A cooperative agent modelling approach for process planning

F.L. Zhao, S.K. Tso *, Paul S.Y. Wu

Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Tat Chee Avenue, Hong Kong, China

Received 30 November 1998; accepted 25 January 1999

Abstract

A well designed computer-aided process planning (CAPP) system bridges the gap between CAD and CAM. A number of systems have recently been developed relying on a standalone expert system. However, because of over-complexity, many such systems cannot be effectively applied to industrial enterprises in practice. Moreover, the modern computer integrated manufacturing system (CIM) requires the CAPP system to be extendible and flexible for practical industrial applications. It is hardly possible to develop the extensive industrial CAPP system by using only one large expert system. To overcome these weaknesses, a new cooperative agent model is presented for process planning in this paper that satisfies five major requirements: Autonomy, Flexibility, Interoperability, Modularity and Scalability. In accordance with this framework proposed, a machining cooperative process planning system (Machining CoCAPP) is specifically developed for demonstration purpose. The system modelling, agent structure design, cooperation and coordination mechanism, and case study of the Machining CoCAPP are presented. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: CAPP; Cooperative agent; Modelling; CIM

1. Introduction

Process planning provides information to the shopfloor on how to produce the designed products. It addresses each part of the product separately and collectively. It defines the process, cost and production lead time under the constraints such as the designed geometry, material, quantity, machine and tooling availability, labour capacity and suitability, etc. In the past, process plans were often generated by human process planners who had plenty of manufacturing domain knowledge and worthy experience. In the recent decades, computer technologies have stimulated the advance toward computer-aided process planning (CAPP).

Generally, there are two CAPP approaches: variant and generative. The variant approach is a data retrieval and editing method. Some standard or mature process plans are collected based on the group technology and stored in a database. When a new part is required to be produced, a similar process plan is retrieved from the database and edited to adjust it to suit the new part. The generative approach is a knowledge-based method which automat-
ically generates a process plan according to the part’s features and manufacturing requirements.

The success of the variant approach depends on the group technology, planner’s experience and a sufficient collection of standard or mature process plans. This method is especially suitable for companies with few product families and a large number of parts per family. Most earlier CAPP tools can be categorized under the variant process planning approach [1]. Typical examples are CAPP [2], MIPLAN [3], etc. The generative process planning approach has attracted more attention in recent years. It offers a potential of producing an optimal plan. Typical examples are APPAS [2] EXCAP [4], KRONOS [5], XCUT [6], QTC (Quick turnaround cell) [2], PART [7], OOPPS (object-oriented process planning system) [8], MePlans [9], COMPLAN Process Planner (CPP) [10], etc. Generative process planning systems are mostly oriented towards the needs of large companies and research organizations, especially those which have a number of products in small lot sizes. However, there is still difficulty in developing a truly generative process planning system which can meet industrial needs and provide an appropriate and compatible generic framework, knowledge representation method, and inference mechanism.

Cooperative agent systems attempt to distribute activities to multiple specialized problem solvers and to coordinate them to solve complex problems [11–14]. A cooperative agent system consists of many individual agents with cooperation engines. Each agent which has its own knowledge base and inference engine is responsible for a specific task. It provides a cooperation interface to communicate with other agents in the cooperative environment. A different language and different knowledge representation may be employed by each agent which may well be located in a different machine. Such a system organization provides an integration environment of heterogeneous and scalable architecture suitable to solving different problems.

2. Process planning problem

A machining process generally involves many machine tools, operations, fixtures, and cutting tools. Its planning requires knowledge from diversified fields. Generally, a machining process planning includes the following parts:

- feature recognition;
- machining operation selection;
- machine selection;
- cutting tool selection;
- fixture selection and design;
- sequencing operation and cost estimate.

The feature recognition part identifies manufacturing features from the product design data. The machining operation selection part selects the relevant machining operation according to the feature characteristics and the manufacturing environment. The required machine equipment is selected for implementing the selected operations after considering the nature of the parts and the machine processing capabilities such as the working volume, accuracy, power, fixturing, and other functions. The fixture selection part chooses the fixtures according to the part geometric shapes and dimensions as well as manufacturing features. The main concerns of the cutting tool selection include the tool types, materials, shapes, and tool dimensions.

3. The proposed method

From the technological viewpoint, CAPP is still a very complex and difficult problem. Many research efforts have focused on CAPP system development, using different methodologies and strategies. However, most systems are developed by using standalone expert systems. Due to the complexity of CAPP, such a system structure is hardly able to solve the problems normally found in the manufacturing industry.

A cooperative CAPP framework is proposed to reduce the limitation of currently available CAPP systems. In particular, it highlights the requirements that a modern CAPP system should meet in order to facilitate practical development: flexibility, modularity, interoperability, autonomy, and scalability [15,16].

- Autonomy means that the CAPP system is developed as an independent system. Once developed, it can readily be integrated into the CAD/CAM
system. Each agent is also treated as an independent and autonomous system.

- **Flexibility** permits new technologies and new methods to be easily added into the CAPP system.
- **Interoperability** permits multiple heterogeneous machines or approaches to work smoothly together in solving problems.
- **Modularity** enables the CAPP to function as an integration of multiple subsystems.
- **Scalability** offers the ability to scale the CAPP system architecture according to the user’s transaction requirements.

The design of the CoCAPP framework is discussed with relation to a machining process planning (Machining CoCAPP). Its structure, implementation, and its application case study are detailed in this paper.

### 3.1. Overview of Machining CoCAPP

The overall structure of the Machining CoCAPP system in the integration environment of CAD/CAPP/CAM is shown in Fig. 1. The following six process planning agents (P-agents) are in general included in the implementation:

- feature recognition agent;
- machining operation selection agent;
- machine selection agent;
- cutting tool selection agent;
- fixture selection and design agent;
- sequencing operation and cost estimate agent.

Each agent may run in a different computer connected to the internet. The B-agent supplies the global state information of the problem and monitors the operational dependencies among all the individual P-agents. Once there is a problem, the B-agent will notify all registered P-agents for actions. The results from the P-agents will be posted on the B-agent. The B-agent maintains the consistency and integrity of the decision space within the given constraints. The product design data are sent to the CoCAPP system through the D-agent from the computer-aided design system. The CoCAPP system gets

![Fig. 1. The structure of the Machining CoCAPP system.](image-url)
the production constraints from the scheduling/shopfloor system. The CoCAPP system generates the process plans according to the product design data and production constraints. If no acceptable results can be obtained, the CoCAPP system will feed the conflict information back to the design department or other relevant departments. The feasible process plan alternatives will eventually be transmitted to the scheduling/shopfloor system for scheduling.

3.2. Agent infrastructure

The general structure of process planning agents (P-agents) is shown in Fig. 2. The agent is composed of four parts: agent controller, inference engine, functional adapters, and application libraries. The configuration file is used to construct the agent. The rules and facts form the application libraries. The agent controller, rule-based engine, internet adapter, file adapter, keyboard adapter, information view adapter, and schedule adapter are the commonly used components for all the P-agents. Other adapters are also shown in Fig. 2.

The solver adapter is a very important adapter in the P-agents because it is used to accomplish proposal generation, conflict resolution and proposal evaluation of the process planning. In order to utilize the knowledge of each domain in the combination of production rules and objects, the adapter is differently implemented for different agents. It must be specifically designed to deal with the knowledge of object-oriented description.

The database adapter is used to store data useful to the B-agent and P-agents, such as problem definitions, proposals, conflicts, evaluations, solutions, etc.

Because KQML [17] is the most commonly used language for communication among agent-based programs, particularly when they are autonomous and asynchronous, the CoCAPP system has chosen a KQML-based communication protocol as a communication language used by each agent. The NetKQML adapter is used to communicate with the B-agent.
According to the requirements of KQML transport, agents are connected by unidirectional communication links that carry discrete messages. These links may have a finite message transport delay associated with them. When an agent receives a message, it knows from which incoming link the message has arrived; when an agent sends a message, it directs it to the intended outgoing link. Messages to a single destination arrive in the order they were sent; message delivery is reliable. The socket concept of Windows 95 is used to implement the NetKQML adapter. The TCP/IP is a protocol for socket communication transport [18].

The information view adapter is used to display information on the monitor.

The time adapter is used to count time in the course of process planning.

The schedule adapter is responsible for scheduling events such as proposal generation, proposal evaluation, and conflict resolution. When an incoming proposal from another agent is detected, the adapter schedules an evaluation event. When a process planning problem is detected, it then schedules a proposal generation event. Both kinds of events will be assigned a priority at the same time of event generation. If a conflict resolution is required, then the adapter schedules a conflict resolution event and assigns it the highest priority. Whenever a negotiation is required, the adapter immediately suspends other events and fires a negotiation effector.

When a problem arrives, the P-agent first invokes the effector ‘WatchProblem’ to examine the problem. Afterwards, it creates a ‘problem examined’ event. At the same time, it generates the facts of the problem. Responding to the ‘proposal generation task’ event, the effector ‘GenerateProposal’ generates a proposal and creates a proposal ‘Generated-Event’. This event results in the facts of the proposal.

Responding to the task event created by the schedule adapter, when the task name is ‘proposal generation’, the incoming proposal is evaluated. The adapter checks if the adapter has already generated an interacting proposal. If yes, it links the two proposals and notifies the proposal originators of its intent to evaluate the new proposal. In this case, it evaluates the already generated proposal. If the proposal adapter has not yet generated an interacting proposal, the adapter searches the scheduled pending tasks for related generation tasks. It checks to see if it has already started working on a proposal or is planning to start. If yes, it links the evaluation and generation tasks. In this case, it evaluates the proposal with the triggering proposal generation. If the triggering proposal is acceptable, it may not be necessary to generate a separate proposal or a proposal can be generated which is tailored to integrate smoothly with the triggering proposal. If there are no current plans to work on a related proposal, the evaluation task only evaluates the incoming proposal according to agent knowledge. The P-agent which originated the triggering proposal is then notified.

If a conflict is notified from another agent, the conflict resolution responds to this event to resolve the conflict. After the adapter responds to the conflict event, it first judges the conflict situation. Then it invokes the relevant strategy to resolve the conflict.

3.3. Knowledge representation

The CoCAPP system’s knowledge is about product representation and agent knowledge. Each P-agent in the CoCAPP system has three types of knowledge: domain knowledge, control knowledge, and conflict resolution knowledge. The B-agent has only control knowledge. The part representation is about the problem description. It is commonly shared by all agents within the CoCAPP system.

3.4. Part representation

A part in the Machining CoCAPP system consists of two kinds of data. One kind is about the constraints to the generating plan. Another kind is about its geometric information. They are described as follows:

- CONSTRAINT (production time, cutting force, machining power);
- PART (Name; Type; MaxSize; Material; InitStatus; HeatTreat; Features).

The Name is the identifier of a part. It must be unique. The Type is the keyword of the part to show its outlook shape. The MaxSize gives the maximum envelope size of the part. The Material is the material kind of the part. The InitStatus is the raw status
of workpiece with which the part can be fabricated. The HeatTreat is about the part heat-treatment condition. The Features are a collection of all features. A feature is represented as follows:

- **FEATURE (Name; Type; Location; FinalSize; InitSize; Hardness; Tolerances)**

  The Name is the identifier of a feature. It must be unique. The Type is the keyword of the feature. The feature Location is about the original position and directional vector of the feature. The feature size, including final size (FinalSize) and initial size (InitSize), is the dimensional value of the feature. The Hardness is about the feature’s hardness. The Tolerances are about dimensional and geometric tolerances. A keyword is assigned to tolerance to distinguish the tolerance content.

### 3.5. Domain knowledge

The domain knowledge of each P-agent is about the descriptions of its process planning capabilities, and used to generate proposals, evaluate proposals, and resolve conflicts. Different agents may have different formats for domain knowledge representations such as databases, analytical algorithms, etc. The domain knowledge can be extracted from manufacturing handbooks such as [19–21]. Each P-agent has a different domain knowledge content.

For example, the operation selection agent is used to generate machining operation alternatives for defined features of parts. For each given feature, there may exist more than one possible operation. Some traditional machining methods such as forging, die casting, drilling, turning, milling, boring, shaping, grinding, lapping, honing, and diamond boring, etc., have been built into the domain knowledge base. Its content includes the relationship between operations with parameters such as machinable feature, workpiece material, tolerances, preparatory operation, time calculation equation. The knowledge is represented as facts in a semantic net as shown in Fig. 3. In the knowledge base, an operation fact is stored in one node. All facts are stored in a list.

The proposal generation, proposal evaluation, and conflict resolution strategies are embedded in the solver adapter of each P-agent.

### 3.6. Conflict resolution knowledge

The conflict resolution strategies include two categories: domain-dependent and domain-independent. The domain-dependent strategies mainly involve how to suggest further measures when one P-agent conflicts with the other P-agent’s proposals. Each P-agent has its own suggestion strategies different from the other P-agent’s. It also includes the explanation to the conflict resolution proposed. The domain-independent strategies are more common and can be the same for all the P-agents. They are designed as a set of conflict resolution facts with some basic guidelines (conflict resolution rules) for deciding which domain-dependent strategy to apply and altering the resolution strategy in order to improve its understanding of the overall problem according to the other P-agent’s action of conflict resolutions. The following strategies are used to resolve conflicts in the CoCAPP system as domain-independent conflict resolution strategies [22].

- **Compromise**: finding an immediate proposal that is within an acceptable range;
- **GenerateConstrainedAlternatives**: generating new alternatives based on constraints that are received.

---

Fig. 3. Semantic net for internal round features.
from an inflexible agent or based on some other agent’s partial solution;
• CasebasedParameterSetRetrieval: finding a previous solution that succeeded in resolving a conflict involving a similar set of parameters.

3.7. Control knowledge

All the P-agents in the CoCAPP system must be able to perform a set of tasks with the same control knowledge:
• generate new proposals;
• evaluate proposals;
• negotiate;
• resolve conflicts.

The above tasks are scheduled before they are performed. Each task is scheduled as an event. The P-agents first respond to an event, then perform the relevant tasks. The control knowledge of the P-agents is used to schedule tasks and to respond to events.

Each task is assigned a priority as an event. As previously discussed, the negotiation is the highest priority task. The conflict resolution shares the second highest priority. The proposal generation has the lowest priority. The scheduled tasks are performed asynchronously. The event is responded to according to the principle of ‘first come, first served’.

3.8. Planning strategy

The CoCAPP system solves the process planning problem, using coordination and cooperation between the P-agents through a commonly shared language with the expectation about how to reach agreements when conflicts occur. Therefore, the system is designed to support cooperative problem solving by providing a communication and conflict resolution structure and the protocol required for cooperative interaction.

3.8.1. Definition

According to Ref. [23], the problem, proposal, evaluation, and conflict are defined as follows: Suppose that a problem is represented as \( P = \{ Or,G,C,I \} \), where \( Or \) denotes the originator of the problem, \( G \) is the set of goals of process planning problem, \( C \) is the constraints for this problem, and \( I \) contains the initial information such as part design data. A proposal is in the form of \( Q = \{ Ow_P, Ac_P, Exp_Q, Cf \} \), where \( Ow_P \) denotes the owner of the proposal, \( Ac_P \) is a set of actions to solve the given process planning problem, \( Exp_Q \) is the explanation to the proposal, and \( Cf \) is the confidence factor for the proposal. An evaluation result can be represented by \( Ev = \{ Ow_Ev, Id_Q, Ac_Ev, Ra, Re \} \), where \( Ow_Ev \) denotes the owner of the evaluation, \( Id_Q \) is the identity of the proposal \( Q \), \( Ac_Ev \) is the set of evaluation actions for the proposal, \( Ra \) is the set of ratings for each action, and \( Re \) is the overall result for the proposal which is either disagreement or agreement. When \( Re \) says disagreement, a conflict appears, expressed in the form \( Cr = \{ Ow_Cr, Id_P, Ac_Cr, Exp_Cr \} \), where \( Ow_Cr \) denotes the owner of the conflict, \( Id_P \) is the identity of the related proposal, \( Ac_Cr \) is the set of conflict actions, and \( Exp_Cr \) is the explanation for the conflict actions.

The problem solving is initiated by the D-agent accepting a problem and putting the problem definition \( P = \{ Or,G,C,I \} \) into the problem area of the B-agent. All registered P-agents are notified of the problem information. The interested P-agents will examine the problem definition and start producing planning proposals related to their expertise, knowledge and viewpoints. When a P-agent generates a proposal \( Q = \{ Ow_P, Ac_P, Exp_Q, Cf \} \), it is put into the proposal area of the B-agent. The proposal is posted to other registered P-agents. A P-agent is not interrupted if it is already working on another proposal, but immediately triggers an evaluation thread. This thread first determines whether it is going to criticize the proposal. If not, it will go to sleep and wait for another proposal to be asserted. If the P-agent is interested in the proposal, it then evaluates the proposal and posts the evaluation result \( Ev = \{ Ow_Ev, Id_Q, Ac_Ev, Ra, Re \} \) into the evaluation area of the B-agent. If there is a conflict, the result \( Cr = \{ Ow_Cr, Id_P, Ac_Cr, Exp_Cr \} \) will be examined in order to obtain the final evaluation result.

After all the interested P-agents finish evaluating the newly asserted proposal, those P-agents which identify the proposal under consideration as conflict work together to resolve the conflict \( Cr = \{ Ow_Cr, Id_P, Ac_Cr, Exp_Cr \} \). The result of the conflict resolution is either a revision or an abandon-
ment of the proposal scheme. When none of the interested P-agents detect any conflict related to the proposal, the partial planning template residing in the solution area is updated. The planning process continues until the planning template meets the requirements, including planning goals and constraints.

3.8.2. Coordination

The cooperative problem solving is carried out among the B-agent and P-agents. The B-agent is responsible for storing and announcing the public information involved in each process planning problem. It is partitioned into four areas for four distinct groups of information: problem, proposal, evaluation and solution. The problem area contains the initial problem definition and the overall requirements of the process planning problem. The proposal area stores partial and complete proposals at several layers of abstraction issued by the P-agents. A proposal from a P-agent might be evaluated by other P-agents. If there is any inaccurate or incomplete process in the proposal, other P-agents can put their critiques related with the proposal to the evaluation area of the B-agent. The evaluation area stores the conflicts that occur during the process planning, and provides a means of communication among the P-agents who are involved in the conflict. The evaluation results and conflict resolution recommendations issued by the P-agents are also recorded in the evaluation area. The solution area includes the evolving process planning template to which non-conflicting process commitments produced by the P-agents are added. The final solution is recorded into the solution area of the B-agent. The organization of the B-agent is shown in Fig. 4. The inference engine provides the control of information flow among the four areas. The knowledge base contains the event scheduling knowledge for cooperative problem solving and the justifications of the solutions to the problems.

The B-agent monitors the data of the four areas. Once a proposal is received from a P-agent, it will be placed in the proposal area. At the same time, the P-agent will check if the number of proposals for the same problem is greater than a fixed value. If yes, the B-agent will choose the best proposal from the proposal list in the proposal area as a possible solution. The process planning for the problem will be
terminated. If not, the proposal will be posted to other registered P-agents for their review. Once an evaluation is received from a P-agent, the B-agent will check if the evaluation has a conflict result. If there is any conflict, the owner of the proposal will be notified. The B-agent will coordinate these P-agents having conflict to settle by negotiation. If there is no conflict, the B-agent will check if all the registered P-agents have agreed with the proposal. If yes, the proposal will be evolved into a solution to the problem.

3.8.3. Cooperation

As mentioned previously, the CoCAPP system environment is organized as a community of cooperative problem-solving agents, where each P-agent is a relatively independent and autonomous knowledge-based expert system. The P-agent solves problems in its limited domain independently. Therefore, it should have the capabilities to act as a member of a community. These capabilities include:

- a shared communication language with other agents;
- internal knowledge representations which capture sufficient goal and history information to allow for solution revision to be carried out cooperatively;
- provisions for sharing information in a timely manner for problem-solving;
- mechanisms for incorporating externally produced partial solutions;
- mechanisms for negotiation to settle conflicts;
- the ability to coordinate an internal agenda with external events.

Each P-agent communicates with other agents by using a common shared language. The proposal generation and evaluation, solution, and conflict generation and resolution are produced according to the internal domain knowledge of the P-agent.

4. Software implementation

The IBM ABE Toolkit [24] is chosen as the development environment of the CoCAPP system. The Visual C++ language is chosen as the implementation language. The system can run in the platform of Windows 95 or Windows NT.

![Diagram of Conflict resolution of P-agents.](image)
As mentioned, each P-agent in the CoCAPP system makes use of three types of knowledge: domain knowledge, control knowledge, and conflict resolution knowledge. In order to enhance the flexibility and scalability, the domain knowledge is further classified into universal-level, shop-level, and machine-level knowledge. The universal-level knowledge is applicable to any status without considering individual companies and is established when the CoCAPP system is in development and is often fixed after the system has been constructed. The shop-level and machine-level knowledge can be added and modified by individual companies when the system is scaled. The machine-level knowledge is only applicable to a specific machine. The knowledge of each agent is implemented as a database or file.

The conflict resolution handler is shown in Fig. 5. It consists of two effectors: ‘WatchConflict’ and ‘ResolveConflict’, as well as one conflict resolution facts file and one conflict resolution rules file. The ‘WatchConflict’ is used to map out the conflict problem space and conflict situation. The ‘ResolveConflict’ is used to resolve conflicts. The conflict resolution strategies are implemented in this effector.

5. Case study

The case study is used to illustrate the characteristics of the Machining CoCAPP system. The first example demonstrates the feedback due to an unreasonable product design. In this example, as an unreasonable part design is provided, the CoCAPP system generates an ‘unresolved conflict’ output and reports to the D-agent the locations and causes of the conflict. The part is shown in Fig. 6, a bar with an envelope size of $50 \times 40 \times 30$ mm$^3$.

5.1. Initial Part Design Data

timeless:7200;
PartOne;
Rectangular;
50:40:30;
MildSteel;
Bar;
;
Face1,FlatFace, 0:0:0:90:0:0, 50:40, 50:40, DimTol:0.04;
Face2,FlatFace, 0:50:0:90:0, 40:30, 40:30, DimTol:0.02;
Face3,FlatFace, −30:0:0:−90:0:0, 50:40, 50:40, DimTol:0.02;
Face4,FlatFace, 0:0:0:−90:0, 40:30, 40:30, DimTol:0.04;
Face5,FlatFace, 0:0:0:0:0:90, 50:30, 50:30, DimTol:0.03;
Face6,FlatFace, 0:0:40:0:0:−90, 50:30, 50:30, DimTol:0.03;
Slot1,ThroughSqSlot, 0:20:25:0:90:90, 20:30:15, 0:0:0, DimTol:0.02;
Slot2,ThroughSqSlot, −5:50:25:90:0:90, 20:15:10, 0:0:0, DimTol:0.02;
Hole, ThroughHole, −15:10:0:90:90:0, 10:40, 0:0, DimTol:0.01;

5.2. Planning result based on initial design data

== = UNRESOLVED CONFLICT == ==
Conflict1Op;
Op-agent;
ProposalCOp;
Slot2:operation::,notavailable;
Slot2:operation::,notavailable
== == AND CONFLICT == ==
Conflict0Cutter
== == CONFLICT IS == ==
Conflict0Cutter;
Cutter-agent;
Proposal100Op;
Face2:operation:Broaching:,notsupportforop;
Face3:operation:Broaching:,notsupportforop;
Slot1:operation:Broaching:,notsupportforop;
Slot2:operation:FFGrinding:,notsupportforop;
Slot2:operation:Broaching:,notsupportforop
== == UNRESOLVED CONFLICT == ==
Conflict3Op;
Op-agent;
Resolution1Op;
Hole:operation::,notavailable
== == AND CONFLICT == ==
Conflict1Machine
== == CONFLICT IS == ==
Conflict1Machine;
Machine-agent;
Resolution1Op;
Hole:operation:RHoning:,notsupportforop

In this example, only three P-agents are involved for purpose of simpler illustration, and cooperative process planning revealing unresolved conflicts among the P-agents is demonstrated. The B-agent receives the product design data from the D-agent and forms a problem ‘Problem0’. The ‘Problem0’ is sent to the Op-agent, Cutter-agent, and Machine-agent for them to generate proposals. The Op-agent generates a proposal ‘Proposal0Op’ and sends this proposal to the B-agent. The B-agent posts this proposal to the other two P-agents: Cutter-agent and Machine-agent for them to generate proposals. The Op-agent generates a proposal ‘Proposal0Op’ and sends this proposal to the B-agent. The B-agent posts this proposal to the other two P-agents: Cutter-agent and Machine-agent. The Cutter-agent finds a conflict ‘Conflict0Cutter’ with ‘Proposal0Op’. The Machine-agent also disagrees with ‘Proposal0Op’ and finds a conflict ‘Conflict0Machine’. Both conflicts are posted to the Op-agent. After reviewing ‘Conflict0Cutter’, the Op-agent finds that it cannot resolve this conflict, and generates an ‘unresolved conflict’ reply ‘Conflic1Op’ to ‘Conflict0Cutter’. Both conflicts ‘Conflict0Cutter’ and ‘Conflict0Op’ are fed back to the D-agent. At the same time, the Op-agent generates a resolution ‘Resolution1Op’ in reply to ‘Conflict0Machine’. But the Machine-agent cannot agree with the resolution ‘Resolution1Op’, and a conflict reply ‘Conflict1Machine’ is posted to the Op-agent. After reviewing this conflict and former conflict from the Machine-agent, the Op-agent cannot resolve the conflict ‘Conflict1Machine’ and generates an unresolved conflict ‘Conflict3Op’. Both conflict ‘Conflict3Op’ and ‘Conflict1Machine’ are fed back to the D-agent. The problem solving activity is terminated. In this example, due to the unresolved conflict existing, no plan is obtained. Only the unresolved conflicts are fed back to the D-agent. The results are summarized in Table 1.

5.3. Modified data

A different design is next used to demonstrate a successful planning process output. The part design has the same dimensional size as the previous example, but has different dimensional tolerance for the feature ‘Slot1,Slot2’ and Hole. Again three P-agents are involved in this example.

| Table 1 |
| Proposal0Op | owner | disagreement | disagreement |

5.4. Planning result based on modified data

Plan19980718154047;
Machining;
CoCAPP;
PartOne;
feature:Face1.operation:FFMilling:Face1Op2:::,
machine:VMillingMachine:XH715:15.:1500.:7.5:
600.,cutter:MFEndMill:MillingCutter5
a1:3.:0.5:
600.:3 0.,explanation:Cutter-agent:cuttingto:50.:40.;
feature:Face1.operation:RFMilling:Face1Op1:::,
machine:VMillingMachine:XH7 15:15.:1500.:7.5:
The B-agent receives the product design data from the D-agent and forms a problem Problem0. The Problem0 is sent to the Op-agent, Cutter-agent, and Machine-agent for them to generate proposals. The Op-agent generates a proposal Proposal0Op and
The B-agent posts this proposal to the other two P-agents: Cutter-agent and Machine-agent. The Machine-agent agrees with Proposal0Op. The Cutter-agent finds a conflict Conflict0Cutter with Proposal0Op. The Op-agent is informed of the conflict Conflict0Cutter. After reviewing Conflict0Cutter, a resolution Resolution0Op is generated in reply to Conflict0Cutter. Resolution0Op is posted to the Cutter-agent. This time the Cutter-agent agrees with the resolution. After the Op-agent receives resolution agreement to Resolution0Op, it starts forming a new proposal Proposal1Op based on the resolution results. Proposal1Op is posted to the other P-agents for their evaluation. After the Machine-agent and Cutter-agent add their views to the problem solving, finally, all three P-agents agree with the proposal Proposal2Machine. The B-agent starts composing a solution to Problem0. A solution Plan19980718150407 is sent to the D-agent. The B-agent also sends the message Complete to all the P-agents to inform them of the end of the problem solving stage. The evolution from a proposal to the solution is shown in Table 2.

It is shown that the cooperative process planning is carried out among the P-agents. Once a problem definition arrives in the CoCAPP system, the B-agent immediately posts it to all the registered P-agents. When a P-agent generates a proposal to the problem, it will immediately post it to the B-agent. The B-agent informs the other P-agents of the proposal after it examines the proposal. An interested P-agent will evaluate the proposal and give an evaluation result: either agreement, conflict, or agreement with an added new proposal. The proposal owner will try to resolve the conflict. The presented problem is either resolved, or unresolved. The unresolved conflict will be fed back to the D-agent. If the conflict is resolved and the method of resolution is agreed, a new proposal will be proposed based on the resolution result.

If the resolution method is still not agreed by the conflict owner, both P-agents involved will trigger a negotiation program to handle this conflict. If a proposal is agreed by all the P-agents, the B-agent will form a solution based on this proposal. Once a solution is generated, the B-agent will inform the P-agents to end the process planning. Any new scheduling task will be terminated. The solution to a problem is generated in a cooperative way. No one P-agent can generate a full solution. Each P-agent can only contribute a partial solution. During negotiation, only the conflict owner and proposal owner involved are invited to carry out the conflict resolution or evaluation. This reduces the difficulty of problem solving. From the two examples given, it is seen that the Machining CoCAPP system can successfully deal with the process planning problem. The designed system is able to meet the proposed five requirements: Autonomy, Flexibility, Interoperability, Modularity, and Scalability.

6. Conclusion

The cooperative agent model for CAPP was introduced in this paper. The model makes use of intelligent agents and tries to satisfy five major requirements simultaneously: Autonomy, Flexibility, Interoperability, Modularity, and Scalability. An experimental Machining CoCAPP system has been developed by using the proposed model. The developed CoCAPP system is different from other CAPP systems available; it utilizes cooperative and coordination mechanisms built into distributed agents with their own expert systems. Each agent in this system deals with a relatively independent domain of process planning. This is in sharp contrast to other CAPP systems utilizing a single standalone expert system to perform the entire process planning. This system is hence flexible and upgradable. This feature is especially useful as the change of process planning methods or revision due to technology advances is increasingly more common and frequent.

In the paper, a typical mechanical component is considered to test the performance of the Machining CoCAPP system. The experimental results show that the system can effectively deal with the process planning.
planning problems. It can generate process plans according to the product design data and available manufacturing resources. The system has met the proposed design requirements. In particular, the CoCAPP system has the following characteristics.

- Its P-agents can be added and deleted at any time without affecting system operation, and can be individually updated without affecting the others, thus reflecting the modularity and flexibility features embodied in the system.
- Each P-agent only generates a partial solution related to its knowledge; therefore, a complex problem can be decomposed into many simpler sub-problems on a modular basis.
- The whole solution to a problem is obtained by integrating each P-agent’s proposal together with other proposals.
- Each P-agent can be an individually developed expert system or an analytical program with cooperation knowledge included according to demand. This autonomous feature greatly simplifies the implementation of the CAPP system.

This paper presents the method to model the process planning agent (P-agent) by using intelligent agent technology. According to this model, each agent is interoperable and not confined to any machine platform. The model for conflict resolution strategy is developed to suit the CoCAPP system. In addition, the process-planning knowledge base is divided into multiple knowledge bases which are independently established. This has greatly reduced the search space of each inference engine. Based on the proposed CoCAPP system, the optimization of process plans would be feasible and easier to obtain. The experimental results have shown that the CoCAPP framework can be easily integrated into the concurrent engineering environment to implement integrated product design; it can deal with unreasonable part designs. The proposed CoCAPP framework opens up a new approach to the CAPP development. It provides an open framework. It is very suitable for distributed CAPP system development. Further investigations should focus on the improvement and extension of the system. Currently, the experimental Machining CoCAPP system only includes three P-agents. Other P-agents such as feature recognition, etc. may be added to the system in the work of future development.

References


Dr. S.K. Tso, obtained his BSc (Eng.) degree from the University of Hong Kong, and his MSc and PhD degrees from the University of Birmingham, UK. He has taken up the Professorship of Mechatronics and Automation in the City University of Hong Kong after a long service at the University of Hong Kong. He is currently the Director of the Centre for Intelligent Design, Automation and Manufacturing. He is a Fellow of IEE and Senior Member of IEEE.

Dr. Paul S.Y. Wu, received the BS degree from Cheng-Kung University, the MS degree from New York University, and PhD degree from the University of Cincinnati, all in mechanical engineering. He is a member of ASME and SPIE.

F.L. Zhao, obtained his BSc and MSc degrees from Zhejiang University, P.R. China. He is currently a PhD candidate at the City University of Hong Kong.