

CODA: Coordinating Distributed Human Planners

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

Artificial Intelligence Center
SRI International
333 Ravenswood Ave.,
Menlo Park, CA 94025

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 system 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 changes 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 planning large-scale 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.

For complex domains of this type, *plan authoring* tools are being introduced to improve the quality and process of plan development (Valente *et al.* 1999; Knoblock *et al.* 2001; GTE 2000). Plan authoring tools provide a structured set of plan editing operations that support users in exploring and developing large-scale plans. While these tools may

provide limited automated capabilities, 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 inter-planner 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 information about 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 a distributed framework called CODA (Coordination of Distributed Activities) that 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 to reflect coherent strategy, to coordinate actions, and to share limited resources.

Within CODA, each planner declares the kinds of plan changes 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 and present formal definitions for matching them to evolving plans. Finally, we compare our approach to related work in distributed AI, concurrent engineering, and active databases.

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 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 following plan ontology. Each plan object is characterized by a unique *id* and may optionally have a (not necessarily unique) *name*.

Objective goal to be achieved within a plan

Action activity that can be performed to achieve objectives

Effect world-state condition denoting expected result of plan activities

Event world-state condition denoting an expected externality

Decision Point condition for branching among subplans

Role a required capacity within a plan

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

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

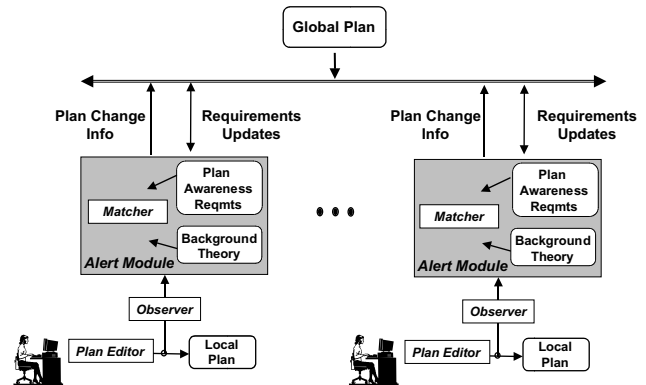


Figure 1: CODA Architecture

or more *assembly-point* roles that denote places where evacuees should assemble to be evacuated. Planners often know early on that assembly-points with certain capacities are required, but the exact physical locations may not be chosen until late 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 (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 bounded quantification (e.g., quantification with respect to an enumerable type).

CODA Architecture

Figure 1 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 process history of editing operations (as required for appropriate interpretation of editing operations such as redo and undo). As events are logged, a semantically grounded representation of the local plan is built within CODA. This internal representation of the plan 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 Planner 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 Planner, 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 Planner 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. CODA 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 distance and compass bearing between geographical points, and capacity analysis functions.

Plan Awareness Requirements

CODA supports two types of plan awareness requirements: *plan-state* and *transition*. Plan-state awareness requirements describe conditions of a plan; in contrast, transition awareness requirements 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 language:

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

Matching of a plan-state PAR occurs when a modification results in 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. Symbols preceded by ? (e.g., ?x) denote variables.

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

Instance Creation Creation PARs are used to declare interest in the addition of an object to a plan that satisfies stated conditions. They are defined by an object schema $\langle\sigma\rangle$ that describes characteristics of the plan object to be created. For example, interest in the

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 Deletion PARs are used to declare interest in the removal of an object from a plan that satisfies stated conditions. They are defined by an object schema $\langle\sigma\rangle$ 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 can be used to declare interest in the modification of an object that satisfies stated conditions. They are defined by an object schema, $\langle \sigma \rangle$, which 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 (OPELEMENT ?X))))
```

Attribute Refinement Attribute Refinement PARs can be used to 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 Decommitment Similar to Attribute Refinement PARs, Decommitment PARs can be used to 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 Object 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

```
 $\sigma$  (any ?X (AND (EFFECT ?X)
                  (= (TYPE ?X) Secure)
                  (= (OPERAND ?X) Church)))
A TIME
 $\delta$  (DELAY-OF 1 :HOURS)
```

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:

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

would be represented by

```
 $\Phi$  (setof ?X (AND (MOVE ?X)
                   (= UH-60 (ASSETTYPE ?X))))
A ASSETCOUNT
 $\pi$  SUM
 $\delta$  (DECREASE-OF 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 $HOLDS(\phi)$ indicates that the plan-state formula ϕ is a logical consequence of the background theory, while $HOLDS(\phi, P)$ 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 .

Definition 1 (Object Schema Domain) The domain of an object schema $\sigma = (ANY x \phi[x])$ for a plan P , denoted by $DOMAIN(\sigma, P)$, consists of the set of plan objects $\{k_1 \dots k_n\}$ within P for which $HOLDS(\phi[k_i], P)$.

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

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 $k \in DOMAIN(\sigma, P')$.

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

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

- $k \in DOMAIN(\sigma, P)$, $k \in P'$,
- $HOLDS(ATTR(k, A) = v, P)$
- $HOLDS(ATTR(k, A) = v', P')$
- $HOLDS(v \neq v')$

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

- $k \in DOMAIN(\sigma, P)$, $k \in P'$,
- $HOLDS(ATTR(k, A) = v', P')$
- there is no v for which $HOLDS(ATTR(k, A) = v, P)$

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

- $k \in DOMAIN(\sigma, P)$, $k \in P'$
- $HOLDS(ATTR(k, A) = v, P)$
- there is no v' for which $HOLDS(ATTR(k, A) = v', P')$

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 $\langle \Phi, A, \pi, \delta \rangle$:

- $v = \begin{cases} \pi^* (\{\text{ATTR}(x, A) \mid x \in \text{EXTENT}(\Phi, P)\}) & \text{if } A \neq \emptyset \\ \pi^* (\text{EXTENT}(\Phi, P)) & \text{if } A = \emptyset \end{cases}$
- $v' = \begin{cases} \pi^* (\{\text{ATTR}(x', A) \mid x' \in \text{EXTENT}(\Phi, P')\}) & \text{if } A \neq \emptyset \\ \pi^* (\text{EXTENT}(\Phi, P')) & \text{if } A = \emptyset \end{cases}$
- $\text{HOLDS}(v \neq v')$
- if $\delta \neq \emptyset$ then $\text{HOLDS}(\delta[v, v'])$

where π^* is defined to be the identify function when $\pi = \emptyset$ and π otherwise.

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, 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 more strict interpretation of change for two reasons. First, it seemed to match more closely intuitions for creation and deletion. Second, changes that correspond to the more liberal interpretation can be expressed as Aggregate Modification PARs of the form $\langle \Phi, \emptyset, \emptyset, \emptyset \rangle$, which captures the (somewhat more general) notion of changes to membership in the set of objects denoted by Φ .

Matching Process

CODA supports two control modes, based on the frequency with which users are notified of matches to their registered plan awareness requirements.

In *real-time* mode, PARs are checked after every plan edit operation (i.e., $E = \{e\}$ in Definition 3), thus providing planners with immediate notification of relevant plan changes. Real-Time 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 at that time would be counterproductive. CODA's *batch* mode provides users with summaries of matches to their PARs for a sequence of plan modifications performed on a designated 'checkpoint' plan. In this way, batch mode can support coordination of distributed planners

earlier in the planning process. Note that the checking process need not be concerned with the nature of the intervening plan modifications, since the semantics for matching only compare the original and current plans.

PAR Authoring

CODA includes an interactive *PAR authoring tool* designed to help users quickly and easily declare plan changes in which they are interested. This tool builds on the Adaptive Forms technology (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 using an English-like syntax; as users incrementally specify PARs, the remaining options change in accord with 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; additionally, 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 commonly used *idioms* within the SOF planning domain (e.g., delays to operations of a particular type, decrease in the use of a particular resource). The specialized forms build in values that users would have to specify in the general case, thus simplifying and shortening the specification process. The idiomatic forms serve as the primary mechanism for specifying PARs, with the general-purpose forms used for PARs that lie beyond the scope of the predefined idioms.

Over time, we anticipate that users will develop libraries of PARs for standard plan changes in which they are generally interested. During a particular planning session, a planner would draw on this library of predefined PARs to form the basis of their registered interests, augmenting them as necessary with declarations tied to the specifics of the situation at hand and the current plan.

Plan awareness requirements can be registered or unregistered dynamically throughout a session. This ability 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 doesn't require the cargo, then he would unregister interest in available capacity.

Related Work

Distributed AI Several distributed AI planning systems have been built that provide coordination mechanisms to share information among distributed planners. In contrast to the work reported here, these systems link automated planning modules that construct plans with comprehensive causal structures. Analysis of plan dependencies within the causal structure forms the basis for determining what information should be shared 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. Rules for constraint propagation are used to maintain consistency among regions.

In the DSIPE distributed planning framework (Wolverton & desJardins 1998), automated planners exchange information at planning time to ensure overall consistency. 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. A planner determines whether a given planning decision establishes effects that unify with published relevant predicates from other planners, and notifies them as appropriate.

The DSIPE approach can be viewed as a counterpart of the CODA framework in which PARs are generated automatically through analysis of a correct and complete causal plan structure rather than authored by humans. The generation, however, relies on the well-definedness of plans relative to an underlying set of planning operators, and hence cannot be used when planning is controlled by humans rather than by validated planning algorithms.

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 CODA. First, CODA monitors changes to plans, which have richer structure than do relational or object-oriented databases. Second, CODA emphasizes 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 technologies for the detection and resolution of conflicts that arise during the design of a complex artifact by a large, distributed team (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, CODA avoids such onerous runtime documentation requirements, but leaves the task of ascertaining the ramifications of PAR matches to the user.

Conclusions

CODA provides a practical solution to the problem of coordinating the activities of distributed human planners. By having human planners explicitly declare those aspects of the overall planning process that interest them, CODA provides timely and focused distribution of information that can expedite and improve the quality of coordinated problem solving.

While CODA is a general-purpose coordination tool that can be linked to a variety of plan authoring frameworks, its development was strongly influenced by the real-world challenges of SOF mission planning. Several domain experts provided guidance in the formulation CODA to ensure that it meets the expressivity and usability needs of potential users. We recently delivered CODA to a select set of operational users, who are conducting evaluations of its suitability for deployment within the SOF planning community.

References

- Conklin, E. J., and Yakemovic, K. B. 1991. A process-oriented approach to design rationale. *Human-Computer Interaction* 6:357–391.
- 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.
- 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.