model M and the domain specification D, by ensuring that M is a valid representation of the specification, i.e. it encodes all the aspects that are correct and relevant for the domain. Conversely, completeness involves the solution plans P in that it means that M enables the generation of all (and only) solution plans that are correct with regards to the domain specification D.

Pragmatic Quality: this covers how the model externalisation is activated, i.e. the way in which the exploitation of the model, maybe within a larger framework, can affect the domain specification and, in a broader sense, the application domain itself. In principle, activation and articulation can be seen as a co-design cycle, where feedback drives the improvement of the overall understanding of the domain and its specification; and as the model externalisation is evolved and refined, this has potential impacts on the efficiency of the planning engine, on the interpretation, and on current knowledge. To some extent, pragmatic quality is driven by the comprehension of the domain specification, and of the model, by the different stakeholders.

Operational Quality: this quality covers the ability of the selected planning engine E to reason upon the model externalisation M to generate P. This quality aspect incorporates two perspectives. First, the shape of solution plans that E allows to generate. On this matter there may be preferences in terms of number of actions involved, or makespan, or cost of the actions that are considered. It may also be the case that, for the specific application domain, only *optimal* solution plans are acceptable. Second, the resource bounds that can be used by E to solve a problem instance. In this context, acceptable resource bounds can be defined in terms of runtime, memory usage, number of CPUs, etc. Resource bounds can be specified in the domain specification D, or may be derived by the interpretation I, or by the current shared knowledge.

5 EVALUATION OF TOOL SUPPORT ENVIRONMENTS

The proposed quality framework can provide a valuable means for relating existing environments with the process they are aimed to support, and to the quality aspects a specific tool can contribute to. Here we consider a range of knowledge engineering tools environments for planning, focusing on those that lead to the generation of PDDL models. This appears to be a timely exercise, as the last edition of the International Competition on Knowledge Engineering for Planning and Scheduling (Chrpa et al., 2017) highlighted that most teams did not use any knowledge engineering tool (except text editors), and thus relied only on their expertise. Some teams were not aware of the *existence* of such tools support.

We selected four tools environments, aiming at including very different approaches and techniques. The first two tools are well-cited early examples, and the second two are more recently produced environments.

- GIPO: The Graphical Interface for Planning with Objects (Simpson et al., 2007) is based on users encoding in object-centred languages *OCL* and the HTN variant OCL_h, with a translation to PDDL in the background. These formal languages exploit the idea that a set of possible states of objects are defined first, before action (operator) definition. This allows the integration of a number of tools covering automated domain acquisition, consistency checking and plan animation.
- itSIMPLE (Vaquero et al., 2013b) provides a graphical environment that enables knowledge engineers to encode knowledge of the domain by using the Unified Modelling Language (UML). Object classes, predicates, action schema are modelled by UML diagrams allowing users to visually inspect knowledge models. itSIMPLE incorporates a model checking tool based on Petri Nets that are used to check invariants or analyse dynamic aspects of the knowledge models. Standard planning engines are integrated for dynamic testing.

Table 1: Fully and partially supported processes by the considered Knowledge Engineering tools for automated planning.

Tool	Supported Processes	
	Fully	Partially
GIPO	Articulation	Execution, Changes
KEWI	Articulation	Changes
itSIMPLE	Articulation	Execution, Activation, Evolution, Changes
VS-studio	–	Articulation, Execution, Evolution

- KEWI (Wickler et al., 2014) is a tool for encoding planning knowledge collaboratively by domain and planning experts, in a high level language called *AIS-DDL*, originally for use within Industrial Drilling domains. It features a domain expert-friendly set of tools for verification and validation, based around a layered *domain ontology*.
- VS-studio (Long et al., 2018). There is a class of PDDL editors that provide a range of different ways for supporting the knowledge encoding process, such as syntax highlight, auto-completion, plan visualisation, etc. This class includes tools such as PDDL studio (Plch et al., 2012), `planning.domains`, and more recently VS-studio (and its integration with the `planning.domains` editor). For the sake of readability, we will focus on VS-studio, as its set of functionalities appears to subsume the functionalities of the other members of the class.

In terms of the various kinds of Quality Concepts discussed above, all the environments aim to assist in physical and syntactical qualities. The first three environments employ a user-oriented language (OCL_h, UML, *AIS-DDL* respectively) to act as an intermediary language between domain requirements and externalisation, which enhances support for semantic quality. On the other hand, the planning engines that they are connected to are not fully integrated (in part, this is so the environments can be flexible in enabling users to embed a choice of planners). The VS-studio environment, however, is aimed at supporting operational quality, with its planning engine embedded centrally, and its focus on dynamic testing. In the following, we analyse the considered tools with regards to the support they provide for each of the processes described in the quality framework. Table 1 shows an overview of the analysis. Tools are partially supporting processes if they provide means to support some aspects of the considered process, but not the whole process.

With regards to *Articulation*, all the considered tools support it. itSIMPLE, GIPO, and KEWI provide high level, structured languages that can be used to represent the knowledge about the domain, from which the model externalisation M can be generated (automatically or semi-automatically). In that, these tools support both the steps of the overall Articulation process: from the domain D to the interpretation I, and from I to a model M. VS-studio instead supports, in a very thorough way, only the last step of the process. No support is provided for obtaining the interpretation, but the tool includes a wide range of techniques for supporting the generation of the model M, and for making sure that it is syntactically correct.

Reflection is not appropriately supported by any of the tools. Either they translate domain knowledge to a single version of the PDDL language, or they do not provide means for measuring the impact of different languages L on the model externalisation M and on the engine(s) E. As a matter of fact, the ability of human experts to select the best language L (or version of the language) that is sufficiently expressive for encoding the model externalisation M, and to select the most appropriate planning engine(s), is still somewhat of an art. This indicates the need for an extension to the tools which feature high level input languages, to facilitate Reflection.

The *Execution* process is partially supported by most of the tools. VS-studio incorporates a single predefined planning engine, but provides a remarkable interface for the visualisation and analysis of the generated plans P and of the way in which the planning engine has explored the search space. There are also means for testing the shape of plans, and for generating test sets of problems, which characteristics can be specified in a meta language. itSIMPLE incorporates a wide range of planning engines, but provides a limited support for analysing and comparing plans generated by the different engines, or for compare the engines' performance. GIPO incorporates a *plan animator* that is used to view and query object state changes over plan execution.

Activation is partially supported by the latest version of itSIMPLE, only. By including a dedicated module (Vaquero et al., 2013a), itSIMPLE supports the post-design analysis of the model externalisation M, and of the plans P. Such analysis allows to identify, among other model-specific issues, missing requirements in the domain, for instance. In fact, it would be more precise to state that the analysis is done with regards to the interpretation I: however, changes in the interpretation can then be reflected in the domain requirements.

itSIMPLE, GIPO and KEWI support *Changes* partially, in the sense that their high level interfaces are built to support maintenance throughout opera-

tion, with the engineer changing the high level language, and the new externalisation being automatically generated. For example, in the case of GIPO, the diagrams defining object classes can be edited, and updated PDDL will be generated automatically.

The *Evolution* process is partially supported by itSIMPLE via the previously mentioned post-design analysis module. Such analysis my lead to the identification of aspects of M that can be represented in a different way. Evolution is partially supported also by VS-studio: the environment provides the means for making sure that changes made in part of the knowledge models are correctly reflected in problem instances.

All the environments provide some limited support for encoding additional knowledge K. Users of KEWI encode a dedicated domain-specific ontology and a corresponding set of relations. With VS-studio the user can encode in a pre-defined programming language the characteristics of instances to be solved, and the expected properties of solutions P. Both GIPO and itSIMPLE allow the user to encode axioms, although their purpose is for static domain analysis rather than for influencing the interpretation function and hence the externalisation. VS-studio provides a mean for encoding some additional knowledge that could be exploited for testing purposes, and for the automated population of planning problems.

Summarising, the knowledge engineering environments sampled provide support for engineering automated planning knowledge models in major aspect such as actualisation, but have major gaps as highlighted by the application of the framework.

6 BENEFITS AND DISCUSSION

This section provides an overview of potential benefits that the use of the proposed framework can bring to the AI planning field and, to some extent, to the more general AI area.

Use of an Over-arching Framework and Terminology. The main benefit of the framework presented is to replace disparate notions of quality with a connected, composite and over-arching notion, that can be used within all work relating to this endeavour. Embedding automated planning, and in particular automated planning engines, within applications software, is becoming more widespread, as evidenced in part by the growth of applications papers and relevant workshops in academic conferences. This points to the immediate need for a general framework for considering the quality of the various components involved in the life cycle of the planning knowledge model. Applications papers tend to take an ad hoc approach to analysing such issues, but a framework such as the one proposed would lead to more structured and rationalised experience reports about applied AI planning. While we have shown how the framework can be used to evaluate tools environments, it would also help clear up ambiguous or vague terminology in current use –a prime example being the term "domain" itself. This term is sometimes used to mean the model M, and sometimes used to mean domain specification D. Clarifying terminology at the very least would improve communication within development processes, as well as the clearer communication of research.

Insights from Previous Framework Research. The research surrounding the foundations of the SEQUAL framework can be used to provide insights into planning developments using the proposed quality framework. For example, the modelling process should terminate not when the model is perfect, i.e. when semantic quality goals such as accuracy and completeness are satisfied, but when the model has reached a state where further modelling is less beneficial than exploiting it (Krogstie et al., 1995). Feasibility introduces a trade-off between the benefits and the required resources for achieving a given overall quality of the knowledge model. Benefits and reasonable resource bounds are themselves part of the specification of the domain D, as they are a key component of the modelling process. In that, the "right" feasible quality of a planning knowledge model has to be decided via a domain-specific analysis, and can not be generalised over different domains and scenarios.

Casting Previous Work into a Wider Framework. As remarked above, work addressing quality in the automated planning area has largely addressed semantic quality, motivated by software engineering concerns about the verification and validation of a completed system, as well as knowledge engineering concerns about accuracy and completeness of the knowledge model (Biundo et al., 2003). Taking action model learning as an example, there has been much work in developing machine learning procedures which learn parts of the knowledge model, e.g. (Zhuo et al., 2010). In order to evaluate the quality of these learning procedures, it is natural to want to measure the quality of what is learned –such as by comparing the learned knowledge with what has been hand crafted (evaluating physical and semantic quality), or by checking that the new knowledge can

be used by a planner (evaluating operational quality). Similarly, we can classify the type of learning by its inputs, such as training examples taken from the domain specification D. In this way, the activity of using machine learning to learn parts of the model externalisation M can be cast within the Framework to clarify quality goals and classifications of procedure, leading to a sounder footing for the evaluation of such domain model learning techniques.

Generalisation to Other Areas of Automated Reasoning. It may be argued that the introduced framework, as well as some of the peculiarities of planning knowledge models described in the corresponding section, are shared by other approaches for automated reasoning. This is, to some extent, correct. Answer Set Programming (ASP) and Satisfiability (SAT) components can be seen as components of larger frameworks, potentially providing support and automated reasoning capabilities to autonomous systems. However, a significant difference lies in the fact that in areas such as ASP and SAT, there is less emphasis on the shape of generated solutions. In fact, ASP solvers are usually bounded to return all the possible solutions. In SAT, instead, it is important to show that there exists a solution, or to demonstrate that a formula is unsatisfiable. Sub-areas of SAT, such as MaxSAT, can accommodate preferences about the structure of solutions, but solvers tend to be more focused on optimality –by including preferences in the overall quality of generated solutions. From a language perspective, both SAT and ASP show a smaller degree of variability than planning: the selection of the language is therefore much easier and less important for the overall modelling process. Therefore, components such as P, and L are less relevant for the ASP and SAT modelling process.

7 CONCLUSIONS AND FUTURE WORK

The engineering of automated planning applications –and in particular the knowledge model– is of great importance as research advancements lead to applications, particularly in autonomous systems. To support this, a deeper understanding of quality in the development process needs to be derived. Utilising general frameworks for conceptual modelling, we have introduced a quality framework for use in viewing the development of the knowledge model in automated planning. This links previously distinct research efforts in areas such as post-design analysis, automated model acquisition, model debugging, and tool development.

We have shown how past research in this area can be given a more holistic setting within this framework, and have exploited the framework for assessing the support provided by existing knowledge engineering tools for automated planning, and, finally, we pointed out a list of potential benefits to the planning community. The results of the performed analysis, beside demonstrating the usefulness of the introduced framework, confirms the lack of support for engineering automated planning knowledge models of state-of-the-art tools and indicates the areas where further work is needed. For instance, the analysis put a spotlight on the lack of support for the *Reflection* process, and on the limited ability of existing tools in supporting the encoding of additional knowledge that is not strictly part of the models.

For future work, we are interested in synthesising metrics that can be used for quantitatively evaluating processes and components of the quality framework, and to develop approaches that, by leveraging on the introduced processes and quality concepts, can help in addressing robustness issues of planning engines (Vallati and Chrpa, 2019).

REFERENCES

Biundo, S., Aylett, R., Beetz, M., Borrajo, D., Cesta, A., Grant, T., McCluskey, T., Milani, A., and Verfaillie, G. (2003). PLANET roadmap. *http://www. planet-noe.org/service/Resources/Roadmap/Roadmap2.pdf*.

Chrpa, L., McCluskey, T. L., Vallati, M., and Vaquero, T. (2017). The fifth international competition on knowledge engineering for planning and scheduling: Summary and trends. *AI Magazine*, 38(1):104–106.

Frank, J. (2013). The challenges of verification and validation of automated planning systems. *Proceedings of ASE*, page 2.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory & Practice*. Morgan Kaufmann.

Krogstie, J. (2012). Quality of business process models. In *The Practice of Enterprise Modeling - 5th IFIP WG 8.1 Working Conference (PoEM)*, pages 76–90.

Krogstie, J. (2013). A semiotic approach to data quality. In *Proceedings of the International Conference on Enterprise, Business-Process and Information Systems Modeling (BPMDS/EMMSAD)*, pages 395–410.

Krogstie, J. and de Flon Arnesen, S. (2004). Assessing enterprise modeling languages using a generic quality framework. *Information modeling methods and methodologies*, (1537-9299):63–79.

Krogstie, J. and Jørgensen, H. D. (2002). Quality of interactive models. In *Advanced Conceptual Modeling Techniques, ER 2002 Workshops: ECDM*, pages 351–363.

Krogstie, J., Lindland, O. I., and Sindre, G. (1995). Defining quality aspects for conceptual models. In *Proc. of ISCO*, pages 216–231.

Krogstie, J., Sindre, G., and Jørgensen, H. D. (2006). Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15(1):91–102.

Lindland, O. I., Sindre, G., and Sølvberg, A. (1994). Understanding quality in conceptual modeling. *IEEE Software*, 11(2):42–49.

Long, D., Dolejsi, J., and Fox, M. (2018). Building Support for PDDL as a Modelling Tool. In *Proc. of KEPS*.

McCluskey, T. L. (2002). Knowledge Engineering: Issues for the AI Planning Community. In *Proc. of (KEPS)*.

McCluskey, T. L., Vaquero, T. S., and Vallati, M. (2017). Engineering knowledge for automated planning: Towards a notion of quality. In *Proc. of K-CAP*, pages 14:1–14:8.

Plch, T., Chomut, M., Brom, C., and Barták, R. (2012). Inspect, edit and debug PDDL documents: Simply and efficiently with PDDL studio. In *Proc of ICAPS*.

Simpson, R. M., Kitchin, D. E., and McCluskey, T. L. (2007). Planning domain definition using gipo. *The Knowledge Engineering Review*, 22(2):117–134.

Stamper, R. (1996). Signs, information, norms and systems. *Signs of work*.

Vallati, M. and Chrpa, L. (2019). On the robustness of domain-independent planning engines: The impact of poorly-engineered knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP*, pages 197–204.

Vallati, M. and Kitchin, D. E., editors (2020). *Knowledge Engineering Tools and Techniques for AI Planning*. Springer.

Vaquero, T., Silva, J., and Beck, J. (2013a). Post-design analysis for building and refining ai planning systems. *Eng. App. of AI*, (26(8)):1967–1979.

Vaquero, T. S., Silva, J. R., Tonidandel, F., and Beck, J. C. (2013b). itsimple: towards an integrated design system for real planning applications. *Knowl. Eng. Rev.*, 28(2):215–230.

Venturelli, D., Do, M., Rieffel, E. G., and Frank, J. (2017). Temporal planning for compilation of quantum approximate optimization circuits. In *Proc. of IJCAI*.

Wegner, P. and Goldin, D. (1999). Interaction as a framework for modeling. In *Conceptual Modeling: Current Issues and Future Directions*, pages 243–257.

Wickler, G., Chrpa, L., and McCluskey, T. L. (2014). Kewi: A knowledge engineering tool for modelling ai planning tasks. In *Proc. of KEOD*.

Zhuo, H. H., Yang, Q., Hu, D. H., and Li, L. (2010). Learning complex action models with quantifiers and logical implications. *Artificial Intelligence*, 174(18):1540 – 1569.