In Defence of Design Patterns for AI Planning Knowledge Models[⋆]

Mauro Vallati$^{1[0000-0002-8429-3570]}$ and Lee McCluskey$^{1[0000-0001-8181-8127]}$

University of Huddersfield, Huddersfield, UK
{m.vallati,lee}@hud.ac.uk

Abstract. Design patterns are widely used in various areas of computer science, the most notable example being software engineering. They have been introduced also for supporting the encoding of automated planning knowledge models, but with little success.
In this paper, we argue about the merits of design patterns, as an example of the broader class of reusable abstractions, in the automated planning context; particularly, we aim at drawing attention to their potential usefulness for the explainability of domain-independent planning systems.

Keywords: Automated Planning · Knowledge Engineering · Design Patterns · Explainability.

1 A Knowledge Engineering (Historical) Perspective

Automated planning has been successfully applied in challenging real-world domains, including drilling [7], transport [16], smart grid [23], UAV control [19], e-learning [9] and mining [13].

Undoubtedly, the intensive development of domain-independent planning techniques has contributed to the advancement of planning technology. In domain-independent planning there is a decoupling between the planning logic, that is embodied in the planning engine, and the domain knowledge, that comes under the form of knowledge models. As the two components rely on a well-defined interface language, they can be substituted by other approaches without any changes to the rest of the framework.

A critical aspect of domain-independent planning, is the application knowledge that must be added to the planner to create a complete planning application. This is made explicit in (i) a domain model, which is a formal representation of the persistent domain knowledge, and (ii) an associated problem instance, containing the details of the particular problem to be solved. Both these components are used by automated planning engines for reasoning, in order to synthesise a solution plan. Formulating knowledge for use in planning engines is currently

[⋆] Copyright ©2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

something of an ad-hoc process, where the skills of knowledge engineers significantly influence the quality of the resulting planning application [4].

Studies on Knowledge Engineering for Planning and Scheduling (KEPS) have led to the creation of several tools and techniques to support the design of domain knowledge structures, and the use of planners for real-world problems [15, 10]. However, tools are still rarely used when experts have to engineer new knowledge models [4]. It may be due to the fact that experts have to adapt the encoding and refining processes according to the functionalities of a given tool. Further, the learning curve can be particularly steep for this kind of tools.

A different yet complementary approach relies on the notion of reusable abstractions (see, for instance, [18]): instead of encoding planning knowledge models directly in a low-level language such as PDDL, this approach envisages the use of more abstract level descriptions, that foster the design and use of modular structures. This would support re-use of similar structures, and provide a mean for justifying some decisions made during the knowledge acquisition and encoding processes. One of such reusable abstraction techniques for planning was proposed in the early 2000s, and leveraged on the notion of design patterns [20]. Design patterns [1] facilitate reuse of good design practices, as they describe a recurrent problem and a well-tested solution. Since their introduction, design patterns have been swiftly and successfully adopted in most areas of computer science, most notably software engineering, as a mean for fostering reusable and easy to generalise solutions [8]. The approach introduced by Simpson *et al.* [20] proposed a tool for organising and managing patterns, and a way for generating the corresponding solution either in OCL [14] or in PDDL, to be then incorporated in knowledge models that can be used by domain-independent planning engines. Examples of the patterns proposed include a general approach for moving elements, and to describe structures such as maps, sets, or sequences.

Design patterns remove the mentioned issue of KEPS tools, particularly the need to learn to use a dedicated tool, and can actively support the widespread use of best practices in the field. Notably, a rudimentary form of design patterns is implemented in the well-known online editor *planning.domains*,[1] as the "Misc PDDL Generators" plugin. This plugin provides the PDDL code for representing some common structures, such as grids and maps. There is not, however, a description of the problem that such structures are aimed at solving, and there is no justification for the specific encoding that is suggested.

Unfortunately, aside from the mentioned *planning.domains* example, design patterns are not widely used by the AI planning community. A reason for that may be the lack of a centralised repository for managing such design patterns, the lack of a commonly agreed way for encoding them, and the fact that the encoding of planning knowledge models is still seen as a sort of ad-hoc artisan process, whose shortcomings can be dealt with by using high-performance planning engines rather than a proper engineering process.

[1] http://editor.planning.domains/

2 Beyond Knowledge Encoding: Explainability

Here we would like to highlight the important role that reusable abstractions, and particularly design patterns can play for AI planning knowledge models, beside the support for knowledge encoding and acquisition.

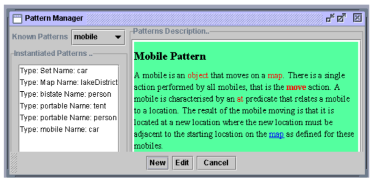
An area of growing importance for the automated planning field, as well as for AI in general, is that of explainability. The idea being that an AI system should be able to explain, to some extent, its behaviour to stakeholders. Focusing on the narrower topic dubbed XAIP, as for EXplainable AI Planning, a number of recent works considered the problem of defining what "explainable" means for an automated planning system, explaining and describing generated plans, bridging the gap between machines and human stakeholders, and designing approaches to explain the behaviour of planning systems [6, 5, 3, 22, 21, 2].

As noted in the previous section, domain-independent planning systems are heavily dependent on the provided knowledge models; this is also true when explainability is concerned. The behaviour of a planning system can not transcend the knowledge model, and the importance of the planning knowledge model has been well-argued [17]. In fact, in the XAIP field, the knowledge model is more and more regarded as a source of knowledge that can explain the behaviour of the planning system [24, 11]. This is also because, during the knowledge engineering process of encoding specifications into an appropriate planning knowledge model a number of design decisions have to be made. Some of those decisions are due to the experience of the encoder, while others are due to either common or specific knowledge about the application context. Clearly, other aspects of the knowledge model support explainability also: the fact that dynamic knowledge is stated with both a procedural and declarative interpretation means that parts of the knowledge can be used in isolation within an explanation. Even namings within the model can be used as an understandable unit of the explanation.

It should therefore come as no surprise that design patterns can play a significant role in the XAIP field. Design patterns can provide a valuable and well-defined mean for supporting explanations. This is because, when design patterns are implemented in a planning model, the design pattern description becomes a valuable source of additional knowledge, that is not encoded in the model. Further, there is not even the need for finding a way to "attach" such additional knowledge to the standard planning models. The management of additional knowledge is a potential issue for planning systems [24]. Instead, having a centralised repository for planning design patterns (something similar to what *Source Making*[2] does for software engineering, for instance) would allow planning models to be encoded and circulated as they are right now, and would support explanation engines in looking for patterns and descriptions in a structured manner. In fact, the XAIP field can provide new momentum to the use and management of design patterns for planning.

In a sense, the use of patterns becomes then an explicit way for providing explanations, that can of course be complemented by the line of work that de-

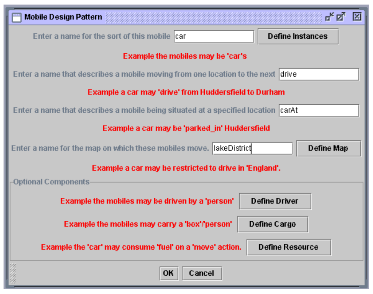
[2] https://sourcemaking.com/



Type: Set Name: car
Type: Map Name: lakeDistrict
Type: bistate Name: person
Type: portable Name: tent
Type: portable Name: person
Type: mobile Name: car

Mobile Pattern

A mobile is an object that moves on a map. There is a single action performed by all mobiles, that is the **move** action. A mobile is characterised by an at predicate that relates a mobile to a location. The result of the mobile moving is that it is located at a new location where the new location must be adjacent to the starting location on the map as defined for these mobiles.



Enter a name for the sort of this mobile: car

Example the mobiles may be 'car's

Enter a name that describes a mobile moving from one location to the next: drive

Example a car may 'drive' from Huddersfield to Durham

Enter a name that describes a mobile being situated at a specified location: carAt

Example a car may be 'parked_in' Huddersfield

Enter a name for the map on which these mobiles move.: lakeDistrict

Example a car may be restricted to drive in 'England'.

Example the mobiles may be driven by a 'person'

Example the mobiles may carry a 'box'/'person'

Example the 'car' may consume 'fuel' on a 'move' action.

Fig. 1: The Mobile design pattern, described via a dedicated interface (Pictures taken from [20]).

vises techniques for identifying known substructures of planning models, and link them to an expected behaviour (see for instance [12]). It is naive to believe that a complex planning model can be composed only by design patterns, but at least the core elements are expected to be encoded using some form of patterns. Additional sources of domain-specific knowledge can then be sought via ontologies or similar structures [24].

3 Example: The Mobile Design Pattern

For exemplifying our vision for the use of design patterns for explainability, let us consider the `Mobile` design pattern introduced by Simpson *et al.* [20], and partly shown in Figure 1. The Mobile pattern deals with the problem of encoding mobile elements (typically vehicles) in the models, that can be used to carry goods or people, and may require a driver to be moved.

The problem description, the use of appropriate naming convention, and the optional components can allow to deal with the classes of explanations that Vallati, McCluskey and Chrpa defined as *context-related* and *assumption-related* [18]. The former class refers to explanations that relate to contextual knowledge about the domains, the latter class requires information about assumptions and decisions made during the knowledge encoding process.

A context-related question that can be answered using knowledge that comes with the Mobile design pattern would be: *Why can the vehicle not move from location X to location Y?* The pattern description includes the answer to this question, by stating that the movement can be allowed between adjacent locations in a map. Similarly, taking into account the optional components, the use of the design pattern can help to answer questions like *Why did the vehicle not move immediately at the start of the plan?* Here the fact that a vehicle needs a driver to move can help in addressing this question. Assumptions-related questions can then relate to the maximum capacity of the vehicle, for instance, or to the fuel consumption of the vehicle.

Given the provided example, it should be clear that to exploit design patterns for supporting explainability, there is the need for structuring the way in which the corresponding problem description, optional components, etc. are encoded in a repository, and for dedicated approaches able to identify the most relevant knowledge with regards to a given query. However, the planning community can take inspiration from the work done in other research areas, for instance Knowledge Engineering and Knowledge Management, that have decades of experience in addressing similar challenges.

4 Conclusion

Reusable abstractions, mostly under the form of design patterns, have been introduced for supporting the knowledge encoding and acquisition processes. However, they did not have great success in the KEPS field.

In this paper, we point to the fact that design patterns can play a significant role also for supporting the explainability of the behaviour of a planning system. The benefit for Knowledge engineers derived by using design patterns in planning models can therefore be twofold: (i) design patterns allow to deal with common problems by reusing best practices, and (ii) they directly support the explanaibility of the knowledge models and, indirectly, of the overall planning system.

Future work is envisaged in identifying suitable way for storing and managing design patterns for automated planning, with focus given to the way in which the problem they address is described in an explainable-friendly manner, and to the planning languages that are supported. A general repository of design patterns for the planning community, as it is currently done for benchmarks, could foster their use and give momentum to this important area of research.

References

1. Alexander, C.: A pattern language: towns, buildings, construction. Oxford university press (1977)
2. Caminada, M.W., Kutlak, R., Oren, N., Vasconcelos, W.W.: Scrutable plan enactment via argumentation and natural language generation. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. pp. 1625–1626 (2014)
3. Chakraborti, T., Kulkarni, A., Sreedharan, S., Smith, D.E., Kambhampati, S.: Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In: Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS. pp. 86–96 (2019)
4. Chrpa, L., McCluskey, T.L., Vallati, M., Vaquero, T.S.: The fifth international competition on knowledge engineering for planning and scheduling: Summary and trends. AI Magazine **38**(1), 104–106 (2017)
5. Eifler, R., Cashmore, M., Hoffmann, J., Magazzeni, D., Steinmetz, M.: A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI. pp. 9818–9826 (2020)

6. Fox, M., Long, D., Magazzeni, D.: Explainable planning. CoRR **abs/1709.10256** (2017)
7. Fox, M., Long, D., Tamboise, G., Isangulov, R.: Creating and executing a well construction/operation plan (2018), uS Patent App. 15/541,381
8. Gamma, E., Helm, R., Vlissides, J., Johnson, R.: Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley (1997)
9. Garrido, A., Morales, L., Serina, I.: Using AI planning to enhance e-learning processes. In: Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS. AAAI (2012)
10. Jilani, R., Crampton, A., Kitchin, D., Vallati, M.: Automated knowledge engineering tools in planning: State-of-the-art and future challenges. In: Proceedings of KEPS (2014)
11. Lindsay, A.: Towards exploiting generic problem structures in explanations for automated planning. In: Proceedings of the 10th International Conference on Knowledge Capture, K-CAP. pp. 235–238 (2019)
12. Lindsay, A.: Using generic subproblems for understanding and answering queries in xaip. In: Proceedings of KEPS (2020)
13. Lipovetzky, N., Burt, C.N., Pearce, A.R., Stuckey, P.J.: Planning for mining operations with time and resource constraints. In: Proceedings of the International Conference on Automated Planning and Scheduling (2014)
14. Liu, D., McCluskey, T.: The ocl language manual. Tech. rep., Version 1.2. Technical report, Department of Computing and Mathematical ... (2000)
15. McCluskey, T.L., Aler, R., Borrajo, D., Garagnani, M., Haslum, P., Jarvis, P., Refanidis, I., Scholz, U.: Knowledge Engineering for Planning Roadmap. http://scom.hud.ac.uk/planet/home (2003)
16. McCluskey, T.L., Vallati, M.: Embedding automated planning within urban traffic management operations. In: Proceedings of the International Conference on Automated Planning and Scheduling (2017)
17. McCluskey, T.L., Vaquero, T.S., Vallati, M.: Engineering knowledge for automated planning: Towards a notion of quality. In: Proceedings of the Knowledge Capture Conference, K-CAP. pp. 14:1–14:8 (2017)
18. McCluskey, T., Simpson, R.: Towards an algebraic formulation of domain definitions using parameterised machines. In: Proceedings of the Annual UK PLANSIG Workshop (2005)
19. Ramírez, M., Papasimeon, M., Lipovetzky, N., Benke, L., Miller, T., Pearce, A.R., Scala, E., Zamani, M.: Integrated hybrid planning and programmed control for real time UAV maneuvering. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS. pp. 1318–1326 (2018)
20. Simpson, R., Mccluskey, T., Long, D., Fox, M.: Generic types as design patterns for planning domain specification. In: AIPS2002 Workshop on Knowledge Engineering Tools and Techniques for AI Planning (2002)
21. Sohrabi, S., Baier, J.A., McIlraith, S.A.: Preferred explanations: Theory and generation via planning. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
22. Sreedharan, S., Srivastava, S., Smith, D., Kambhampati, S.: Why can't you do that hal? explaining unsolvability of planning tasks. In: International Joint Conference on Artificial Intelligence (2019)
23. Thiébaux, S., Coffrin, C., Hijazi, H., Slaney, J.: Planning with mip for supply restoration in power distribution systems. In: Proceedings of the International Joint Conference on Artificial Intelligence (2013)

24. Vallati, M., McCluskey, L., Chrpa, L.: Towards explanation-supportive knowledge engineering for planning. In: Proceedings of XAIP (2018)