TF Method: An Initial Framework for Modelling and Analysing Planning Domains

Austin Tate†, Stephen T. Polyak‡, Peter Jarvis†

† Artificial Intelligence Applications Institute (AIAI)
‡ Department of Artificial Intelligence
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN United Kingdom
a.tate@ed.ac.uk, Steve_Polyak@ed.ac.uk, paj@aiai.ed.ac.uk

Abstract

Early work on the NONLIN and O-Plan projects indicated a need for a defined methodology which would guide users performing various roles in the acquisition and analysis of domain requirements for planning. This work included links to a requirement analysis methodology, CORE (COntrolled Requirements Expression), tool support via an intelligent assistant as part of the Task Formalism (TF) Workstation and an initial collection of guidelines and checklists to aid in using the TF domain description language. This paper describes work underway to follow-on from this past research and to infuse it with knowledge gained from recent research related to planning domain development, knowledge modelling, design rationale and ontological and requirements engineering.

Introduction

The activities involved in discovering, engineering, documenting, and maintaining a set of domain constructs for most AI planning-based projects can be considered ad hoc and disorganised, at best. The current sources for advice on the process of writing AI planning domain descriptions have been summarised as

> "... it is the most neglected aspect of planning, and there is not an established software-engineering methodology to guide this job". (Erol 1995)

Domain capture and modelling has been an issue in Edinburgh-based planning research as early as the work on the NONLIN (Tate 1977) planner. In fact, the original O-Plan overall architecture and system design, which dates from 1983, outlined a need for a defined methodology which would guide users performing various roles in the acquisition and analysis of domain requirements for planning (Currie & Tate 1991). This planning life-cycle methodology was envisioned as encompassing a set of standardised activities and methods which had well-defined design criteria, techniques, and tools. This was proposed to assist in transforming planning domain development from a craft towards more of an engineering activity.

The domain description language used by both the NONLIN and O-Plan planners is the Task Formalism (TF) (Tate 1977; Tate, Drabble, & Dalton 1994a). Early prototyping efforts on a PERQ-based TF Workstation (Tate & Currie 1984; 1985) demonstrated tool-support for the domain modellers (an expert providing the structure of the domain and specialists providing the details) and planners (acting in any one of a range of roles). This tool was designed to include an "intelligent assistant" which would interact with the user via a structured dialogue which was tied to a specific domain development methodology. Research was conducted into a requirements engineering methodology which could be adapted for use in this way. The Controlled Requirements Expression (CORE) (Mullery 1979; Curwen 1991) was proposed for structuring these domain management activities. It is hoped that an adaptation of this method, combined with experience in working with TF, would help to drive the development of planning domains in a more reliable fashion.

In this paper, we review past research efforts related to a move towards an overall TF Method framework. This includes a sampling of the guidelines and checklist included in the TF manual, advice on the use of TF condition types, work on prototype tool-support via the TF workstation and past research on links to the CORE methodology. These ideas form a base foundation upon which new efforts from the AI planning related domain modelling research community may be added.

In section 2, we present these components of the initial TF Method. Section 3 mainly reports on experience gained using this work in the development of domain models for the construction industry. A sampling of some of the existing research on domain development tools and approaches from the AI planning community is provided in section 4. Finally, in section 5, we widen the scope and discuss possible links to research in related fields.

Towards a TF Method

Guidelines for Writing TF

Initial work on pulling together the experience gained in coding specific domains in the Task Formalism domain description language resulted in a section on "guidelines for writing TF" which is part of the TF manual (Tate, Drabble, & Dalton 1994a). This section provides advice on the use of various TF forms and elements, which can be seen as a start towards a general framework for the development of a methodology which would structure the domain design activities.

This advice is rooted in a project management perspective which describes the need for preparatory steps and uses role identification prior to engaging in the development-oriented activities. The central controlling role was identified as the **Domain Expert** who is in charge of managing the scope of the domain and introducing a "top level" description (e.g. in a house building domain this person might be an architect with overall responsibilities for a project). For large domain engineering efforts, a partitioning of the domain development responsibilities was recommended. These modular sections of the domain were viewed as "detailed" aspects of the top level descriptions which were provided by the domain expert. **Domain Specialists** would then be assigned to particular domain partitions and would have responsibility for providing specifications of activities, objects (resources), events and effects which were relevant to their particular needs. The specialists may be subject matter experts (e.g. in a house building domain they might represent a plumber, or electrician, etc.). More likely, the domain expert and specialists may be knowledge engineers who have performed the required knowledge elicitation and acquisition activities from those with knowledge of the domain.

The guidelines point toward necessary project management decisions such as the choice between one of two "main approaches" toward modelling a domain: hierarchical action expansion or goal achievement (conditions on world states). While these approaches can be mixed in the specification of the domain, experience had shown that it is useful, if not important, to specify what the main approach will be for a particular domain development process and to treat the other approach as secondary to it.

Another important management decision considers the selection of a method for structuring the domain specifications. A level-oriented approach to domain modelling is proposed in this work whereby actions, events, effects, and resources are all separated into a series of defined and increasingly detailed levels. This helps to avoid the commonly experienced problem of "hierarchical promiscuity" (Wilkins 1988) or "level promiscuity" which is characterised by the inconsistent usage of various domain elements at varying areas in the overall domain description.

This level-oriented approach is further detailed via a checklist of activities which may suit either the action expansion approach or the goal achievement approach, depending on the ordering of the defined activities. This checklist includes the following activities:

- Identify the main actions (and events) that will appear at the top of a task or plan.
- Develop the detailed actions (and events) for lower action levels.
- Think about what world statements will be needed (effects) at which levels.
- Consider the conditions for actions. Ensure they are introduced at a level which is at or below the level at which the related effects are introduced.
- Add type information to restrict usage of conditions. Types are primarily used to differentiate what a condition means. This will lead to differences in which condition satisfaction methods apply. Consult the definitions of TF condition types (see section 2.2).
- Add resources at each level.
- Consider time restrictions and related information.

There are also notes on specific aspects/techniques

- Functional expression of properties
- Conditional actions
- Conditional effects
- Variable typing
- Modelling reusable, non-sharable resources (using conditions and effects)

TF Condition Types

The guidelines on the use of condition types described in the TF manual were detailed in (Tate, Drabble, & Dalton 1994b). While the advice found in this work is oriented more towards the search effects in the planning system, it has also provided a useful perspective for domain modellers working with levels to constrain the use of condition types. Experience has shown that condition types, such as Supervised, Unsupervised, and Only_use_if map to domain expertise. A verification step, which would take the specific condition types into account, would help to ensure that the modelling levels are valid and that the modeller was not misusing conditions or unsuitable effects by specifying them at the wrong level. This would assist the domain modeller with a careful consideration of the reasons why effects were introduced and conditions placed at a particular level. A consistent, verified model, extracted from this step, would address a major part of the "hierarchical promiscuity" problem.

CORE (COntrolled Requirements Expression)

COntrolled Requirements Expression (CORE) was a method developed by British Aerospace (Warton) and systems designers in the late 70's (Mullery 1979). Over time, the method has evolved and CORE now provides techniques for requirements capture, analysis and specification (Curwen 1991). The method can be used to partition problems into manageable modules which can be assessed using CORE analytical techniques. This ensures that the requirements for a specification are complete and consistent. Some of the strengths of this methodology include decomposability of requirements and traceability mechanisms between different levels of requirements.

The CORE specifications are expressed in terms of graphics, structured text and mathematically based notations. These resultant requirements models start from operational requirements which influence functional requirements and, in turn, impact implementation requirements (with non-functional requirements acting as functional and implementation constraints). Viewpoints are used as logical partitionings of the system under consideration. These are divided into **bounding viewpoints**, which can be viewed from a planning context as providers of unsupervised conditions and **defining viewpoints** which are analogous to activities which can achieve supervised conditions. Viewpoint decompositions correspond to node expansions. The CORE notion of "scope" addresses choices between elements which may be included in the domain, and breaks them down into "local scopes" which designate responsibilities for domain specialists.

It is envisaged that an adaptation of the CORE methods can be used to structure the activities of users acting in particular roles throughout the life-cycle of a domain. For example, a domain expert divides a domain into a series of tasks to be completed by specialists. A domain specialist can list the assumptions he/she will be making (e.g. walls have been built and foundation laid). Specialists can retrieve previous plans to modify. For each plan, a viewpoint decomposition process is applied to it. This includes some checking based on CORE analysis techniques:

- Does every node have at least one precursor?
- For every node which has a precondition, is the precondition satisfied by the current network or by another node at the same level or higher?
- Do precursor and successor assignments match?

CORE provides specialised techniques for inspecting the evolving specification/domain. One example is the "viewpoint to viewpoint role-playing" technique. Using this approach, a structured document is produced which defines a particular perspective within the domain (e.g. between a builder and a floor installation procedure, or between a carpet layer and a floor installation procedure, etc.) Techniques such as this one aid in combining the viewpoints by showing where conflicting requirements are present. CORE has been used previously as the controlling methodology for an expert system-based requirements analysis tool (Stephens & Whitehead 1984)[1]. This tool utilised knowledge of CORE via stored relations, entities, rules, and could answer questions related to a requirement specification such as: how, why, and why not.

Future work will seek to adapt the CORE methodology and to provide tool-based support for it in the structuring of planning domain development activities.

TF Workstation

The original O-Plan design described the development of an intelligent, graphical user interface between an AI planning system and its users. This tool was called the TF workstation (Tate & Currie 1984)[2]. The users of the TF workstation were separated into: those who describe the application domain; and those who require plans to operate within the domain.

During domain building, the workstation assisted in building up the details of the alternative actions possible in the domain, the resource or time constraints on the actions, and the ways in which actions can be combined, etc. In this role, it could communicate with a domain expert and possibly several domain specialists to elicit their knowledge about the applications domain.

The TF workstation also acted as the interface between a human planner and the AI planner. In

[1] Joint work with the O-Plan team in the mid 1980s explored the use of O-Plan as a planning assistant within the Analyst Workbench

[2] An example screen shot from the TF workstation is shown in appendix A

this role, which can be thought of as a coordination activity, the workstation sought details of the task for which a plan is required. It checked that sufficient domain knowledge was available to enable a solution to be found (if necessary, the system pointed omissions out to the human planner, domain expert, or domain specialist) and acted as an intermediary to enable the human and AI planners to jointly generate a valid plan.

A hook for an expert system-style agent interface was provided to perform various services such as searching for close matches for terminological differences or incomplete information, etc. Preliminary work on the use of the CORE methodology within the TF workstation was performed (Wilson 1984). Unfortunately, this research was set aside once the initial prototype was completed. Research is currently underway to extend the original TF workstation/methodology ideas as part of the Common Process Editor (CPE) which is a component in a framework for applying AI planning to manufacturing, military and business process management[3].

TF Compiler

The O-Plan TF compiler converts the Task Formalism language (coming from a file or from a domain editing tool) into the internal domain information used by the O-Plan planner. The compiler can be run incrementally and will add to or modify the existing domain information available to the planner. It is anticipated that facilities to change specific aspects of forms previously submitted will be provided, along with the current facility of simply replacing old forms or adding new ones. There is an interaction between the facilities provided by the compiler and the possible activities performed in a domain life-cycle methodology. Future work on a richer interface to the TF compiler will facilitate steps in domain knowledge management which may overlap with planning, replanning, execution, etc.

TF Method Experience

Domain Description Development for the Construction Industry

The initial TF Method components were used during a research project at The University of Brighton to guide the development of a TF encoding of planning knowledge elicited from the construction industry (Jarvis & Winstanley 1998; Jarvis 1997; Jarvis & Winstanley 1996a; 1996b). This section outlines this work to relate industrial experiences of the TF Method from individuals who were at the time independent of the O-Plan design team.

[3]An example screen shot from the Common Process Editor (CPE) is shown in appendix A

Planning the Development of a Domain Description

The first stage of the TF Method calls for a planned approach to the development of a domain description. It advises the identification of an overall domain expert to scope and structure the domain and a number of domain specialists to "fill-in" the structure with detailed knowledge. This approach worked well in the construction industry. The senior director used in the role of domain expert provided an overview of the planning process. Managers further down the organisation used in the role of domain specialist provided detailed knowledge about the areas in which they work and their interactions with other specialists.

The different views of the domain expert and domain specialists complemented one another. The expert understood the overall process and the relationships between each stage but not the detail of how each stage was performed. The specialists understood the detail of their area but not the complete context in which they worked. Reconciling these two views added a beneficial cross check to the modelling process. Mismatches were traced to one of two causes. Either the knowledge engineer had misunderstood a specialist's or expert's comments or an organisational problem had been encountered. In the former case, the mismatch provided a useful tool for prompting both specialist and expert to clarify their comments. In the latter case, the mismatch motivated the specialist and the expert to meet and clarify their perceptions of the actual planning process they engaged in.

Selecting between Action Expansion and Goal Achievement

The second stage of the TF Method recommends a conscious commitment to either action expansion or goal achievement as the primary modelling approach to a domain. Experimental modelling with both approaches was used to inform this decision. This experimentation categorised planning knowledge in the construction industry as being structured around the components of a building and the trades or specialists used to construct related groups of these components. A plumber, for example, is responsible for the installation of a building's bathroom fittings and a scaffolder is responsible for erecting the scaffolding that supports bricklayers in the task of constructing walls. This structure was readily mapped to the hierarchy of schemata inherent in the action expansion approach. Considering the earlier example, the overall task of installing a building's services was encapsulated within a single schema. This schema then refined to two schemas at a lower modelling level with the first

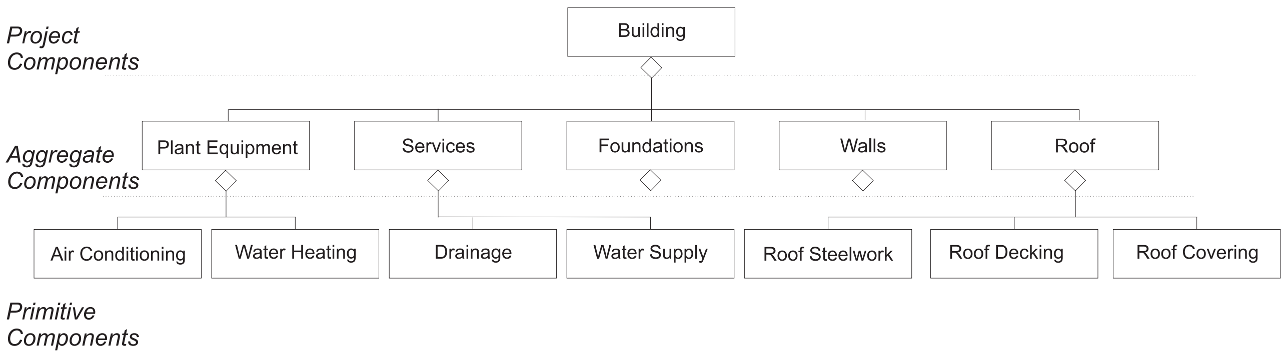

Figure 1: Construction Domain Model Partitioned into Modelling Levels

describing the work of the plumbing specialist and the second the electrician specialist.

This experience with the Task Formalism provides evidence to support Drummond's thesis (Drummond 1994) that industrial planning problems are more readily addressed by action expansion than by goal achievement techniques.

Developing the TF Schemata

The third stage of the TF Method suggests that each schema expansion level should hold some meaning within the domain under consideration. Figure 1 shows a section of the building subcomponent hierarchy developed from the meetings with domain experts. In the figure, a building is shown as being decomposed into a number of subcomponents (Plant Equipment through to a Roof). These components are decomposed further until the level of detail required for producing a construction plan is reached. This structure allows experts to reason at different levels of abstraction. The assignment of the construction of the roof component to a contractor would, for example, assume that the contractor would then be responsible for the construction of all the roof's subcomponents (Roof Steelwork, Roof Decking and Roof Covering).

Part of the TF encoding of the components in figure 1 is shown in figure 2. Figure 1 is partitioned through the dashed horizontal lines into the modelling levels: project components, aggregate components, and primitive components. These modelling levels were used to guide the encoding process. Considering the schemata in figure 2, the building component at the project modelling level in figure 1 is encoded as the initial task plan_buildings_construction. The transition from the project modelling level to the aggregate modelling level via the subcomponent relationship is achieved in the TF encoding through the schema refinement mechanism. The schema build_building refines the initial task (plan_buildings_construction) and it introduces a node for each subcomponent of the building that resides within the aggregate modelling level. The transition from the aggregate modelling level to the primitive modelling level is also achieved through schema refinement as demonstrated by the schema erect_roof. The schema, which will be used to refine node 2 in the build_building schema, contains an action for each subcomponent of the Roof component.

The encoding shown in figure 2 preserves both the subcomponent structure and the modelling levels elicited from the domain. Figure 3 shows part of the subcomponent structure from figure 1 augmented with the scope of the schemata that describe that structure in the TF encoding. The dashed lines represent modelling levels and the dotted lines the scope of each schema.

```
task plan_buildings_construction; ;; modelling level project componets
  nodes 1 task {build ?building};
end_task;

schema build_building; ;; modelling level aggregate components
  expands {build ?building}
  nodes 1 action {lay ?foundations},
        2 action {erect ?roof},
        3 ...
end_schema;

schema erect_roof; ;; modelling level primitive components
  expands {erect ?roof};
  nodes 1 action {install ?roof_steelwork},
        2 action {lay ?roof_decking},
        3 action {lay ?roof_covering};
end_schema;
```

Figure 2: Schemas build_building and erect_roof

As advised by the TF Method, the effects produced by actions were considered before the conditions required by actions. Each component was considered to determine the effect(s) that would result from its construction. The components at the higher modelling levels produce effects that describe the overall result of constructing their subcomponents. Constructing The Foundations component, for example, adds the effect {State_Of Foundations} = laid. The components at the lower modelling levels produce effects that describe their own construction. Constructing the

Roof Steelwork component, for example, adds the effect {State_Of _Roof_Steelwork} = erected. Figure 3 positions each effect within the same modelling level as the component which will produce it. Figure 4 shows how the schemata developed in figure 2 were modified to include these effects. The newly added TF elements in figure 4 are highlighted in bold.

Figure 3 contains both the effect levelling and schema scope information that is required to follow the guidelines on encoding action conditions described in (Tate, Drabble, & Dalton 1994b). Consider the roof decking component in figure 3. This component requires the roof steelwork to be in place before its own construction is started as the roof steelwork supports it. The scope and levelling information in figure 3 informs us that both the roof steelwork and the roof covering are introduced by the same schema. An ordering constraint and a supervised condition may therefore be placed between the actions to describe this. This encoding is shown in figure 4 within the erect_roof schema as the ordering constraint "1 $\rightarrow$ 2" and the condition "supervised {State_of Roof_Steelwork} = installed at 2 from [1]". The levelling information shown in figure 3 informs us that the actions are at the same modelling level. This situation conforms to the guideline that a supervised condition must be placed at the same or at a lower modelling level than the effect that satisfies it.

Consider the arrow between the roof steelwork and pile components within figure 3. The arrow is depicting the knowledge that the roof steelwork is supported by the pile. Figure 3 shows that these components are described in different schemas. Hence, an unsupervised condition must be used to describe the relationship. This knowledge is encoded within the schema erect_roof as the condition "unsupervised {State_Of Pile} = laid at [1]".

```
schema build_building; ;; modelling level aggregate components
  expands {build ?building}
  nodes 1 action {lay ?foundations},
        2 action {erect ?roof},
        3 ...
  only_use_for_effects
        {state_of foundations} = laid at 1,
        {state_of roof} = erected at 2.
        ...
end_schema;

schema erect_roof; ;; modelling level primitive components
  expands {erect ?roof};
  nodes 1 action {install ?roof_steelwork},
        2 action {lay ?roof_decking},
        3 action {lay ?roof_covering};
  orderings 1-->2;
  condtions
        supervised {state_of roof_steelwork} = installed at 2 from [1],
        unsupervised {state_of pile} = laid at [1];
  only_use_for_effects
        {state_of roof_steelwork} = installed at 1,
        {state_of roof_decking} = installed at 2,
        {state_of roof_covering} = laid at 3;
end_schema;
```

Figure 4: Schemas build_building and erect_roof augmented with action conditions and effects

Conclusions Drawn from this Experience

The TF Method provided a principled set of guidelines that aided the development of a TF representation of an aspect of the construction industry. The division of domain experts into the roles of expert and specialist mapped to the different views on planning knowledge that were held by people working in the domain. Reconciling these views provided a useful cross check that encouraged the knowledge engineer to clarify knowledge as it was elicited and domain experts to meet to clarify their own understandings of their domain.

The method made clear the importance of mapping schema expansions to modelling levels within the domain and it provided guidelines for ensuring the appropriate positioning of action conditions and effects within those levels. These guidelines assisted the development of a principled model of the domain.

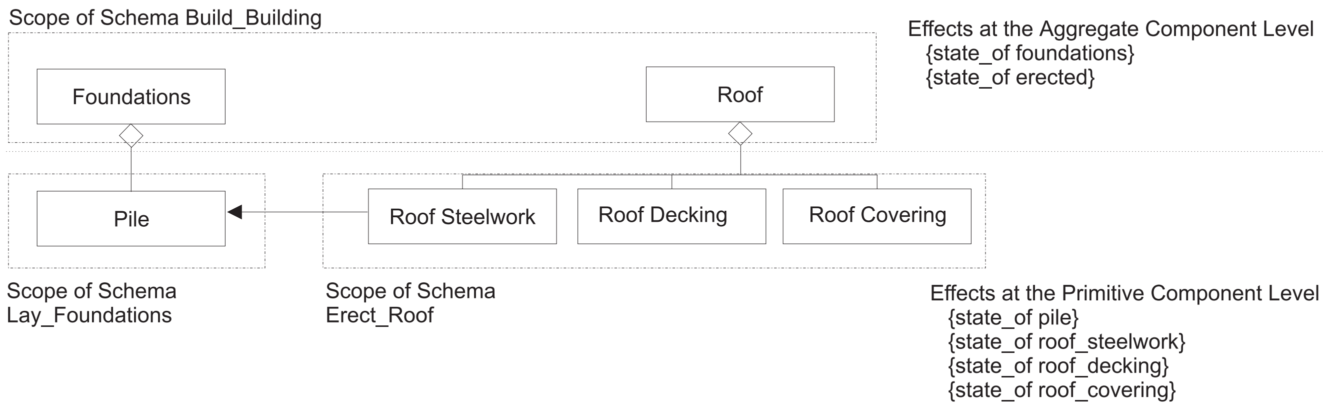

Figure 3: Schema Scope, Effects and Condition Types

The weakness of the method is the absence of tool support. The knowledge engineer must use pencil and paper to construct and maintain the figures shown in this section. Tools can be provided to automatically show the scope of schemas, highlight the levels of effects relative to a particular condition, and warn the knowledge engineer when the guidelines for relating condition types to modelling levels are violated.

Common Process Editor

Recent research on a Common Process Framework (CPF) is seeking to facilitate process management in a business and manufacturing application using AI planning representations. This framework includes tool support via a Common Process Editor (CPE) which acts as both the process visualisation and domain management tool for users. An example screen shot from the Common Process Editor (CPE) is shown in appendix A. Connection to an intelligent planning agent (e.g. O-Plan) allows for system-supported generation of business and manufacturing processes. A Common Process Assistant (CPA), which is also accessed as an agent, is used to perform analyses of the processes.

The language used to communicate between the CPE and the planning agent is currently the Task Formalism. This has provided us with insights into the use of TF as part of an integrated process management system. A defined TF Method may be adapted for use in structuring of activities related to the design, modelling, and maintenance of these processes. This tool-based assistance will help address the missing support mentioned in section 3.5.

Related Domain Research

A number of recent efforts in the AI planning research community have produced a variety of representations, approaches, tools, and architectures for working with AI planning domains. These range from machine learning approaches to user-based knowledge acquisition tools. This section samples some of the scope of ideas which may be utilised to provide a more effective methodology. We briefly present each approach in terms of its contribution and then discuss some of the possible issues.

A Formalisation of HTN Planning (Erol 1995)

- *Contributions*: Formal representation of Hierarchical Task Network (HTN) planning that gives a clear understanding of what the different constructs and condition types mean. This gives the knowledge engineer a formal underpinning which they may consult to clarify precisely the operation of different facets of an HTN planner and how the constructs supported by HTN representational devices affect this operation. This work also presents a list of steps to follow when encoding a domain description.

- *Issues*: The work is only accessible to AI planning specialists and cannot be readily understood by domain experts. It does, however, provide a foundation for understanding HTN planning that planning specialists can use to guide them in the writing of user oriented methods like the guidelines in the TF manual.

An Object-centred Specification Approach (McCluskey & Porteous 1997)

- *Contributions*: The authors seek to provide support for constructing planning domain descriptions by adapting methodological steps and notations of the object-oriented community. This approach utilises the notion of "lifting" domain representation from the level of the literal to the level of the object. Once a domain has been described in terms of a state transition graph, the author's algorithms compile the diagram into a STRIPS (Fikes & Nilsson 1971) style action representation.

- Issues: This work assumes that a domain can be described as a state transition graph (STG). The technique cannot currently generate HTN representations. This might be possible if it is extended to include techniques which use hierarchies of STGs. However, there does not appear to be a mechanism for inferring condition types.

Domain Analysis Techniques and Tools (Chien 1996)

- *Contributions*: Chien provides two types of tools for planning knowledge base development: static KB analysis techniques to detect certain classes of syntactic errors and completion analysis techniques to iteratively debug the planning knowledge base. This tool set supports typical user questions when investigating these types of error.

- *Issues*: The tool set can only be used after a significant proportion of a domain description has been elicited. It doesn't directly address how this initial description is to be constructed. Some AI planners may already perform such forms of domain checking during domain compilation.

Automatically Learning Operators (Wang 1996)

- *Contribution*: Takes a set of example plans described in terms of the actions in each plan and the state of the world before and after each action. The system examines these examples and generates the preconditions and effects of operator descriptions.

- *Issues*: The technique assumes that the user can provide example plans described in terms of the state of the world before and after each action. It provides no assistance for the construction of these example plans. Again, the technique is only applicable to STRIPS style planning not HTN.

A number of other contributions from the AI planning community may be useful sources for the development of the TF Method as well. These works include architectures, such as the EXPECT knowledge acquisition architecture (Swartout & Gil 1996) which dynamically forms expectations about the knowledge that needs to be acquired by the system and then uses these expectations to interactively guide the user through the knowledge acquisition process. There are are also specialised techniques, for example, knowledge acquisition on the fly (i.e. during planning) (desJardins 1996) and tools for editing operators and domain knowledge (e.g. Act editor (Myers & Wilkins 1997), Operator editor (desJardins 1996), etc.).

Integrating with Other Research Areas

An increasing number of requirements are being placed on both domain representations and the processes in which these artifacts are created, maintained etc. as we forge ahead toward future implementations of artificial intelligence planning systems. Domain development methods require solid modelling techniques and well-defined, accepted concepts and terminology. Aspects of the domain may be linked to a specific set of possibly dynamic requirements. Modifications to the domain throughout its life-cycle may require contextual knowledge which expresses the rationale for particular domain design decisions.

Some of these issues facing applied planning efforts are being addressed by related research areas. These areas may provide sources of techniques, methods and guidelines which can be combined with AI domain development approaches to provide a more robust methodology. We briefly outline four possible research areas: knowledge modelling, ontology engineering, requirements engineering, and design rationale.

Knowledge Modelling

Several approaches have been developed to tackle AI planning problems (Allen, Hendler, & Tate 1990). While the result is a rich corpus of techniques and methods, it is proving to be a very difficult task to compare and contrast each approach. Some researchers believe the best way is to chart these results with detailed algorithmic treatment (Kambhampati, Knoblock, & Q. 1995). Barros, Valente, and Benjamins present a differing perspective whereby the focus is on an abstract analysis which highlights the capabilities of the system and the way it represents and uses knowledge (Barros, Valente, & Benjamins 1996).

This knowledge modelling research utilises the CommonKADS (Wielinga *et al.* 1992; Breuker & van de Velde 1994) methodology which outlines a set of detailed models to be created for an analysis. The AI planning community has gained a more informed perspective on the ways knowledge is used in various planning systems via the application of these methods (Jarvis 1997; Barros, Valente, & Benjamins 1996; Kingston, Shadbolt, & Tate 1996; Cottam *et al.* 1995; Valente 1995). A hybrid domain life-cycle methodology that integrates these model-building techniques along with the current methods and guidelines from AI planning domain development could aid in lifting the domain engineers level of interaction with the domain and improve the overall construction process.

Ontology Engineering

Planning domain ontologies are specifications of the concepts, terms, relations, etc. that form the basic language used to describe a domain (Valente 1995). These specifications or definitions, expressed either informally or formally (Uschold & Gruninger 1996), help to clarify the semantics of the planning domain concepts. Domain ontologies, along with domain-independent ontologies (cf. (Tate 1996c; 1996b)), characterise elements in the planning world model separately from any particular system that is reasoned about (generative planning system, plan evaluation system, etc.). Shared domain ontologies (i.e. two or more systems/groups agree to defined terminology) assist in breaking down some of the arbitrary differences at the knowledge level and facilitate knowledge sharing (Neches *et al.* 1991).

A methodology which seeks to address the construction of plan domain models in an environment where knowledge sharing is required must somehow be connected or combined with a methodology for building a shared domain ontology. Recent ontological engineering research has begun to address the design and development of such methodologies (Fernández, Gómez-Pérez, & Juristo 1997; Gómez-Pérez, Fernández, & Vicente 1996; Mizoguchi, Vanwelkenhuysen, & Ikeda 1995). For example, Gómez-Pérez et. al. propose the following set of phases (Gómez-Pérez, Fernández, & Vicente 1996)

- Acquire Knowledge
- Build a requirements specification document

- Conceptualise the ontology
- Implement the ontology
- Evaluation during each phase
- Documentation after each phase

Some researchers propose general guidelines or techniques, such as a "middle-out" approach (Uschold & Gruninger 1996) in which a glossary of terms is used to define an initial set of primitive concepts which, in turn, are used to define new ones. Other researchers propose more domain-specific approaches such as the ontology building process utilised for disturbance diagnosis and service recovery planning in electrical networks (Bernaras, Laresgoiti, & Corera 1996). Techniques developed in these projects may be candidates for integration into a planning domain development tool-box.

Requirements Engineering

Significant work in requirements engineering has been made since the early O-Plan research into adopting the CORE methodology for use in planning domain development. This includes work on viewpoint management and stake-holder analysis (Easterbrook & Nuseibeh 1996; Kotonya & Somerville 1996; Finkelstein *et al.* 1994), as well as work on various methodologies, techniques, and guidelines (Sommerville & Sawyer 1997; van Lamsweerde & Letier 1998) for eliciting, recording, and managing requirements.

Connecting domain aspects to their underlying requirements may assist in managing domain modifications which are the result of changing needs of an organisation. Clearly defined roles and responsibilities at the requirement level will help to organise the activities at the domain level. This will help to address one of the major impediments which has prevented the adoption of AI planning tools and techniques in applied settings: a lack of organisational context.

Design Rationale

A design rationale is a representation of the reasoning behind the design of a system (Shum 1991). It is essentially the explicit recording of the issues, alternatives and justifications that were relevant to elements in the design of an artifact. Examples of design rationale implementations include: QOC (MacLean *et al.* 1991), DRL (Lee 1990), gIBIS (Conklin & Begeman 1988).

Large plan domains utilised within organisations can be viewed as complexly designed artifacts. These artifacts are managed, reviewed, and maintained just as information systems are. A methodology which encompasses the development of such artifacts may need to support the recording and replay of the rationale for the decisions taken during its design. In a recent review of planning rationale, we described a method for incorporating design rationale in planning (Polyak & Tate 1998). We believe that the benefits of a design rationale approach (Moran & Carroll 1996), will aid in the reasoning, analysis and communication of planning domain knowledge.

Summary

This paper has presented perspectives on an initial framework which will assist in the process of modelling and analysing planning domains. These perspectives are based on past and present research efforts in: the TF guidelines; TF workstation; and integrating with a requirements engineering methodology, experience acquired in working with TF domains and insights gained through the other efforts of planning, knowledge modelling, ontology and requirements engineering, and design rationale research groups. We believe that a synthesis of the techniques and methods found in these works will be essential for improving the quality of AI planning domain management throughout its organisational life-cycle.

Acknowledgements

The O-Plan project is sponsored by the Defence Advanced Research Projects Agency (DARPA) and the U.S. Air Force Research Laboratory (AFRL), under grant number F30602-95-1-0022. The O-Plan project is monitored by Dr. Northrup Fowler III at AFRL (Rome). One author is sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-96-1-0348 – an AASERT award monitored by Dr. Abe Waksman and associated with the O-Plan project. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA, AFRL or the U.S. Government.

References

Allen, J.; Hendler, J.; and Tate, A., eds. 1990. *Readings in Planning.* Palo Alto, CA: Morgan Kaufmann.

Barros, L.; Valente, A.; and Benjamins, R. 1996. Modeling planning tasks. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, 11–18. Edinburgh, Scotland: Morgan Kaufmann.

Bernaras, A.; Laresgoiti, I.; and Corera, J. 1996. Building and reusing ontologies for electrical network applications. In *Proceedings of the European Conference on Artifical Intelligence (ECAI) '96*, 298–302.

Breuker, J., and van de Velde, W. 1994. *The CommonKADS Library for Expertise Modelling: reusable components for artificial problem solving.* Amsterdam, Tokyo: IOS Press.

Chien, S. 1996. Static and completion analysis for planning knowledge base development and verification. In Drabble (1996), 53–61.

Conklin, E., and Begeman, M. L. 1988. gIBIS: A hypertext tool for explanatory policy discussion. *ACM Transactions on Office Information Systems* 6:303–331.

Cottam, H.; Shadbolt, N.; Kingston, J.; Beck, H.; and Tate, A. 1995. Knowledge level planning in the search and rescue domain. In *Research and Development in Expert Systems XII, proceedings of BCS Expert Systems'95.*

Currie, K., and Tate, A. 1991. O-Plan: the open planning architecture. *Artificial Intelligence* 52:49–86.

Curwen, P. 1991. System development using the CORE method. Military Aircraft Ltd. BAe/WIT/ML/GEN/SWE/1227, British Aerospace, PLC, Warton Aerodrome, Preston, UK.

desJardins, M. 1996. Knowledge acquisition tools for planning systems. In Tate (1996a), 53–61.

Drabble, B., ed. 1996. *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96).* Edinburgh, Scotland: Morgan Kaufmann.

Drummond, M. 1994. On precondition achievement and the computational economics of automated planning. In Backstrom, C., and Sandewall, E., eds., *Current Trends in AI Planning.* IOS Press. 6–13.

Easterbrook, S., and Nuseibeh, B. 1996. Using viewpoints for inconsistency management. *Soft. Engin. Journ.* January.

Erol, K. 1995. *Hierarchical Task Network Planning: Formalisation, Analysis, and Implementation.* Department of computer science, University of Maryland, College Park, USA.

Fernández, M.; Gómez-Pérez, A.; and Juristo, N. 1997. Methontology: From ontological art towards ontological engineering. In *Workshop on Ontological Engineering, Spring Symposium Series, AAAI97.*

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Finkelstein, A.; Gabbay, D.; Hunter, A.; Kramer, J.; and Nuseibeh, B. 1994. Inconsistency handling in mulit-perspective specifications. *Trans Software Eng* 20(8):569–578.

Gómez-Pérez, A.; Fernández, M.; and Vicente, A. D. 1996. Towards a method to conceptualize domain ontologies. In *Workshop on Ontological Engineering, ECAI'96*, 41–51.

Jarvis, P., and Winstanley, G. 1996a. Dynamically assessed and reasoned task (DART) networks. In *Proceedings of the Sixteenth Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems (ES-96), Cambridge, UK.*

Jarvis, P., and Winstanley, G. 1996b. Objects and objectives: the merging of object and planning technologies. In *Proceedings of the Fifteenth Workshop of the UK Planning and Scheduling Special Interest Group, Liverpool, UK.*

Jarvis, P., and Winstanley, G. 1998. Reducing the semantic gap between application domains and AI planning technology: a compilation based approach. In *Workshop on Knowledge Acquisition and Knowledge Elicitation, to be held within the Fourth International Conference on Artificial Intelligence Planning Systems, Pittsburgh, USA.*

Jarvis, P. 1997. *Integration of Classical and Model–Based Planning.* PhD thesis, School of Computing and Mathematical Sciences, University of Brighton, Sussex, UK.

Kambhampati, S.; Knoblock, C.; and Q., Y. 1995. Planning as refinement search: a unified framework for evaluating design tradeoffs in partial–order planning. *Artificial Intelligence* 76.

Kingston, J.; Shadbolt, N.; and Tate, A. 1996. CommonKADS models for knowledge based planning. Artificial Intelligence Application Institute AIAI-TR-199, University of Edinburgh, Edinburgh, Scotland.

Kotonya, G., and Somerville, I. 1996. Requirements engineering with viewpoints. *Soft. Engin. Journ.* 11(1).

Lee, J. 1990. SIBYL: A qualitative decision management system. In Winston, P., and Shellard, S., eds., *Artificial Intelligence at MIT: Expanding Frontiers.* MIT Press. 104–133.

MacLean, A.; Young, R.; Bellotti, V.; and Moran, T. 1991. Design space analysis: Bridging from theory to practice via design rationale. In *Proceedings of Esprit '91*, 720–730.

McCluskey, T., and Porteous, J. 1997. Engineering and compiling planning domain models to promote validity and efficiency. *Artificial Intelligence* 95(1):1–65.

Mizoguchi, R.; Vanwelkenhuysen, J.; and Ikeda, M. 1995. Task ontology for reuse of problem solving knowledge. In *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing.* IOS Press. 46–59.

Moran, T., and Carroll, J., eds. 1996. *Design Rationale: Concepts, Techniques, and Use.* Lawrence Erlbaum Associates.

Mullery, G. 1979. CORE: A method for controlled requirements specification. In *Proceedings of the 4th International Conference on Software Engineering.*

Myers, K., and Wilkins, D. 1997. The act-editor user's guide: A manual for version 2.2. SRI International Artificial Intelligence Center, Stanford University, Menlo Park, CA.

Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; and Swartout, W. 1991. Enabling technology for knowledge sharing. *AI Magazine* Fall.

Polyak, S., and Tate, A. 1998. Rationale in planning: Causality, dependencies, and decisions. *Knowledge Engineering Review* 13(2):1–16.

Shum, S. 1991. Cognitive dimensions of design rationale. In Diaper, D., and Hammond, N., eds., *People and Computers VI*, 1–13. Cambridge: Cambridge University Press.

Sommerville, I., and Sawyer, P. 1997. *Requirements Engineering: A Good Practice Guide.* John Wiley and Sons.

Stephens, J., and Whitehead, R. 1984. The analyst – an expert system approach to requirements analysis. In *Proceeding of the third seminar on Application of Machine Intelligence to Defence Systems.*

Swartout, W., and Gil, Y. 1996. EXPECT: A user-centered environment for the development and adaptation of knowledge-based planning aids. In Tate (1996a), 250–258.

Tate, A., and Currie, K. 1984. The O-Plan task formalism workstation. Artificial Intelligence Applications Institute (AIAI) AIAI-TR-7, University of Edinburgh.

Tate, A., and Currie, K. 1985. The O-Plan task formalism workstation. In *Proceedings of the Third Workshop of the UK Alvey Programme's Planning Special Interest Group.* London, UK: Institute of Electrical Engineers.

Tate, A.; Drabble, B.; and Dalton, J. 1994a. Task formalism manual. Artificial Intelligence Applications Institute AIAI-TF-Manual, University of Edinburgh, Edinburgh, UK ftp://ftp.aiai.ed.ac.uk/pub/documents/ANY/oplan-tf-manual.ps.gz.

Tate, A.; Drabble, B.; and Dalton, J. 1994b. The use of condition types to restrict search in an AI planner. In *Proceedings of Twelfth National Conference on AI (AAAI-94), Seattle.*

Tate, A. 1977. Generating project networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-77)*, 888–893.

Tate, A., ed. 1996a. *Advanced Planning Technology: Technological Advancements of the ARPA/Rome Laboratory Planning Initiative.* Menlo Park, CA: AAAI Press.

Tate, A. 1996b. Representing plans as a set of constraints – the < I-N-OVA > model. In Drabble (1996), 221–228.

Tate, A. 1996c. Towards a plan ontology. *AI*IA Notiziqe (Publication of the Associazione Italiana per l'Intelligenza Artificiale), Special Issue on Aspects of Planning Research* 9(1):19–26.

Uschold, M., and Gruninger, M. 1996. Ontologies: Principles, methods and applications. *Knowledge Engineering Review* 11(2).

Valente, A. 1995. Knowledge-level analysis of planning systems. SIGART *Bulletin* 6(1).

van Lamsweerde, A., and Letier, E. 1998. Integrating obstacles in goal-driven requirements engineering. In *Proceedings (ICSE'98) – 20th International Conference on Software Engineering, (IEEE-ACM).*

Wang, X. 1996. Planning while learning operators. In Drabble (1996), 229–236.

Wielinga, B.; van de Velde, W.; Schriber, G.; and Akkermans, H. 1992. The KADS knowledge modelling approach. In *Proceedings of the Japanese Knowledge Acquisition Workshop.*

Wilkins, D. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann.

Wilson, A. 1984. *Information for Planning.* M.Sc. Thesis, Department of Artificial Intelligence, University of Edinburgh, UK.

Sample Graphical User Interface Screens

These two screen shots are examples from the TF workstation (top, 1984) and the Common Process Editor (CPE) (bottom, 1998) which provide tool-supported assistance for plan/process management.