A Mixed-Initiative Approach to Computer Aided Process Planning

Martín G. Marchetta *
Facultad de Ingeniería, Universidad Nacional de Cuyo
Mendoza, Centro Universitario - 5500, Argentina
mmarchetta@fing.uncu.edu.ar

and

Raymundo Q. Forradellas
Facultad de Ingeniería, Universidad Nacional de Cuyo
Mendoza, Centro Universitario - 5500, Argentina
kike@uncu.edu.ar

Abstract

Several approaches have been proposed in order to develop intelligent applications on Computer Aided Process Planning (CAPP) domain. These approaches range from historic designs storage for later recovery, to generative synthesis of process plans. Although these approaches present advantages over traditional methods, they have several drawbacks derived specially from the under and over-automation of the decision processes. In this paper a mixed-initiative model for CAPP systems is proposed, that integrates plan recognition of user's intentions, with planning techniques in the context of artificial intelligence, in order to synthesize new designs that fulfill process planner's inferred intentions, thus improving the usability and usefulness of such intelligent assistants.

Keywords: Computer Aided Process Planning, Mixed-Initiative System, Planning, Intelligent Agent

1 INTRODUCTION

In the manufacturing context, process planning is the definition of manufacturing and assembly operations needed to produce the different parts, along with the machines, tools and fixturing required for these tasks [8]. Parts need to go through a set of manufacturing processes in order to be produced. This set of manufacturing processes is a process plan. Each manufacturing process within a process plan can be performed by machines of some type or family, and also requires certain type of tools. Besides, each part has its own Bill Of Materials (BOM). The BOM of a part to be manufactured is the set of raw materials and intermediate products, needed by each manufacturing process in the corresponding process plan in order to produce the part.

Manual process planning is an intense and time consuming activity. Besides, the quality of its results is very dependant on process planner's experience. The characteristics above mentioned yield the need of computer support to process planning activities. There are basically two approaches to CAPP, although there exist others that are special cases of these. The first one is called *variant CAPP*. This approach makes use of the concept of Group Technology (GT), in which parts with similar

*CONICET PhD fellow

features are grouped together. Then, a standard process plan is created for each group and is stored as a process plan template. When a manufacturing engineer is creating a process plan for a new part, he can retrieve the design template of the part's group, and adapt this standard process plan to the specific case he is working on.

The second approach for incorporating computer assistance to process planning, is called *generative CAPP*. This approach uses knowledge based systems, since the parts to be manufactured are described as a feature model. A feature model is a formal description of a part, that contains features and values for these features. Having an "open" description of a part to be manufactured, the manufacturing processes, the capabilities of machines and tools and a rule set indicating which process should be used to produce each feature value, the automatic synthesis of new design processes can be achieved.

The two mentioned approaches to CAPP have some drawbacks. On the one hand, variant CAPP does not assist process planners while they adapt the part family process plan template, to the case of the specific part they are working on. Besides, the manufacturing engineer must be conscious of the existence of a process plan template for that particular part's family, and needs to know which is the corresponding family in order to retrieve the correct process plan template.

Another important consideration is that the number of different parts the enterprise manufactures could make this approach unfeasible. Consider the case where the enterprise manufactures thousands of different parts for its products. In this scenario, there are two alternatives: many families, each one grouping a reasonable number of parts, or few families with a big number of parts each one. The former case has the advantage that each template needs less adaptation to special cases, since it corresponds to a small group of very similar parts. However, the identification of the correct part family is very complicated, tedious and error prone, since there exist the risk of a bad template selection. The bad selection of the design template to use could make the adaptation process more difficult and time consuming. On the other hand, if there are few part families, the retrieval process could be easy and fast, but the adaptation process would be longer and more difficult than the previous case.

Finally, when parts that do not fit on previously known families must be produced, no previous knowledge could be automatically exploited, since as we said before, there is no support for the adaptation process.

Generative CAPP, on the other hand, requires a tight integration with CAD tools, since the feature model of a part should be the output of the CAD system (or the output of some component that converts the CAD output into a feature model). Besides, the extraction of complete feature models from CAD data is very difficult (in most cases, either feature models or process plans must be completed by hand).

Another problem with this approach is that a great amount of knowledge must be represented in a formal way, since not only knowledge of manufacturing processes, machines and tools (which constitutes the basic enterprise's capabilities) must be represented formally, but also a set of rules indicating which process, machine and tool should be used to produce some value for a feature. Most of the systems based on generative CAPP follows expert systems approach, which not only requires the intervention of a domain expert, but also of a knowledge representation expert.

Finally, the manufacturing engineer's experience about process planning could not be interactively incorporated to each planning process, since in generative approaches the complete process is automated, from the feature model to the complete process plan, so the process planner can modify or adapt the process plan only when it has been completely synthesized. Each modification of the decision rules that guide the process plan synthesis is a knowledge representation task. In [5], this problem is pointed out by saying that, in general, if the user of one of these systems can answer the questions "what" and "how", probably he can not answer the question "why".

In this paper we propose a mixed-initiative approach that combines the exploitation of knowledge acquired from previous designs, the recognition of manufacturing engineering's design intentions, and a generative approach to generate new designs that fulfill these inferred intentions, in an interactive fashion, where both the user and the agent collaborates dynamically in order to produce a solution to the problem. The rest of this paper is organized as follows. In the next section, we present our mixed-initiative approach to CAPP, along with a brief overview of some plan recognition concepts. Section 3 presents the proposed execution model. In section 4 we summarize some related works on CAPP. Finally, section 5 gives some conclusions and future work.

2 MIXED-INITIATIVE COMPUTER AIDED PROCESS PLANNING

As mentioned in section 1, CAPP includes tasks that are time consuming, very intensive and where the results are sensitive to the process planner experience (even using variant CAPP). On the other hand, its complete automation is difficult, requires much knowledge representation, and does not allow the incorporation of the manufacturing engineer's experience and preferences during the process planning. In this context, it would be useful to have an intelligent agent that contributes to the process planning task with the advantages of automatic planning (in order to reduce the workload of manufacturing engineers), but which allows the process planner to incorporate his experience, preferences, restrictions and knowledge about certain aspects of the manufacturing processes that are not taken into account by the agent.

Thus, in this paper we propose a mixed-initiative approach that combines the exploitation of knowledge acquired from previous designs, the identification of manufacturing engineer's design intentions, and a generative approach to generate new designs that fulfill these inferred intentions, in an interactive fashion.

2.1 Mixed-initiative Systems

Mixed-initiative systems are those where each participating agent contributes with its best capabilities during the interaction, in order to find a solution. Thus, the problem solving and tasks execution on this kind of systems is seen as a collaborative process, where each agent can initiate an interaction, can propose solutions or give advise to the other, in contrast with single-initiative systems in which the agent that has the initiative in the interaction is previously specified [4].

Mixed-initiative systems, where both human and computer agents participate in the interaction, have the advantage that they do not automate completely the decision processes (since the human user is actively involved in the tasks execution and decisions taking), and at the same time they provide automated assistance in order to achieve user's goals in a better way. Thus, mixed-initiative interactions provides a mean for tackling the problems mentioned in section 1, specially the problem of reducing the user's workload on a complex knowledge-rich domain as CAPP, and at the same time allow him to introduce his experience interactively during the decision processes. Figure 1 shows the basic architecture of a mixed-initiative system.

In this paper, we present an approach where mixed-initiative is supported by a plan recognition component, integrated with an automated planner. The proposed approach learns and updates a plan library from previous designs created by the human process planners, uses plan recognition in order to infer his intentions and integrates these techniques with automatic planning for synthesizing new designs in accordance with the engineer's intentions.

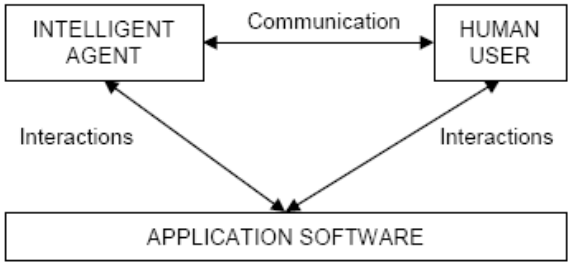

Figure 1: A Mixed-initiative system's basic architecture

2.2 Plan Recognition and Mixed-initiative Systems

Mixed-initiative systems require each participant to understand each other's possible goals, plans and capabilities, in order to make better contributions. In this context, plan recognition techniques are useful because they provide an intelligent assistant with a mean for inferring the user's intentions, by observing its behavior.

2.2.1 Plan Recognition

Plan recognition is the problem of inferring an observed agent's goals and intentions, sufficient to explain his observed behavior. Plan recognition techniques have been applied in many domains (natural language disambiguation, intelligent interfaces, military simulations, etc). Our interest in these techniques is because they are useful whenever a human interacts with an intelligent agent. Independently of the specific domain, if an agent is going to assist a user in his tasks, it must understand what the user is doing or trying to do. The communication between the agent and the user can be done in several ways. The user could explicitly ask the agent to do something, or they could communicate with each other implicitly, using some kind of shared user interface. Plan recognition techniques allow an assisting agent to automatically infer the user's intentions, reducing the need of explicit communication of his intentions. Figure 2 shows a similar architecture to that presented in figure 1, but including a plan recognition component in the intelligent agent.

Most of the existing plan recognition techniques rely on a plan library in order to infer user's intentions. These plan libraries contain information about the objectives the user may be pursuing, the tasks needed to achieve these objectives, constraints on tasks and parameters (such as orderings, causal links, parameter values propagation, etc), and possibly probabilistic information. During the recognition process, primitive actions are observed in the user's behavior, and using these observations

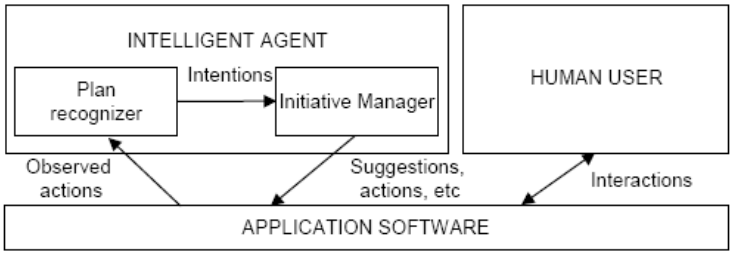

Figure 2: Architecture of a mixed-initiative system with a plan recognition component

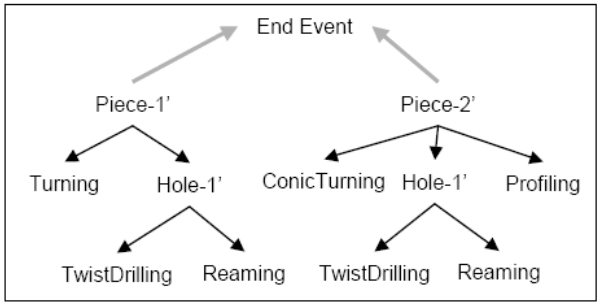

Figure 3: Architecture of a mixed-initiative system with a plan recognition component

and the information in the plan library, the plan recognizer can infer the plan (or plans) the user may be executing.

Basically, there exist two kind of approaches to plan recognition: consistency based and probability based. Consistency based approaches try to narrow the possible plans and goals the user may be pursuing, by isolating those that are consistent with the observed user's actions. Probability based approaches, on the other hand, use an explicit representation of probabilities in order to determine which are the most plausible ones. Once the user's intentions have been identified, the intelligent agent can assist the user in several ways, such as giving him advise, detecting errors or executing actions on his behalf. There are several algorithms and formalisms in the literature that uses consistency based and/or probability based approaches. A complete description of these algorithms and formal models is beyond the scope of this paper (for more details on a specific formal model of consistency based plan recognition and plan library representations, see [9] and [10]).

As an example, figure 3 depicts a simplified plan library for the CAPP domain, based on the formal model proposed in [10]. Under this representation, nodes represent events (machining processes in our domain). Thick gray arrows represents abstractions ("is a" relationships), and thin black ones represent decompositions ("part of" relationships). Consider the case where a user includes *TwistDrilling* and *Reaming* machining processes into the process plan he is building for a piece to be manufactured. In a consistency based approach, a plan recognizer could narrow the possible user's intentions identifying the minimum set of plans that are consistent with the observations. In this example, the plan recognizer can infer that the user is including the machining process of making *Hole-1* as part of his plan, and that he is creating a process plan for *Piece-1* or *Piece-2*. If the user includes the *ConicTurning* process later, the recognizer can infer that he is creating a plan for *Piece-2*, even when *Profiling* has not been yet observed.

2.2.2 Plan Libraries Learning

Until some years ago, plan libraries for plan recognition were completely hand-coded. Creating plan libraries manually is a complex and difficult task. It requires not only a domain expert (in our case, a CAPP expert), but also a knowledge representation expert. As mentioned in [11], a plan library has an external representation (related to conceptual aspects), and and internal one (concerning implementation details). There are several external plan library representation schemes. Here we adopt a formal model similar to the commonly used one proposed in [10].

Under the adopted representation, plan libraries contain a hierarchical tree of goals and plans, that includes not only the primitive observable actions (and their corresponding data), but also every possible goal and plan the user may be trying to execute. As will be seen in later sections, in our CAPP

domain the primitive actions and their associated data are the basic capabilities of a manufacturing company, so we can expect that these actions do not change very frequently. However, there are many combinations of these primitive actions, that represents different manufacturing procedures. So the hand-coding of all allowed possibilities is very expensive in time and is very error prone. Moreover, if the combinations of manufacturing processes change over time, this task must be done several times.

The reasons mentioned before have motivated several research efforts in order to produce algorithms for generating these plan libraries automatically. Basically, the approaches to automatically generate plan libraries, with a representation model similar to that mentioned before (i.e, without learning or storing probability information, but only learning symbolic structures), are supervised or not supervised. Supervised approaches require example labeling in order to work, while unsupervised approaches can learn from unlabeled examples.

There are several works on learning plan libraries (specially supervised approaches), but we mention two relevant ones here to exemplify. In [11], a technique is proposed that generates all possible plans and goals in the domain, and filters out only those plans and goals that do not satisfy some imposed bias. That work uses a formal representation of plan actions, and goal predicates as a knowledge support for generating all possible goals and plans in the domain. The problem with this approach is that it requires the specification of biases for plans and goals, and the generated libraries are sensitive to these biases. Besides, it does not generate hierarchical libraries, and generates all possible plans and goals that satisfy the biases, instead of only considering those that the user actually pursues. Bauer [1] proposes an alternative approach, where plan libraries are generated by abstracting the concrete plans that are observed. The problem with this work is that it requires labeled examples as input to the learning algorithms. Moreover, it uses simple abstraction mechanisms, and these mechanisms require an abstraction hierarchy of concepts in order to produce good results.

The works mentioned (as most of the works in this area), require labeled examples, which implies an explicit supervised training, or uses biased-driven plan library generation. In our domain, we need a more precise approach, in order to generate plan libraries that contain plans that the user actually pursues, and at the same time we need not require the user (a manufacturing engineer) to explicitly train the system. In [12] and [13], we presented an algorithm that learns hierarchical plan libraries similar to that proposed in [10] (although some details of these libraries are not yet learned). Our approach is to use observed cases from a user working on a CAPP application (for example sequences of manufacturing processes that form a process plan), to learn a hierarchical decomposition of plans, similar to that shown in figure 3. Thus, this unsupervised algorithm learns "in background" as the user works on real cases, rather than requiring labeled training sets. The mentioned algorithm also supports the case of interleaved plans in the input cases, and provides corrective mechanisms for some situations where the learning component produces a concept that includes two sub-concepts that are not related to each other. As an example, the plan library shown in figure 3 was learned by our algorithm, from the following two action sequences, each one representing a process plan:

$$AS_0 = \{conicTurning, profiling, twistDrilling, reaming\}$$
$$AS_1 = \{turning, twistDrilling, reaming\}$$

Basically, the algorithm identifies common processes in the observations (also called *events* or *tasks* in the context of plan recognition), and it creates a hierarchical decomposition based on this similarities. In the previous example, the processes *TwistDrilling* and *Reaming* are shared by the two processed process plans, and this yields the plan library shown in figure 3 (see [12] and [13] for more details about the learning algorithm).

```
Action(GrooveMilling(p: Part, m: MachineFamily, t: ToolFamily),
        Precond: MachineType(m, Miller) ^
                 ToolType(t, GrooveMillingTool) ^
                 ¬Grooved(p),
        Eff: Grooved(p) )
```

Figure 4: An example of a manufacturing process represented in the ADL language

2.3 Integrating Plan Recognition and Planning in a Mixed-initiative Approach

Plan recognition is only a component within a mixed-initiative system. Mixed-initiative interactions require that agents not only understand each other, but also contribute with the common plan they share. Thus, for an intelligent assistant, the identification of user's intentions and plans is useful to guide an active participation in the solution generation procedure.

2.3.1 Managing Initiative

In order to give the agent initiative capabilities, we propose the use of a planning algorithm. Several kind of planners could be used, such as reactive planners (e.g. PRS), or generative planners (e.g. UC-POP, Graphplan, etc). PRS [3, 14, 18] is a reactive planner specially intended for real-time environments. It uses a kind of plan library (KA library for some versions and ACT library for other ones) that contains information regarding goals and plans for these goals, from which the planner takes procedures to achieve its goals (posted by the user or generated by world events). The contents of plan libraries used by PRS are similar to that included in some plan libraries for plan recognition. However they are intended for generating (or selecting) plans to achieve goals, not for infer goals from observed plans, and therefore some aspects are not taken into account (such as abstractions, probabilities, etc).

Generative planning algorithms on the other hand, can take an initial state description, a goal to achieve, and a set of operators (or actions), and synthesize a solution in the form of a plan that, when executed in the initial state, can achieve the requested goals. Generative planners use primitive actions as building blocks in order to produce a solution, rather than search for a known solution stored in the plan library. Reactive planners (such as PRS) could be adapted in order to share a plan library with the plan recognizer component. However this approach would only allow the planner to complete the inferred user's plan, and not to synthesize other alternatives. Even when our approach does not prescribe a specific planning algorithm, we propose the use of a generative planner since it is more flexible for solutions generation, thus giving the agent better assistance capabilities.

2.3.2 Plan Recognizer-Planner Interface

One important aspect to be addressed is that of the integration of the plan recognizer and the planner. The output of the plan recognizer must be useful for the planner, in order for it to produce a new solution. We propose the adoption of a commonly used language, such as ADL [15] (Actions Description Language), for the description of actions for the planner, the plan recognizer and the plan library learning components, though our approach can accommodate different planning languages (such as STRIPS, ACT, etc). Figure 4 shows an example of a manufacturing process represented in ADL.

Thus, we propose that the plan recognition component, the plan library learner and the planning system share a common set of information (primitive actions, their parameters, preconditions, effects, and some descriptions of certain type of resources, such as raw materials, machines, etc). Using

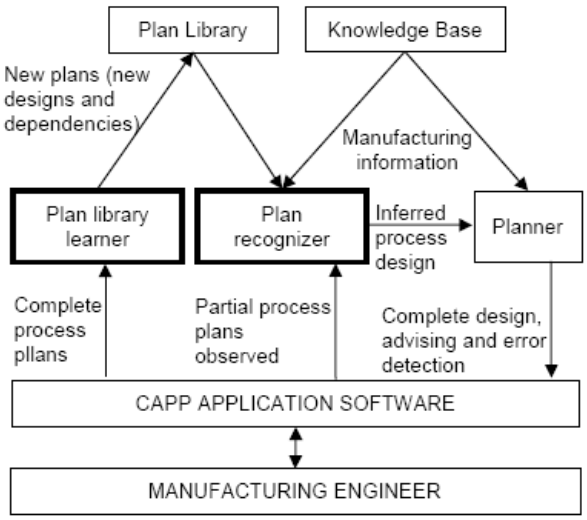

Figure 5: The proposed mixed-initiative CAPP architecture

this information as a shared resource, all components in the proposed mixed-initiative system can communicate with each other.

Figure 5 shows the complete architecture. We now point out some important details of this approach. As proposed in [12], if a plan library learner knows not only the name (or some kind of name or identification) of the primitive actions that could appear in observed cases, but also has detailed knowledge of these actions, their parameters, etc (i.e, information that is already needed for a planning component, as mentioned before, and that was exemplified in figure 4), it can not only generate a hierarchical decomposition of actions, but also it can learn more deep and useful information, such as abstractions of actions and alternative decompositions. Abstractions of actions and alternative decompositions have not been successfully addressed in previous works on plan libraries learning for plan recognition, and has almost not been even considered. Thus, the addition of knowledge representation to the primitive actions that are observable can significantly increase the knowledge that can be learned. Besides, this knowledge representation is already needed if we assume that a planner will be part of the mixed-initiative system.

Another important consideration, is that not only the learning process is enriched with the addition of knowledge representation of actions. If primitive actions have a knowledge-rich description (including preconditions, effects and parameters), the output of the plan recognizer can be improved, in the sense that the plan recognizer is not limited to just inform which of a set of known goals the user is trying to achieve, but it can generate an open logic representation of that goal.

The decomposition of a non-primitive action (also called a non-primitive event), is a set of simpler actions that must be done in order to achieve it. Thus, the actions that constitutes the decomposition of another one have an implicit conjunction relationship between each other. For example, the decomposition of *Hole-1* in figure 3, is the set of actions composed by *TwistDrilling* and *Reaming*, so these two actions have and implicit conjunction relationship in order to achieve *Hole-1* (i.e, both actions must be done in order to consider *Hole-1* as done). Now suppose that we have an open logic representation of primitive actions (in this case, *TwistDrilling* and *Reaming*), such that shown in figure 4. In this situation, when a set of actions is observed during the recognition process, as well as during the learning process, the system can compute the preconditions and effects of the non-primitive actions recognized, as the conjunction of the preconditions and effects of their individual components

observed.

Formally, let A_i be the i-th part of a non-primitive event A, and let P_i and E_i be the preconditions and effects of A_i. Let P and E be the preconditions and effects of the non-primitive event A. Then, since the components A_i have a conjunction relationship within A, P is the conjunction of the preconditions P_i, and similarly E is the conjunction of the effects E_i:

$$P \Longleftrightarrow P_1 \wedge P_2 \wedge \cdots \wedge P_i$$
$$E \Longleftrightarrow E_1 \wedge E_2 \wedge \cdots \wedge E_i$$

For example, the system can infer the preconditions and effects of *Hole-1* (which is not directly observable, and whose knowledge representation is not hand-coded), as the conjunction of the preconditions and effects of *TwistDrilling* and *Reaming* (which are primitive actions whose knowledge representation is assumed to be defined). For the plan library learner this means that, even when it can not assign a significative name to the new abstract concepts it learns after each observation is processed, it can know which preconditions and effects are associated with it. Besides, a similar approach can be taken with primitive actions' parameters (they can be propagated up in the tree hierarchy). During the recognition process, the plan recognizer can also take advantage of this information in order to improve the recognition process.

Additionally, once the plan recognizer has identified the user goal (or possible goals), it can build a logic representation of this goal (using the same idea mentioned before). The approach usually taken for the intelligent assistant's initiative, is to complete the rest of the plan the user is working on using only the information of the plan library, since in most cases the events included in the plan library do not have a knowledge-rich representation.

We propose to use generative capabilities for completing the plan and/or synthesizing other alternatives, using the knowledge-rich representation of the user's goals as input to the planner. A planner could then generate alternative solutions (plans) that achieve the same goals as the plan (perhaps partially) built by the user. The advantage of this approach is that the planner could generate plans optimizing different criteria, and could suggest to the user some modifications (i.e, it can give advice to the user), including giving him explanations about the planning decisions. Thus, the system does not have to restrict its actions to what it knows about the possible plans in the plan library (i.e, the solutions the user has already created in the past), but it has now the capability of building new solutions based on the knowledge it can extract from the user behavior.

As can be seen, the proposed mixed-initiative approach to CAPP can give the generative advantages of a planning system (or a knowledge based system), with the possibility to allow the user to contribute to the decision process with his experience.

3 EXECUTION MODEL

In this section we explain the execution model of the proposed approach. We assume that the basic knowledge has already been represented in the system (primitive actions with parameters, preconditions and effects, as shown in figure 4, and representation of the necessary resources, such as machine families, raw materials, etc).

At the beginning, the system has an empty plan library, so it can not recognize users intentions during the process plan elaboration. As the manufacturing engineer uses the system, the plan library learning component updates the plan library with new information extracted from observed action sequences. The action sequences processed by the plan library learner corresponds to complete process plans, so after the user completes each plan this component processes it.

As the plan library evolves, the plan recognition module is able to infer user's intentions more and more precisely. When the plan recognizer identifies the goal or goals the user may be pursuing, it can use this information to build a formal representation of these goals, based on the information about preconditions and effects of the recognized goals and plans, as was exposed in section 2.3. The plan recognizer can use a consistency based approach, as well as a probability based approach (our proposal does not restrict the system to use one kind of plan recognizer or another).

Once the plan recognizer has identified user's intentions and goals, and has built a knowledge representation of these intentions, it can provide this information to an automated planner. Thus, the planner can generate alternative solutions to assist the user in diverse forms. Is important to note that the plan recognition process as well as the planning process, can be performed even when the user has not finished the creation of the plan. Even in the situation where the process plan the user is working on is a completely new one, the plan recognizer can identify sub-plans already known and assist the user on them.

The agent can give assistance to the user in diverse forms. Once the plan recognizer has identified the user's intentions, the planner can generate alternative solutions and use them as suggestions to the user. Another useful assistance is the execution of tasks on the user's behalf, for example automatically completing the process plan the engineer is working on. This particular assistance is very useful, as it allows the user to actively participate during the planning process while can significantly reduce the workload of this task (this situation could be seen as some sort of mixed-initiative planning). Finally, the system can suggest the user specific modifications for improving the already created process plan portions, or to correct potential errors. The type of assistance each user needs or prefers on each situation is beyond the scope of this research (see for example [16]).

Consider an example. At the beginning, the system's plan library is empty. After the user creates the following simple process plans:

$$AS_0 = \{conicTurning, profiling, twistDrilling, reaming\}$$
$$AS_1 = \{turning, twistDrilling, reaming\}$$

the plan library state is that depicted in figure 3. When the user works on a new process plan, the plan recognition component will try to identify his goals, and if it can, it will build an open representation of that goal (i.e, instead of restricting the recognition's output to, for example, *Hole-1*, the plan recognizer can output something like $MakeHole(H) \wedge FinishHole(H, FinishType1)$). With this knowledge, a planner can synthesize other process plans that achieve the same goals, and can propose this alternatives to the user. Alternatively, the processes already included by the user in the process plan could be used as restrictions during the planning process.

The execution of the learning algorithm is performed after the process plan is completed, so the learning component processes only complete action sequences. The plan recognition and the planning components instead, are executed "in background" during the process plan creation. Thus, the system can infer user's intentions, and proactively act in order to reduce his workload and errors. At the same time, the system adjust its own knowledge during execution, so new plans are learned as the user starts considering them, thus eliminating the restriction of having a complete plan library before the use of the system.

4 RELATED WORK

Some existing works in CAPP address some of the problems mentioned in section 1, by proposing interactive methods supported by machine learning techniques in order to achieve interactive intelligent

assistance, reducing the need of knowledge representation experts (see [5], [6] and [7]). One of the contributions of these papers is the identification of problems when implementing CAPP systems that over or under-automate the process planning activities. They also propose the use of CAPP systems as communication tools between experts that must work together (for example, experts on machines and machining processes, and experts on process planning), or to help inexperienced manufacturing engineers. In the mentioned works, the focus is set on the analysis of the problems that arises when human's intentions need to be stored on an internal representation. The authors point out that one of the biggest problems in implementing successfully CAPP systems is the complexity of the communication between the users and the problem solving procedures. They also allow many types of resources to be saved in the knowledge base (multimedia resources, plain text, formally represented knowledge, etc), so even when these works propose novel ideas from the conceptual point of view, the specific proposals lacks of a solid intelligent support. The above mentioned works are focused on the architectural and conceptual ideas, and not on algorithms and data structures, so they does not include information for implementation.

In [2] a multi-agent architecture is proposed for CAPP systems. As in those already mentioned, the focus of this work is on the architecture, and not on specific techniques and algorithms. An interesting contribution is a comparison between the amount of automation level achieved so far in CAD/CAM tools and the automation of CAPP applications.

In [17], an expert system approach to generative CAPP is proposed. The authors propose a system where manufacturing processes, machines and tools are formally represented, as well as a set of rules for generating process plans based on an input feature model. This approach has some drawbacks, such as the big amount of knowledge that must be represented (it does not use a planning approach to generate process plans, but a rule-based approach), it lacks of learning and it requires a complete feature model of the parts to be manufactured.

5 CONCLUSIONS AND FUTURE WORK

CAPP is a knowledge rich domain in which intelligent assistance would be very useful. A description of the known problems of current CAPP systems was exposed in this paper. We also proposed a mixed-initiative approach to CAPP domains, that aims at solve these problems.

At this moment we have an implementation of the learning algorithms we proposed in [12] and [13]. We are currently doing some work in order to test them with real data. In order to achieve the mixed-initiative system we proposed here, future work must not only validate the mentioned learning algorithms, but also extend them in order to learn complete plan libraries for plan recognition. Besides, special plan recognition and planning algorithms must be developed and integrated with the plan library learner in order to test the complete system in a real CAPP application.

REFERENCES

[1] M. Bauer. Towards the Automatic acquisition of Plan Libraries. In Proc. of the 13th European Conference on Artificial Intelligence. 1998.

[2] P. Dépincé, H. Amara and J-Y. Hascoët. Multi-Agent environment for process planning elaboration. In Proc. 8th International Conference on Emerging Technologies and Factory Automation. 2001.

[3] M.P. Georgeff, F.F. Ingrand. Decision-Making in an embedded reasoning system. 11th International Joint Conference On Artificial Intelligence. 1989.

[4] M.A. Hearst. Trends & Controversies: Mixed-initiative interaction. IEEE Intelligent Systems. Vol 14, N 5:14-23. 1999.

[5] L. Horváth and I. J. Rudas. Human Computer Interactions at Decisión Making and Knowledge Acquisition in Computer Aided Process Planning Systems. In Proc. International Conference on Systems, Man and Cybernetics. 1994.

[6] L. Horváth and I. J. Rudas. A Machine Learning Based Approach to Manufacturing Process Planning. In Proc. International Symposium on Industrial Electronics, ISIE'93. 1993.

[7] L. Horváth and I. J. Rudas. Modeling Human-Computer Interactions in Collaborative Design and Planning. International Conference on Systems, Man and Cybernetics. 'Intelligent Systems for the 21st century'. 1995.

[8] S. Kalpakjian and S. R. Schmid. Manufacturing Engineering and Technology. Addison-Wesley. 4th ed. 2002

[9] H. Kautz. A Formal Theory of Plan Recognition and its Implementation. In Reasoning About Plans, chapter 2, Morgan Kaufmann. 1991.

[10] H. Kautz. A Formal Theory of Plan Recognition. PhD Thesis. University of Rochester. 1987.

[11] N. Lesh. Scalable and Adaptive Goal Recognition. PhD thesis. University of Washington. 1998.

[12] M. Marchetta and R. Forradellas. A New Model for Automatic Generation of Plan Libraries for Plan Recognition. Proc. Third International Conference On Production Research Americas' Region (ICPR-AM06). 2006.

[13] M. Marchetta and R. Forradellas. Supporting Interleaved Plans in Learning Hierarchical Plan Libraries for Plan Recognition. Proc. 8th Argentine Symposium on Artificial Intelligence. 2006.

[14] K.L. Myers. User Guide for the Procedural Reasoning System. Technical Report, Artificial Intelligence Center, SRI International. 1997.

[15] E.P.D. Pednault. Formulating multi-agent, dynamic-world problems in the classical planning framework. En Proc. Workshop on Reasoning About Actions and Plans. 1986.

[16] S.N. Schiaffino. Personalization of User Interface Agent Interaction. PhD Thesis. Universidad Nacional del Centro de la Pcia. de Bs. As. 2004.

[17] D. Sormaz and B. Khoshnevis. Knowledge Representation for Automated Process Planning. In Proc. International Symposium on Assembly and Task Planning. 1995.

[18] D. Wilkins, K.L. Myers, J.D. Lowrance, L. Wesley. Planning and Reacting in Uncertain and Dynamic Environments. Journal of Experimental and Theoretical Artificial Intelligence. volumne 6, 1994.