A Problem-Solving Method for Plan Evaluation and Critiquing

Jim Blythe and Yolanda Gil

USC, Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292 USA
blythe@isi.edu, gil@isi.edu

Abstract

As intelligent systems become larger and work in increasingly more complex and knowledge intensive environments, it becomes impractical to develop knowledge bases from scratch. A practical alternative is to develop knowledge bases out of reusable components, either ontologies or domain-independent problem solving methods. This paper describes a reusable knowledge base of general principles about plan evaluation and critiquing. It includes ontologies to represent plans and critiques, an ontology of evaluation criteria, and problem-solving knowledge about how to evaluate plans with respect to those criteria. Currently, the evaluation criteria analyze various aspects of the plan structure and its use of resources. We performed a study of three evaluation and critiquing task domains, and found that our framework covers approximately 75 percent of the critiquing criteria related to plan structure and resource use (and over 90 percent within the framework's declared scope). Our framework also enables a systematic approach to the development of plan evaluation tools by following the organization of the criteria in the method.

1 Introduction

As intelligent systems tackle large-scale knowledge intensive tasks, it becomes impractical or even impossible to develop knowledge bases from scratch. The cost of developing a large knowledge base is high, and it would be useful to transfer as much of its contents as possible when building a new system to perform a related task. Many researchers advocate the development of large, reusable knowledge components, including ontologies (Neches *et al.*, 1991), and Problem-Solving Methods (PSMs) for tasks such as design and diagnosis (McDermott, 1988; Wielinga *et al.*, 1992). These components can be organized into libraries and can be combined together to develop the core of new knowledge-based systems, which is then augmented with knowledge specific to the domain and application at hand.

There are several benefits of reusable PSMs. First, they support the initial stages of KBS design by providing a systematic approach to eliciting knowledge from experts about the task and the domain and by supporting a principled implementation of the evaluation and critiquing system. Second, they support the development of systems in new domains by 1) reuse of ontologies and problem solvers to implement parts of the new system and 2) enabling knowledge acquisition tools to help users add task knowledge that is specific to the new domain. Third, they support knowledge acquisition to populate the knowledge base based on the ontologies associated with the PSM that specify the information required to do problem solving.

We are developing a domain-independent and reusable knowledge base that can be used to develop plan evaluation and critiquing systems. The knowledge base is composed of ontologies and associated problem-solving knowledge to reason with them. Unlike other PSMs, our PSM is implemented and has been used to develop a Course of Action Critiquer that reuses the ontologies and problem solving knowledge in the PSM. We implemented this PSM in the EXPECT framework (Gil & Swartout, 1994), which provides a language to express problem-solving knowledge that is tightly coupled with ontology representations. Our implementation follows the philosophy described in (Gil & Melz, 1996) to represent the propose-and-revise PSM in EXPECT.

EXPECT's Plan Evaluation PSM is designed for analytical plan evaluation and plan critiquing tasks. It is given an input plan that has been generated by a user, a system, or in mixed-initiative mode. The input may include information about the initial world state assumed, about the desired objectives, and about any constraints that the plan should comply with. The plan is analyzed with respect to the set of principles embodied in the knowledge base. EXPECT's Plan Evaluation PSM focuses on two types of evaluations: detecting ill-formed plan descriptions and the use of resources. The PSM includes ontologies and problem-solving knowledge. The ontologies include an ontology of dimensions for evaluation and types of critiques, and an ontology of resources and resource types. In addition, there is a plan ontology that captures the assumptions made by the PSM about the plan representation. The problem-solving knowledge is represented in EXPECT methods, which include methods that structure and control the execution of the critiques, methods that implement critiques that are constant across domains, and methods that describe general principles that can be customized to particular domains using EXPECT's knowledge acquisition tools.

This work builds in part on research on planning and scheduling algorithms and on practical experience in evaluation in planning domains. We are drawing from past experience in building plan evaluation systems in real-world domains, notably in military domains such as logistics transportation and air campaign planning but also in manufacturing. In these domains, the plan creation process is mostly manual and as a result it is prone to errors that a plan evaluation and critiquing tool can help detect and correct. A first group of evaluation criteria in our PSM is concerned with *plan structure*. Generative planners such as SIPE, NOAH, and NONLIN (Wilkins, 1988; Sacerdoti, 1977; Tate, 1977) use critics that detect problems in the plans that they generate while planning. These critics detect, for example, harmful interactions between goals and inadequate resource assignments. Some of our evaluations are inspired by these critics. In addition, plans that are manually created have other potential flaws that the plans generated by these tools do not have by definition (e.g. an unnecessary step). Our PSM checks such potential flaws. Formal analyses of partial-order and hierarchical planning algorithms define some desirable properties of plans, such as justifiability and correctness (Kambhampati *et al.*, 1995; Yang, 1990; Tate, 1996), that can be checked by a plan evaluation and critiquing tool. The second group of evaluation criteria in our method is *resource use*, an important dimension for plan evaluation in many domains and a central component of scheduling systems and in some cases of generative planners (Smith & Lassila, 1994; Wilkins, 1988). Theories and models of resources have been developed that characterize the nature of resources (e.g., reusable, linear, local) so they can be analyzed accordingly to evaluate their use (Smith *et al.*, 1996). Other recurring principles for plan evaluation that can be generalized across domains include assessment of risk, execution properties, assessment in adversarial situations, and comparison of alternatives based on a battery of evaluations of the plan under different perspectives.

In the next section we present some general principles of plan evaluation that are useful across a number of planning domains, and in the subsequent section describe the PSM for plan evaluation based on these principles. We then describe the ontologies that are associated with this PSM, including a plan ontology, a resource ontology, and an evaluation ontology. Next we present our experiences customizing the PSM for three different plan evaluation domains. Finally we discuss related work and present the conclusions of this work.

2 Capture of General principles

We define some domain-independent principles of plan evaluation that will structure the PSM. These are divided into two types, the first concerned with checking a plan's structure and the elements of its description, and the second concerned with its use of available resources.

2.1 Checking plan and action structure

The PSM contains evaluations concerned with debugging ill-formed plan descriptions. They detect problems in the subcomponents that describe the plan. These subcomponents include the goals that are achieved by the plan, the actions taken in a plan, their interdependency relations, parent/child relations, and ordering constraints. Notice that in different domains the representation of a plan may use only some of these components. For obvious reasons, these evaluations are usually considered before other aspects of the plan are evaluated.

2.1.1 Statement structure: checking the description of an individual plan component

complete statements: checking that all the necessary information is present in a description (ex: stating an action but not specifying an agent to achieve it, as required.)
clear statements: checking that descriptions are not vague or inappropriate (ex: stating "conduct operations" is a vacuous specification of what is to be done.)
correct statements: checking that no inappropriate item (agent, object, action description, etc.) are used as part of a description (ex: stating that an attack mission will be done with a refueling aircraft such as a tanker).
coherent statements: checking that descriptions refer to something that exists and/or makes sense given the world state (ex: stating that a company will perform its task in an area outside of its assigned area of responsibility)

2.1.2 Interdependency links: checking the links among components

Links include decomposition links (in hierarchical plans), sequencing links that specify orderings, and establishment links that indicate that a subplan relies on a previous subplan to achieve (or establish) something.

correct parent/child link: checking an assertion that a subtask is art of a higher-level task. Note that some parent/child links are analyzed when checking whether plans are unjustified or incomplete (see below)
missing parent/child link: checking that every task accomplishes a higher-level task and is decomposed into subtasks when it is not a primitive task.
correct causal link: checking that assertions that a condition needed for a task is achieved by a previous task is true given the nature of both tasks.
missing causal link: checking that there are links between every task and the tasks that achieve the conditions that are needed by it.
correct ordering link: checking that all orderings are consistent.
missing ordering links: checking that all necessary orderings and precedence relations are specified.

2.1.3 Plan structure problems: checking the global composition of the plan.

complete plan: checking that the plan achieves the given objectives and includes all the required tasks.
clear plan: checking that the plan is clear, simple, and understandable.
correct plan: checking that the plan complies with given constraints and preferences (for example, with rules of engagement or user preferences).

consistent plan: checking that the plan is consistent with respect to the state of the world assumed (for example, that the plan does not assume that some assets are the location where they will be used if the assets are initially assumed to be in a central location).

justified plan: One can check whether some part of the plan is unnecessary, because it does not achieve given objectives or because the intended effects are already true of the world state or achieved by another task, thus making the plan suboptimal. However, the requirement of a single task to achieve a goal may be too strict in some domains where redundancy is desired, so the notion is generalized to allow a varying weighted sum of tasks for a goal (see Figure 1). This check may be as complex as a probabilistic analysis of goal achievement or as simple as counting the tasks that contribute to a goal.

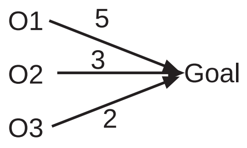

FIGURE 1: Weighted links for threshold justifications on a step.

task reuse: Just as the justified plan check looks at the in-degree of nodes in the the structural graph of the plan, so the task reuse check looks at their out-degree. In the two plan fragments if Figure 2, the goals **G1** and **G2** are achieved by two separate steps in the first case and by one step in the second. Which is preferable depends on the domain: the one-step case might be viewed as economical or as placing the goals under too much jeopardy.

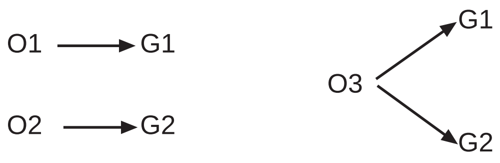

FIGURE 2: Reuse structures

Other structural evaluations may be specific to the domain at hand and not correspond to any of the above categories. For example, checking that an objective is decomposed appropriately given domain knowledge about what are correct decompositions.

2.2 Checking a plan's use of resources

The Plan Evaluation PSM contains knowledge to evaluate the use of resources by a plan. Resource use is central in planning domains because plans are built within given resource constraints. It is useful to identify the types of objects in a domain that can be considered as resources and perform an evaluation with respect to them. Evaluating a plan with respect to its use of resources can unveil aspects of the plan that are unfeasible because they incorrectly assume the availability of some resource, and can also determine if the plan is using the given resources adequately. Note that time can be and often is viewed as a resource.

Evaluating resource use can be broken down similarly to the structural checks on a plan:

Task-level resources: checking the use of resources for individual tasks in the plan (eg the time taken by each task). The same components apply as for statement structure, for example completeness means checking that necessary resources are identified

Interdependency links among resources: one task can be used to supply a needed resource for another task. When this relation is explicit we can check correctness and completeness.

Plan-level resources: checking the use of resources across the entire plan (eg the time taken to execute the whole plan).

Some evaluations detect problems in the use of resources (for example, a non-shareable resource that cannot be used by two concurrent actions). Other evaluations do not point out problems but rather generate an estimate of the use of resources in the plan that is interesting to report (for example, "the total fuel consumption is 500 gallons"). The kinds of resources that are encountered in plan evaluation are represented in a separate resource ontology, described in section 4.4.

3 Explicit Representations of PSMs in EXPECT

This section gives a short overview of how PSMs can be represented in EXPECT. (Gil & Melz, 1996) provide a more detailed account using propose-and-revise as an example. In EXPECT, both factual knowledge and problem-solving knowledge are represented explicitly. Factual knowledge is represented in Loom (MacGregor, 1991), a state-of-the-art knowledge representation system based on description logic. Factual knowledge includes concepts, instances, and the relations among them.

Problem-solving knowledge is represented in a procedural-style language that was developed for EES (Swartout *et al.*, 1991), a predecessor of EXPECT. This language is tightly integrated with the Loom representations. Subgoals that arise during problem solving are solved by methods. Each method description specifies: 1) the goal that the method can achieve, 2) the type of result that the method returns, and 3) the method body that contains the procedure that must be followed in order to achieve the method's goal. A method body can contain nested expressions, including subgoal expressions that need to be resolved by other methods; control expressions such as conditional statements and some forms of iteration; and relational expressions to retrieve the fillers of a relation over a concept. Some method bodies are calls to Lisp functions that are executed without further subgoaling.

For example, Figure 3 shows an EXPECT method to estimate the amount of some linear resource needed by a plan. The `capability` field describes this ability in a machine-readable form, and the `result-type` field shows that the method will return a number. The `method-body` field finds the amount of the resource used by each task in the plan, and adds those amounts (by definition of a linear resource is one whose task requirements can be summed in this way).

```
(name estimate-linear-resource)
(capability (estimate
              (obj (?f is (spec-of amount-used)))
              (of (?r is (inst-of linear-resource)))
              (needed-by (?p is (inst-of plan)))))
(result-type (inst-of number))
(method-body (add (obj (estimate
                         (obj amount-used)
                         (of (checked-resource ?r))
                         (needed-by (plan-tasks ?p))))))
```

FIGURE 3: Representing knowledge in EXPECT.

4 A Problem Solving Method for Plan Evaluation

The general principles of plan evaluation described earlier form the basis of our PSM. In some frameworks, the planning ontology would be considered the input ontology and the evaluation ontology would be called the output ontology. We do not make these distinctions explicitly in EXPECT, since the system can analyze the problem solving methods and infer this information. Furthermore, a given ontology may contain inputs, outputs, or terms internal to the PSM, and we have not found a need to set strict boundaries in our KBs.

4.1 PLANET: An Ontology of Planning Knowledge

Descriptions of plans and their features are captured in PLANET, an ontology of planning knowledge. PLANET is used as a knowledge component within our PSM, but goes beyond the needs specific to plan evaluation. It was designed to be reused as an independent component, for example we are using PLANET as a knowledge component within an ontology-based translation service for planning agents. PLANET complements recent efforts on formalizing, organizing, and unifying AI planning algorithms (Kambhampati *et al.*, 1995; Tate, 1996; Yang, 1990; de Barros *et al.*, 1997) by focusing on the representation of plans, and adds a practical perspective in that it is designed to accomodate a diverse range of real-world plans (including manually created ones). A detailed description of PLANET can be found in (Blythe & Gil, 1999). A few interesting features of PLANET may be worth mentioning. First, planning contexts that refer to domain information and constraints that form the background of a planning problem are represented explicitly. Planning problems, which supplement the context with information about the initial state of the world and the goals, are represented explicitly and are accessible from the context. Alternative plans themselves are then accessible from each planning problem for which they are relevant. Second, PLANET maintains an explicit distinction between *external constraints*, which are imposed on a context or planning problem externally to a planning agent, and *commitments* which are constraints that the planning agent elects to add as a partial specification of a plan (for example, a step ordering commitment). Plans themselves are represented as a set of commitments in addition to the constraints that specify the planning domain and problem.

Figure 4 shows a diagram of the major concepts and relations in the ontology. We describe here the portion that is more relevant to the plan evaluation PSM. **Plan task descriptions** represent the actions or operations that can be taken in the world state. These include templates and instantiations of them, and can be abstract or specific. AI planning systems often refer to these as operators or task decompositions. A plan task description models one or more **actions** in the external world. A **plan task** is a subclass of **plan task description** and represents an instantiation

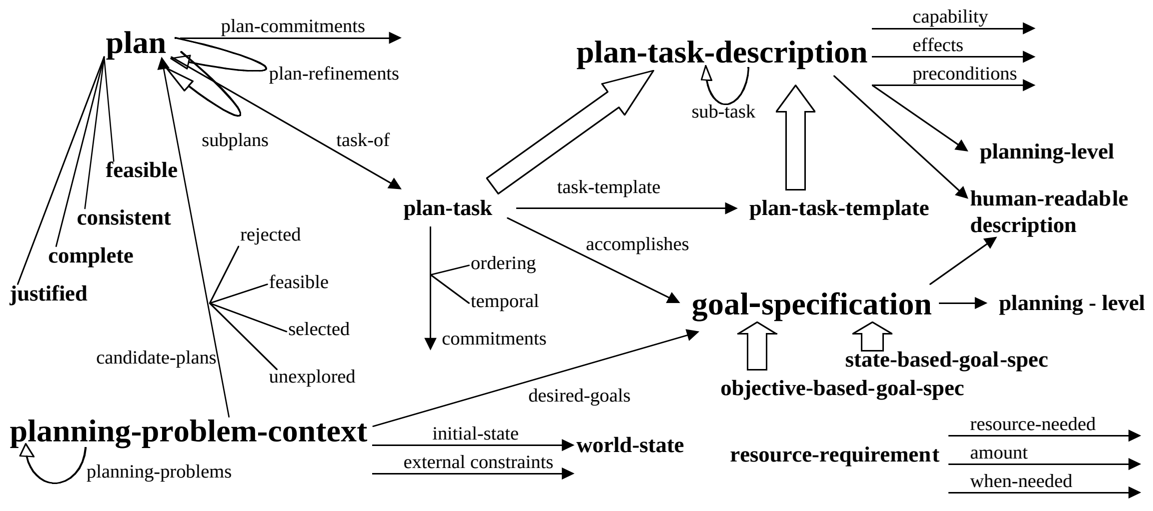

FIGURE 4: An overview of the PLANET ontology.

of a task as it appears in a plan. It can be a partial or full instantiation. A **plan task template** is a subclass of **plan task description** that denotes an action or set of actions that can be performed in the world state. In some AI planners the two classes correspond to operator instances and operator schemas respectively, and in others they are called tasks and task decomposition patterns.

Plan task descriptions have a set of preconditions, a set of effects, a capability, and can be decomposed into a set of subtasks. Not all these properties need to be specified for a given task description, and typically planners represent tasks differently depending on their approach to reasoning about action. The **capability** of a task or task template describes a goal for which the task can be used. A **precondition** is a necessary condition for the task. If the task is executed, its **effects** take place in the given world state. Tasks can be decomposed into **subtasks** that are also task descriptions. Hierarchical task network planners use task decomposition or operator templates (represented here as plan task templates) and instantiate them to generate a plan. Each template includes a statement of the kind of goal it can achieve (represented here as a **capability**)and a decomposition network into **subtasks**, each of which is matched against the task templates all the way down to primitive templates (represented here as **primitive plan task descriptions**. Other planners compose plans as an ordered set of primitive **plan steps** (often called operators, as in STRIPS and UCPOP (Weld, 1994)). Plan steps are specializations of primitive plan task descriptions that must have some set of effects, since they are typically used in means-ends analysis planners.

A **goal specification** represents anything that gets accomplished by a plan, subplan or task. This is used to represent both capabilities and effects of actions and tasks (i.e., they are both subtypes of goal specification), as well as posted goals and objectives. Goals may be variabilized or instantiated. **State-based goal specifications** are a subclass of goal specifications that typically represent goals that refer to some predicate used to describe the state of the world, for example 'achieve (at Jim LAX)', 'deny (at Red-Brigade South-Pass)' or 'maintain (temperature Room5 30)'. **Objective-based goal specifications** are a subclass of goal specifications that are typically stated as verb- or action-based expressions, such as 'transport brigade5 to Ryad'.

A **plan** represents a set of commitments to actions taken by an agent in order to achieve some specified goals. We define the following subclasses of plans: superset A **feasible plan** P is one for which *there exists* some plan that has a consistent superset of the commitments in P and will successfully achieve the goals. A **justified plan** is a plan with a minimal set of commitments. A **consistent plan** is one whose commitments are consistent with each other, with what is known about the state and with the model of action. A **complete plan** is one that includes the tasks necessary to achieve the goals to the required level of detail (this depends on the planning agent's concerns). These definitions are useful to describe properties of plans and to accomodate different approaches that planners use. For example, we can represent that a hierarchical planner generates a feasible plan at each planning level, while generating a complete plan only at the lowest level. A partial-order planner, such as UCPOP, would successively refine feasible plans (called candidate plans) until finding a solution which is a complete plan. There is no requirement for a plan to be justified or consistent in order for it to be represented in PLANET. This is important because we can represent not only machine-generated plans but also human-generated plans, which are likely to contain errors. Notice that plans can be represented as instances or classes, the ontology does not commit to either and leaves the option open.

In searching or designing a plan, a number of choices typically need to be made to find or create a solution. At a given choice point, several alternatives may be considered, and one (or more) chosen as selected. Such choices are represented in PLANET as a type of commitment. **Commitments** can be made in both plans and tasks. **Plan commitments** are a subclass of commitments on the plan as a whole. Commitments may be in the form of actions at variously detailed levels of specification, orderings among actions, a decision that a certain action will be used to establish a particular condition and other requirements on a plan such as a cost profile. The tasks that will form part of the plan are represented as a subset of the commitments made by the plan. **Task commitments** are a subclass of commitments that affect individual tasks or pairs of tasks. An **ordering commitment** is a relation between tasks such as (before A B). A **temporal commitment** is a commitment on a task with respect to time, such as (before ?task ?time-stamp). Another kind of commitment is the selection of a plan task description because it **accomplishes** a goal specification. This relation records the intent of the planning agent for the task, and is used in PLANET to represent causal links.

We have used PLANET to represent plans in three different domains: Air Campaign Planning (ACP), Army courses of action (COA), and enemy workarounds to target damage (WA) (Blythe & Gil, 1999). Although all three are military domains, the plans are of a radically different nature in each case. In two domains, the plans were built manually by users and needed to be represented as given, i.e., containing potential flaws and often serious errors. In the ACP domain, plans are hierarchically decomposed and have verb-based objectives. Information about causal links and task decomposition templates is not provided. The COA domain has plans with a hierarchical flavor that is not always explicitly represented in the plan. In the WA domain, plans were generated automatically by an AI planner and are not hierarchical, consisting of a set of partially ordered steps and causal information in terms of the enabling conditions and achieved effects of each step.

4.2 An Ontology of Evaluations

This ontology is used to represent the results of the evaluations and captures a wide range of evaluation results, including but not limited to the evaluations that can be generated using our PSM. Generally speaking, plan evaluations yield a wide range of conceptually different results. Sometimes, an evaluation is made to derive a property of the plan that is not explicitly stated initially, such as how many units of a certain resource are used in the plan. These evaluations range from simple inferences to very detailed analysis. Other times, an evaluation is done through a simulation of the execution of the plan. Another aspect of what this kind of ontology needs to capture is that there may be more than one value for a given evaluation criterion, since the evaluation system may have more than one algorithm at its disposal for providing an estimate for

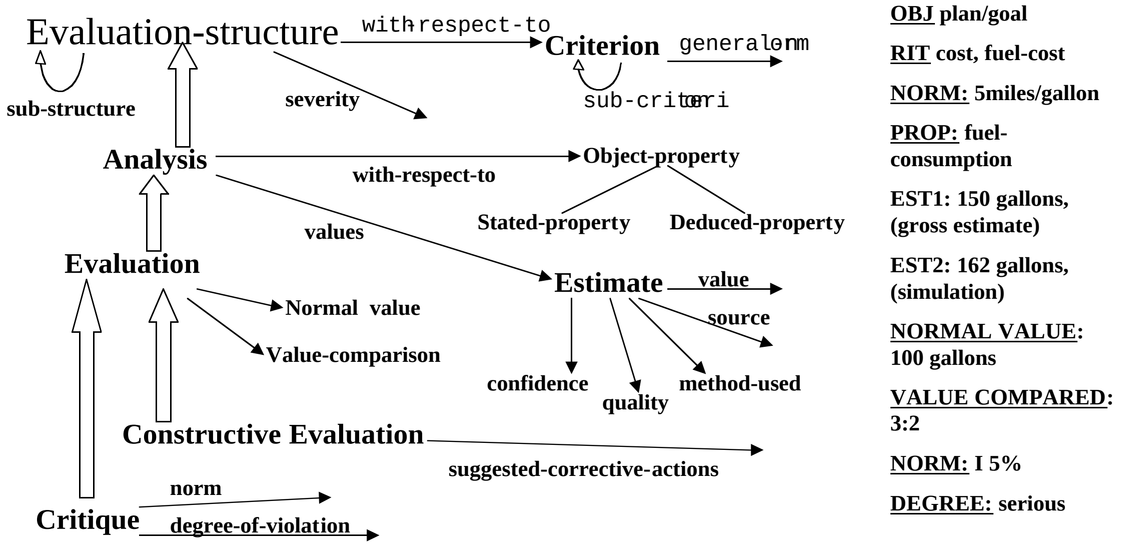

FIGURE 5: An overview of the ontology of evaluations.

that value. For example, a system may be able to provide a quick initial estimate of fuel usage based on back-of-the-envelope calculations, as well as a finer estimate based on a detailed slower simulation of the plan execution.

Figure 5 shows an overview of the main entities in this ontology, together with an example on the right-hand side. In this ontology, an **evaluation structure** indicates the **criterion** that the evaluation is addressing (e.g., fuel cost), the **general norm** that the plan is expected to follow (e.g., 5 miles per gallon), and the **severity** or relative importance of this criterion. Notice that the criteria can be grouped, so the fuel consumption evaluation can be a **sub-structure** of the cost. An **analysis** is done of several properties of the plan, for example the actual fuel consumption. The overall result may be derived from several of these analyses, for example fuel cost can be derived from an estimate of fuel consumption when vehicles have no load combined with an estimate of the expected load to be moved. In some domains, this analysis may be sufficient as an evaluation of the plan (Gil & Swartout, 1994). Often times, the estimated value is compared to the norm, and the result is what we call here an **evaluation** properly. For instance, we might expect some plan to use 100 gallons of fuel and find that it actually uses 150 gallons, giving a comparison ratio of 3:2. A **constructive evaluation** would go one step further and include **suggested corrective actions** that can be taken to modify the plan so that the estimate comes out closer to what may be desirable. Sometimes in evaluating a plan we want to be alerted if some evaluation has exceeded certain limits from its normal value, and this is expressed in this ontology by **critique**, which is a kind of evaluation. This entity has a norm, that typically indicates a range around the normal value that is acceptable, and a **degree of violation**, that records by how much, if at all, the critique is violated. For example, we may state that fuel usage may deviate from the normal amount by 15levels of deviation represent a violation. Critiques may or may not be constructive evaluations.

4.3 An Ontology of Checks on Plan Structure

The different kinds of checks on plan structure exist as entities in this ontology, with a class hierarchy that follows the generic kinds of evaluation, as shown in Figure 6. Within the PSM there are generic sub-methods that are associated to each class. The sub-methods described earlier perform the evaluation of a plan based on any critique that is an instance of the class.

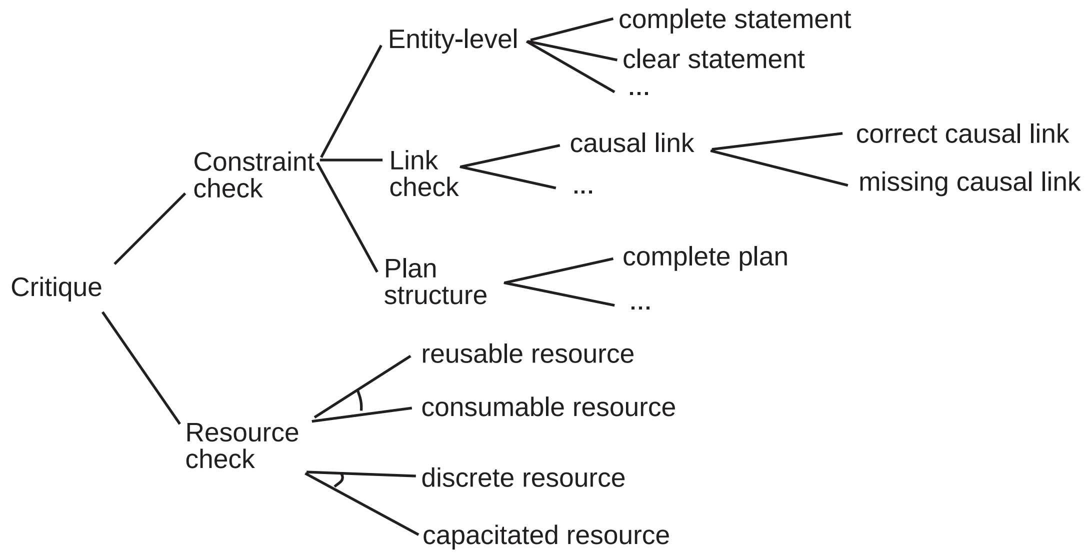

FIGURE 6: The class hierarchy of plan checks is based on the generic kinds of evaluation from the previous section. Reusable sub-methods in the PSM are attached to each critique which can be specialized for a new domain by creating instances and/or subclasses of this ontology. For reasons of space, only a subset of the classes is shown here.

4.4 An Ontology of Resources

This ontology builds on the OZONE ontology (Smith & Becker, 1997), an ontology of resources that was developed for a scheduling tool. We identified several distinct patterns of resource usage that are used to build general resource checking mechanisms. A *capacitated resource* can exist in different quantities, and a numeric check of the amount required against the amount available is appropriate. For example to move a ship over some distance one must have a certain amount of fuel. A *discrete resource* is simply present or absent and no numeric check is made. For example a key is needed to open a door, but ten keys are usually no better than one.

The standard notions of consumable and reusable resources have been generalized to improve their applicability. Typically a *consumable resource* is no longer available after it has been used by some task (for example fuel) and a *reusable resource* becomes available again after the task using it has completed (for example a spanner). We generalize these two types of resource by considering which pairs of tasks could both use the same resource. In the case of fuel there are no such pairs, while the reusability of the spanner is represented by all pairs of tasks that do not overlap in time. This more precise representation allows us to capture other important cases with precise semantics. For example some resources, like information, are *universally reusable* in the sense that any pair of tasks can use the resource once it is acquired. Other resources, like ship berths, are partially reusable in the sense that for two tasks to use the same resource, they must not only have no temporal overlap but also must be spatially co-located.

4.5 CHECKERS: A Suite of Fine-Grained Plan Evaluation Sub-Methods

Problem-solving knowledge is represented explicitly as described in Section 3. Currently, the plan evaluation PSM contains 35 sub-methods. We include here a few examples that illustrate the range and variety of the sub-methods in this PSM.

Some sub-methods do specific structural checks. For example, this is a method to check that there are parent links for all subtasks within the plan:

```
(capability (evaluate (obj (?t is (inst-of plan-task)))
                      (with-respect-to (missing-parent-link))))
(result-type (inst-of boolean))
(method-body (determine-whether-there-are (obj (plan-task-parents ?t))))
```

EXPECT achieves the sub-goal in this method body by using one of its general-purpose primitive methods that takes a set of objects and determines whether it is empty. Notice that this method applies directly to plans in any domains, i.e., there is no need for the user to add domain-specific knowledge in order for this evaluation to be done with this sub-method.

Other sub-methods do checks on resources. For example, this is a method that evaluates the use of resources that are both consumable and capacitated by checking whether the amount used in the plan is greater than the amount available:

```
(capability (evaluate (obj (?p is (inst-of plan)))
                      (with-respect-to (?r is (inst-of (and resource consumable-resource
                                                            capacitated-resource))))))
(result-type (inst-of boolean))
(method-body (is-greater-or-equal
               (obj (estimate (obj amount-used) (of ?r) (by ?p)))
               (than (estimate (obj amount-available) (of ?r) (for ?p)))))
```

Notice that this method applies directly to resources in any domains without being modified. The only information that a user needs to specify in order for the method to work is to place the resources in the domain within the classes in the resource ontology (i.e., whether it is consumable, capacitated, etc.) The user needs to also add two pieces of domain-specific problem-solving knowledge: how to estimate the amount used by a task and how to find out how much is available for that task. We will show in the next section how this is done.

Finally, there is a group of sub-methods that captures the top-level structure of the PSM by organizing the subgoals to evaluate the plan with respect to specific criteria. For example, the following sub-method states that in order to evaluate a plan, one evaluates a plan with respect to each of the evaluation criteria as defined for the particular application domain:

```
(capability (evaluate (obj (?p is (inst-of plan)))
                      (with-respect-to (?a is (inst-of evaluation-structure)))))
(result-type (inst-of evaluation-structure))
(method-body (combine (obj (evaluate (obj ?p)
                                     (with-respect-to (sub-structure ?a))))
                      (with (evaluate (obj ?p)
                                      (with-respect-to (factor ?a))))
```

5 Developing Domain-Specific Plan Evaluation Tools with the PSM

This section illustrates how the PSM can be specialized in a new application domain. The user typically starts by defining a new evaluation in a domain by defining the criterion of the evaluation as a subclass of some existing criterion class. When this new class is defined, it allows more specific sub-methods to be created for part or all of the evaluation. Through these the user can complete or specialize parts of the process of performing the evaluation. In some cases, the generic sub-methods are sufficient and no new sub-methods need to be added to complete the evaluation.

As an example, we create a new critique in a transportation planning domain. Plans in this domain must move a given payload, distributed across a number of cities, to some destination by air or sea. The critique we add measures “sealift”, which is the capacity by weight available for transportation by sea — in other words the sum of the load capacity of all available transport ships. A transport plan will contain tasks moving packages by sea, where a package is a unit of payload at a seaport. For each package we will ensure that there is enough sealift capacity at that seaport.

We create the class **sealift-capacity** as a sub-class of **capacitated-resource** and **sealift-available** as a class of critique that checks this resource. This makes problem-solving knowledge available in the PSM to compare for any task an amount of **sealift-capacity** required with an amount available. Sub-methods are also available to report violations of the new critique by gathering the set of tasks for which demand exceeds supply. We add two new sub-methods, as shown in Figure 7, for computing how much **sealift-capacity** a task needs and how much is available. The amount needed is simply the weight of the package to be moved, and the amount available is the sum of the load capacities of the ships at the seaport where the package is located. By using the generic sub-methods of the PSM, we can create an application that critiques a plan and warns a planner of any violations with only these two new sub-methods. Six generic sub-methods from the PSM are used to create the application, by finding relevant critiques, checking individual task or global properties as appropriate, collecting problems raised by the critiques and communicating them to the user.

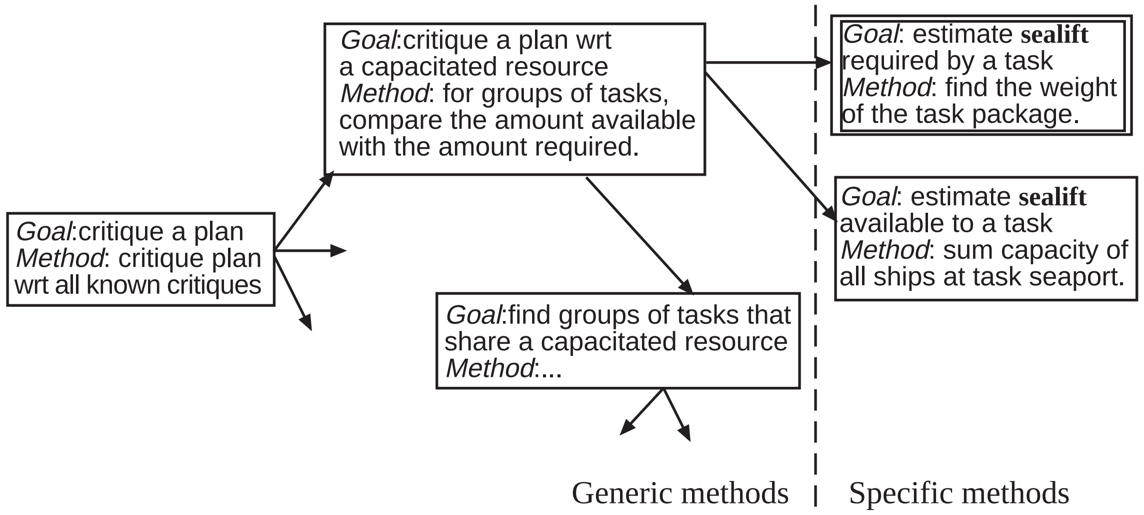
Goal:critique a plan
Method: critique plan wrt all known critiques
Goal:critique a plan wrt a capacitated resource
Method: for groups of tasks, compare the amount available with the amount required.
Goal:find groups of tasks that share a capacitated resource
Method:...
Goal: estimate **sealift** required by a task
Method: find the weight of the task package.
Goal: estimate **sealift** available to a task
Method: sum capacity of all ships at task seaport.
Generic methods
Specific methods

FIGURE 7: Generic and specific methods used to implement the critiqe **sealift-available** in the transportation planning domain.

As it stands, the critique checks the **sealift-capacity** for individual tasks. We would like to extend it to check the combined demand from tasks that leave the same seaport at the same time, to make sure there is enough load capacity for all of them. This can be achieved by specializing the default behaviour to determine how reusable a resource is, which assumes that the resource is completely reusable, i.e. that no pair of tasks impedes each other's use of the resource. We specialize this sub-method, as shown in Figure 8, to say that two tasks must share the same resource if they leave the same seaport on the same day. Now our critiquer both checks the **sealift-capacity** used per tasks and also checks the daily useage at each seaport. It does this by finding groups of tasks that must share the same capacitated resource and summing their demand, using two generic sub-methods that are not shown.

Goal:determine if two tasks must share the same **sealift-capacity**
Method:Do the tasks leave on the same day from the same seaport?

FIGURE 8: A method to determine whether two tasks must share the same pool of **sealift-capacity**

In this section we saw how the PSM provides an infrastructure for performing an evaluation based on a subclass of a partially reusable capacitated resource. Generic sub-methods also exist for the other evaluation criteria in the library. EXPECT's KA capabilities can be brought to bear on the process by prompting the user for missing knowledge, both problem-solving knowledge and domain knowledge. For instance if the problem solver is run after defining **sealift-capacity** and **sealift-available** but nothing else, EXPECT's analyzer will detect the subgoals that it is unable to solve and ask the user to supply them, allowing the appropriate methods to be created by analogy from already existing ones. Also, once the method in Figure 8 is added, EXPECT will check all the tasks it knows about to see if it knows the time and seaport they leave from, and bring missing information to the user's attention.

6 Experiences with the evaluation PSM

The section shows the results of various analyses that we performed on three different plan evaluation task domains. In the COA domain, the plan is an Army Course of Action at the Division level and is manually developed. The evaluation criteria range from simple structural checks (e.g., *a task has no stated purpose*) to complex ones that require sophisticated tactical and spatial reasoning (for example, *the COA does not focus combat power at the decisive point of battle.*) Our starting point was a reference document (Jones, 1998) that published the results from extensive interviews with domain experts describing a total of 62 evaluation criteria.[1] In the air-campaign planning (ACP) domain, users start with high-level objectives and decompose them into subobjectives using an editing tool. Evaluation criteria were gathered in knowledge elicitation sessions with air force personnel. The third domain critiques military transportation plans for crisis situations.

6.1 Coverage

To test the coverage of the evaluation criteria contained in the PSM, we analyzed the evaluation and critiquing criteria that had been suggested by domain experts in each domain to determine which ones could be implemented as specializations of the evaluation criteria in the PSM. The results are shown in Table 1.

The results are very encouraging: around 75% of the criteria in these domains were special cases of the general criteria that were defined for the PSM. If we restrict attention to criteria related to statement, link and plan structure and resources, coverage is 95%. In the COA domain, 44 out of the 62 criteria could be well described as specializations of the domain-independent evaluation criteria in the PSM described earlier. Of the remaining 18, 11 are concerned with adversarial planning

[1]Some of the original criteria were split to ensure they are all of the same granularity.

	Type of evaluation	COA	ACP	TRANSP
Statement Structure	complete statement	20	3	0
	clear statement	2	0	0
	correct statement	1	2	0
	coherent statement	6	4	0
Interdependency links	correct causal link	4	0	0
	missing causal link	0	0	0
	correct parent/child link	0	2	0
	missing parent/child link	0	1	0
	correct sequence link	0	1	0
	missing sequence link	0	0	0
Plan Structure	justified plan	1	0	0
	step reuse	0	1	0
	complete plan	1	2	0
	clear plan	2	0	0
	correct plan	4	0	0
	consistent plan	0	0	0
Use of resources	task-level resources	0	6	0
	resource links	0	0	0
	plan-level resources	3	0	12
	Criteria covered	44	22	12
	Criteria not covered	18	0	1
	Total criteria	62	22	13

TABLE 1: Domain-independent evaluations in the PSM and the number of evaluations that instantiate them in three different task domains.

(analyzing risk from the enemy or the use of deception in the COA) and 5 are concerned with spatial reasoning. Both of these topics are currently beyond the scope of the PSM. The remaining 2 were criteria that we think perform some useful abstraction over the plan that users find useful to look at in this particular domain, and do not seem to generalize or correspond to any general principles that could be captured in the method. All the critiques in the ACP domain were covered. In the Transportation domain, there was one evaluation that was not covered by the method and that seemed to also express an abstraction that users find useful in this domain but that does not correspond to any general principles that could be included in our PSM.

6.2 Systematic Design of Critiques

The framework of evaluation criteria defined in the PSM provides a structure that can be used to design a plan evaluation tool in a new domain. The structure can be used as the basis to interview experts by going systematically through each evaluation criterion of the method and asking the domain experts whether it applies to the domain. If a criterion applies, then the general-purpose methods and ontologies are useful to extract the domain-specific knowledge that complements the general-purpose principles captured by the method.

The evaluation PSM can also be used to evaluate the completeness of the criteria used in a plan evaluation tool that is under development, as is the case in our three domains. This allows users to calibrate the results of plan evaluation tools, and increase their confidence on plans that have have been checked for tools that are pretty thorough and complete.

Table 1 shows that none of the three cases include all the criteria that our PSM suggests. In the COA domain, for example, a large number of criteria (20) are concerned with checking the completeness of individual statements of the COA. This concentration of evaluation types suggests that more evaluations of other types might be useful. For example, checking that the operations in the plan are adequately sequenced, or checking that there are appropriate resources used by each individual operation. In the ACP domain, an interesting thing that we realized is that the evaluation criteria were gathered in two separate knowledge elicitation sessions. One was with high-level planning experts that design the overall air campaign and one was with logistics experts that make sure that they can provide the resources to support the operations. The criteria gathered from high-level planning experts were uniformly spread among the categories in our framework while the criteria gathered from logistics experts were heavily concentrated in criteria based on individual objective statements. This is not surprising, since the logistician's job is to guarantee the feasibility of the air campaign plan for execution rather than to consider the plan's overall structure or its ability to achieve the desired goals. It is interesting to point out that there was no need in this domain for the critiquer to check for clear statements, because it turns out that when we designed this application we extended the plan editor so that it would prevent users to specify unclear statements in the first place.

6.3 Supporting the Acquisition of Domain-Specific Knowledge

With coverage of new criteria around 75% in complex domains, the PSM should reduce the amount of effort to develop a new plan evaluation or critiquing tool by reuse and adaptation of its ontologies and methods. As we described in the previous section, a knowledge acquisition tool can use the PSM to guide users to add the domain-specific knowledge needed. We are presently using the PSM to design a critiquing tool that can be extended in this way. An interesting evaluation to conduct in this regard is to estimate how much effort is saved when the PSM is used to support users in creating the plan evaluation tool.

6.4 Reuse

We are using this PSM to develop a plan evaluation tool in a new application domain to critique Army Courses of Action (COAs). Our preliminary results show that each critique added can be modelled by the PSM, and of the 52 chunks of domain-specific problem-solving knowledge used, half (26) are new to the domain and half (26) are generic. Of the reused generic chunks, 11 come from the PSM and 15 are utility knowledge, such as arithmetic.

7 Discussion

Comprehensive ontology: A distinguishing feature of EXPECT that is not present in other frameworks is that we represent terms and objects that capture useful notions related to problem solving but are not data being consumed or produced by the method. For example, a problem-solving task such as "estimate the duration of a plan" would be represented in most other approaches as a predicate, perhaps "(estimate-duration ?plan)". In EXPECT this would be modeled by defining "estimate", "duration", and "plan" individually, and then the goal would be stated as "(estimate (obj (spec-of duration)) (of ?p is (instance-of plan)))". The shorter predicate representation confounds the type of task (estimate), a qualification of it (it is a duration estimation) and how this task description relates to the actual data being passed to the method (the plan). This knowledge is explicitly captured in EXPECT. All of these individual terms end up represented in the knowledge bases and ontologies, and are referenced in the problem-solving methods. As a result, the ontologies that EXPECT uses associated with a PSM tend to capture explicitly more information about the problem-solving activity. For more details, see (Swartout & Gil, 1995).

Explicit constraints: The ontological definitions of the terms used in methods are used by EXPECT during problem solving. For example, if an ordered-set-of-numbers is a term defined in the Loom ontology as a set composed of numbers and its definition indicates which sorting function to use, then EXPECT would only use a method to add an ordered-set-of-numbers for sets that have the properties indicated in the definition. In effect, Loom is providing a language to state constraints between different inputs and outputs to the individual methods.

PSM code compilation: EXPECT generates automatically the code for the overall, larger PSM for plan evaluation when given on a generic top-level task, in this case "(evaluate (instance-of plan))" (see (Swartout & Gil, 1995)). This is done through a kind of partial evaluation that includes goal reformulation and incorporates relevant ontological knowledge. This creates a variabilized problem-solving tree that can be analyzed by EXPECT's KA tools to determine what information is needed for the PSM (Gil & Melz, 1996). In addition, executable code for the PSM (e.g., Lisp code) can be generated by walking the problem-solving tree. The resulting piece of code can be executed independently from the EXPECT framework, and we often use this capability to support efficient problem execution.

Knowledge representation continuum: When the PSM is extended with domain-specific knowledge, domain-specific ontologies and problem-solving knowledge are added, using the same kind of approach followed for domain-independent knowledge. In EXPECT, both factual and problem-solving knowledge can be domain dependent or domain independent. There is a continuum between the representation of domain-dependent and domain-independent factual knowledge in EXPECT. They are represented in the same language, yet they can be defined and maintained separately. Domain-specific definitions can be subclasses of domain-independent definitions, inheriting general constraints and properties. For more details, see (Gil & Melz, 1996).

Knowledge roles: The method shown above that describes the use of consumable capacitated resources brings up an interesting issue with respect to the roles that domain-specific knowledge plays in a PSM. Typically, knowledge roles in a PSM are mapped to domain objects, such as a constraint or a fix. Here, the domain-specific knowledge to be added is a sub-method. In EXPECT, *knowledge roles can be filled with domain-specific problem-solving knowledge*, which we believe is a unique feature of our approach.

8 Related Work

There is a component for assessment and evaluation as part of the CommonKADS library (Breuker & de Velde, 1994). The assessment and evaluation strategies are not implemented, but instead are intended to provide a methodology to design systems to assess and evaluate any kind of object. Our PSM is tailored to the evaluation of plans, and it has been implemented. Our PSM draws from the CommonKADS work in terms of the output ontology, making important distinctions between creating abstractions, evaluating with respect to a normal value, and critiquing with respect to a norm or threshold. Related work on problem-solving methods for plan generation (Valente *et al.*, 1998; de Barros *et al.*, 1997)analyzes AI planning algorithms (or planning methods) and identifies the typical knowledge roles that characterize the domain knowledge used by these planning methods. This study makes an interesting distinction between static and dynamic roles. Static roles are filled by knowledge that is constant for the given domain, for example the planning task templates are considered static plan composition knowledge. Plans are dynamic knowledge roles that consist of plan steps, ordering constraints, auxiliary constraints (which group temporal constraints and causal links), and variable binding constraints. It is easy to see the connection with the various roles of the class plan in PLANET. Goals are also considered dynamic roles, and can be either conditions or actions to be accomplished. Thus, they map directly to PLANET's goal specifications. The main knowledge roles in this study map directly to classes in PLANET. PLANET adds many more relations between the roles and contains many more classes and axioms. PLANET was also designed from the perspective of planning environments where plans are manually created (instead of representing only plans of AI planning systems), and as a result can also represent the errors and flaws that these plans often contain. It would be useful to add the static vs dynamic distinctions from this study to PLANET.

There has been relatively little work on machine-aided evaluation of plans compared with the amount of work on generation. Pérez and Carbonell designed a plan generation system to produce plans of high quality (Pérez & Carbonell, 1994). They divided quality measures into three groups: plan execution cost, plan robustness or its ability to respond well to changing or uncertain conditions, and other factors. Of these, execution cost was modelled in the greatest detail. This was described as the sum of the execution costs of individual operators, and thus could be modelled in our PSM as a linear consumable capacitated resource.

In PLANET we have drawn from previous work on languages to represent plans and planning knowledge (Ghallab *et al.*, 1998; Wilkins & Myers, 1995; Kambhampati *et al.*, 1995; Tate, 1996; Yang, 1990). These languages are often constrained by the reasoning capabilities that are provided by AI planning systems. PLANET is an ontology, and as such does not make specific commitments about the language in which various items are expressed. The planning knowledge represented in these languages can be mapped into PLANET. PLANET also accomodates plans that are not created by AI planning systems, and provides a representation for the context of the planning problems that are given to these systems.

9 Conclusions

We are developing a reusable, general-purpose PSM for plan evaluation and critiquing. The Plan Evaluation PSM captures domain-independent knowledge of general principles to evaluate plans from two perspectives: plan structure and resource use. In a study of plan evaluation criteria used in three real-world domains, we estimate that the evaluation criteria included in the PSM cover roughly 75 % of the criteria, and around 95 % of those criteria related to plan structure and resources. The benefits of using a reusable PSM are in knowledge acquisition and knowledge reuse. The PSM can guide interviews with domain experts by systematically following its evaluation criteria. It can also be useful to assess the completeness of a plan evaluation tool and raise the confidence of a user in the plan being evaluated. Since domain-dependent knowledge for plan evaluation changes, plan evaluation tools need to be customizable and adaptable by users. Tools are needed that can acquire this type of knowledge, a central reason for the use of plan evaluation tasks in the last few years for knowledge acquisition work within the EXPECT framework (Swartout & Gil, 1996). EXPECT's KA tools use the PSM to guide the user through the process of creating a new critique or modifying an existing one because of the explicit representation of critique classes.

We would like to extend our PSM to include additional perspectives to evaluate a plan. We plan to analyze the adversarial planning and spatial reasoning critiques that we found in the COA domain and extend our PSM to cover these areas. Other important criteria to evaluate plans include balance, parsimony, robustness, adequacy for execution, and plan understandability. In our experience, they are either hard to pin down and formalize or their evaluation is very specific to the domain. In either case we have found it hard to extract principles that we could incorporate in a general-purpose and reusable PSM. We continue to study and explore plan evaluation issues so we can make our evaluation PSM and associated knowledge acquisition capabilities more comprehensive and useful.

Acknowledgments

We thank all the members of the Expect project, including Andre Valente, Jihie Kim, Marcelo Tallis, Bill Swartout and Surya Ramachandran for many useful discussions about this work. We gratefully acknowledge the support of DARPA with contract DABT63-95-C-0059 as part of the DARPA/Rome Laboratory Planning Initiative, and with grant F30602-97-1-0195 as part of the DARPA High Performance Knowledge Bases Program.

Blythe, J. & Gil, Y. (1999). Planet: A shareable and reusable ontology for representing plans. Technical report, Expect internal report.

Breuker, J. & de Velde, W. V., (Eds.) (1994). *CommonKADS Library for Expertise Modelling.* IOS Press.

de Barros, L. N., Hendler, J., & Benjamins, V. (1997). AI planning versus manufacturing-operation planning: A case study. In *Proc. 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan. Morgan Kaufmann.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). Pddl — the planning domain definition language. Technical report. Available at http://www.cs.yale.edu/pub/mcdermott/software/pddl.tar.gz.

Gil, Y. & Melz, E. (1996). Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proc. Thirteenth National Conference on Artificial Intelligence.* AAAI Press.

Gil, Y. & Swartout, W. R. (1994). Expect: A reflective architecture for knowledge acquisition. In *Proceedings of the 1994 Workshop of the ARPA-Rome Laboratory Knowledge-Based Planning and Scheduling Initiative*, Tucson, AZ.

Jones, E. (1998). Hpkb course of action challenge problem specification. Technical report, Alphatech, Inc., Burlington, MA.

Kambhampati, S., Knoblock, C., & Yang, Q. (1995). Planing as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence*, 76:167–238.

MacGregor, R. M. (1991). *The Evolving Technology of Classification-Based Knowledge Representation Systems.* San Mateo, CA, Morgan Kaufmann.

McDermott, J. (1988). *Preliminary Steps Towards a Taxonomy of Problem-Solving Methods.* Kluwer Academic Publishers.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. (1991). Enabling technology for knowledge sharing. *AI Magazine*, pages 36–56.

Pérez, M. A. & Carbonell, J. (1994). Control knowledge to improve plan quality. In Hammond, K., (Ed.), *Proc. Second International Conference on Artificial Intelligence Planning Systems*, pages 323–328, University of Chicago, Illinois. AAAI Press.

Sacerdoti, E. (1977). *A Structure for Plans and Behavior.* New York, Elsevier.

Smith, S. F. & Becker, M. (1997). An ontology for constructing scheduling systems. In *AAAI Spring*

Symposium on Ontological Engineering, Stanford University.

Smith, S. F. & Lassila, O. (1994). Toward the development of mixed-initiative scheduling systems. In *Proceedings ARPA-Rome Laboratory Planning Initiative Workshop*, Tucson, AZ.

Smith, S. F., Lassila, O., & Becker, M. (1996). Configurable, mixed-initiative systems for planning and scheduling. In Tate, A., (Ed.), *Advanced Planning Technology*, Edinburgh, UK.

Swartout, W. R. & Gil, Y. (1995). Expect: Explicit representations for flexible acquisition. In *Proc. Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta.

Swartout, W. R. & Gil, Y. (1996). Expect: A user-centered environment for the development and adaptation of knowledge-based planning aids. In Tate, A., (Ed.), *Advanced Planning Technology*, Edinburgh, UK.

Swartout, W. R., Paris, C. L., & Moore, J. D. (1991). Design for explainable expert systems. *IEEE Expert*, 6(3):58–64.

Tate, A. (1977). Generating project networks. In *International Joint Conference on Artificial Intelligence*.

Tate, A. (1996). Representing plans as a set of constraints – the ¡i-n-ova¿ model. In Drabble, B., (Ed.), *Proc. Third International Conference on Artificial Intelligence Planning Systems*, University of Edinburgh. AAAI Press. Available as Pointer.

Valente, A., Benjamins, R., & de Barros, L. N. (1998). A library of system-derived problem-solving methods for planning. *International Journal of Human-Computer Studies*, 48:417–447.

Weld, D. (1994). A gentle introduction to least-commitment planning. *AI Magazine*.

Wielinga, B. J., Schreiber, A. T., & Breuker, A. (1992). Kads: A modelling approach to knowledge acquisition. *Knowledge Acquisition*, 4(1):5–54.

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

Wilkins, D. E. & Myers, K. L. (1995). A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation*, 5(6):731–761.

Yang, Q. (1990). Formalizing planning knowledge for hierarchical planning. *Computational Intelligence*, 6(1):12–24.