

Active Coordination of Distributed Human Planners

Karen L. Myers Peter A. Jarvis Thomas J. Lee

Artificial Intelligence Center
SRI International
333 Ravenswood Ave.,
Menlo Park, CA 94025
{myers, jarvis, tomlee}@ai.sri.com

Abstract

Effective coordination of distributed human planners requires timely communication of relevant information to ensure the overall coherence of activities and the compatibility of assumptions. This paper presents a framework called CODA that provides targeted information dissemination among distributed human planners as a way of improving coordination. Within CODA, each planner declares interest in different types of plan change that could impact his or her local plan development. As individuals develop plans using a plan authoring tool, their activities are monitored; changes that match declared interests trigger automatic notification of appropriate planners. In this way, distributed planners can receive focused, real-time updates of plan changes that are relevant to their local planning efforts.

Introduction

The scope and complexity of large-scale planning tasks generally requires cooperation among multiple planners with differing areas of expertise, each of whom contributes portions of the overall plan. These planners may be distributed both geographically and temporally, further complicating coordination.

As a concrete illustration, special operations forces (SOF) mission planning involves numerous people working on separate but interconnected facets (e.g., strategic, logistical, medical, intel) of an overall plan. The SOF planning process itself is time constrained, concurrent, and iterative. Individual planners construct subplans based on their expectations for the operating environment and requirements. As the overall plan develops, these expectations change and modifications must be made to reflect new information. Currently, such changes are communicated informally by word of mouth, or transmitted in batch mode at regularly scheduled coordination sessions. This approach can lead to omissions and delays that reduce the effectiveness of the overall planning process and the quality of the resulting plans.

The SOF planning domain lies well beyond the range of current automated planning technologies. Moreover, fully automated solutions are unlikely ever to succeed, due to two

main factors. First, effective planning for this domain involves a huge strategic component that requires evaluating options with respect to expected plan quality. This type of knowledge is extremely difficult to capture and model, being grounded in a combination of intuition, vast experience, and common sense. Planning knowledge bases constructed to date mostly ignore strategic issues, focusing instead on modeling preconditions/postconditions for action applicability and effects. Second, SOF planning tasks (e.g., disaster relief, counterterrorism) tend to be unique and distinctive, making it difficult to formulate reusable background knowledge with adequate coverage. As an additional consideration, human planners in this area are reluctant to cede full control to automated planning systems, even in situations where automation is possible.

The above characteristics apply to a range of planning domains, including more general military operations planning, workflow management, and certain types of space missions. Techniques from the AI planning community can still contribute to complex problem solving in these domains. In particular, *plan authoring* tools that build on AI planning concepts are being introduced to improve the quality and process of plan development (Valente *et al.* 1999; GTE 2000; Muñoz-Avila *et al.* 1999; Knoblock *et al.* 2001; desJardins *et al.* 2002). Plan authoring tools provide a set of plan editing operations and manipulation capabilities that support users in exploring and developing plans. These tools may provide some automated capabilities; however, their main role is to augment rather than replace human planning skills. Plan authoring tools introduce a degree of structure to the planning process, yielding principled representations of plans with well-defined semantics. Planning aids that reason over these structures can be defined, including tools to support interplanner coordination.

Three main considerations drive the development of coordination techniques within this type of setting.

Selectivity Coordination strategies must strike a balance between disseminating too little and too much information. Failure to communicate enough information could be disastrous, if planners do not receive notification of critical changes. Conversely, flooding planners with much extraneous information can lead to cognitive overload, with critical changes being lost in a sea of irrelevant updates. Overcommunication can be a further problem for

situations where bandwidth is limited (e.g., wireless devices) or communication is expensive.

Timeliness Information about plan changes must be received in a timely manner to ensure adequate time for human planners to assess impact on their local plans and develop appropriate repairs.

Nonintrusiveness Users are generally reluctant to embrace technologies that require changes to traditional work habits (Conklin & Yakemovic 1991). In particular, a user is unlikely to perform activities beyond his normal sphere of responsibility unless there are immediate and substantial benefits for him. Thus, coordination tools that impose demands on users must incentivize their participation.

This paper describes the CODA (Coordination of Distributed Activities) framework, which provides automated support for focused information sharing during collaborative plan development by a team of humans. While motivated by the SOF planning problem, CODA more generally targets applications where distributed human planners are assigned responsibility for developing subportions of a global plan. These subplans are expected to have a moderate degree of coupling due to the need for coherent strategy, coordinated actions, and sharing of limited resources.

Within CODA, each planner declares the kinds of plan change that are of interest to him or her; we call these declarations *plan awareness requirements* (or *PARs*). As users develop plans with a plan authoring tool, their activities are monitored. Changes that match plan awareness requirements are forwarded automatically to the person who declared interest in them. In this way, distributed planners can receive focused, real-time updates of plan changes that are relevant to their local planning efforts.

Because local planners declare precisely the information in which they are interested, CODA satisfies the criteria for selectivity stated above. The declaration of plan awareness requirements constitutes the only additional effort required by the human planners above and beyond their normal operation. Because this declaration process will produce direct benefits to the planner (namely, notification of changes in which they are interested), the motivation for this specification phase is strong. Finally, the real-time nature of the plan updates within CODA addresses the timeliness concern.

The paper begins with an overview of the planning model within CODA, followed by a description of the CODA architecture. Next, we define our language for expressing plan awareness requirements along with a formal semantics and algorithms for matching them to evolving plans. We then describe an implementation of the CODA framework, focusing on different matching modes that it supports and tools for creating and registering plan awareness requirements. Finally, we compare our approach to related work in distributed AI, active databases, and concurrent engineering.

Plan Model

The generative planning community typically models a plan as a partial order of actions, and planning as an action selection problem. More generally, planning can be viewed

Objective a goal to be achieved within a plan

Action an activity that can be performed to achieve objectives

Effect a world-state condition denoting an expected result of plan activities

Event a world-state condition denoting an expected externality

Decision Point a condition for branching among subplans

Role a required capacity within a plan

Relation a semantic connection among plan objects (e.g., an Action *supports* an Objective or *enables* an effect)

Figure 1: Plan Ontology for CODA

as a *design task* that involves creating, refining, and linking objects to produce a composite structure that satisfies stated design objectives. The plan authoring model underlying CODA embodies this more general model of planning.

Plan Ontology and Structure A CODA plan is composed of a collection of objects drawn from the plan ontology presented in Figure 1. Each plan object is characterized by a unique *id* and may optionally have a (not necessarily unique) *name*. A collection of *domain objects* augments the plan ontology, representing the basic entities within a specific domain.

While most of the ontology elements correspond to standard concepts in AI planning, *roles* are somewhat different. A role defines an abstract capacity within a plan, independent of the specific object that will eventually fill that capacity. For example, *place* roles are used frequently within the SOF domain: in a disaster relief plan, there may be one or more *assembly-point* roles that denote places where evacuees should assemble to be evacuated. A planner often knows early on that assembly-points with certain capacities are required, but the exact physical locations for them and the actions that make use of them may not be chosen until later in the planning process.

Planning Process The planning process for CODA involves specifying roles required within a plan, selecting objects to instantiate those roles, selecting particular actions to be executed, declaring expected effects and events, and asserting relationships among objects and actions. The planning process is incremental in nature, involving the selection, extension, and refinement of selected objects and actions until the plan meets the human planner's criteria for adequacy with respect to stated objectives.

Plan Query Language The language for expressing plan awareness requirements (described in a subsequent section) builds on a general-purpose plan query language for the CODA plan ontology. The query language consists of a typed first-order language specialized to the Generic Frame Protocol (GFP) model of frame representation systems (i.e., it includes the full range of GFP-defined functions and predicates for manipulating classes, instances, and relationships

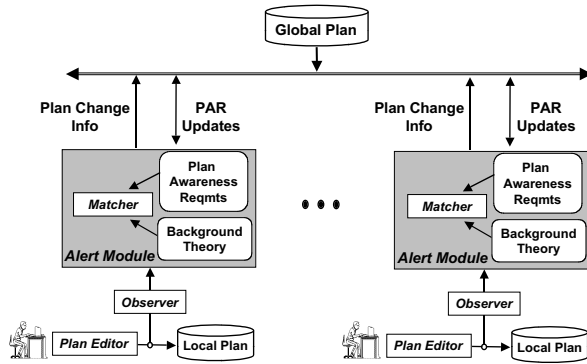


Figure 2: CODA Architecture

(Karp, Myers, & Gruber 1995)). Thus, for example, the language supports predicates of the form $(C \langle x \rangle)$ and terms of the form $(Attr \langle x \rangle)$ for every class C and attribute $Attr$ defined within the ontology. The language further includes equality, term constructors for lists and intensionally defined sets, and quantification with respect to an enumerable type.

CODA Architecture

Figure 2 presents the architecture of CODA. Within the context of a global plan, individuals work independently to produce local plans for their assigned tasks. Plans are developed using a structured *plan editor*, which supports a broad range of plan manipulation capabilities. User interactions with the plan editor are tracked by an *observer* module, which maintains a complete history of editing operations. As events are logged, a semantically grounded representation of the local plan is built within CODA. This internal representation can be annotated and used for reasoning, independent of the plan editor GUI.

The *matcher* provides the main inferential capability within CODA, being responsible for linking observed plan changes to declared PARs. The matching process may involve reasoning with a *background theory*, whose role is to bridge the gap between low-level plan edits and PARs expressed in high-level languages. When matches are detected, notification is sent to the local planner who registered the matched plan awareness requirement.

SOFTools Temporal Plan Editor CODA could be linked to a variety of manual and automated planning tools. Currently, it is connected to a specific plan editor, the SOFTools Temporal Plan Editor, which was developed to support graphical editing of SOF plans (GTE 2000). SOF plans involve the coordination of large numbers of activities and resources, subject to complex temporal synchronization constraints. CODA's event monitoring for the Temporal Plan Editor covers most of the available editing operations, including creation, modification and deletion of objects, modification of object attributes, temporal shifting of activities, and resource assignment.

Background Theory The background theory extends the inferential capabilities of the matching process by allowing definition of PARs in a high-level language whose expressivity extends beyond the basic plan query language. As such, the background theory enables a greater separation between observable plan modifications and the vocabulary for expressing PARs.

The background theory encompasses three types of information. The first specializes the CODA plan ontology with domain-specific information. For example, each domain would have its own specialization of the plan ontology class *Event*, which could be organized into a hierarchical structure of subclasses (e.g., *Weather-Event*, *Equipment-Event*) and instances (e.g., *Hurricane*). The second type consists of the domain objects, such as locations and resources, which are also structured in hierarchical fashion. Our CODA system stores these two types of background information within the Ocelot frame representation system (Karp, Chaudhri, & Paley 1999). The third type extends the underlying plan language with functions and relations defined over plan and domain objects. For example, our SOF application of CODA includes functions for computing distance and compass bearing between geographical points, and capacity analysis functions.

To appreciate the value of the background theory, consider the class of plan changes corresponding to *The addition of insertion points within two miles of an existing insertion point*. Our representation of plans supports the creation and manipulation of specific types of insertion points (e.g., *Helicopter-Landing-Zones*, *Drop-Points*) with a range of attributes that includes geographical location. The background theory in this case provides the ability to refer to a collection of instances (i.e., the elements of the type *Insertion-Point*), as well as to measure geographic distance between objects. The above plan change would be cumbersome to describe without these predefined background concepts.

Plan Awareness Requirements

We define two types of plan awareness requirement: *plan-state* and *transition*. Plan-state PARs describe conditions of a plan; in contrast, transition PARs describe *changes* to a plan.

Plan-State PARs

A plan-state PAR describes conditions of a plan and is modeled in terms of a well-formed formula in the plan query language. For example, a plan-state PAR for

There is an arrival to FSB Gold scheduled for after 8 AM

would be represented by the following formula in the plan query language¹:

```
(EXISTS (?X ?Y)
  (AND (MOVE ?X) (FSB ?Y)
    (= (NAME ?Y) GOLD))
    (= (DESTINATION ?X) ?Y)
    (> (ARRIVAL-TIME ?X) 800))
```

¹Symbols preceded by ? (e.g., ?x) denote variables.

Matching of a plan-state PAR occurs when a modification yields a plan that satisfies the associated plan query.

Plan Transition PARs

In its most general form, a plan transition PAR describes two plan states P and P' , and changes that transform the former into the latter. Transition PARs are grounded in the modification of individual or groups of objects. For this reason, transition PARs are defined in terms of an *object schema* or a *set specification*.

An object schema denotes a plan object with designated characteristics. Object schemas are specified using the syntax $(ANY ?x \phi[?x])$ where ϕ is an arbitrary formula in the plan query language. A set specification, given by the syntax $(SET ?x \phi[?x])$, designates a collection of objects with specified characteristics. We use the symbol σ to denote object schemas and the symbol Φ to denote set specifications.

We distinguish several categories of transition PAR, based on the nature of the underlying plan changes. These categories correspond to the creation, deletion or modification of plan objects, the refinement or generalization of plan object attributes, changes to specific attributes of a plan object, and changes to a collection of plan objects.

Instance Creation Instance Creation PARs declare interest in the addition of an object to a plan that satisfies stated conditions. An Instance Creation PAR is defined by an object schema σ that describes characteristics of the plan object to be created. For example, interest in

Addition of decision points related to weather calls

would be represented by an Instance Creation PAR with object schema

```
(ANY ?X (AND (DECISION-POINT ?X)
              (= (TYPE ?X) Weather-Call)))
```

Instance Deletion Instance Deletion PARs declare interest in the removal of an object from a plan that satisfies stated conditions. An Instance Deletion PAR is defined by an object schema that describes characteristics of the plan object to be deleted. For example, interest in

Elimination of a landing zone south of the embassy

would be represented by an Instance Deletion PAR with object schema

```
(ANY ?x (AND (LANDING-ZONE ?x)
              (= South
                 (DIR (POSN ?x)(POSN Embassy))))))
```

Instance Modification This class of PARs declares interest in the modification of an object that satisfies stated conditions. An Instance Modification PAR is defined by an object schema that describes the plan objects to be modified. As an example, interest in

Changes to 4th-platoon movements

would be represented by an Instance Modification PAR with object schema

```
(ANY ?X (AND (MOVE ?X)
              (= 4th-platoon (OPERAND ?X))))
```

Attribute Refinement Attribute Refinement PARs declare interest in the assignment of values to unbound attributes of plan objects. They are defined as a pair $\langle\sigma, A\rangle$ where σ describes a class of plan objects, and A the attribute to be bound.

Attribute Decommittment Similar to Attribute Refinement PARs, Attribute Decommittment PARs declare interest in decommitment from assigned attribute values. They are defined as a pair $\langle\sigma, A\rangle$ where σ describes a class of plan objects, and A the attribute to be decommitted.

Attribute Modification This class of PARs specializes Instance Modification PARs to changes to a specific attribute of a plan object. PARs of this type are defined as a triple $\langle\sigma, A, \delta\rangle$, where σ describes a class of plan objects, A the attribute of interest, and δ an optional *change predicate* that imposes constraints on the nature of the change. The change predicate can restrict the new value (e.g., $(= ?NEW 5)$) or the relationship between the old and new values (e.g., $(> (- ?NEW ?OLD) 3)$). For example, interest in

Delays of > 1 hour in expected time to secure the Church

would be captured by an Attribute Modification PAR $\langle\sigma, A, \delta\rangle$ with the components

```
 $\sigma$  (ANY ?X (AND (EFFECT ?X)
                  (= (TYPE ?X) Secure)
                  (= (OPERAND ?X) Church)))
A TIME
 $\delta$  (> (- ?NEW ?OLD) 1)
```

Aggregate Modification This class of PARs can be used to declare interest in changes to an intensionally defined collection of objects. The change may be to membership in the collection, or to some *aggregation value* defined over the collection. For example, an aggregation could be defined in terms of the movements that involve a particular type of personnel, with the aggregation value defined as the sum of resources employed by those movements.

An Aggregate Modification PAR is defined as a 4-tuple $\langle\Phi, A, \pi, \delta\rangle$, where Φ describes a set of plan objects, A the attribute to change, and δ a change predicate. The *aggregation function* π reduces a set of values to a single value; common aggregation functions include *MIN*, *MAX*, *SUM*, *COUNT*, and *AVERAGE*. As an example, the change

Decrease of more than 2 in the number of UH-60s used

would be represented by an aggregate modification PAR $\langle\Phi, A, \pi, \delta\rangle$ defined as follows:

```
 $\Phi$  (setof ?X (AND (MOVE ?X)
                  (= UH-60 (ASSETTYPE ?X))))
A ASSETCOUNT
 $\pi$  SUM
 $\delta$  (>= (- ?NEW ?OLD) 2)
```

The change predicate, attribute specification, and aggregation functions within Aggregate Modification PARs are each optional. Omission of the change predicate indicates that any change is acceptable. Omission of the attribute indicates that the aggregation function should be applied directly to the objects in the set designated by Φ . Omission

of the aggregation function indicates interest in the composition of the set of objects (or their attribute A); this last case corresponds to a ‘modify set’ semantics.

Match Semantics

We use the notation $\langle E, P, P' \rangle$ to denote a set of plan edit operations $E = \{e_1, \dots, e_m\}$ that maps plan P into a revised plan P' . The notation $\text{Holds}(\phi)$ indicates that the plan-state formula ϕ is a logical consequence of the background theory, while $\text{Holds}(\phi, P)$ indicates that ϕ is a logical consequence of plan P conjoined with the background theory. Finally, $k \in P$ ($k \notin P$) indicates that the plan object k is defined (is not defined) within plan P .

First, we define the concept of *domain* for an *object schema* and a *set specification*.

Definition 1 (Domain: Object Schema, Set Specification) The domain of an object schema $\sigma = (\text{ANY } ?x \phi[?x])$ or set specification $\Phi = (\text{SETOF } ?x \phi[?x])$ for a plan P , denoted by $\text{Domain}(\sigma, P)$ and $\text{Domain}(\Phi, P)$, respectively, consists of the set of plan objects $\{k_1 \dots k_n\}$ within P for which $\text{Holds}(\phi[k_i], P)$.

Definitions 2 and 3 capture the semantics for PAR matching.

Definition 2 (Match of Plan State PAR) A plan state PAR $\langle \phi \rangle$ matches $\langle E, P, P' \rangle$ iff $\text{Holds}(\phi, P')$.

Definition 3 (Match of Transition PAR) A transition PAR matches $\langle E, P, P' \rangle$ under the following conditions:

Instance Creation $\langle \sigma \rangle$: There exists a plan object k such that $k \notin P$ but $k \in \text{Domain}(\sigma, P')$.

Instance Deletion $\langle \sigma \rangle$: There exists a plan object k such that $k \in \text{Domain}(\sigma, P)$ but $k \notin P'$.

Instance Modification $\langle \sigma \rangle$: There exists a plan object k and attribute A of k such that

- $k \in \text{Domain}(\sigma, P), k \in P'$
- $\text{Holds}(\text{ATTR}(k, A) = v, P)$
- $\text{Holds}(\text{ATTR}(k, A) = v', P')$
- $\text{Holds}(v \neq v')$

Attribute Refinement $\langle \sigma, A \rangle$: There exists a plan object k such that

- $k \in \text{Domain}(\sigma, P), k \in P'$
- $\text{Holds}(\text{ATTR}(k, A) = v', P')$
- there is no v for which $\text{Holds}(\text{ATTR}(k, A) = v, P)$

Attribute Decommitment $\langle \sigma, A \rangle$: There exists a plan object k such that

- $k \in \text{Domain}(\sigma, P), k \in P'$
- $\text{Holds}(\text{ATTR}(k, A) = v, P)$
- there is no v' for which $\text{Holds}(\text{ATTR}(k, A) = v', P')$

Attribute Modification $\langle \sigma, A, \delta \rangle$: There exists a plan object k such that

- $k \in \text{Domain}(\sigma, P), k \in P'$
- $\text{Holds}(\text{ATTR}(k, A) = v, P)$
- $\text{Holds}(\text{ATTR}(k, A) = v', P')$
- $\text{Holds}(v \neq v')$
- if $\delta \neq \emptyset$ then $\text{Holds}(\delta[v, v'])$

Aggregate Modification $\langle \Phi, A, \pi, \delta \rangle$: Define π^* to be the identify function when $\pi = \emptyset$ and π otherwise. Let $D = \text{Domain}(\Phi, P)$ and $D' = \text{Domain}(\Phi, P')$. For v and v' defined by

$$v = \begin{cases} \pi^*(\{\text{ATTR}(x, A) \mid x \in D\}) & \text{if } A \neq \emptyset \\ \pi^*(D) & \text{if } A = \emptyset \end{cases}$$

$$v' = \begin{cases} \pi^*(\{\text{ATTR}(x', A) \mid x' \in D'\}) & \text{if } A \neq \emptyset \\ \pi^*(D') & \text{if } A = \emptyset \end{cases}$$

it is the case that

- $\text{Holds}(v \neq v')$
- if $\delta \neq \emptyset$ then $\text{Holds}(\delta[v, v'])$

The semantics for matching an Instance Creation PAR embody a *strict* interpretation of creation that requires the generation of a new object within the domain of the PAR’s object schema σ . A more liberal interpretation would match a plan transition whose changes move an existing object from outside to inside the domain of σ . For example, changing the type of a decision-point from `equipment-check` to `weather-call` could be interpreted as matching an Instance Creation PAR with object schema

```
(ANY ?X (AND (DECISION-POINT ?X)
              (= (TYPE ?X) weather-call)))
```

Similarly for Instance Deletion PARs, one might consider matches to include transitions where an object was modified so that it no longer satisfies the conditions of σ .

We opted for the stricter interpretation of change for two reasons. First, it matches more closely intuitions for creation and deletion. Second, changes that correspond to the more liberal interpretation can be expressed as Aggregate Modification PARs. For example, the more liberal interpretation of an Instance Creation PAR with object schema $\sigma = (\text{ANY } ?x \phi[?x])$ can be expressed by the Aggregate Modification PAR

```
((SETOF ?x \phi[?x]), \emptyset, SUM, (> ?NEW ?OLD))
```

which captures the notion of an increase in the number of objects that satisfy $\phi[?x]$.

Matching Algorithms

The matching of a plan state PAR reduces to the performance of a plan query. Matching of transition PARs is somewhat more complex.

The transition PARs that focus on instances and attributes (namely, Instance Creation/Deletion/Modification and Attribute Refinement/Decommitment/Modification PARs) have a decomposable nature. Testing for matches to these PARs reduces to checking for changes to plan objects within individual plans. As such, these PARs can simply be broadcast to the distributed CODA modules, with matching performed locally and results returned independently to the CODA module that registered the PAR.

Aggregate Modification PARs are not decomposable in this manner because they embody an implicit quantification over the union of the local plans under development. Testing of these PARs requires reasoning about changes across all

subplans. To illustrate, consider the PAR that represents the change

Decrease of more than 2 in the number of UH-60s used

If three individual planners each decrease their usage of UH-60s by one, their collective change results in a match to the PAR. Detecting a match of this type involves reasoning at an aggregate level about changes that have been made by the team of distributed planners.

Fortunately, the matching of Aggregate Modification PARs can be reduced to a combination of local matching and plan querying. For a given Aggregate Modification PAR

$$\gamma = \langle (\text{SETOF } ?x \phi[?x]), A, \pi, \delta \rangle$$

define the following *derived* PARs:

- *Attribute Modification PAR* $\gamma^{AM} = \langle (\text{ANY } ?x \phi[?x]), A, \emptyset \rangle$
- *Instance Creation PAR* $\gamma^{IC} = \langle (\text{ANY } ?x \phi[?x]) \rangle$
- *Instance Deletion PAR* $\gamma^{ID} = \langle (\text{ANY } ?x \phi[?x]) \rangle$

A match to any of the derived PARs in $\Gamma = \{\gamma^{AM}, \gamma^{IC}, \gamma^{ID}\}$ constitutes a necessary but not sufficient condition for match of γ : for γ to match, there must either be a change to one of the objects in the set (captured by γ^{AM}), or to membership in the set (captured by the combination of γ^{IC} and γ^{ID}).

Upon notification of a match to a PAR in Γ , the CODA module that registered γ can confirm a match for γ by aggregating the results from a set of distributed queries to each CODA module L_i . These queries consist of

$$\text{Objects}_{L_i} = \text{Query}[(\text{SETOF } ?x \phi[?x])]$$

$$\text{Values}_{L_i} = \{ \text{Query}[\text{ATTR}(?x, A)] \mid ?x \in \text{Objects}_{L_i} \}$$

for each local planner L_i . Determination of a match then involves evaluating the change predicate δ on the result returned by the application of the aggregation function to the union of these values, that is:

$$\delta(\pi(\cup_{L_i} \text{Values}_{L_i})).$$

CODA System

Figure 3 shows the interface for the CODA system (on the left) beside that of the SOFTools Temporal Plan Editor (on the right). Within the graphical plan representation of the Temporal Plan Editor, labeled horizontal lines correspond to places (e.g., a staging base or the Hospital), arrows correspond to actions, and diamonds denote key effects (e.g., the securing of a building or the attainment of a position).

The interface in the figure corresponds to a state in which a human is developing the maneuver portion of a plan to rescue hostages from a town held by enemy forces. Working from the top downward, the plan consists of two assault teams flying by helicopter from a warship to the Gold and Silver staging bases just outside the town. The assault teams then move by foot to the town and secure the buildings in which the hostages are being held (the Church, Hospital, and Town-Hall). The hostages are collected and taken to a third

staging base (Bronze) from which they are flown to safety in a transport aircraft.

Another human planner, whose interface is not depicted, is developing the fire-support portion of the same plan. The bottom half of CODA's interface displays the PARs registered by these two planners. The window labeled *Own PARs* indicates that the maneuver planner is interested in any delay of more than 15 minutes to the planned time of fire-support assets arriving on station. The window labeled *Others' PARs* shows the fire-support planner's interest in changes to the planned locations of helicopter landing zones (HLZs). The *Matches* window informs the maneuver planner that the fire-support planner has changed his plan in such a way as to delay the time at which a fire-support asset will be on station.

Matching Modes

The CODA system supports two modes for registering PARs, based on the frequency with which users are notified of matches.

PARs registered in *immediate* mode are checked after every plan edit operation (i.e., $E = \{e\}$ in Definition 3), thus providing planners with real-time notification of relevant plan changes. Immediate notification would be suitable for the endstages of planning (when plans are mostly stable and changes are significant), or during execution.

For earlier stages of plan development, frequent and wide-ranging changes to plans would be expected; real-time notification of matches during early plan development would be counterproductive. For PARs registered in *on-demand* mode, matching information is provided only in response to an explicit user request. Such requests produce summaries of matches for the current plan relative to a designated 'checkpoint' plan. On-demand matching can support coordination of distributed planners earlier in the planning process by enabling a given planner to periodically check for changes by other planners that could impact his own efforts.

PAR Definition and Registration

Within the CODA system, PARs can be registered and unregistered dynamically throughout a planning session. This flexibility is important because the informational needs of planners will necessarily evolve in response to changes in guidance, the dynamics of the environment, and the unfolding of the planning process itself. For example, a planner may wish to be notified in the event that spare capacity is available to transport certain extra cargo that would be required for one option under consideration; if he decides to adopt a different strategy that does not require the cargo, then he would unregister interest in available capacity.

PARs could originate from a variety of sources. Within CODA currently, PARs are either (a) selected from predefined libraries, or (b) created by individual planners on an as-needed basis.

PAR Libraries Libraries group PARs for standard types of related plan change that would generally be of interest to planners within a given domain. For example, a team responsible for medical needs during an evacuation would

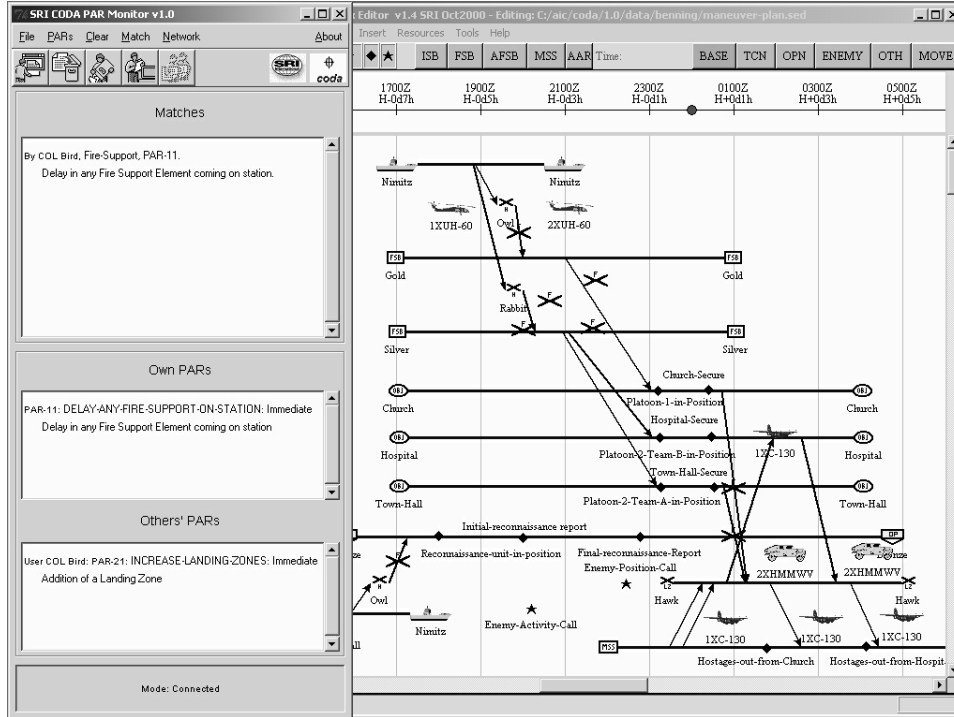


Figure 3: Interfaces for CODA (left) and the Temporal Plan Editor (right)

always be interested in changes to the size of the force and the expected number of evacuees. PARs for these types of change can be stored in a library specialized to the needs of medical planners. During a particular planning session, a planner would draw on libraries of predefined PARs to form the basis of his registered interests, augmenting them as necessary with authored PARs tied to the situation at hand and the current plan.

PAR Authoring CODA includes an interactive *PAR authoring tool* designed to help users quickly define PARs not contained within predefined libraries. This tool was developed using Adaptive Forms (Frank & Szekely 1998), a grammar-based framework that supports the specification of structured data through a form-filling interface that adapts in response to user inputs. With this tool, users create PARs by filling in forms with an English-like syntax; as users incrementally specify PARs, remaining options change in accord with the constraints of the underlying grammar. An internal compiler transforms these high-level specifications into the formal PAR structures required by CODA's matcher.

In designing a specification tool of this type, the competing requirements of expressivity and ease of use must be balanced. Sufficient expressivity is required to ensure coverage of relevant cases; however, support for full expressivity can lead to complex and unintuitive interfaces. To address this issue, CODA's PAR authoring tool provides two sets of forms. First, a set of *general forms* provides the full expressive power of the PAR language, including the ability to construct arbitrary expressions in first-order logic. While powerful, these forms require more effort to complete; in

addition, people unaccustomed to formal languages require training to use them effectively.

For this reason, the tool also includes specializations of the general forms that capture common *idioms* within the SOF planning domain. These specialized forms build in values that users would have to specify in the general case, thus simplifying and shortening the specification process. Parameters within the forms enable customization to a given planning session. SOF planners, for example, are generally interested in delays to activities. The SOF application of CODA includes (among others) the following parameterized PAR idiom:

Delays to any actions of greater than < duration >
Delays to action < action-id >

The first form supports declaration of interest in delays to actions that exceed a duration to be supplied by the user. The second form supports declaration of interest in delays to a user-specified action within a plan. Users can create PARs based on these specialized forms simply by supplying the designated parameters.

The idiomatic forms serve as the primary mechanism for authoring PARs within the CODA system, with the general-purpose forms reserved for defining PARs that lie beyond the scope of the predefined idioms.

Related Work

Distributed AI

Several AI planning systems have been built that provide coordination mechanisms to share information among dis-

tributed planners. In contrast to CODA, these systems link automated planners that construct plans with comprehensive causal structures. Analysis of plan dependencies within these structures forms the basis for determining the information to share among planners.

The COLLAGE planner (Lansky 1998) supports the conceptual partitioning of a planning problem into collections of regions, each of which constitutes a planning problem in its own right. Constraint propagation rules are used to maintain consistency among regions.

In the DSIPE distributed planning framework (Wolverton & desJardins 1998), each planner publishes a list of predicates that are *relevant* to its planning needs. Relevance is determined automatically through a reachability analysis of goals and operators. As planning proceeds, a planner determines whether a given planning decision establishes effects that unify with published relevant predicates from other planners, and notifies them as appropriate.

DSIPE's approach can be viewed as a counterpart to the CODA framework in which PARs are generated automatically. Because this automation relies on a complete and comprehensive set of planning operators, its applicability to plan authoring systems (where such information will not be available) is limited. Furthermore, the PAR representation provides a richer language for describing plan changes, going well beyond the simple effects-oriented approach within DSIPE.

Active Databases

Active databases augment traditional database technology with a set of rules that trigger activities in response to database modifications (Widom & Ceri 1996). Rules have the form $\langle E, C, A \rangle$ where E is an event that triggers invocation of the rule, C is a condition to be satisfied, and A is an action to be performed when C is met.

Although parallels can be drawn between PARs and active database rules, several characteristics distinguish our work. First, our approach involves monitoring changes to plans, which have richer structure than do relational or object-oriented databases. Second, we emphasize user-friendly languages and tools for declaring PARs, while the active database community has focused on automated synthesis of rules to support tasks such as the enforcement of integrity constraints. Finally, active database work does not consider the incorporation of background theories into the matching process.

Concurrent Design

The concurrent design community has developed techniques for detecting and resolving conflicts that arise during large-scale distributed design tasks (Klein 1993; Petri 1993). The general approach involves the documentation of *design rationale* – an explicit representation of design decisions and their interdependencies. As changes are made, the captured rationale supports automatic identification of decisions that might be impacted and should be reconsidered. The design rationale must be specified by users, thus imposing a substantial documentation burden during the design process. In contrast, our approach avoids such onerous documentation

requirements but leaves the task of ascertaining the ramifications of PAR matches to the user.

Conclusions

There has been a recent trend within the field of AI planning to increase relevance by focusing on more realistic and challenging problems drawn from real-world applications. Deeper exploration of several motivating domains (e.g., military operations planning, workflow management, space mission science planning) has revealed the impracticality of fully automated systems for many of these planning tasks. In particular, formulation of the knowledge bases required to support generation of high-quality plans in these domains lies well beyond the reach of current and foreseeable modeling capabilities. Instead, the experience, common sense, and intuitions of the human decision-maker are essential for effective planning in these domains. For this reason, many researchers within the AI planning community have begun developing *plan authoring tools* designed to assist a human decision-maker in constructing high-quality plans rather than to replace him.

The CODA framework provides a practical solution to the problem of coordinating the activities of distributed human planners engaged with such plan authoring tools. By having human planners explicitly declare those aspects of the overall planning process that interest them, CODA enables timely and focused distribution of information that can expedite and improve the quality of coordinated problem solving. The use of a rich, AI-based representation for describing plans, planning changes, and background theories provides the key to this technology.

CODA is a general-purpose coordination framework that can be linked to a variety of plan authoring tools. However, its development has been strongly influenced by the real-world challenges of SOF mission planning. Several domain experts provided guidance in the formulation of CODA to ensure that it meets the expressivity and usability needs of potential users. In addition to our current layering of the CODA system on top of the Temporal Plan Editor, we are looking to link CODA to additional types of plan authoring systems, such as an asset allocator and a map-based planner. Linkage of these tools through CODA will enable a rich and heterogeneous distributed plan authoring environment that provides greatly improved coordination over current practice.

Acknowledgments This work was supported by DARPA under Air Force Research Laboratory contracts F30602-97-C-0067 and F30602-00-C-0058. The authors would like to thank Fred Bobbitt, Doug Dyer, Warren Knouff, and Kelly Snapp for their help in shaping the development of CODA.

References

- Conklin, E. J., and Yakemovic, K. B. 1991. A process-oriented approach to design rationale. *Human-Computer Interaction* 6:357–391.
- desJardins, M.; Myers, K. L.; Tyson, M.; Wolverton, M.; Jarvis, P.; and Lee, T. 2002. PASSAT: From automated

planning to plan authoring. Technical report, Artificial Intelligence Center, SRI International, Menlo Park, CA.

Frank, M. R., and Szekely, P. 1998. Adaptive Forms: An interaction paradigm for entering structured data. In *Proceedings of the ACM International Conference on Intelligent User Interfaces*, 153–160.

GTE. 2000. *SOFTools User Manual*.

Karp, P. D.; Chaudhri, V. K.; and Paley, S. M. 1999. A collaborative environment for authoring large knowledge bases. *Journal of Intelligent Information Systems* 13:155–194.

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*.

Klein, M. 1993. Supporting conflict management in cooperative design teams. *Group Decision and Negotiation* 2:259–278.

Knoblock, C. A.; Minton, S.; Ambite, J. L.; Muslea, M.; Oh, J.; and Frank, M. 2001. Mixed-initiative, multi-source information assistants. International World Wide Web Conference.

Lansky, A. L. 1998. Localized planning with action-based constraints. *Artificial Intelligence* 98(1-2):49–136.

Muñoz-Avila, H.; Aha, D. W.; Breslow, L.; and Nau, D. 1999. HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations. In *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence*, 879–885. AAAI Press.

Petri, C. 1993. The Redux' server. In *Proceedings of the International Conference on Intelligent and Cooperative Information Systems (ICICIS)*.

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*.

Widom, J., and Ceri, S., eds. 1996. *Active Database Systems*. Morgan Kaufmann.

Wolverton, M. J., and desJardins, M. 1998. Controlling communication in distributed planning using irrelevance reasoning. In *Fifteenth National Conference on Artificial Intelligence*. AAAI Press/MIT Press.