PASSAT: A User-centric Planning Framework

Karen L. Myers[1] W. Mabry Tyson[1] Michael J. Wolverton[1]
Peter A. Jarvis[1] Thomas J. Lee[1] Marie desJardins[2]

[1] Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
{myers,tyson,mjw,jarvis,tomlee}@ai.sri.com

[2] University of Maryland, Baltimore County
Dept. of CS and EE
1000 Hilltop Circle
Baltimore, MD 21250
mariedj@cs.umbc.edu

Abstract

We describe a plan-authoring system called PASSAT *(Plan-Authoring System based on Sketches, Advice, and Templates)* that combines interactive tools for constructing plans with a suite of automated and mixed-initiative capabilities designed to complement human planning skills. PASSAT is organized around a library of predefined *templates* that encode task networks describing standard operating procedures and previous cases. Users can select from these templates to apply during plan development, with the system providing various forms of automated assistance. A mixed-initiative *plan sketch* facility helps users refine outlines for plans to complete solutions, by detecting problems and proposing possible fixes. An *advice* capability enables user specification of high-level guidelines for plans that the system helps to enforce. Finally, PASSAT includes *process facilitation* mechanisms designed to help a user track and manage outstanding planning tasks and information requirements, as a means of improving the efficiency and effectiveness of the planning process. PASSAT is designed for applications for which a core of planning knowledge can be captured in predefined action models but where significant user control of the planning process is required.

Introduction

AI planning technology provides powerful tools for solving problems that require the coordination of actions in the pursuit of specified goals. To date, however, there has been limited success in transitioning this technology to significant applications in the commercial, military, or space sectors. A major obstacle to technology transfer lies with the lack of control available to potential users of planning systems. AI planning systems have traditionally been designed to operate as *black boxes*: they take a description of a domain and a set of goals and automatically synthesize a plan for achieving the goals. Human planners, however, are generally reluctant to cede full control to automated planning systems in this manner.

Many potential consumers of planning technology require more *user-centric* tools that are designed to augment human skills rather than replace them. This observation has led, in recent years, to the development of a number of *plan-authoring* frameworks. Plan-authoring systems provide a set of plan editing and manipulation capabilities that support users in developing plans. These systems introduce a degree of structure to the planning process, yielding principled representations of plans with well-defined semantics. Plan-authoring systems can include a range of planning aids that reason over this structure; however, the role of such automated aids is to augment human planning skills by facilitating human-driven plan development. Interest in plan-authoring systems is strong within both the space and military sectors, for their potential to improve the quality and process of plan development without incurring the high knowledge modeling costs and loss of control associated with fully automated planning systems.

This paper describes a plan-authoring system called *PASSAT (Plan-Authoring System based on Sketches, Advice, and Templates)* designed to support user-centric planning. At its heart, PASSAT is a plan-authoring system in which users construct and modify plans interactively. Users can draw upon a library of *templates*, to the extent they desire, to assist with plan development. Templates correspond to a form of hierarchical task network (HTN) [Tate, 1977], and may encode both parameterized standard operating procedures and cases corresponding to actual or notional plans developed for related tasks.

To complement these interactive tools, PASSAT includes a range of automated and mixed-initiative planning capabilities. Users can invoke an automated planning mode based on standard HTN methods to expand any open task within a plan. A *plan sketch* facility enables users to create outlines of plans that are then filled out using templates designed for similar tasks. *Advice* within PASSAT enables users to define high-level policies to be satisfied by both plans and planning processes. Such guidance can be useful both in directing automated components within the system, and in tracking high-level guidelines that a user wants satisfied but may inadvertently violate through his interactive planning choices.

PASSAT also includes *process facilitation* mechanisms designed to aid the user in managing plan development. These mechanisms help the user track open tasks and outstanding information requirements for the current plan. Such assistance is critical in complex applications, as it helps the user stay focused without overlooking important details.

With its combination of interactive and automated capabilities, PASSAT enables a user to quickly develop plans that draw upon past experience encoded in templates but are customized to his individual preferences and the demands of the current situation. PASSAT has been under development for about a year. This paper describes both the current PASSAT system and the more comprehensive plan-authoring system toward which we are working (with all future work noted explicitly as such). We begin with a more detailed discussion of PASSAT and an example of its use, followed by a description of the representational constructs within PASSAT, the user-centric planning capabilities, and the process facilitation mechanisms. The final section discusses related work.

PASSAT Overview

Plan development within PASSAT has been guided by two key principles:

- *Flexible, 'out of the box' planning:* Traditional AI planning systems lock users into a set of solutions, namely, those implied by the predefined action models that underlie plan development. Within PASSAT, templates are viewed as guidelines for performing tasks; the human planner is free to expand the set of solutions defined by the templates. In particular, a user can override constraints, drop tasks, or insert additional tasks in accord with his personal preferences or the demands of the current situation. Such flexibility is critical for domains in which correct and comprehensive collections of templates cannot be provided.
- *Controllable user-centric automation:* Automated capabilities within PASSAT are designed both to complement human planning skills and to be readily directable by a human. Automation would be invoked under user control only in contexts where he feels that it would be beneficial.

Domain Characteristics

PASSAT is generic, domain-independent technology but is tailored toward applications with the following characteristics.

(a) The complexity of the domain precludes full capture of all relevant planning knowledge. However, partial planning models can be developed.

(b) Human input is critical, but some amount of automation would both improve plan quality and reduce overall planning time.

Our motivating application domain, Special Operations Forces (SOF) mission planning, has these characteristics. Standard operating procedures exist for many high- and mid-level activities in the SOF domain, and are readily amenable to encoding within an HTN representation. For example, a hostage rescue operation can be characterized as consisting of the high-level objectives of performing reconnaissance in the areas around the rescue site, establishing a safe haven to which to remove the hostages, undertaking the assault to rescue the hostages, and transporting the hostages to the safe haven. Low-level operations follow standard doctrine and can also be modeled in a relatively straightforward manner.[1] Intermediate strategy decisions pose a bigger challenge. For example, informed selection of areas and methods for reconnaissance requires deep background knowledge of reconnaissance operations, breadth of understanding of the current situation, and significant experience. Capturing and modeling this type of strategic knowledge in full presents a tremendous challenge.

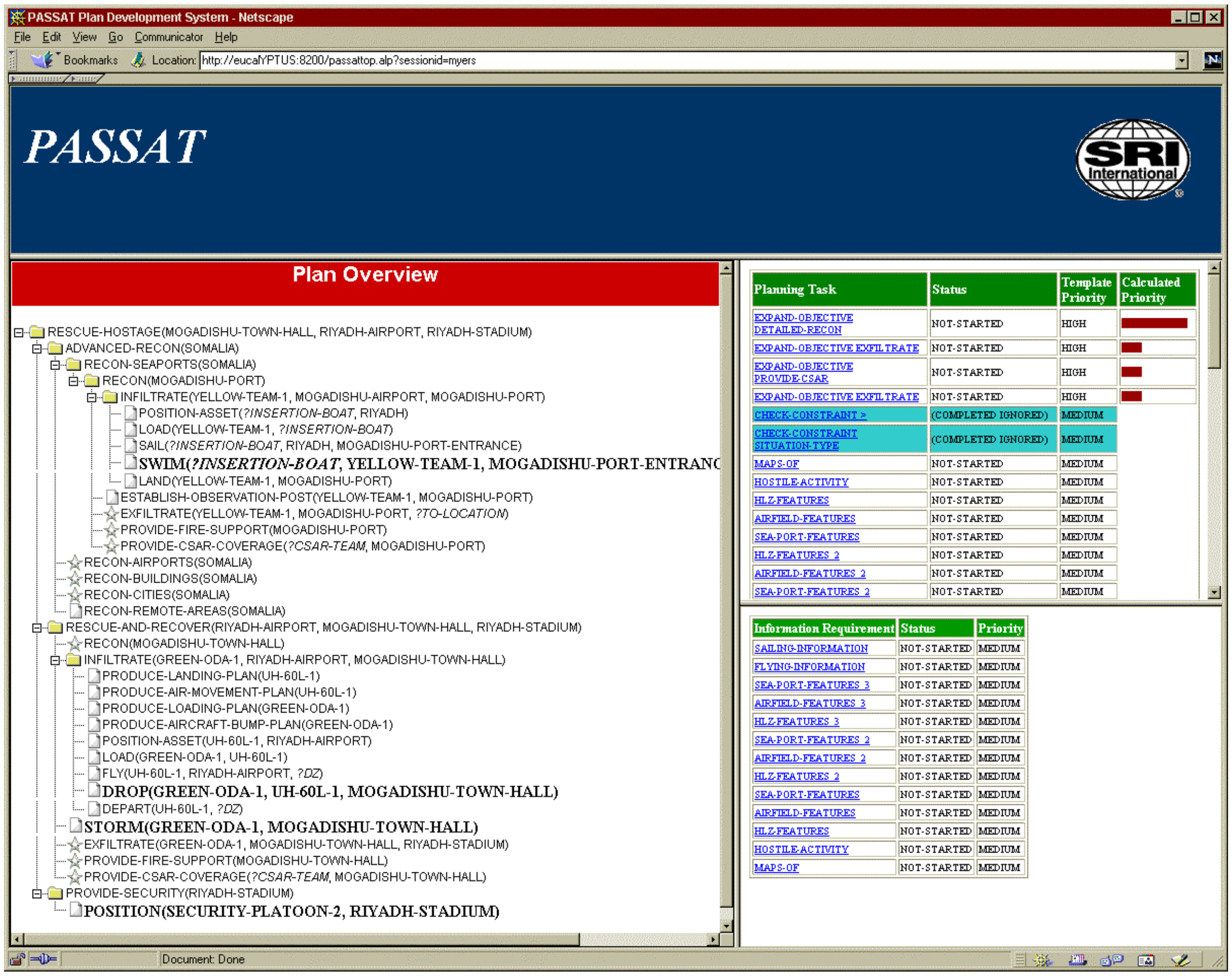
SOF planning lies well beyond the range of current automated planning technologies; moreover, fully automated solutions are unlikely ever to succeed because of the difficulty in formulating strategic knowledge with sufficient fidelity. In contrast, a PASSAT-style plan-authoring system provides a good technological match for the SOF planning domain. Missions arise unexpectedly, resulting in a need to assemble high-quality plans rapidly. Thus, the availability of tools to expedite plan development is important. Because many types of SOF operations can be broadly characterized with predefined templates, knowledge bases can be developed that capture certain portions of the planning process. However, individual operations tend to be highly distinctive, making it important to have tools that enable users to modify and customize plans to suit the needs of a particular situation.

Many potential application domains for planning technology share these characteristics of having partially formalizable domain knowledge and requiring significant user input to produce high-quality, situation-specific plans. On the military side, examples include air operations, disaster relief planning, and noncombatant evacuation operations. Space applications include science mission planning and ground operations planning.

PASSAT Example

Figure 1 shows a snapshot of the PASSAT interface during a planning session. The large frame on the left contains a hierarchical decomposition of the current partial plan. Items next to folder icons are tasks that have been expanded; items next to star icons are tasks that can be expanded further (either through automated template application or interactively); and items next to document

[1] Many of our templates were derived directly from SOF field manuals.


PASSAT Plan Development System - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://eucalYPTUS:8200/passattop.alp?sessionid=myers
PASSAT
SRI International
Document: Done

Plan Overview

```
RESCUE-HOSTAGE(MOGADISHU-TOWN-HALL, RIYADH-AIRPORT, RIYADH-STADIUM)
  ADVANCED-RECON(SOMALIA)
    RECON-SEAPORTS(SOMALIA)
      RECON(MOGADISHU-PORT)
        INFILTRATE(YELLOW-TEAM-1, MOGADISHU-AIRPORT, MOGADISHU-PORT)
          POSITION-ASSET(?INSERTION-BOAT, RIYADH)
          LOAD(YELLOW-TEAM-1, ?INSERTION-BOAT)
          SAIL(?INSERTION-BOAT, RIYADH, MOGADISHU-PORT-ENTRANCE)
          SWIM(?INSERTION-BOAT, YELLOW-TEAM-1, MOGADISHU-PORT-ENTRANC
          LAND(YELLOW-TEAM-1, MOGADISHU-PORT)
        ESTABLISH-OBSERVATION-POST(YELLOW-TEAM-1, MOGADISHU-PORT)
        EXFILTRATE(YELLOW-TEAM-1, MOGADISHU-PORT, ?TO-LOCATION)
        PROVIDE-FIRE-SUPPORT(MOGADISHU-PORT)
        PROVIDE-CSAR-COVERAGE(?CSAR-TEAM, MOGADISHU-PORT)
    RECON-AIRPORTS(SOMALIA)
    RECON-BUILDINGS(SOMALIA)
    RECON-CITIES(SOMALIA)
    RECON-REMOTE-AREAS(SOMALIA)
  RESCUE-AND-RECOVER(RIYADH-AIRPORT, MOGADISHU-TOWN-HALL, RIYADH-STADIUM)
    RECON(MOGADISHU-TOWN-HALL)
    INFILTRATE(GREEN-ODA-1, RIYADH-AIRPORT, MOGADISHU-TOWN-HALL)
      PRODUCE-LANDING-PLAN(UH-60L-1)
      PRODUCE-AIR-MOVEMENT-PLAN(UH-60L-1)
      PRODUCE-LOADING-PLAN(GREEN-ODA-1)
      PRODUCE-AIRCRAFT-BUMP-PLAN(GREEN-ODA-1)
      POSITION-ASSET(UH-60L-1, RIYADH-AIRPORT)
      LOAD(GREEN-ODA-1, UH-60L-1)
      FLY(UH-60L-1, RIYADH-AIRPORT, ?DZ)
      DROP(GREEN-ODA-1, UH-60L-1, MOGADISHU-TOWN-HALL)
      DEPART(UH-60L-1, ?DZ)
    STORM(GREEN-ODA-1, MOGADISHU-TOWN-HALL)
    EXFILTRATE(GREEN-ODA-1, MOGADISHU-TOWN-HALL, RIYADH-STADIUM)
    PROVIDE-FIRE-SUPPORT(MOGADISHU-TOWN-HALL)
    PROVIDE-CSAR-COVERAGE(?CSAR-TEAM, MOGADISHU-TOWN-HALL)
  PROVIDE-SECURITY(RIYADH-STADIUM)
    POSITION(SECURITY-PLATOON-2, RIYADH-STADIUM)
```

Planning Task	Status	Template Priority	Calculated Priority
EXPAND-OBJECTIVE DETAILED-RECON	NOT-STARTED	HIGH	🟥
EXPAND-OBJECTIVE EXFILTRATE	NOT-STARTED	HIGH	🟥
EXPAND-OBJECTIVE PROVIDE-CSAR	NOT-STARTED	HIGH	🟥
EXPAND-OBJECTIVE EXFILTRATE	NOT-STARTED	HIGH	🟥
CHECK-CONSTRAINT >	(COMPLETED IGNORED)	MEDIUM	
CHECK-CONSTRAINT SITUATION-TYPE	(COMPLETED IGNORED)	MEDIUM	
MAPS-OF	NOT-STARTED	MEDIUM	
HOSTILE-ACTIVITY	NOT-STARTED	MEDIUM	
HLZ-FEATURES	NOT-STARTED	MEDIUM	
AIRFIELD-FEATURES	NOT-STARTED	MEDIUM	
SEA-PORT-FEATURES	NOT-STARTED	MEDIUM	
HLZ-FEATURES 2	NOT-STARTED	MEDIUM	
AIRFIELD-FEATURES 2	NOT-STARTED	MEDIUM	
SEA-PORT-FEATURES 2	NOT-STARTED	MEDIUM	

Information Requirement	Status	Priority
SAILING-INFORMATION	NOT-STARTED	MEDIUM
FLYING-INFORMATION	NOT-STARTED	MEDIUM
SEA-PORT-FEATURES 3	NOT-STARTED	MEDIUM
AIRFIELD-FEATURES 3	NOT-STARTED	MEDIUM
HLZ-FEATURES 3	NOT-STARTED	MEDIUM
SEA-PORT-FEATURES 2	NOT-STARTED	MEDIUM
AIRFIELD-FEATURES 2	NOT-STARTED	MEDIUM
HLZ-FEATURES 2	NOT-STARTED	MEDIUM
SEA-PORT-FEATURES	NOT-STARTED	MEDIUM
AIRFIELD-FEATURES	NOT-STARTED	MEDIUM
HLZ-FEATURES	NOT-STARTED	MEDIUM
HOSTILE-ACTIVITY	NOT-STARTED	MEDIUM
MAPS-OF	NOT-STARTED	MEDIUM

Figure 1. PASSAT Interface during Plan Development

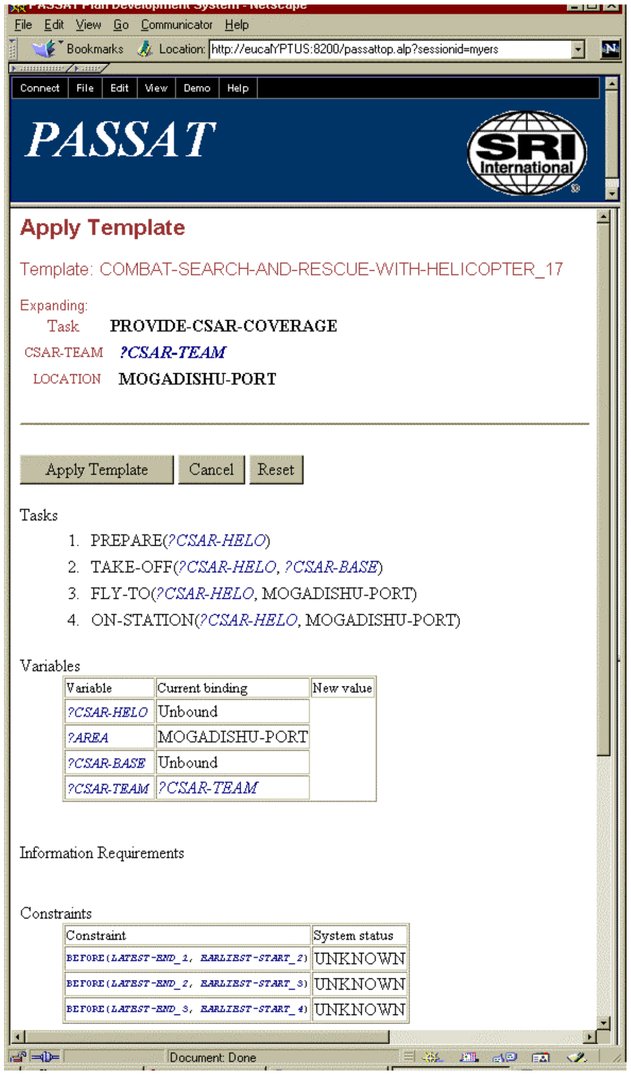
icons are tasks that match no templates. The frame on the upper right shows the current *agenda* – the list of planning steps the user must perform to address outstanding issues. The frame on the lower right shows the list of *information requirements* – sources of information that have been identified by the user or PASSAT's planning knowledge as relevant to various portions of the planning process.

The human planner develops the plan by selecting a planning step from the agenda and performing that step (many of these planning steps are accessible through the plan display as well). If the planning step is to expand the `PROVIDE-CSAR-COVERAGE` task, for example, the planner would be presented with several options: apply one of the templates that matches the task (see Figure 2), enter an expansion manually, or create a sketch for achieving the `PROVIDE-CSAR-COVERAGE` task and work with PASSAT to refine that sketch. Performing this planning step may cause additional planning steps to be added to the agenda (i.e., new tasks, variables, and constraints may have been introduced into the plan) and new information requirements as well.

Plan Representation

PASSAT's representation of plans and tasks is based on a fairly standard HTN model (similar to that of [Erol et al., 1994]), augmented with a rich temporal representation for tasks. Using PASSAT, a user would describe the objective of the plan in the form of one or more *task statements*, each consisting of a *task operator* and *terms* (variables, instances, or functions applied to terms).

Templates A *template* describes one way that a task (i.e., the template's *purpose*) can be decomposed into subtasks. A template consists of a set of these *subtasks*, as well the *variables* used in the template, *constraints* on the applicability of the template, and the *effects* of successfully performing individual tasks and the entire template. Different templates may describe different decompositions for the same task.


PASSAT Plan Development System - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://eucalYPTUS:8200/passattop.alp?sessionid=myers
Connect File Edit View Demo Help
PASSAT
SRI International

Apply Template

Template: COMBAT-SEARCH-AND-RESCUE-WITH-HELICOPTER_17

Expanding:

Task	PROVIDE-CSAR-COVERAGE
CSAR-TEAM	*?CSAR-TEAM*
LOCATION	MOGADISHU-PORT

Apply Template
Cancel
Reset

Tasks

1. PREPARE(*?CSAR-HELO*)
2. TAKE-OFF(*?CSAR-HELO*, *?CSAR-BASE*)
3. FLY-TO(*?CSAR-HELO*, MOGADISHU-PORT)
4. ON-STATION(*?CSAR-HELO*, MOGADISHU-PORT)

Variables

Variable	Current binding	New value
?CSAR-HELO	Unbound	
?AREA	MOGADISHU-PORT	
?CSAR-BASE	Unbound	
?CSAR-TEAM	*?CSAR-TEAM*	

Information Requirements

Constraints

Constraint	System status
BEFORE(*LATEST-END_1, EARLIEST-START_2*)	UNKNOWN
BEFORE(*LATEST-END_2, EARLIEST-START_3*)	UNKNOWN
BEFORE(*LATEST-END_3, EARLIEST-START_4*)	UNKNOWN

Document: Done

Figure 2. A Candidate Template for Task Refinement

PASSAT's template representation supports two features not found in the framework of [Erol et al. 1994], namely *information requirements* (discussed in detail below) and *enumeration* tasks. Enumeration tasks enable the specification of a set of tasks relative to a set of terms that satisfy a designed predicate. For example, the enumeration task

$$\forall\mathtt{?city.DISTANCE(?city,?hostage\text{-}locn)<20} \Rightarrow \mathtt{RECON(?city)}$$

indicates that a RECON task should be performed for each city within the specified distance. Other HTN frameworks (e.g., O-Plan [Currie and Tate, 1991] and SIPE-2 [Wilkins 1993]) provide similar mechanisms for enumerating subtasks relative to a designated constraint.

Constraints Constraints consist of state predicates that denote hard or soft conditions, perhaps due to physical laws or policy rules. PASSAT employs a three-valued logic for constraints, grounded in the values TRUE, FALSE, and UNKNOWN.

Automated constraint checking is performed when constraints are created or modified in the plan. Checking of ground constraints may return a status of UNKNOWN, if the information is not specified in the world state; such constraints would need to be validated explicitly by the user. Checking of nonground constraints occurs only when the number of possible instantiations is less than a predefined threshold, with the system testing whether the constraint is valid or invalid for each (i.e., establishing that the constraint is necessarily true or false independent of the instantiation). Otherwise, the system returns UNKNOWN and the constraint is rechecked when more variables are instantiated.

Unlike in automated planning systems, a constraint with value other than TRUE does not necessarily halt the process or cause backtracking. Instead, a violated constraint is called to the attention of the user, who has the choice of ignoring the violation or changing the step that triggered the violation.

Temporal Representation PASSAT supports the scheduling of tasks via constraints on the earliest and latest possible times for the start and end points of tasks. Temporal constraints typically refer to these end points but may also refer to upper and lower bounds on those time points. Temporal constraints can also be expressed using Allen's interval relations [(Allen, 1984)].[1]

Domain Definition PASSAT utilizes a number of coordinated databases to define its application domain. An *ontology* (based on the Generic Frame Protocol representation [Karp et al., 1995]) defines the hierarchical organization of classes and instances and their properties. *State predicate* and *task statements* are declared, specifying the number and classes of their arguments. *Functions* are similarly declared, with the additional declaration of the class of the function's value. Some predicates and functions are computable (e.g., <, +, and Distance) while others are defined by their extent. The *world state* is defined by a set of ground state predicates.

User-centric Plan Development

PASSAT currently provides two main modes of plan development: *interactive plan refinement* and *plan sketching*. Future versions of PASSAT will also support an advice module to guide plan development.

[1] The temporal reasoning portion of the system is not yet fully implemented.

Interactive Plan Refinement

Interactive plan refinement in PASSAT involves three types of planning step: *expand task*, *instantiate variable*, and *resolve constraint*.

Expand Task When a task is to be expanded, the system offers the user the choice of applying a predefined template, specifying a set of subtasks interactively, sketching a solution (see below), or dropping the task.

When the user chooses a template to apply, the system first unifies the task and the template's purpose, making appropriate substitutions throughout the template. PASSAT adds the (partially instantiated) subtasks and constraints of the template to the plan. In addition, it extends the agenda to include planning steps to expand the new subtasks, to check the new constraints, and to instantiate any unbound variables from the template. The planning step for the parent task is marked as completed and removed from the agenda. In the displayed plan, the parent task is shown with its subtasks.

As the system performs this step, it also checks the status of all new constraints. If one is found to be valid, the planning step to check it is marked as completed and removed from the agenda. If it is found to be invalid, the planning step is flagged.

As the system expands a task, other planning steps may be affected. If the unification results in the assignment of a value to a variable, the planning step for instantiating that variable is removed. The status of constraints that contained that variable might now be resolvable; the system checks those constraints and updates the planning steps, if necessary.

Instantiate Variable The agenda contains a planning step for each unbound variable within the current plan. When the user is ready to instantiate a variable, PASSAT provides the set of possible instantiations that satisfy all relevant constraints; the user can select from this set, provide an alternative value (hence, overriding a relevant constraint), or simply mark some subset of the values as unacceptable. When the variable is instantiated, any impacted constraints are rechecked. A user can optionally provide a justification (currently, a text string) for his actions.

Resolve Constraint As noted above, PASSAT provides automated checking of constraints as part of template application, with the agenda being used to track constraints that the system was unable to validate. *Resolve constraint* steps enable a user to declare that the system can disregard individual constraints with the status of `FALSE` or `UKNOWN` in a given situation. Such declarations do not have assertional import (i.e., they do not change the system's world model); rather, they enable relaxation of constraints from the planning model embodied in the domain templates. A user can declare that a given constraint be ignored for a variety of reasons: (a) he has more recent information that would validate the constraint, (b) he knows that the constraint is overly strong for the current situation, or (c) he wants to explore a *what-if* scenario. PASSAT supports the user in providing a justification (currently, a text string) for such constraint relaxations.

Robust Plan Sketching

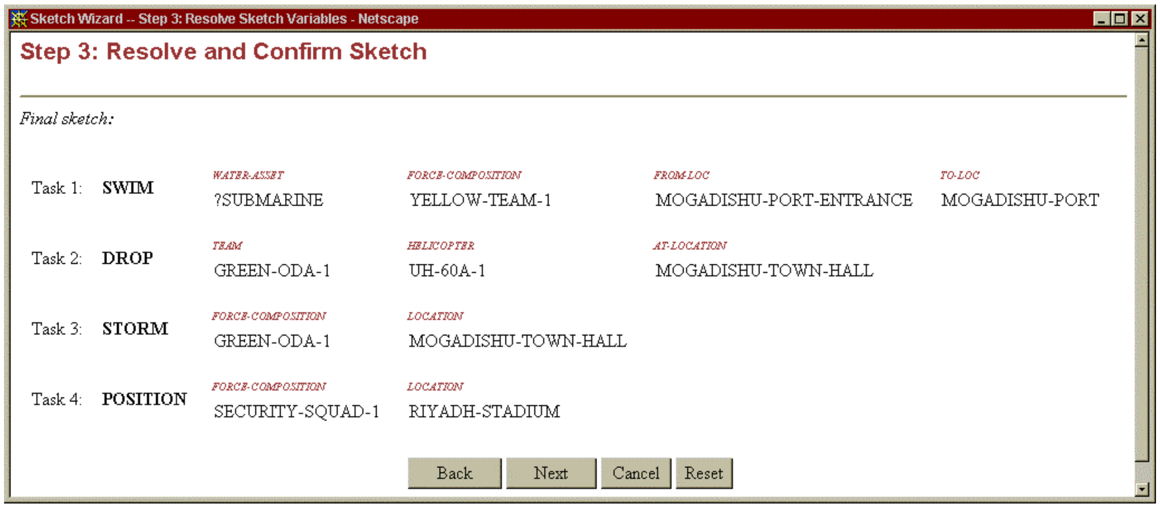
Hierarchical planning systems are designed to support top-down development of plans, taking an initial high-level objective and refining it to increasingly more concrete levels. Human planners, in contrast, often combine refinement-style planning with a more bottom-up approach that identifies specific tasks to be included in a final solution. For example, the planners of a hostage rescue may know where and how they will establish a safe haven without yet having decided on a particular high-level rescue strategy.

Within PASSAT, a user can *sketch* an outline of a plan, with the system providing assistance in expanding the sketch to a full-fledged solution for a particular objective. A sketch consists of a collection of tasks that (1) may be only partially specified, and (2) may occur at various levels of abstraction in the plan hierarchy. When given a sketch, PASSAT generates possible *sketch expansions*, which correspond to least-commitment plan structures that embed the sketch and all derived consequences. The user may choose any of these expansions to continue planning; the agenda will be updated to reflect the derived set of outstanding tasks.

The sketch processing capability within PASSAT builds substantially on the algorithms of [Myers, 1997] but provides robustness through an ability to recognize and respond to *invalid* sketches. By invalid, we mean a sketch for which there is no legal completion relative to the set of defined templates. To provide robustness in the face of invalid sketches, the sketch completion algorithm has been extended to tolerate constraint violations that are classified as *potentially fixable* according to prespecified domain knowledge about constraints and tasks (discussed further below). PASSAT guides the human planner through the process of repairing fixable constraint violations within expansions that he selects. Users can select from two types of repair method: *constraint drop* and *task modification*.

Constraint drop repair involves simply ignoring the violated constraint; this type of repair is appropriate for constraints with a 'soft' interpretation (i.e., they correspond to preferences or guidelines rather than gating conditions). For example, a template for a helicopter airlift may require wind speed below a certain threshold; a planner may decide to drop that constraint in the event that the current wind speed only slightly exceeds the threshold and all other requirements are satisfied. Constraint drop repair can be applied only to constraints that have been explicitly declared as ignorable for the sake of sketch repairs.

Task modification involves changing one or more arguments of a sketch task that are deductively linked to a violated constraint. For instance, consider a sketch that contains two tasks: the establishment of a safe haven at a particular location, and a helicopter airlift to remove


Sketch Wizard -- Step 3: Resolve Sketch Variables - Netscape

Step 3: Resolve and Confirm Sketch

Final sketch:

Task 1:	SWIM	WATER-ASSET ?SUBMARINE	FORCE-COMPOSITION YELLOW-TEAM-1	FROM-LOC MOGADISHU-PORT-ENTRANCE	TO-LOC MOGADISHU-PORT
Task 2:	DROP	TEAM GREEN-ODA-1	HELICOPTER UH-60A-1	AT-LOCATION MOGADISHU-TOWN-HALL	
Task 3:	STORM	FORCE-COMPOSITION GREEN-ODA-1	LOCATION MOGADISHU-TOWN-HALL		
Task 4:	POSITION	FORCE-COMPOSITION SECURITY-SQUAD-1	LOCATION RIYADH-STADIUM		

Figure 3. Sample Plan Sketch for the Hostage Rescue Task

rescued hostages to the safe haven. If the helicopter has insufficient range to reach the safe haven, the user would be given the options of selecting an air asset with appropriate range characteristics, or choosing a closer destination for the safe haven.[1] Domain knowledge restricts the set of arguments that can be modified in service of sketch repair, as a means of limiting the number of options to consider (both by the user and the system).

The robust plan sketch capability within PASSAT is designed to be used iteratively, with a human planner repeatedly refining a sketch in response to detected problems until a solution is found that meets his needs.

Sketch Example To illustrate the sketch-processing capabilities within PASSAT, we consider an example from a hostage rescue scenario in which a group of Americans is being held captive by guerrillas in Mogadishu's town hall. Riyadh Airport has been selected as the jumping-off location for the mission while the hostages are to be evacuated to Riyadh Stadium. The high-level task for this plan is represented as

```
RESCUE-HOSTAGE(MOGADISHU-TOWN-HALL,
               RIYADH-AIRPORT,
               RIYADH-STADIUM)
```

PASSAT provides an interactive editor for specifying a plan sketch. Figure 3 shows a completed sketch consisting of four tasks: (1) having an infiltration team (Yellow-Team-1) swim from a submarine (denoted by the variable `?SUBMARINE`) located at the entrance to Mogadishu Port to the port itself, (2) inserting a combat team (Green-ODA-1) at the Town Hall via a UH-60A helicopter, (3) having the combat team storm the Town Hall, and (4) positioning a security team at the evacuation site. The labels above each task argument identify that argument's 'role' in the task.

Processing of this sketch by PASSAT yields six expansions, with a range of three to four violated constraints in each. The user can select one of these expansions and explore options for repairing its associated constraint violations. Figure 4 summarizes the constraint violations and the hierarchical template structure for one of the expansions.

Figure 5 displays the window that would be presented to a user to assist in the repair of the original sketch. The window summarizes the available repair options for each violation, which may consist of dropping the constraint, changing a parameter for a designated task, or making no repair. Because the use of constraint dropping and task parameter changes is restricted (by predefined domain knowledge about their applicability), these repairs are not necessarily applicable in each case.

To support the user in changing a task parameter, the interface provides a drop-down list of candidate values. This set consists of instances for the type associated with that argument, filtered to remove values that lead to violations of the given constraint (in accord with the deductive linkage from the sketch task to the constraint). This filtering is incomplete: the list may include values that do not fix the detected problem, due to interactions with constraints in other parts of the plan. Future versions of PASSAT will incorporate additional checking to restrict this set further.

To repair the chosen expansion, the user could perform the following repairs:

- drop the constraint *VC1*
- modify the *Helicopter* argument of the `DROP` task to be `UH-60L-1` rather than `UH-60A-1` to

[1] A sketch could also be repaired by changing the type of a task, rather than simply changing the task arguments (e.g., ground-based evacuation rather than an airlift). PASSAT does not currently support this class of sketch repair.

Violated Constraints

VC1. `(SITUATION-TYPE RIYADH-STADIUM HOSTILE)`
VC2. `(DISTANCE-< RIYADH-AIRPORT MOGADISHU-TOWN-HALL (RANGE UH-60A-1)`
VC3. `(> (SEA-TEMPORATURE MOGADISHU-PORT-ENTRANCE) 40)`
VC4. `(PLATOON-SIZED SECURITY-SQUAD-1)`

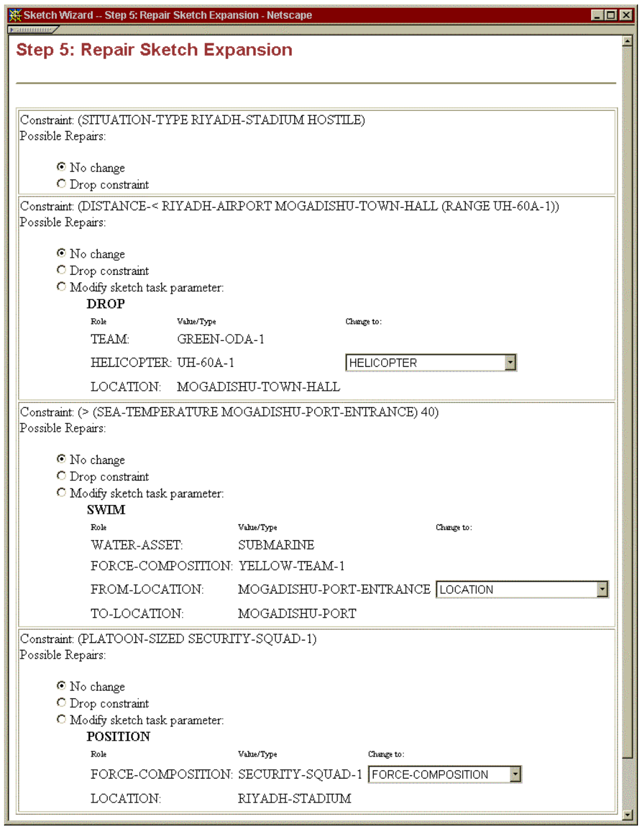
Expansion Template Structure

```
- HOSTAGE-RECOVERY-TO-A-POTENTIALLY-UNSTABLE-AREA
    - ADVANCED-RECON-OF-TARGET-AREA
         - RECON-SEAPORTS-IN-AREA
            - RECON-WITH-COVERT-GROUND-FORCE
                - SWIM-INSERTION-FROM-SUBMARINE
    - RESCUE-AND-RECOVER-HOSTAGES
         - HELICOPTER-INSERTION-ROPE
    - SITE-DEFENSE-LARGE-REACTION-FORCE
```

Figure 4. Violated Constraints and Plan Structure for the Selected Expansion

address the constraint *VC2* (i.e., the UH-60Ls have greater range than the UH-60As)

- drop the constraint *VC3*
- modify the *Force-Composition* argument of the `POSITION` task to be `SECURITY-PLATOON-1` (to address the violated constraint *VC4*)


Sketch Wizard -- Step 5: Repair Sketch Expansion - Netscape

Step 5: Repair Sketch Expansion

Constraint: (SITUATION-TYPE RIYADH-STADIUM HOSTILE)
Possible Repairs:

[✓ No change]
Drop constraint

Constraint: (DISTANCE-< RIYADH-AIRPORT MOGADISHU-TOWN-HALL (RANGE UH-60A-1))
Possible Repairs:

[✓ No change]
Drop constraint
Modify sketch task parameter:

DROP

Role	Value/Type	Change to:
TEAM:	GREEN-ODA-1	
HELICOPTER:	UH-60A-1	HELICOPTER
LOCATION:	MOGADISHU-TOWN-HALL	

Constraint: (> (SEA-TEMPERATURE MOGADISHU-PORT-ENTRANCE) 40)
Possible Repairs:

[✓ No change]
Drop constraint
Modify sketch task parameter:

SWIM

Role	Value/Type	Change to:
WATER-ASSET:	SUBMARINE	
FORCE-COMPOSITION:	YELLOW-TEAM-1	
FROM-LOCATION:	MOGADISHU-PORT-ENTRANCE	LOCATION
TO-LOCATION:	MOGADISHU-PORT	

Constraint: (PLATOON-SIZED SECURITY-SQUAD-1)
Possible Repairs:

[✓ No change]
Drop constraint
Modify sketch task parameter:

POSITION

Role	Value/Type	Change to:
FORCE-COMPOSITION:	SECURITY-SQUAD-1	FORCE-COMPOSITION
LOCATION:	RIYADH-STADIUM	

Figure 5. Sample Repair Options

Given a set of repairs, PASSAT attempts to validate the revised sketch. In this case, the repairs resolve the original problems but introduce a violation of the constraint `(COMBAT-EFFECTIVE SECURITY-PLATOON-1)`. This new problem can be repaired by changing the *Force-Composition* argument to be `SECURITY-PLATOON-2` (i.e., a platoon that has been certified ready for combat). Processing of this revised sketch yields a single expansion with no constraint violations. Figure 1 shows the insertion of that expansion for the original Hostage-Rescue task, with the sketch tasks highlighted in bold font. Constraints that the user chose to drop appear highlighted on the agenda, and are marked as completed but ignored. At this stage, the user could continue planning with the sketch result, using any of PASSAT's capabilities for interactive planning (e.g., applying templates, instantiating variables), or by providing a plan sketch for an unrefined objective.

Advice

In future work, we will extend PASSAT to enable a user to guide and control automated template expansion through the metaphor of *advice* [McCarthy, 1958]. Advice within PASSAT will express user recommendations for characteristics for the desired solution, thus limiting the set of allowed operations (human or automated) in constructing plans. Advice will be *heuristic*, capturing conditions that the user would like satisfied, but that can be relaxed if necessary.

PASSAT will monitor evolving plan content to identify violations of stated advice. Violations will lead to user notification, as well as the posting of appropriate planning task entries on the user's agenda. This work will build on our previous work on giving strategic advice to fully automated planners [Myers, 1996], with adaptations and extensions as required for use within a plan-authoring framework.

Usability Features

We have incorporated several features into PASSAT to facilitate its use within real applications.

Because the development of a plan may span several days or be interrupted by other duties, PASSAT offers the ability to save a plan and to restart it later. As PASSAT is further developed to support multiple planners working on a single plan, this facility will allow parallel efforts to be coordinated in a shared plan repository.

A planner may sometimes develop a part of the plan and realize that the initial idea will not work. The system currently allows the user to *undo* the steps in reverse order. In the future, the user will be able to back out of earlier steps without necessarily losing later, independent steps.

PASSAT is designed to reduce the chance of inadvertent errors. Strong typing for task, function, and predicate definitions enables the checking of inputs for consistency. If a processing error should occur in the system, the undo mechanism can provide recovery to a safe checkpoint.

Process Facilitation

PASSAT facilitates the user's plan-authoring process by helping the user track information that is important to the development of the plan. Process facilitation is supported primarily by two capabilities:

- A prioritized *agenda* of planning steps listing the decisions that the user must make to address problems or incompleteness in the current plan.
- A mechanism for identifying key *information requirements* implicit in the user's partial plan, and for directing the user's attention to relevant plan elements when new information arrives.

Agenda and Prioritization

PASSAT's agenda consists of the open planning steps facing the user given the current state of planning. By ‘planning steps’, we mean decisions and actions that the user makes in the process of developing the plan; these are distinguished from the activities that are part of the plan itself. PASSAT currently supports three types of planning step – *expand task*, *instantiate variable*, and *resolve constraint* – described earlier. The planning steps PASSAT displays in its agenda can be filtered by the user along several dimensions, including step type and completion status. The user can also sort the agenda along several dimensions, including step type, creation time, and alphabetical order. The filtering and sorting facilities can be especially useful for helping the user find a particular step on the agenda.

In real domains, the development of a plan can involve hundreds or even thousands of decisions. Correspondingly, PASSAT's agenda can grow quite long during the planning process. The system provides some basic mechanisms to control agenda growth – instantiating variables during template application, automatic calculation of constraints – and to control information overload in the agenda display – the aforementioned agenda filtering and sorting. However, even with these capabilities, the agenda can frequently reach a size that is overwhelming to the user. In the face of a large number of planning steps, we need a technique for keeping the human planner focused on the most important ones.

To deal with this problem, we have developed mechanisms for prioritizing the planning steps on the agenda, according to some notion of a step's importance to the planning process. Our approach has been to offer a suite of prioritization tools, from which the user may choose given the specific planning situation. Currently, PASSAT supports three prioritization approaches:

Predefined Each subtask, variable, and constraint in a template may be tagged with a qualitative priority (*high*, *medium*, or *low*), corresponding to the importance of making a decision about that entity (expanding the task, instantiating the variable, checking the constraint). Predefined priorities always take precedence over PASSAT's other prioritization methods in ordering the agenda display.

Commitment-based This approach prioritizes each planning step according to the degree that a decision will constrain the rest of the planning process, giving highest priority to the most constraining decisions. This criterion is especially useful in collaborative planning situations, where it is important to make decisions early when they will constrain the alternatives available to other planners. Our technique measures commitment as the expected number of future decisions eliminated by performing the step. We approximate this with a recursive formula that performs a lookahead search through the plan space. While we use some simple heuristics to reduce the size of the search, the current procedure is still reasonably expensive relative to PASSAT's other update calculations. As a result, the current implementation of commitment-based prioritization covers only tasks. In future work, we will investigate techniques for approximating the commitment level of a planning step more efficiently.

Experience-based In contrast to the commitment-based approach, which is an attempt to identify what the planner should do next based on some theoretical model of planning, the experience-based approach bases its prioritization on what real human planners have done first in the past. The experience-based prioritization technique stores preference histories of planning steps, and learns a preference function for them using the online learning algorithm of [Cohen et al., 1998]. Planning steps are indexed by the step type, the object name, and the ‘call stack’ of templates that created the object.

Other possible methods for deriving a step's priority include

- *Urgency-based:* prefer decisions that involve execution tasks that are scheduled to start soon.
- *Backtracking-based:* prefer decisions that are difficult to achieve. This is effectively the prioritization criterion of the Fewest Alternatives First strategy and related heuristics [Pollack et al., 1997] used in automated planning.
- *Depth-first:* prefer steps that derive from the steps most recently performed by the user. This approach assumes that the user wants to remain focused on one area of the plan before moving to another.
- *Breadth-first:* prefer steps that derive from the steps least recently performed by the user.

Information Requirements

In real-world planning, the human planner often makes decisions based on criteria that are too complex or vague to formalize in a predicate. These criteria are often based on external sources of information (e.g., reports, meetings). For example, a SOF planner may want to base his selection of a rendezvous point on an overall assessment of an intelligence report from the relevant region, though it may be virtually impossible to formalize the exact set of conditions the planner is looking for within that report. In a plan-authoring system, we want to be able to capture these criteria and information sources, and record the connection between them and the relevant elements of the plan. PASSAT accomplishes this through the use of *information requirements*.

In addition to specifying the method for expanding a task, a template may also include one or more information requirements. An information requirement specifies a monitoring condition on an information source that may be useful for determining the applicability of the template, for selecting variable instantiations, or for resolving the template's constraints.

Currently, information requirements are used in PASSAT to make explicit to the user the connection between plan elements (e.g., *variables, constraints*) and information sources. When a planner activates an information requirement in a template, the system creates a link between the information described in the information requirement and an element or elements in the plan. When the information arrives, PASSAT calls the planner's attention to the relevant plan element by creating a high-priority item on the agenda to revisit that element. PASSAT's current method of detecting when information has arrived is to be told explicitly by a user, but one could imagine more sophisticated automated *sentinels* that would, for example, monitor data sources (e.g., Web pages, databases) for specific updates.

For example, a user planning a SOF mission may make a tentative assignment to a variable `?RENDEZ-POINT` based on the sketchy information available to him. At the same time, he may activate an information requirement representing an intelligence report on the region in question and attach it to the variable `?RENDEZ-POINT`. When the intelligence report comes in, PASSAT will notify the planner by putting the Instantiate Variable step for `?RENDEZ-POINT` back on the active agenda, giving it a high priority, and highlighting the element on the planner's agenda display.

Related Work

In its effort to increase relevance to real problems, the field of AI planning has recently produced a number of more human-centric technologies that incorporate both interactive and automated planning capabilities. Work in this area has progressed on two fronts: (a) the incorporation of more sophisticated reasoning into simple plan specification tools, and (b) the addition of interactive and mixed-initiative capabilities into existing automated planning systems. The first category includes systems such as the SOFTools Temporal Plan Editor, APGEN, and INSPECT. Examples in the second category include O-Plan, Heracles, and TRIPS.

The SOFTools Temporal Plan Editor [GTE, 2000] supports the graphical specification of a collection of activities on a series of timelines; its automated capabilities are limited to simple syntactic checking (e.g., action start times precede end times). The APGEN system [Maldague et al., 1997] provides a timeline-oriented interface for creating mission sequences as well as automated validation of predefined flight constraints. There is currently an effort under way to link APGEN to the RAX-PS planner [Jonsson et al., 2000] to enable the automated synthesis of plans. The resulting system will facilitate user-driven exploration of options, as automation enables candidate plans to be generated rapidly. INSPECT [Valente et al., 1999] provides an interactive planning environment in which users can create plans by drawing on predefined knowledge bases of planning operators. A knowledge-based critic looks for problems in user-formulated plans, both syntactic and (in limited cases) semantic.

Within Heracles [Knoblock et al., 2001], a user can construct plans by interactive selection and instantiation of predefined HTN-style templates. Heracles provides constraint reasoning that facilitates the planning process by focusing users on choices that are guaranteed compatible with earlier decisions. Plans must instantiate the predefined templates, thus preventing users from exploring ‘out of the box’ solutions. The TRIPS system [Ferguson and Allen, 1998] provides a dialog-based interface to a temporal planner that enables users to interactively guide the construction and execution of a plan through a cooperative, mixed-initiative effort.

O-Plan was developed initially as a fully automated HTN planning system but has been modified to incorporate interactive capabilities such as user support for operator selection and variable instantiation [Drabble and Tate, 1995], and human-driven exploration of multiple courses of action [Tate et al., 1998]. PASSAT lacks somem of the automated planning capabilities within O-Plan (i.e., there is no infrastructure to support automated search with backtracking), being focused instead on more human-centric planning methods (interactive planning, sketching, advisability). O-Plan contains a task agenda similar to that in PASSAT, but no prioritization methods. Furthermore, it does not include information requirements, or capabilities related to sketching or advice.

Conclusions

Our long-term objective for PASSAT is to provide a planning environment that covers the range from purely interactive through mixed-initiative to user-controllable automated planning capabilities. At all times, automation would be readily controllable and understandable by a human planner, enabling humans to determine when automation is used, to control how automation applies, and to validate or override any automated decisions.

PASSAT currently provides a strong base of interactive, template-based plan authoring and robust sketch-based planning. Our main next steps on PASSAT are (a) to increase the flexibility of the interactive planning, and (b) to implement the advisability module for imposing high-level constraints on a plan that can be validated automatically.

Acknowledgments. This work was supported by DARPA under Air Force Research Laboratory Contract F30602-00-C-0058.

References

Allen, J. F. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence* 23.

Cohen, W. W., Schapire, R. E., and Singer, Y. (1998). Learning to Order Things. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds). *Advances in Neural Information Processing Systems*, The MIT Press.

Currie, K., and Tate, A. (1991). O-Plan: The open planning architecture. *Artificial Intelligence*, 32(1).

Drabble, B., and Tate, A. (1995). O-Plan Mixed Initiative Planning Capabilities and Protocols, Technical Report, University of Edinburgh.

Erol, K., Hendler, J., and Nau, D. (1994). Semantics for Hierarchical Task-Network Planning. Technical Report CS-TR-3239, Computer Science Department, University of Maryland.

Ferguson, G., and Allen, J. (1998). TRIPS: Towards a Mixed-Initiative Planning Assistant. In *Proceedings of the AIPS Workshop on Interactive and Collaborative Planning*.

GTE. (2000). *SOFTools User Manual*.

Jonsson, A. K., Morris, P. H., Muscettola, N., Rajan, K., and Smith, B. (2000). Planning in Interplanetary Space: Theory and Practice. In *Proceedings of the Fifth International Conference on AI Planning Systems*.

Karp, P. D., Myers, K. L., and Gruber, T. (1995). The Generic Frame Protocol. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*.

Knoblock, C. A., Minton, S., Ambite, J. L., Muslea, M., Oh, J., and Frank, M. (2001). Mixed-initiative, Multi-source Information Assistants. In *Proceedings of the International World Wide Web Conference*.

Maldague, P., Ko, A. Y., Page, D. N., and Starbird, T. W. (1997). APGEN: A Multi-Mission Semi-Automated Planning Tool. In *Proceedings of the 1st NASA Planning and Scheduling Workshop*.

McCarthy, J. (1958). Programs with Common Sense. *Symposium on the Mechanization of Thought Processes*.

Myers, K. L. (1996). Strategic Advice for Hierarchical Planners, pp. 112-123. In L. C., and S. C Aiello, J. Doyle. Shapiro (Eds): *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, Morgan Kaufmann Publishers.

Myers, K. L. (1997). Abductive Completion of Plan Sketches. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI Press.

Pollack, M. E., Joslin, D., and Paolucci, M. (1997). Flaw Selection Strategies for Partial-Order Planning. *Journal of Artificial Intelligence Research* 6, 223-262.

Tate, A. (1977) Generating Project Networks, in Proceedings of the Fifth International Joint Conference on Artificial Intelligence.

Tate, A., Dalton, J., and Levine, J. (1998). Generation of Multiple Qualitatively Different Plans. In *Proceedings of the Fourth International Conference on AI Planning Systems*, Pittsburgh, PA.

Valente, A., Blythe, J., Gil, Y., and Swartout, W. (1999). On the Role of Humans in Enterprise Control Systems: The Experience of INSPECT. In *Proceedings of the DARPA-JFACC Symposium on Advances in Enterprise Control*.

Wilkins, D. E. (1993). Using the SIPE-2 Planning System: A Manual for Version 4.3, Artificial Intelligence Center, SRI International, Menlo Park, CA.