Combining Specialized Reasoners and General Purpose Planners: A Case Study

Subbarao Kambhampati[1] Mark Cutkosky[2] Marty Tenenbaum[3] Soo Hong Lee[2]

[1]Center for Design Research
Dept. of Computer Science
Stanford University
Stanford, CA 94305-4026

[2]Center for Design Research
Dept. of Mechanical Engg.
Stanford University
Stanford, CA 94305-4026

[3]Center for Integrated Systems
Dept. Computer Science
Stanford University
Stanford, CA 94305-4070 *

Abstract

Many real-world planning problems involve substantial amounts of domain-specific reasoning that is either awkward or inefficient to encode in a general purpose planner. Previous approaches for planning in such domains have either been largely domain specific or have employed shallow models of the domain-specific considerations. In this paper we investigate a hybrid planning model that utilizes a set of specialists to complement both the overall expressiveness and the reasoning power of a traditional hierarchical planner. Such a model retains the flexibility and generality of classical planning framework while allowing deeper and more efficient domain-specific reasoning through specialists. We describe a preliminary implementation of a planning architecture based on this model in a manufacturing planning domain, and use it to explore issues regarding the effect of the specialists on the planning, and the interactions and interfaces between them and the planner.

1 Introduction

Many realistic planning problems require significant amounts of deep domain-specific reasoning. As an example, process planning for machining involves extensive reasoning about geometry, kinematics and cutting and clamping forces. The classical planning framework, in which the planner is modeled as an isolated module with all knowledge relevant to plan generation at its disposal, is inadequate for addressing such problems because it is impractical to encode deep models of specialized considerations in the constrained-action representations used by classical planners. While extending the action representation sufficiently to encode these considerations is possible, the cost of planning becomes prohibitive as the expressiveness of the domain models increases [14]. Most previous approaches for planning in such situations have dealt with these issues either through very domain specific planning algorithms (e.g. [5]), or by restricting themselves to shallow models of the specialized considerations (e.g. [3, 16]).

*We acknowledge the support of Office of Naval Research under contract N00014-88-K-0620. The authors' *email* addresses are *rao@cs.stanford.edu*, *marty@cis.stanford.edu*, *cutkosky@sunrise.stanford.edu* and *lee@sunrise.stanford.edu*.

In this paper, we investigate an alternative approach: a hybrid planning model that utilizes a set of specialists to complement the expressiveness and reasoning power of a traditional hierarchical planner. Such a model allows us to retain the flexibility and generality of the classical planning framework, while allowing deeper and more efficient domain-specific reasoning through the specialists. It can provide better computational efficiency since the specialists can employ methods that are best suited for particular kinds of analyses. It also facilitates better modularity by avoiding duplication of capabilities between the planner and the specialists.

Planning in such a hybrid model does however place several constraints on the operation of the planner (and the specialists), and raises many important issues regarding the exact role of the specialists, and the interfaces between them and the planner. To begin with, the specialists may be used to detect interactions that the planner itself cannot detect, or to extend the plan to make it satisfy additional constraints not modeled in the planner's own domain model. Further, some of these specialists may be involved in their own specialized planning (synthesis) activities. The analyses of the specialists may be dependent on the state of the plan, and the commitments made by the specialists may in turn have a direct bearing on the plan. Consequently, the planner and the specialists must each keep track of the constraints imposed on their decisions because of commitments made by the others, to avoid inconsistent commitments that could lead to costly inter-module backtracking.

As the planner and the specialists may employ disparate reasoning mechanisms and representations, a complete understanding of the operations of one by the other is not possible. This necessitates design of interfaces between the planner and the specialists that are at the right level of abstraction to enable each to recognize the constraints placed on their results because of commitments made by the other.

Planning in such architectures also has implications for the internal operation of the planner and the specialists. For example, hierarchical abstraction, and the ability to represent plans with partial commitment (partial ordering etc.) are important for allowing the specialists maximum latitude in specializing the plan according to their considerations. More importantly, since inconsistent commitments between planner and the specialists cannot be completely avoided, *incremen-*

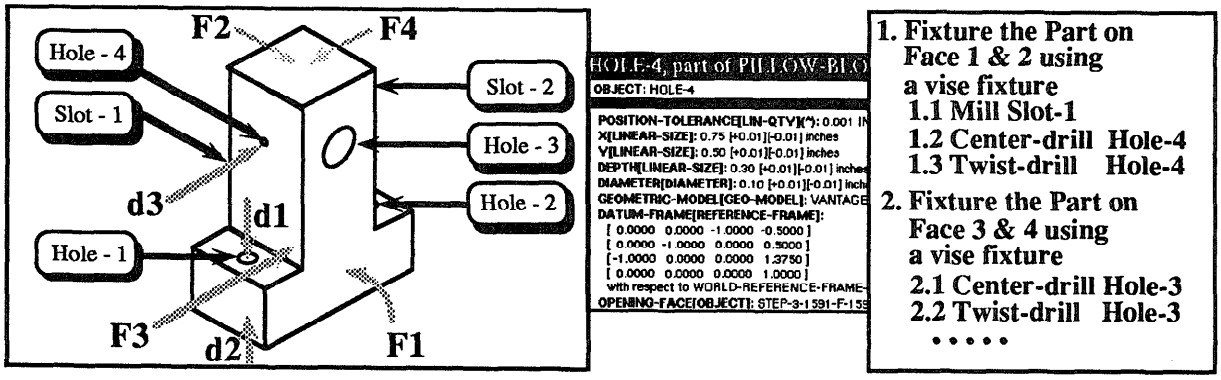

Figure 1: Geometric and feature-based specification of a part and a fragment of the plan for machining it

tal operation, in terms of the ability to reuse previous results while accommodating new constraints [7, 9, 10], is essential for efficiency. When the planner detects (through the interfaces) inconsistencies between its decisions and the commitments made by the specialists, it should be able to update the plan to resolve the inconsistencies. Moreover, to avoid affecting other specialists unnecessarily, this modification must be *conservative*, i.e., preserve as much of the previous plan as feasible. (We note that in contrast to classical planning model, where such replanning ability is justified purely in terms of the internal efficiency of the planner, here it is also motivated by the desire to promote efficient interaction between the planner and the specialists.)

The objective of this paper is to explore some of the complexities involved in planing in a hybrid planning architecture. We will do this through a case study of process planning in the NEXT-CUT concurrent design environment. We start by describing the particular characteristics of this domain that make it a good candidate for hybrid planning. In Section 3 we present the hybrid architecture that we have implemented for planning in this domain, and discuss the operation of the planner and the specialists, as well as the interfaces between them. Section 4 presents details of planning and plan revision in this architecture. In Section 5 we discuss some of the important limitations of this simple architecture and explore directions for overcoming them. Section 6 contains a brief discussion of the related work.

2 The domain characteristics

The domain that we are concerned with is process planning for machined parts. The planner is part of a prototype concurrent design system called NEXT-CUT, in which planning and analysis are performed step-by-step as a designer constructs or modifies a design [2]. In such a system, the planner serves two purposes: it generates plans for machining parts, and it provides designers feedback about the manufacturing implications of design decisions.

The input to the planner consists of the description of a part in terms of features, dimensions, tolerances and corresponding geometric models. Figure 1 shows part of the description of a simple component – which we shall refer to as `pillow-block` (a component used for supporting a shaft) – in terms of its geometry and features. It also shows a fragment of the process plan for machining `pillow-block`. The process plan includes a sequence of "setups" (particular orientations in which the work-piece should be restrained using fixturing devices such as a `vise` or `strap-clamps`), the set of machining operations (such as `drilling, milling, boring`) that should be carried out during each setup, and the tools (such as `0.25in-dia-twist-drill`) to be used during each machining operation.

There are several complexities involved in planning in this domain: First, there are typically interactions between different features such that machining one feature first may make it difficult or impossible to machine subsequent ones. What makes these interactions difficult from a classical planning point of view is that most are geometric in nature, and detecting them requires sophisticated geometric reasoning. Similarly, determining the necessary setups involves considerable reasoning about the intermediate geometry of the part, as well as kinematic and force equilibrium analyses.

Encoding the geometric and force knowledge required for these analyses in a general purpose planner is impractical, both because of the awkwardness of translating the analytical procedures underlying such analyses into the planner's representation, and because of the subsequent inefficiency of planning with such detailed models. Previous approaches for planning in this domain side-stepped these difficulties either by requiring that the input specification involve a description of all possible interactions (e.g. GARI, PROPEL [3, 16]), or by relying on domain-dependent algorithms to do planning (e.g. MACHINIST [5]).

3 Planning Architecture in Next-Cut

Figure 2 shows the schematic of the planning architecture in the NEXT-CUT environment. A general purpose planner is used for selecting appropriate machining processes and tools and composing them into a machining plan. A geometry specialist is used to detect and resolve geometric interactions that arise during machining, and a fixturing specialist is used to decide the orientations and clamping forces for holding the part during machining. There are two forms of communication among the planner and the specialists in the NEXT-CUT environment. The first, and more straightforward, is through the shared central model. The central model contains a description of the part in terms of its component fea-

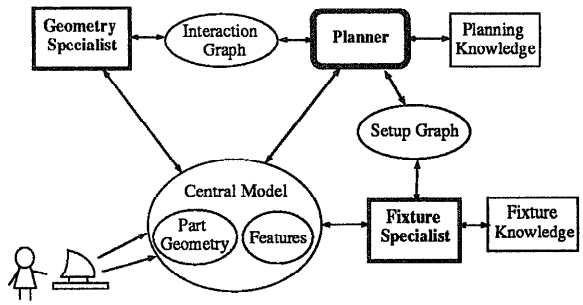

Figure 2: Schematic Diagram of the Planning Architecture in NEXT-CUT

tures, the attributes of the features and their geometry, which all modules can access and modify. The planner and specialists can also communicate through specialized interfaces (e.g. interaction graph, setup graph).

3.1 Planner

The planner is a hierarchical nonlinear planner similar to NONLIN [15]. It represents machining knowledge to select the processes and tools for machining individual features in terms of task reduction schemata. The plans are represented as partially ordered networks of tasks at successive levels of abstraction. Planning consists of reducing the abstract tasks to more concrete level subtasks with the help of the task reduction schemata, and resolving any consequent interactions. As a classical hierarchical planner, the planner only detects the interactions that become evident in terms of clobbered preconditions. (See [8] for a more detailed description of the planner).

As pointed out in Section 1, the planner needs the ability to modify its plans incrementally both to promote efficient interactions with the specialists and to deal with user-imposed changes in the design of the part. Our planner supports incremental plan modification by maintaining the causal dependencies among the individual steps of a plan, and the decisions underlying the development of that plan, in a representation called "validation structure." It utilizes the PRIAR modification framework [7, 8, 9] for carrying out the modification.

3.2 Specialists

The specialists in our framework either augment the specification of the problem as seen by the planner and detect interactions that the planner itself cannot detect, or utilize the generated plan to make their own further commitments. In our system, the geometry specialist is of the former type, while the fixturing specialist is of the latter. The analyses by the specialists impose implicit constraints on the plan developed by the planner (and vice versa). The interfaces – the interaction graph, and the setup graph – help the modules in keeping track of these constraints.

1. Geometry Specialist: The geometry specialist in the NEXT-CUT environment uses solid models of the part and features to detect a variety of geometric interactions that may affect the machining or fixturing of parts. Examples of such interactions include interferences between the tool paths for machining a feature, and the volumes of other features or the part itself. In the case of the `pillow-block` shown in Figure 1, the tool access path for machining `hole-4` (shown by the shaded arrow `d3` in the figure) interferes with the feature volume of `slot-1`. Window I in Figure 4 shows a description of the interference detected in this particular case. Once such interferences are detected, appropriate actions must be taken to resolve them (if possible). The geometry specialist checks to see if the volume of the detected interference is wholly subsumed by the volumes of some subset of other features in the part. If this is the case, then the interference can be avoided by machining those features first. Finally, the geometry specialist conveys these orderings to the planner by constructing (or updating) the interaction graph (see Section 4).

2. Fixturing Specialist: The objective of the fixturing specialist is to decide which operations of the plan will be done in which setup, and to arrive at fixture arrangements for locating and restraining the part as it is machined. An important consideration is to reduce the number of setups. The operation of the fixturing specialist can be understood to consist of two phases; with the first phase consisting of proposing setups and the second phase consisting of testing them, employing geometric, kinematic and force calculations. In the first phase, the fixturing specialist merges the steps of the machining plan based on the expected orientation of the part (and tool approach direction) during those steps. In the second phase, it checks if the part can actually be fixtured in the proposed setups, and selects fixture elements for restraining the part during machining. This involves selecting a particular sequence (total ordering[1]) of the proposed setups (consistent with the ordering constraints among plan steps that comprise the setup groups), and ensuring that the geometry of the work-piece at the start of each setup allows it to be fixtured satisfactorily. The specific sequence of fixturing groups that are tested by the fixturing specialist then constitutes the fixturing plan. The setup graph, which contains information about the chosen setup groupings, and the ordering relations among them, acts as the interface between the fixturing specialist and the planner (see Section 4).

4 The Planning Cycle

In this section, we discuss how the planner and the specialists interact through the interfaces to produce and revise plans. Figure 3 shows a high level description of the planning cycle, and Figure 4 shows the results of planning to produce the part shown in Figure 1.

When the specification of a part, such as that of `pillow-block` as shown in Figure 1, is entered for the first time, the geometry specialist computes the possible geometric interactions between its features(as shown by the example in Window I). Specific ordering constraints to avoid these interactions are then conveyed to the planner via the interaction graph (Window II).

[1]The need to ground the fixturing checks relative to the particular (intermediate) geometry of the part, and the difficulty of generating and maintaining partial geometries, are the main reasons why the fixturing specialist is forced to select a specific total ordering.

Given the plan representation discussed in Section 3.1, the **interaction graph** can be seen as an augmentation to the top-level specification of the problem. In particular, the interaction graph can be represented by a directed acyclic graph (DAG) $\mathcal{G} : \langle F, O_g \rangle$ whose nodes are the individual features of the part, whose edges define a partial ordering on the machining of different features.

The effect of the analysis by the geometry specialist is that instead of starting with unordered goals, the planner orders them according to the restrictions imposed by the interaction graph. In particular, the planner starts with an initial task network $\langle T', O' \rangle$, with T' containing the set of tasks of the form t_i : $Achieve(feature_i)$, and orderings of type $[t_i : Achieve(feature_i)] \prec_{O'} [t_j : Achieve(feature_j)]$ if and only if $feature_i \prec_{O_g} feature_j$. The final plan thus incorporates the orderings imposed by the planner, as well as those inherited from the interaction graph. The machining plan for `pillow-block` is shown in Window III. Notice in particular that the machining steps for `slot-1` and `hole-4` (in the lowest branch of the plan in Window III) are ordered according to the constraints specified by the interaction graph (Window II in Figure 4).

Next, based on this plan, the fixturing specialist chooses setups for fixturing. From the planner's view point, the fixturing specialist merges the plan steps based on a set of equivalence classes defined in terms of the expected orientations of the part during plan execution. As discussed in Section 3.2, this partitioning is followed by checks to ensure that some consistent sequence of these setups can actually be fixtured.

The **setup graph** can thus be formalized as a DAG $\mathcal{S} : \langle \Omega, O_f \rangle$ where each member $\omega \in \Omega$ is a set of plan steps that can be machined in a particular setup, and O_f is a total ordering on the setups.

The constraints on the setup-graph from the planner's viewpoint are that Ω be a set of mutually exclusive and exhaustive subsets of tasks in T, such that the partitioning is consistent with the partial ordering among the tasks. To ensure the latter, the following two constraints must be satisfied: (i) $\forall \omega \in \Omega$, $\forall t_1, t_2 \in \omega \nexists t \in T$ s.t. $t \notin \omega \wedge (t_1 \prec t \prec t_2)$ and (ii) $\forall \omega_1, \omega_2 \in \Omega$ if there exists a task $t_1 \in \omega_1$ and $t_2 \in \omega_2$ such that $t_1 \prec t_2$ in the plan, then it should necessarily be the case that $\omega_1 \prec_{O_f} \omega_2$.

From the point of view of fixturing specialist, each $\omega \in \Omega$ is a fixturing group. In general, once the fixturing specialist makes a merging of the plan steps according to the above constraints, there is an implicit partial ordering among the fixturing groups (as stated in the condition *ii* above). From the standpoint of fixturing, this merging is consistent as long as the fixturing specialist can find a sequence of the setup groups consistent with this partial ordering, which satisfies the fixturing constraints (see Section 3.2).

For the `pillow-block` example, Window IV-A shows the setup group mergings computed, and Window IV-B shows the description of the individual plan steps merged under each setup group. Notice that the graph is partially ordered at this point. The fixturing specialist selects one total ordering (shown in Window V) consistent with this graph that is satisfactory from the fixturing viewpoint, and computes a fixturing plan. It then updates the setup graph with additional orderings corresponding to the selected sequence. Figure 5 shows the fixturing details for the setup group `select-fixture-2` (highlighted in Window V, Figure 4). At this point, we have a complete process plan for machining `pillow-block` (see Section 3).

Given a new or changed specification:

1. **Geometry Specialist:** (*Input*: The solid model of the part and the features)
   Compute geometric interferences and update interaction graph
2. **Planner:** (*Input*: Feature specification, interaction graph, setup graph)
   (a) If no machining plan exists, generate one using the feature specification and the interaction graph
   (b) If a machining plan exists, modify it to accommodate the new specifications (changes in feature attributes, interaction graph or setup graph), while respecting any implicit constraints imposed by the setup graph and the interaction graph (see Section 4.2)
3. **Fixturing Specialist:** (*Input*: Machining plan, feature geometry, setup graph)
   (a) If a fixturing plan does not exist, construct the setup graph by merging steps of the machining plan. Select a setup sequence and compute the fixturing details for it. If no such total ordering is found, backtrack to the planner (see Section 4.1).
   (b) If a fixturing plan does exist, update the setup graph to reflect changes (if any) in the machining plan. Use it to incrementally revise the existing fixturing plan. Update the setup graph.

Figure 3: High level description of the planning cycle

4.1 Backtracking

When inconsistencies arise between the commitments made by the planner and the specialists, the linear control flow discussed above is disrupted, and backtracking is necessitated. When this happens, there are in general a variety of backtracking alternatives, some intra-module, and some inter-module, each presenting a different set of tradeoffs.

Consider, for example, the case where the fixturing specialist fails to find a fixturing arrangement to accommodate all the machining steps in a particular merged group ω in the setup graph. In such a situation, it will first try splitting ω into two or more setups $\{\omega_1, \ldots, \omega_m\}$ such that these groups can be fixtured individually. Note that splitting an existing setup group this way will not necessitate any revision in the machining plan (in particular the constraints *i* and *ii* on setup graph, discussed in Section 4 are not violated).

Sometimes, however, there may be a particular machining step which cannot be made in the chosen orientation without running into fixturing difficulties. At this point, there are two options: The first is to try an alternative tool approach direction for the feature associated with that machining step, and merge the operation for that feature with some other steps in the plan. Changing the orientation this way may cause new geometric inter-

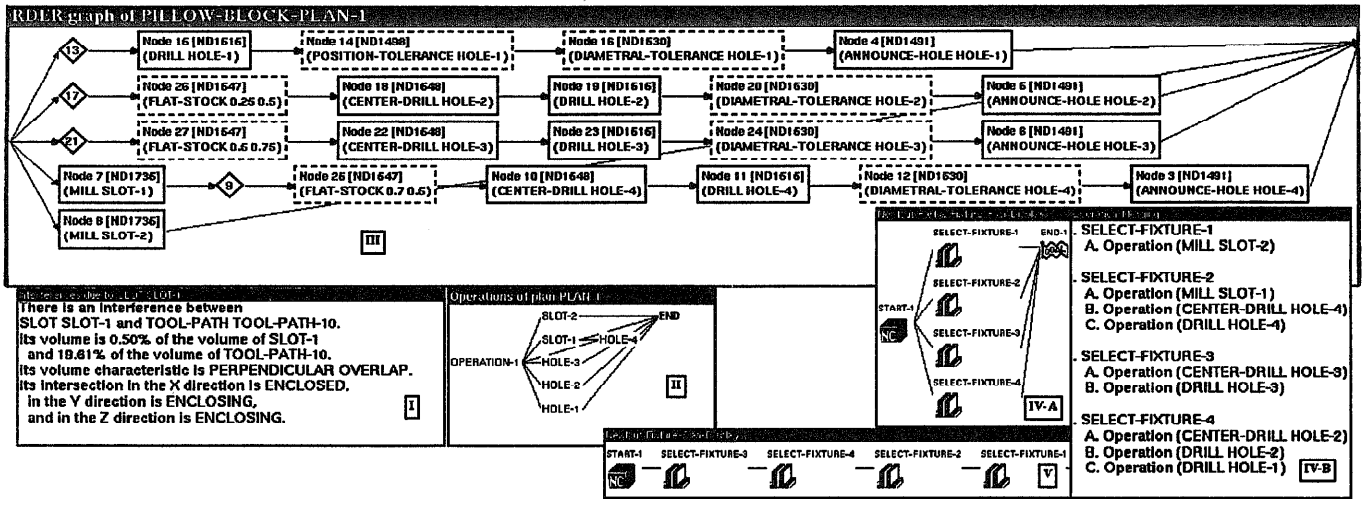

Figure 4: Planning for `pillow-block`: An example session in NEXT-CUT planning environment.

actions and may indirectly impose new ordering relations on the machining plan by changing the interaction graph. (For example, if the fixturing specialist decides to make `hole-1` in Figure 1 in direction `d1` instead of `d2`, then the geometry specialist will detect a new interaction between `hole-1` and `slot-1`.) The second option is for the fixturing agent to test a different total order on the setup graph (see Section 4), such that the machining steps corresponding to the problematic feature appear earlier or later in the sequence (so that the part geometry will be different when the feature is made).

In the first option, since there may be new interactions between the features, the machining plan would need to be revised, taking into account any updates in the interaction graph. In comparison, the second option involves only additional fixturing analyses. The tradeoff is not however as straightforward as this – analyses by the fixturing agent are typically more time consuming than any incremental analysis by the planner. So, currently our system prefers the first option (even though it causes inter-module backtracking). To deal with such tradeoffs in a more domain-independent fashion, the modules need to have some idea about the cost of violating individual constraints (see Section 5).

4.2 Plan Revision

We have seen that inconsistent commitments by specialists may necessitate revision of the machining plan. Similar revision is also necessitated in response to design changes. In both cases, the revision needs to be conservative both to ensure internal efficiency of planning, as well as to contain run-away ripple effects (see Section 1). As mentioned earlier, the planner uses the PRIAR modification framework [7, 9] to carry out this revision. There are however some additional difficulties which arise in revising plans in this architecture, that merit discussion. To begin with, we are no longer concerned solely with the internal consistency of the revised plan (as in [7, 9]), but with the global consistency – both the planner and the specialists must be satisfied with the current

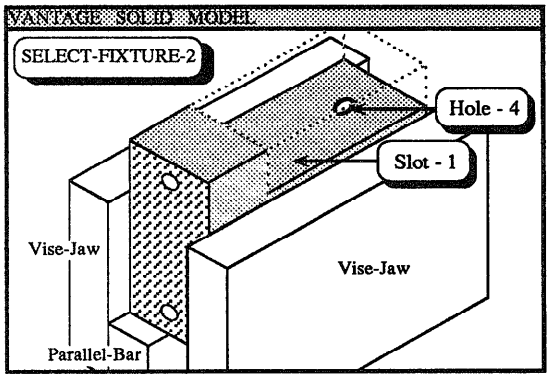

Figure 5: Details of a fixturing

state of the overall plan.

In particular, to avoid costly ripple effects, the planner must keep track of any implicit constraints imposed by the specialists, through the interfaces, and respect them during any plan revision. At the end of a normal planning cycle (discussed above), there are three types of ordering constraints among the steps of the plan:

(*i*) Orderings inherited from constraints imposed by the geometry specialist. In the example that we are following (see Window III of Figure 4), the ordering `(mill slot-1)` $\prec$ `(drill hole-4)` is of this type.

(*ii*) Orderings imposed by the planner during the planning (i.e., $t_i \prec_O t_j$) (e.g. `(center-drill hole-2)` $\prec$ `(drill hole-2)` in the example).

(*iii*) Orderings imposed by the fixture specialist (i.e., t_i belongs to the setup ω_i and t_j belongs to the setup ω_j such that $\omega_i \prec_{O_f} \omega_j$). E.g., `(drill-hole1)` $\prec$ `(mill slot-1)` in the example (since the step `(drill hole-1)` is included in setup `select-fixture-4` which precedes the setup `select-fixture-2` that includes `(mill slot-1)`)

During plan revision, planner is only capable of reasoning about the ramifications of violating the the orderings of type *ii*. Violating the other two types would lead to inter-module backtracking. In particular, violat-

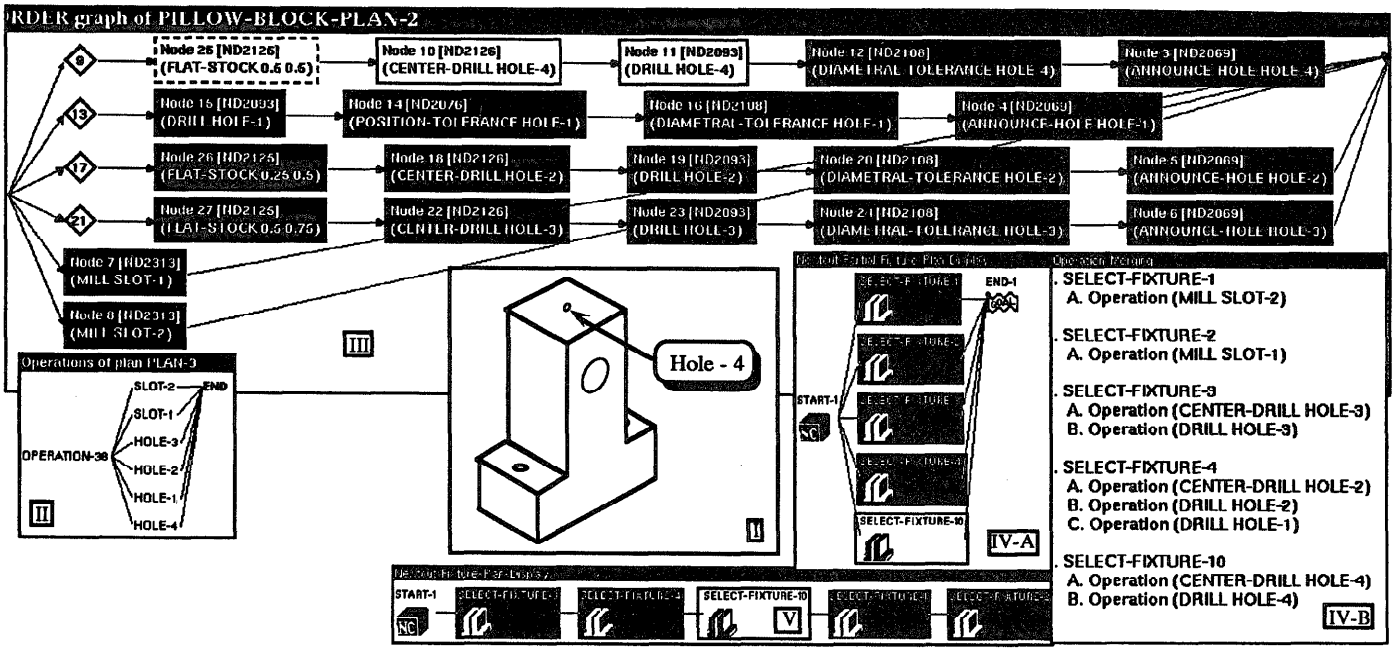

Figure 6: Revising the plan in response to external constraints

ing orderings of type *i* will lead to geometric interactions, while violating orderings of type *ii* will lead to costly refixturing analyses. To avoid these difficulties, the planner currently considers the externally imposed orderings to be non-negotiable. However, this may adversely affect the flexibility of modification (see Section 5).

Example: In the `pillow-block` example, suppose the designer changes the specification of the part, moving the set-screw hole, `hole-4`, from side to the top of the part, and increasing its depth slightly (as shown in Window I in Figure 6). This change has an effect on the interactions detected by the geometry specialist. In particular, the geometry specialist updates the interaction graph (see Window II in Figure 6) with the information that the ordering between `hole-4` and `slot-1` is not required since there is no longer an interference between them (note that left to itself, the planner would not have been able to detect this ramification of the design change). The updated interaction graph, and the changed specification of `hole-4`, now become the new specification for the planner. Since a machining plan already exists, the planner uses its incremental modification capability (see above) to accommodate these new specifications into the existing plan. Window III in Figure 6 shows the revised plan that the planner produces by accommodating this change. The black nodes in the figure represent the parts of the original plan (shown in Window III, Figure 4), while the white ones correspond to the newly added parts. In particular, the orderings between the machining steps of `hole-4` and those of `slot-1` are removed. Next, the fixturing specialist finds that it cannot merge the machining steps of `hole-4` and `slot-1` in the same setup (because the new orientation of `hole-4` is perpendicular to its old orientation). So it decides to split the corresponding setup group (`select-fixture-2` in Window IV-B of Figure 4) into two parts (`select-fixture-2` and `select-fixture-10` in Window IV-B). Using this updated setup graph (shown in Window IV-A in Figure 6), the fixturing specialist then revises the fixture plan (Window V) (details of fixture plan revision can be found in [12]). Once again, the black nodes represent the parts of the fixture plan that are salvaged from the original plan, while the white ones represent the results of new analysis. Notice that the planner's ability to revise the machining plan conservatively allows the fixturing specialist to reuse much of its analysis in turn, leading to a significant overall savings in computation.

5 Discussion of Limitations

In this section we look at some of the limitations of our current model, and discuss the directions that we are exploring to overcome them. To begin with, while the interfaces described in the previous sections allow the planner to keep track of the externally imposed constraints on the plan, they do not provide any indication of the reasons for the particular constraint, or the cost (in terms of additional processing by the specialists) that would be incurred if those constraints are violated. At present, we get around this problem by assuming that the external constraints are *non-negotiable* (see Section 4.2). However, such an assumption is too inflexible in that short of starting from scratch again, there may not be any way of conservatively revising the plan to resolve an inconsistency without violating any of the external constraints.

Consequently, we are exploring a framework where the external constraints are accompanied with an explanation structure–"window of applicability"–which provides a rationale for the constraint and the circumstances under which computed results would remain valid [8, 10]. It could, for example, document whether the constraint is a hard one or a soft preference; provide a cost measure associated with violating the constraint ; and/or attach some conditions under which the constraint is jus-

tified. Once again, the rationale needs to be at a level of detail that is commensurate with the planner's model of the domain. Such a framework will allow the planner to make educated decisions as to which constraints can be relaxed during plan revision process.

A related issue is the level of interfaces: In the current implementation, the specialist interfaces essentially impose external ordering relations on the plan. Within the classical planing framework, we could also accommodate interfaces that augment the specification of the planning problem. For example, the geometry specialist could provide a high-level description of the interference to the planner, and allow it to resolve the interactions itself. This may sometimes provide a finer grained interaction between the specialist and the planner. We are currently in the process of experimenting with this type of interface between the planner and the geometry specialist.

Finally, in our current implementation we have implicitly assumed a *sequential* control strategy (see Figure 3). We believe that more flexible interfaces (of the type discussed above) may allow us to exploit the inherent parallelism in the planning model through more parallel control regimes.

6 Related Work

There are several commonalities between the model of planning that we have explored here and work in multi-agent planning (e.g. [11]) distributed planning (e.g. [4]), black board based systems (e.g. [6]), and task-specific architectures [1]. In comparison to distributed planning approaches, which typically assume a common vocabulary among modules, and are concerned about coordinating a set of homogeneous planners working on different subgoals of a single problem, our hybrid architecture is concerned about the issues of cooperation between a general purpose planner and a set of specialists (with possibly disparate vocabularies and domain models). Constructing appropriate interfaces to facilitate effective interaction between the planner and the specialists is of critical importance in this model. Hybrid architectures similar to ours have been studied previously in automated reasoning – Miller and Schubert [13] describe a reasoning system that interfaces a general purpose theorem prover with a set of specialists to accelerate the general reasoning. Here, typically the general purpose reasoner already has a complete model of the reasoning carried out by the specialists. In contrast, in our model, specialists complement both the expressiveness and efficiency of the general purpose planner.

7 Conclusion

In this paper, we have explored a hybrid planning architecture which utilizes a set of specialists to complement both the overall expressiveness and reasoning power of a traditional hierarchical planner. We have described our preliminary implementation of this model in a manufacturing planning domain, and discussed several issues concerning the interfaces and interaction management between the planner and the specialists. The results of the implementation have been encouraging: Our architecture allowed effective interaction between the planner and the specialists, without binding the planner too tightly to the internal operations or the domain specific knowledge of the specialists. We are currently extending the architecture in several directions as discussed in Section 5. Given the current status of AI planning techniques, we believe that hybrid methodologies such as the one explored here offer a promising avenue of research for dealing with realistic planning domains.

Acknowledgements: Andrew Philpot helped in developing the planner and the interfaces. Amy Lansky provided several valuable criticisms on a previous draft. The AAAI reviewers made helpful suggestions to improve the clarity of the paper. To all, our thanks.

References

[1] B. Chandrasekaran. Design problem solving: A task analysis. *AI Magazine*, Winter, 1990.

[2] M. R. Cutkosky and J. M. Tenenbaum. A methodology and computational framework for concurrent product and process design. *Mechanism and Machine Theory*, 23(5), 1990.

[3] Y. Descotte and J. C. Latombe. Making compromises among antagonist constraints in a planner. *Artificial Intelligence*, 27:183–217, 1985.

[4] E.H. Durfee and V.R. Lesser. Predictability versus responsiveness: Coordinating problem solvers in dynamic domains. In *Proc.7th AAAI*, 1988.

[5] C. Hayes. Using goal interactions to guide planning. In *Proc. 6th AAAI*, 1987.

[6] B. Hayes-Roth. Dynamic control planning in adaptive intelligent systems. In *Proc. DARPA Knowledge-Based Planning Workshop*, 1987.

[7] S. Kambhampati. A theory of plan modification. In *Proc. 8th AAAI*, 1990.

[8] S. Kambhampati and M. R. Cutkosky. An approach toward incremental and interactive planning for concurrent product and process design. In *Proc. ASME WAM on Computer Based Approaches to Concurrent Engineering*, 1990.

[9] S. Kambhampati and J.A. Hendler. A validation structure based theory of plan modification and reuse. Tech. Rep. STAN-CS-90-1312, Comp. Sci., Stanford Univ., 1990. (To appear in *Artificial Intelligence*).

[10] S. Kambhampati and J.M. Tenenbaum. Planning in concurrent domains. In *DARPA Wkshp. on Innovative Approaches to Planning, Scheduling and Control*, 1990.

[11] A. Lansky. Localized event based reasoning for multiagent domains. *Computational Intelligence Journal*, 4(4), 1988.

[12] S.H. Lee and M.R. Cutkosky. Incremental and interactive geometric reasoning for part fixturing in concurrent product and process design. Tech. rep., Center for Design Research, Stanford Univ., 1991.

[13] S.A. Miller and L.K. Schubert. Using specialists to accelerate general reasoning. In *Proc.7th AAAI*, 1988.

[14] R. Simmons and R. Davis. Generate, test and debug: Combining associational rules and causal models. In *Proc. 10th IJCAI*, 1987.

[15] A. Tate. Generating project networks. In *Proc. 5th IJCAI*, 1977.

[16] J.P. Tsang. Propel: An expert system for generating process plans. In *Proc. SIGMAN Wkshp. on Manuf. Planning*, 11th IJCAI, 1989.