KA Tool and domain construction for AI planning applications

Ruth Aylett, Christophe Doniat
Centre for Virtual Environments
University of Salford, Salford, UK

Abstract

KA Tool embodies work aimed at allowing domain experts to generate a domain model for an AI planning system, carried out as part of a larger project to build an integrated set of tools for supporting AI planning. The paper outlines the overall methodology and describes how KA Tool supports it. A Domain model is generated in which can be represented by cluster of constraints shaping an Ontology of each studied case. Progress has been made towards automatic conversion into the modelling language OCL and integration with the OCL tool GIPO. We illustrate the methodology by applying it in two examples of planning

1. Introduction and motivation

The effort required to construct a domain model for an AI planning system has long been recognised as a major barrier to the take-up of this technology outside the AI planning community. Researchers at the Universities of Huddersfield, Salford and Durham [7] have been tackling this problem in the EPSRC-funded project PLANFORM. Its aim is to research, develop and evaluate a method and supporting high level research platform for the systematic construction of planner domain models and abstract specifications of planning algorithms, and their automated synthesis into sound, efficient programs that generate and execute plans. Figure 1 shows the high-level architecture of the PLANFORM system.

Within this project, the domain model is represented in the object-oriented sorted first-order logic language OCL [22] which supports validation and checking tools as well as translation to other formalisms such as PDDL, the standard domain representation language used in the international Planning Competition. The toolset GIPO [23] is used to support the iterative construction and validation of an OCL model: however this still requires too much specialist knowledge of OCCL and AI planning to be a suitable interface for a domain expert – one who understands the domain in which planning is to take place but lacks any specific expertise in AI planning. The KA-Tool discussed here is aimed at such domain experts.

The problem of supporting knowledge acquisition directly from the domain expert, without the intervention of a knowledge engineer, has been discussed in the field of Knowledge-Based Systems (KBS) for many years [1,3]. A consensus has been reached that this may be feasible where a skeletal domain model can be provided to guide the knowledge acquisition process and both the skeleton model and the process itself can be defined through a methodology embodied in the knowledge acquisition tool [1]. The key components of the skeleton model are seen as domain

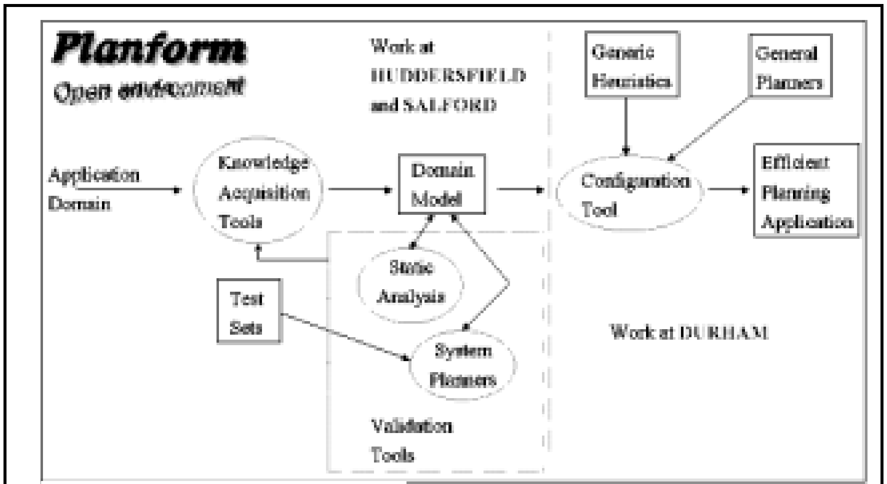

Figure 1: Planform architecture

ontologies combined with domain-independent problem-solving methods which have often been thought of as *generic tasks*. The best-known – but far from only – example of such a methodology is Common KADS [14], offering libraries of configurable problem-solving components together with stereotypical configurations which can be thought of as corresponding to types of abstract problem-solving tasks such as *diagnosis by heuristic classification* or *interpretation*.

It is noticeable that AI Planning has rarely been considered as part of this research (see Valente [3] for a rare exception). While in theory planning could be considered as one or more generic tasks, in practice the Knowledge Engineering community has concentrated on other generic tasks – diagnosis in particular – while AI Planning researchers have hardly been involved at all, tending to concentrate on the development of planning algorithms.

The approach discussed here draws on this work in the KBS community, seeing the combination of ontologies, logics and generic problem-solving methods as a way of addressing knowledge acquisition for planning [1]. It supports the capture and structuring of relevant knowledge about a domain and its intelligent behaviours [2] because they play an important role in the choice of an appropriate problem-solving method, possibly configured from complex components held in a library [3].

1.1. Knowledge Acquisition Process

The knowledge acquisition process embodied in the KA-Tool can be seen in Figure 2. A question-driven process works on protocols and on a skeleton theory derived from an ontologies library. By protocol we mean raw domain knowledge - transcripts, documents, interviews, observations[1], created by a problem-solving episode in which experts are provided with a real-AI Planning problem, of the kind that they normally deal with, and are asked to solve it. As they do so, they are required to describe each step, and their reasons for doing what they do. The transcript of their verbal and/or text account is, in this case, called a protocol.

[1] We will used the term 'transcript' in the next paragraphs to mean a combination of transcripts, documents, interviews, observations as a whole.

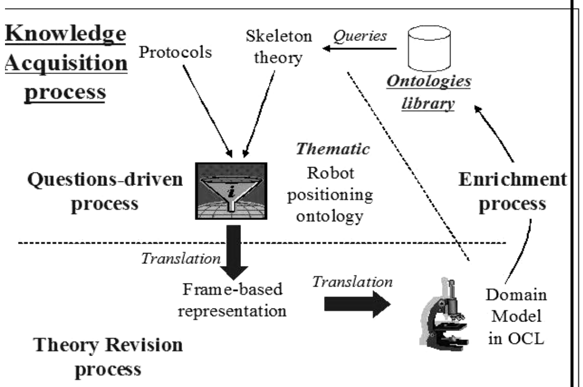

Figure 2. Knowledge acquisition process[1].

By problem specification we mean a definition or description of an application domain represented as a set of choices at a particular level of abstraction in an ontological hierarchy. Thus 'Entertaining a foreign visitor' and 'Drumstore', the domains used for the experiments reported later, are problem specifications.

The second stage in the process incorporates a theory revision process which produces a conceptual model using a hierarchical frame system which allows easy representation of inheritance between *sorts* (the relationship *kind-of*) and/or aggregation between *sorts* (the relationship *part-of*) for instance. Translation into a sorted first-order logic such as that used by OCL is straightforward. Frames have an advantage over a first-order logic in that both structure and behaviour can be embodied in one generic entity.

An ontology is defined [6] as a rigorous specification of a set of specialised vocabulary terms sufficient to describe and reason about the range of situations of interest in a particular domain - a conceptual representation of the domain entities, events, and relationships. Two primary relationships of interest are abstraction (kind-of) and composition (part-of). Thus an ontology provides a grounding of the key concepts within a domain. In principle we need both an ontology of planning problem domains and of planning software to carry out knowledge acquisition since the premise is that the conceptual framework of the problem domain is not the same as that of the planning software – otherwise there would be little problem for the domain expert.

The question-driven process is also used to construct a Domain dictionary, in effect a partial ontology - using the experimental approach, it is hard to make an exhaustive analysis of all domain objects. Nevertheless, the problem specification can be used to define relevant objects and relationships, using macroscopic properties that support appropriate choices. The Domain Dictionary is associated with (i) a particular domain, (ii) specification of a problem or problems that we want to solve, (iii) the reasoning that belongs to the studied domain and allows the specified problem to be solved.

1.2. Overview of PLANFORM-KA Tool architecture

Figure 3 shows the main architecture of the PLANFORM-KA tool – an intelligent system that contains the KA process. The user applies the *module of domain model building* to a particular problem specification. The building of a new conceptual model might be carried out with or without an existing problem specification from the *Domain model library*. The result is recorded in this library. On the right-hand side, the overall knowledge base consists of the conceptual model of the knowledge

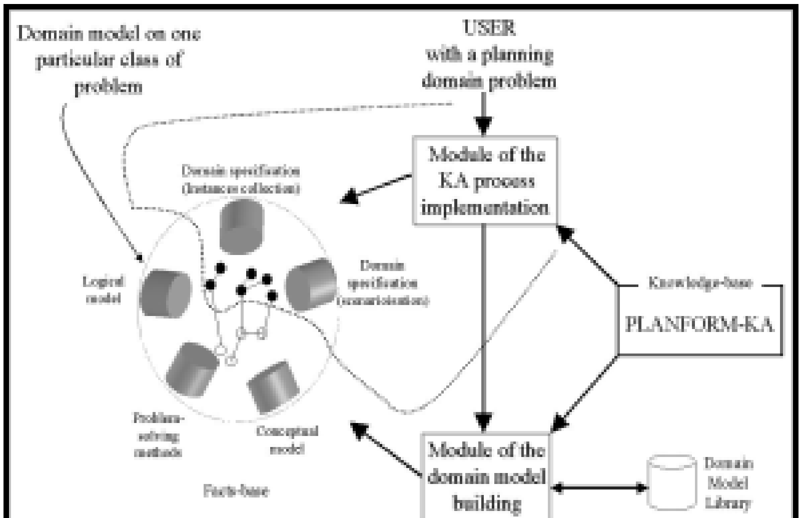

Figure 3: Architecture of the KA-Tool

acquisition process itself, called PLANFORM-KA and the *KA-Expertise* belonging to the particular conceptual model being constructed.

2. Case studies and methodology

In this section, we present two case studies using our methodology, based on the problem specifications: (i) 'EVentus: Entertaining a foreign visitor to your lab at the weekend' and (ii) 'Drumstore: a logistics problem in a nuclear waste factory'. We conducted these experiments, respectively with ten people and six people who verbalised their knowledge about how they would solve this problem during interviews. We chose EVentus because it was a planning problem drawing on general rather than specialised knowledge that was not difficult to collect. Drumstore was chosen because it had already been implemented as an AI planning domain within the group. The interviews contained the unstructured knowledge (discourse) and sometimes some notes such as graphics, plans and other material describing knowledge and activity (explicitly/implicitly) both about the case studies and the KA process itself.

It is important to understand the level of abstraction at which such a sample problem must work. The PLANFORM toolkit as a whole will be used to create a domain model within which a number of specific tasks can be planned. Thus the experiment does not start with a specific task, but with the generic problem specification. Subjects were asked to explore the generic domain model that would be needed to plan within the domain of the problem specification and to support the solving of a number of specific tasks. Note that a more abstract version of this problem would be to replace 'your lab' with 'a lab' where this might be anywhere in the world potentially. An instance of a specific task would be something like 'Professor Stein from DFKI Germany is to be entertained on Saturday May 9th'.

2.1. Building of a domain dictionary

The first phase gives us a domain dictionary (Table 1) that puts together a set of terms according to the problem specification.

Drumstore	EVentus
Robot Thing Gripper Object Relation Reference	Thing Activity Context Visitor Capability

Table 1. Domain dictionary

Next, we built a set of scenarios with the shared knowledge of these domain experts to find out how each expert defines reasoning strategies to solve the problem specification. We used a part of the KOD (Knowledge Oriented Design) [8] method to obtain an accurate process for knowledge acquisition and to build the conceptual model through the set of examples and scenarios (see section 2.2). Table 2 and 3 show the number of instances of each term in each scenario. We will call these outcomes *instance coverage*.

Drumstore	Terms[2]					
	R	T	G	O	Rel	Ref
1	5	1	3	7	2	3
2	10	2	2	5	2	3
3	20	5	5	12	1	3
4	10	3	3	5	1	3
5	5	1	4	7	2	2
6	6	1	3	13	2	2
7	8	1	5	7	2	3
8	7	1	4	11	2	2

Table 2.Instance coverage of Drumstore.

This shows that knowledge about this particular specification varies between domain experts giving different number of examples of each term. This coverage gives us an idea of the experts' practice so as to build the interface of the future intelligent system.

2.2. Building of conceptual/epistemological model

The second extraction gives us first a conceptual model containing semantic relationships, objects and actions. We then add an epistemological model - the definition of concepts, hierarchy and structuring relationships (behaviours). A domain model is thus defined by these representations in our methodology by using a frame system as in Figure 4.

EVentus	Terms[3]				
	T	A	C	V	Ca
1	9	4	1	1	3
2	5	6	1	1	2
3	8	7	2	2	2
4	5	5	3	1	4
5	13	7	1	2	6

Table 3.Instance coverage of EVentus.

`Drumstore` relies on the nine following generic concepts: `Thing` is a root of the domain model and describes two mobile things: `Robot` and `Object`. `Robot` depicts a real robot, which can navigate and has equipment – `Gripper` – to bring and carry some `Object` according to a `Relation/Reference` address pair (e.g. `(Object,at,beacon1)`). `Primitives` depict a set of generic concepts like `Drum` (`Object`), `At`, `Near` (`Relation`) and `Beacon` (`Reference`). `Substate` and

[2] Each Drumstore scenario is designed through the six terms as follows: Robot (R), Thing (T), Gripper (G), Object (O), Relation (Rel) and Reference (Ref).
[3] Each EVentus scenario is designed through the four terms as follows: Thing (T), Activities (A), Context (C), Visitor (V) and Capability (Ca).

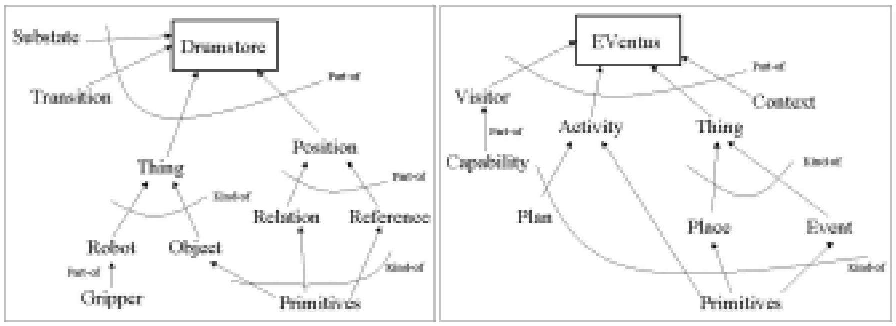

Figure 4. Frame systems of Drumstore and EVentus

`Transition` depict respectively the conditions in which Robot does some tasks and the state of each task when it has taken place.

`EVentus` contains the nine following generic concepts: `Visitor` is a locus of the domain model and describes a real visitor according to her/his real capacities, which are depicted by `Capacity`. `Activity` and `Context` describe behaviours of a visitor, `Plan` describes a set of alternative plans used by a visitor. `Thing` describes `Places` and `Events` used during the activity. Finally, `Primitives` depicts a set of generic concepts like a restaurant, a town (place) or an exhibition (event).

2.3. Summary

A KA process has been carried out to capture knowledge and build two domain models for particular problem specifications through two case studies: `Drumstore` and `EVentus`. The generic concept `Thing` is defined in both domain models with different semantics. In `Drumstore`, this concept represents an abstraction of *mobiles* but in `EVentus`, it represents an abstraction of *locations*.

Categories	***Drumstore***	***EVentus***
Agent	*Thing*	*Visitor*
Object	*Position*	*Thing*
Task	*Substate* *Transition*	*Context* *Activity* *Plan*

Table 4. Abstraction similarity between Drumstore and EVentus

Table 4 shows the similarity between `Drumstore` and `EVentus` concepts using three main categories: `Agent`, `Object` and `Task` as a skeleton ontology for planning domains [9]. Note that the `Task` category is divided into two semantic sub-categories: (i) the `Drumstore` task is state-based and the `EVentus` task is action-based. This represents a first step towards an epistemological model.

3. An Intelligent system: PLANFORM-KA

In this section we discuss the Planform-KA tool in more detail – see Figure 5 for its conceptual model. As outlined above, the process component of the tool can be decomposed into a set of refinement processes – called phases – carried out by the domain expert according to an expertise. We envisage supporting it with a generic ontology like the Upper Cyc Ontology [10] (though in this work we have

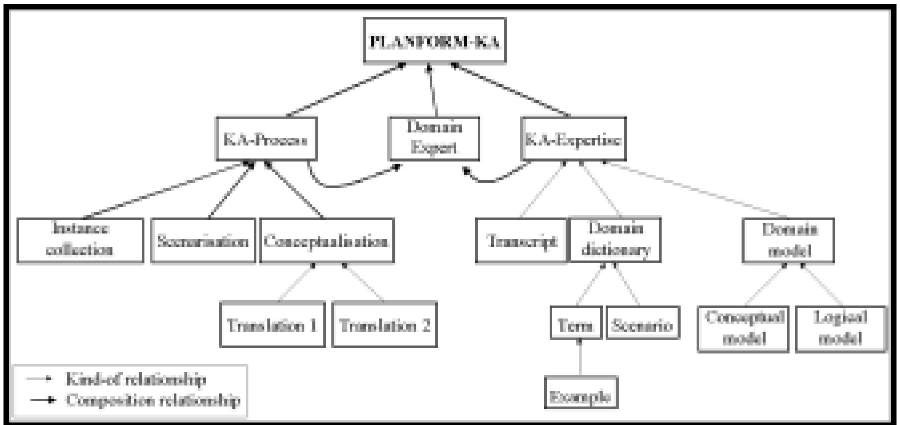

Figure 5. The PLANFORM-KA conceptual model.

constructed a small ontology ourselves) to start instance collection. This Ontology provides a sufficient common grounding for applications. Some concepts such as `Actor` or `Plan` are supplied in it as generic definitions, which should help the domain expert. It also includes definitions of `Object` and `Agent` categories (as in section 2.3) and a fragmentary definition of the `Task` category. That is the case for `Drumstore` for instance where there are `State` and `Transition` generic concepts as parts of OCL.

3.1 Conceptual model of PLANFORM-KA

The conceptual model of Planform-KA (Figure 5) relies on several interrelated generic concepts. The `domain expert` generic concept depicts the subject acquiring the knowledge model, the `KA-expertise` generic concept features the knowledge required to build the knowledge, the `KA-Process` generic concept describes the behaviour carried out by the domain expert with the tool. The KOD method was again used to elaborate a frame system to be implemented inside the KA Tool.

3.2. Frame representations

For reasons of space we discuss only a subset of the frame representations for these concepts – those for Domain expert, KA-Expertise and KA-Process.

3.2.1 Domain expert

We consider the domain expert (DE) as the cognitive agent carrying out the process of knowledge acquisition. DE has a mental model of the real world expressed in concepts. The `domain expert` generic concept represents the properties of this agent in relation to the carrying out of the KA-Process and is central to the overall conceptual model since there are composition relationships with concepts KA-Process and KA-expertise.

3.2.2 KA-Expertise

The `KA-Expertise` generic concept represents the memory of our domain expert. This holds three knowledge categories: transcripts from a case study, and the related domain dictionary and domain model. The `Transcript` generic concept represents the properties of documents such as free-text or graphics collected in a case study. The `Domain dictionary` generic concept represents the properties of a domain

<u>specification</u> expressed as a set of choices – <u>terms</u> – themselves organised into a set of <u>scenarios</u> (figure 6).

The `Domain model` generic concept represents the properties of a conceptualisation as a set of conceptual/epistemological and logical representations .

3.2.3 KA-Process

The `KA-Process` generic concept represents the process which drives knowledge acquisition and its refinement phases. The KA process starts with an <u>instance collection</u> phase, i.e. the explaining of each term by providing examples of it. For example, `Robot`, a term of the terms in `Drumstore`, contain the following instances:

```
Robot R3 navigates from location S3 towards beacon B14
Robot R3 docks at beacon B14
Robot R2 grabs from beacon B15 drum D12
```

This phase continues until the expert has provided instances for each newly defined term. The process continues with a <u>scenario creation</u> phase: the description of several scenarios – particular problems to be solved – within the scope of the given global goal (for example: entertaining a foreign visitor; a logistics problem in a nuclear waste factory) using the previously defined instances. Finally, a scenario can be seen as a set of facts (predicates), which will be used to define some properties, constraints, plan and goal states samples at the conceptual level. The outcome is a terminology, i.e. a set of terms and a set of scenarios. The built-in ontology is used to prompt the expert during this phase.

This bottom-up approach has also been supplemented by a top-down approach in which the ontological categories `agent`, `object` and `action`, are used to drive a question cycle in which new terms are extracted from the expert. Questions move between the categories, so that if the expert provides an `agent` term (for example `robot`), they are then prompted for actions carried out by that agent and objects involved in the action. At the conceptual/epistemological level, first of all, the process automatically carries out a <u>translation</u> phase into the frame-based representation, so that each defined term becomes a frame. Next, the domain expert defines by hand, or through the agent-object-action question cycle, the properties of each frame. Thus, the term Robot becomes the `Robot` frame and belongs to the `Concept`[4] `superframe`.

In the same way, a `Visitor` frame – from `EVentus` – can be defined, though space does not permit us to show it. This references a CAPABILITY frame holding the properties of natural abilities and skills that make the visitor able to do some activities. A visitor could have either at least seven {Status, gender, age, budget, type, quality, nationality} or several further capabilities such as {like to try new things, accompanying other people, swim, has a budget, other}.

The conceptualisation finishes with a second <u>translation</u> phase from the frame-based representation into sorted first-order logic, using the language OCL. In OCL, substate and transition substate concepts describe respectively, the conditions before the transformation of each task and the transition when an object changes from one substate to another substate. This translation is automatic: each frame ‡[5] sort, each

---

[4] SuperFrame CONCEPT is the generic frame, which is the root/father of all frames in the frame system.

[5] ‡ means 'is translated into the type of…'

instance of frame ⧧ object, each attribute ⧧ predicate and each *part-of* relationship with its related arity ⧧ a defined predicate called 'belongs_to'. For example, table 5 shows the `Robot` frame and its translation into OCL where gripper – equipment – of the robot. The arity of this slot (column Arity, bottom) is defined by `(1)`, i.e. this slot takes one frame gripper in the relationship at the same time and `(1,1)`, i.e. this slot allows the obligatory instantiating of one gripper's instance. As a result, the relationship and its arity of this slot translates into invariant predicates (bottom) the constraint that one robot has to have one gripper only.

Epistemological level		**Logical level**
ROBOT Frame and its slots	**Arity**	**OCL**
Kind-of value THING Name domain String = {r1,r2,r3,r4} If-add <GRIPPER, create-instance(),($equipment)> Sense_on domain Boolean = {true,false} Can_sense domain tuple(ROBOT,OBJECT,RELATION,REFERENCE) Equipment domain GRIPPER	(1) (1,1) (1) (1,1) (1) (1,1) (3) (1,1) (1) (1,1)	sorts(thing,[robot]) sorts(primitive_sorts,[robot]) objects(robot,[r1,r2,r3,r4]) sense_on(robot) can_sense(robot,object,relation, reference) belongs_to(gripper,robot) belongs_to(r1,g1) belongs_to(r2,g2) belongs_to(r3,g3) belongs_to(r4,g4)

Table 5. Translation from epistemological level to logical level.

3.3 Evaluation and results

A first demonstrator has been implemented to validate the approach of PLANFORM-KA.

Figure 6 shows the main graphical user interface during the creation of the `Robot` generic concept in the `Drumstore` domain model.

We have also generated the logical model seen in Appendix 1 with OCL semantics and syntax through a first version of a translator:

Generalising over the different phases of the KA process, we have formulated the notion of `Constraint`. Thus the `Term` generic concept – in the instance collection phase – is a kind of constraint which allows the domain expert to make a set of choices to justify the domain specification.

Next, the `Scenario` generic concept is also a kind of constraint, allowing choices in the design of task representations. Thus the task could be state-based, action-based and so forth.

In the same way, the `Relationship` generic concept – in the conceptualisation phase – is a kind of constraint (Figure 7), which structures each concept. In addition, the `Arity` and `Daemon` generic concepts – from the epistemological phase – are also kinds of constraints (Figure 8) on the problem-solving methods (PSM) and heuristics

Finally the `Proposition` generic concept – from the logical phase – is also a kind of constraint, representing the chosen logical language. The `Constraint` is then described as something that must be `true`.

Thus in the KA-process we define a cluster of constraints (Figure 9) across the several representation levels.

Figure 6 Creating the yerm ROBOT

3.4 Capturing actions

The creation of a strong methodological framework for the Planform-KA tool was seen as a priority, and this has been accomplished. What is required now is to incorporate the planning-specific conceptual framework of agent, object and task in a more direct fashion. Work has begun on the generation planning operators and translation into OCL, through the question-driven agent-action-object dialogue.Given that Planform-KA sits within the overall Planform architecture, even the generation of skeletal operators would allow use of GIPO's refinement mechanisms to fill them out into a complete form. This would require an AI planning expert to supplement the role of the domain expert but would at least automate the basic knowledge acquisition process from the expert.

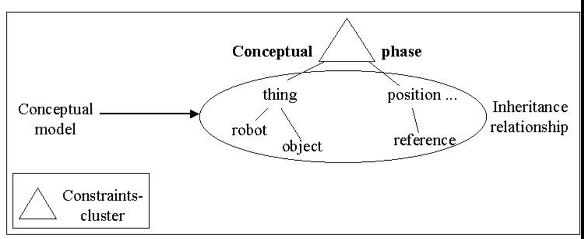

Figure 7. Constraints-cluster on conceptual model

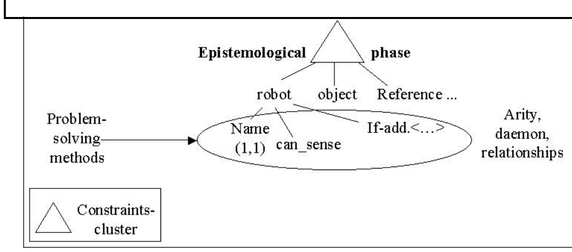

Figure 8. Constraints-cluster on epistemological model

4. Related work

Many specific approaches propose a set of solutions for the acquisition, the representation and the sharing/reusing of knowledge using libraries and/or strategies, since this topic has been studied extensively in the KBS community since the

1980s. Some of them are more specialised in the first extraction of knowledge proposing a generic surrogate to capture knowledge. Protégé [10] includes a suite of tools for editing ontologies, which can automatically generate customised editors that are accessible to domain experts. The Protégé library includes the problem-solving strategies (diagnosis) and also methods ontologies that describe the kinds of domain-independent knowledge used in strategies.

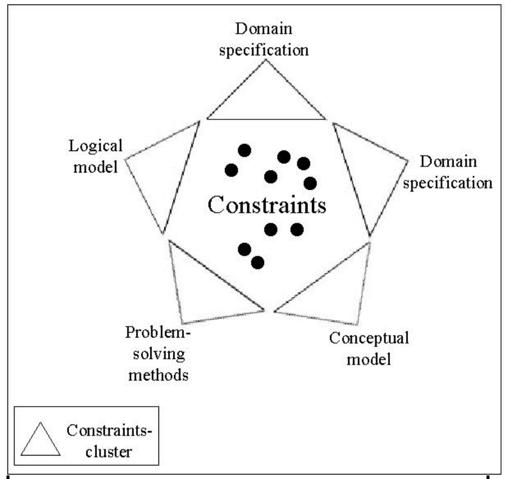

Figure 9. Clustered constraints define the KA model and process

Recently, researchers have investigated this topic through theory revision [25]. As Poole mentioned in [26] belief states of robots are uncertain when it adopts conditional plans. Fox and Long [27] propose the categorisation of domains into domain classes (e.g. transportation domain) is a kind of ontology (i.e. a complex sort) deduced during the planning process to generate Generic Types. McCluskey [28] suggests as well a inductive process with OpMaker.

The Protégé planning system is supported by a tool that captures new ontologies, and offers a library of problem-solving methods. EXPECT [11,12] used explicit representations of problem-solving strategies (propose-and-revise strategy for the configuration design task, for example) to support flexible approaches to knowledge acquisition. Blythe [24] uses expectation theory (task, critique) to capture data, object classes and preferences. Williams [29] proposes an intelligent system called SATEN to deal with theory extraction and revision from inconsistent ranking. KRAKEN is a toolset [30], which addresses several problems from domain experts who are non-logicians. PLANET [13] is an ontology for the representation of plans in the AI Planning field and is very relevant to the more extended framework discussed here. In other approaches, the answer for a given problem is built through a combined set of different techniques (AI methodologies, for example KOD, KADS [14]) according the major aim (diagnosis for example [15][16]).

5. Conclusion and further work

Surprisingly, given the amount of work in the KBS community in general, knowledge acquisition has not been widely studied in AI planning. Yet applying planning systems to real-world problems requires a systematic approach to knowledge acquisition and a methodology supporting reuse rather than ad-hoc adaptations of specific planning systems by particular individuals whose expertise remains private and invisible. The work discussed here represents some steps in this direction.

5.1. Conclusion

Our work consisted in demonstrating the value of the methodology called PLANFORM-KA in supporting a knowledge acquisition process.

First of all, we have presented the basic steps of a methodology to build a representation of AI Planning case studies according to a given problem specifications. We have described how a cluster of constraints could help domain experts during the knowledge acquisition process and how the configuration of a cluster at any representation level can formalise the knowledge of a domain expert.

Second, we have validated our KA process through the building of the case studies such as Drumstore and EVentus and shown some results as follows:

- Instance coverage. This allows us to study the interaction with the domain expert,
- Two frame system. These introduce different abstraction levels of knowledge.
- Three AI Planning categories: `Agent`, which is a mobile thing like `Robot` or `Visitor`, `Object`, for example location (`Position`, `Place`, `Event`) and `Task`, which is specialised into action-based and state-based representations.
- The Constraint generic concept. It features an abstraction of several constraints defined at different representation levels.

Finally, we are building on the question-driven interface and expect soon to generate at least outline planning operators.

Putting this in the context of the overall PLANFORM project, different processes from GIPO/Opmaker, STAN/TIM and KA-TOOL offer a multistrategy toolset meant to deal with the acquisition challenge, with theory revision as a backbone making it possible to build a neat underlying logical model.

5.2. Further work

So far, we have built a framework for an intelligent system to solve a set of issues concerning the knowledge acquisition in AI Planning. We will make a systematic survey – at the epistemological level – of other approaches like PROTÉGÉ, EXPECT or PLANET, for instance, which focus on a similar approach with respect to reuse of ontology. A particular direction is to explore the use of generic types, [18] formulated by Planform co-researchers Fox and Long, within the question-driven acquisition module. Currently, generic types are extracted from PDDL domain models, but the FSM definitions used for this might be moved towards the domain expert through incorporation into Planform-KA. Thus once an expert identifies a mobile agent for example, the system could actively prompt for the possibility of route-following. Further case-study examples will be explored in order to assess the coverage Planform-KA is able to provide for domains where a domain model has already been created by hand. Finally, supporting the expert with a much larger ontology – possibly a specialised version of the CYC Upper ontology – will also be explored. This would then enable much more widespread trials of the system

References

1. M. Musen. Modern Architectures for Intelligent Systems: Reusable Ontologies and Problem-Solving Methods. In Chute (Eds), AMIA Annual Symposium, 46-52. 1998.
2. B. Hayes-Roth and F. Hayes-Roth. A Cognitive Model of Planning. Representation and Reasoning. Readings in Planning. Morgan Kaufman Publishers. 1990.
3. A. Valente. Planning models for the CommonKADS librairy. ESPRIT Project KADS-II. 1993. Available at http://www.swi.psy.uva.nl/usr/andre/publications.html.
4. J.F. Sowa. Knowledge representation: logical, philosophical and computational foundations. Brooks/Cole (eds). 2000.
5. T.L. McCluskey and Porteous J. M. Engineering and compiling planning domain models to promote validity and efficiency, Artificial Intelligence, pp.1-65. 1997.

6. T.R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, **2**(5), 1993.
7. Planform. An Open environment for building planners.
Available at http://helios.hud.ac.uk/planform. 1999.
8. R.S. Aylett, S. Jones. Planner and Domain: Domain Configuration for a Task Planner. Int. Journal of Expert Systems, **9**(2), 279-318, 1996.
9. C. Vogel. Le genie cognitif. Masson (Eds). 1988.
10. The Upper Cyc Ontology, available at http://www.cyc.com/cyc-2-1/cover.html.
11. N. Fridman-Noy et al. The knowledge model of Protégé-2000: combining interoperability and flexibility. 2th Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW). Juan-les-Pins (France) 2000.
12. Y. Gil and J. Blythe. How Can a Structured Representation of Capabilities Help in Planning? Proceedings of the AAAI – Workshop on Representational Issues for Real-world Planning Systems. 2000.
13. Y. Gil and J. Blythe. PLANET: A Shareable and Reusable Ontology for Representing Plan. Proceedings of the AAAI – Workshop on Representational Issues for Real-world Planning Systems. 2000.
14. Schreiber, G., Wielinga, B., de Hoog, R., Akkermans, H. and Van de Velde, W. (1994). CommonKADS: A Comprehensive Methodology for KBS Development. IEEE Expert, 9 (6), pp. 28-37..
15. J.M. Mercantini et al. Safety previsional analysis method of an urban industrial site. Scientific Journal of the Finnish Institute of Occupational Health, serie: People and Work, safety in modern society, pp. 105-109, 33. 2000.
16. N. Mercantini et al. Etude d'un systeme d'aide au diagnostic des accidents de la securite routiere. IC'99. Palaiseau (France). 1999.
17. J. L. Cybulski. Application of Software Reuse Methods to Requirements Elicitation from Informal Requirements Texts. PhD Thesis. http://www.dis.unimelb.edu.au/staff/jacob/publications/home.htm#Theses. 2001.
18. M. Fox, and D. Long. Automatic Synthesis and use of Generic Types in Planning. AIPS 2000 - Workshop on Analysis and Exploiting Domain Knowledge for Efficient Planning.
19. D. McDermott, et al. PDDL --- The Planning Domain Definition Language. In Machine Intelligence 4. D. Michie, ed., Ellis Horwood, Chichester (UK). 1998.
20. Breuker, J. and Wielinga, B. (1989). Models of Expertise in Knowledge Acquisition. G. Guida and C. Tasso (eds). Topics in Expert Systems Design: methodologies and tools. North Holland Publishing Company, Amsterdam, The Netherlands.
21. Benjamins, V. R. (1993). Problem Solving Methods for Diagnosis. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands.
22. D. Liu and T. L. McCluskey, The Object Centred Language Manual - OCLh - Version 1.2. Technical report, School of Computing and Mathematics, University of Huddersfield, 1999.
23. R. M. Simpson, T. L. McCluskey, W. Zhao, R. S. Aylett and C. Doniat 2001 An Integrated Graphical Tool to support Knowledge Engineering in AI Planning. Proceedings, 2001 European Conference on Planning, Toledo, Spain.
24. Blythe, J. 2001: Integrating expectations for different sources to help end users acquire procedural knowledge. IJCAI.
25. Grenier, R. 1998: The complexity of theory revision. Available at http://www.cs.ualberta.ca/~grenier.
26. Poole, D. 1998: Decision theory, the situation calculus and conditional plans in computer and information science. Vol. 3, nr 8.
27. Clark, M. 2001: Construction domains: a generic type solved. PLANSIG'01. Workshop of UK Planning and Scheduling. Edinburgh.
28. McCluskey L. & Richardson B. 2001: The Induction Operator Descriptions from Examples and Structural Domain Knowledge. Edinburgh.

29. Williams, M.A.: 1997: Implementing Belief Revision in G. Antoniou, Nonmonotonic reasoning. MIT Press.
30. CycCorp. 1999: more details about the KRAKEN project are available at http://reliant.teknowledge.com/RKF/proposals/Cycorp.html.

Appendix 1 – OCL model

```
domain_name(drumstore).

% Sorts
sorts(non_primitive_sorts,[thing,position]).
sorts(primitive_sorts,[robot,gripper,object,relation,reference]).
Sorts(thing,[robot,object]).

% Objects
objects(robot,[r1,r2,r3,r4]).
objects(gripper,[g1,g2,g3,g4]).
objects(object,[d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12]).
objects(relation,[near,at]).
objects(reference,[s1,s2,s3,s4,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b1
2,b13,b14,b15,b16]).

% Predicates
predicates([
     can_sense(robot,object,relation,reference),
     sense_on(robot),
     position(thing,relation,reference),
     full(gripper),
     empty(gripper),
     belongs_to(robot,gripper),
     in(object,gripper),
     released(object),
     in_range(reference,reference)]).

% Atomic Invariants
atomic_invariants([
     position(r1,at,d12),position(d9,at,d4),position(r2,near,s2),

belongs_to(r1,g1),belongs_to(r2,g2),belongs_to(r3,g3),belongs_to(r4,
g4),
     in_range(s1,b12),in_range(b12,s1),
     in_range(s2,b15),in_range(b15,s2),
     in_range(s3,b14),in_range(b14,s3),
     in_range(s4,b13),in_range(b13,s4),
     in_range(b13,b1),in_range(b1,b13),
     in_range(b15,b13),in_range(b13,b15),
     in_range(b12,b14),in_range(b14,b12),
     in_range(b14,b16),in_range(b16,b14)])
```