99-38

Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2000), *14*, 289–304. Printed in the USA.

Generating and evaluating designs and plans for microwave modules

DANA NAU,[1,2] MICHAEL BALL,[2,3] JOHN BARAS,[4] ABDUR CHOWDHURY,[5] EDWARD LIN,[2] JEFF MEYER,[6] RAVI RAJAMANI,[7] JOHN SPLAIN,[8] AND VINAI TRICHUR[9]

[1]Department of Computer Science, University of Maryland, College Park, MD 20742 USA
[2]Institute for Systems Research, University of Maryland, College Park, MD 20742 USA
[3]Robert H. Smith School of Business, University of Maryland, College Park, MD 20742 USA
[4]Electrical and Computer Engineering Department, University of Maryland, College Park, MD 20742 USA
[5]IITRI, Rockville, MD, USA
[6]GTE/BBN Technologies
[7]RWD Technologies
[8]Mitretek Systems
[9]i2 Technologies

(RECEIVED September 1, 1999; ACCEPTED January 31, 2000)

Abstract

This paper describes the process planning techniques we developed for use in an Integrated Product and Process Design (IPPD) tool for the design and manufacture of microwave transmit/receive modules. Given a collection of data about the design of a microwave module, the IPPD tool uses a combination of AI planning and OR trade-off analysis to produce a collection of alternative designs and alternative process plans that have Pareto optimal values for manufacturing and purchasing lead time, process yield, cost, and number of suppliers. The IPPD tool provides facilities to enable the user to generate and examine these Pareto optimal alternatives in real time, in order to provide immediate feedback on how to modify the design to improve its cost and productivity.

Keywords: AI Planning; Integer Programming; Integrated Product and Process Design (IPPD); Process Planning; Tradeoff Analysis

1. INTRODUCTION

AI planning technology is coming of age: it is finally becoming capable enough to be useful in a wide variety of practical settings (Aarup et al., 1994; Wilkins & Desimone, 1994; Agosta, 1995; Nau et al., 1995, 1998; Smith et al., 1996; Smith, 1998; Munoz et al., 1999*a*, 1999*b*). However, this very success raises several issues that have not yet been addressed adequately by previous research:

- Planning systems for real-world planning problems will need to have ways whereby people who are not AI planning experts can understand, modify, and maintain them.
- They will also need to have interfaces to other software modules, so that they can operate effectively in an embedded fashion as part of larger systems.

This paper describes our work towards the above goals, in the domain of manufacturing planning. Our specific application task is the development of an Integrated Product and Process Design (IPPD) tool for the development of microwave transmit-receive modules, which are complex electronic devices that operate in the 1–20 GHz range. The system was developed as part of a contract with Northrop Grumman Corporation's Electronic Sensors and Systems Division (ESSD) division in Baltimore. In developing the IPPD tool, we had the following goals:

- To give feedback to designers about how well the design meets the following design objectives: product cost (including both the cost of the parts to be used in the

Reprint requests to: Dana Nau, Department of Computer Science, University of Maryland, College Park, MD 20742, USA. E-mail: nau@cs.umd.edu

design and the cost of the manufacturing processes), lead time, yield, and number of suppliers.
- To help the designer modify the design to improve the design objectives.

To address these goals, the IPPD tool includes the following elements:

1. ways to launch two systems that are external to the IPPD tool: 1) a commercial design system for producing the initial design of a microwave module, and 2) a database system containing information about the parts to be used in the design and alternative parts suitable for substitution into the design in place of the parts specified by the designer;
2. a module that generates alternative process plan elements for each alternative part, and a graphic user interface (GUI) for use in maintaining the process-planning module's knowledge base;
3. a module that takes the output of the process-planning module as its input to generate alternative Pareto optimal designs and process plans (i.e., combinations of parts and plan elements that produce Pareto optimal values for cost, lead time, yield, and number of suppliers);
4. a user interface for real-time user control of each of the above.

This paper concentrates on the AI planning functionality described in Item 2 above. We describe how we designed a planning architecture to provide a combination of high performance, ease of understandability by manufacturing personnel, ease of maintenance, and integration with the other elements of the IPPD tool.

2. MICROWAVE MODULES

Most commercial electronic products operate in the 10 kHz–1 GHz radio frequency spectrum. However, in the telecommunications arena, the range of operation frequency has been increasing at a tremendous pace. For scientific and commercial long-range defense applications—such as radar, satellite communications, and long-distance television and telephone signal transmissions—radio frequencies prove unsuitable, primarily because of the high noise-to-signal ratio associated with radio frequencies. Moreover, the lower-frequency bands have become overcrowded because of the overuse of these bands for commercial communications applications (Trinogga et al., 1991). Consequently, in contrast to other commercial electronic products, most modern telecommunications systems operate in the 1–20 GHz microwave range, and modules of such systems are called *microwave modules*.

Figure 1 shows a typical microwave module, which includes a number of *parts* mounted on a *substrate*. One factor influencing the construction of such modules is the large effect that mechanical characteristics can have on the electronic performance of microwave devices. To achieve proper performance, the lengths and widths of the printed wires must be chosen carefully; and to minimize vibration, the substrate is made of aluminum with a teflon coating to provide electrical insulation.

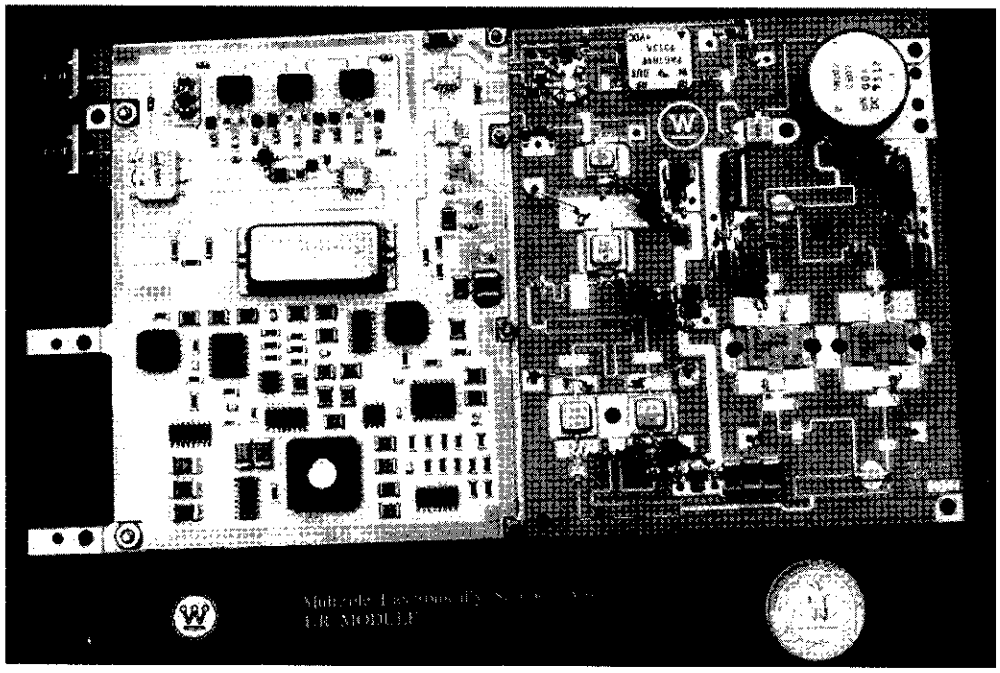


Fig. 1. A typical microwave transmit/receive module.

Figure 2 illustrates the design and manufacturing cycle for microwave modules, which is highly interdisciplinary in nature. Electronic designers develop the detailed circuitry; mechanical designers design the device to resist shock and vibrational loadings, and develop the assemblies, the heat removal systems, and the housing of the device; and manufacturing engineers apply electronic manufacturing processes (such as lithography, soldering, cleaning, and testing), and mechanical manufacturing processes (such as drilling and milling) to manufacture the end product.

In the design of a microwave module, designers and manufacturing engineers may need to choose among a large number of parts and processes to meet system requirements, such as cost, lead times, quality, and so forth. (Boothroyd, 1992, Hebbar et al., 1996). Parts could potentially be available in many forms (e.g., a resistor could be available either with wire leads for through-hole mounting or with tabs for surface mounting), and could be offered by a number of vendors with differing cost and quality attributes. Each of these different forms of a part could require different processes. The choice of these manufacturing processes depends on several factors, such as the type of dielectric material and the degree of integration of functional elements of the design.

When designing a microwave module, designers and manufacturing engineers are faced with a large number of choices. For each part that the designer specifies for use in the design, there may be several alternative parts that are suitable to be substituted for that part; and some combinations of alternatives may possibly produce better values for some of the design and manufacturing criteria. Thus, to select a good combination of parts to use in the design, the design and engineering team might need to go through a large number of iterations between alternative designs and process plans. The IPPD tool is intended to aid them in making these choices.

3. RELATED RESEARCH

3.1. Prior work by others

Process planning can be defined as the act of preparing detailed operating instructions that transform an engineering design into a final part (Chang & Wysk, 1985). Most work on Computer-Aided Process Planning (CAPP) has focused on the development of process plans for mechanical parts. CAPP systems have been traditionally classified as *variant* or *generative*; these are described below.

Variant process planning (which is the basis for most commercial CAPP systems) is based on the use of Group Technology (GT) coding schemes (Chang & Wysk, 1985). The purpose of a GT coding scheme is to assign a fixed-lengh alphanumeric code to each design in such a way that if two designs receive the same GT code, they will require similar manufacturing processes. Given a new design, the user computes its GT code, and uses this code as a database index to retrieve a process plan for some other design having the same GT code. The user then modifies this plan by hand, to produce a process plan for the new design. In *generative* process planning, the process plan is developed automatically by the computer. The development of generative systems has been a subject of much research (for a comprehensive review, see Shah et al. (1994)), but because of the difficulty of the problem, few successful commercial systems exist.

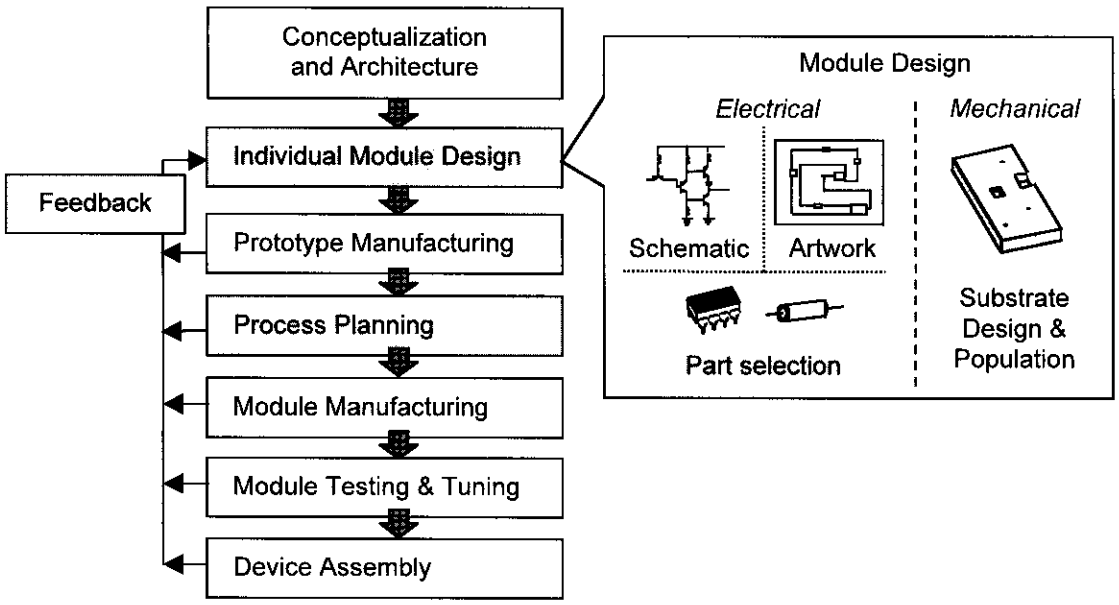


Fig. 2. The design and manufacturing cycle for microwave modules. The scope of our work includes the shaded portions of the figure: part selection, process planning, and feedback to the designer.

Some efforts have focused on CAPP for electronic applications (for a review, see Maria & Srihari (1992)). The PWA Planner (Chang & Terwilliger, 1987) is a rule-based system that performs planning for assembly of parts on placement machines. Sanii and Liau (1993) and others have used AI approaches to develop plans for assembling PCBs; and Liau and Young (1993) have developed a process planning and concurrent engineering system for PCBs that represents process knowledge as constraints and provides manufacturability feedback on the design.

3.2. Our prior work

The IPPD tool described in this paper grew out of the merger of two previous projects at the University of Maryland: the EDAPS project (Hebbar et al., 1996, Smith et al., 1997) and the EXTRA project (Karne et al. 1998):

- EDAPS (Electro-Mechanical Design And Planning System) was an integrated design and process-planning system for microwave modules, that incorporated interfaces to electronic and commercial CAD tools, generated process plans, and provided feedback about manufacturability, cost, and lead time. EDAPS's process-planning module is a predecessor of the one described in Section 5.1.
- EXTRA (EXpert T/R module Analyst) was intended to provide an integration of enterprise-wide product database management with a trade-off analysis optimization mechanism (Ball et al., 1995). EXTRA's trade-off-analysis mechanism is a predecessor of the one described in Section 5.2.

EDAPS's planning module was based on the use of Hierarchical Task Network (HTN) planning, an AI planning methodology that creates plans by *task decomposition*. Because the IPPD tool is based only loosely on HTN planning, a detailed discussion of HTN planning is beyond the scope of this paper—but below we give a brief overview of how it works.

HTN planning systems produce plans by decomposing *tasks* into smaller and smaller subtasks (see Fig. 3 for an example), until primitive tasks are found that can be performed directly. HTN planning systems have knowledge bases containing *methods*. Each method includes a prescription for how to decompose some task into a set of subtasks, with various restrictions that must be satisfied for the method to be applicable, and various constraints on the subtasks and the relationships among them. Given a task to accomplish, the planner chooses an applicable method, instantiates it to decompose the task into subtasks, and then chooses and instantiates other methods to decompose the subtasks even further. If the constraints on the subtasks or the interactions among them prevent the plan from being feasible, the planning system will backtrack and try other methods. For descriptions of the original work on HTN planning, see Sacerdoti (1977) and Tate (1977). Russell and Norvig (1995) give a tutorial overview, and Nau et al. (1998) describe some recent advances in HTN planning technology.

4. THE IPPD TOOL

Our objective in developing the IPPD tool described in this paper was to create a system to help the user perform the following tasks (Fig. 4), starting from the alternatives that are available for each part in the design:

1. For each alternative part, generate alternative "plan fragments", that is, alternative collections of manufacturing processes to use on that part.
2. Find Pareto optimal designs, that is, combinations of parts and plan fragments that produce Pareto optimal values for the following criteria: cost, lead time, yield, and number of suppliers.

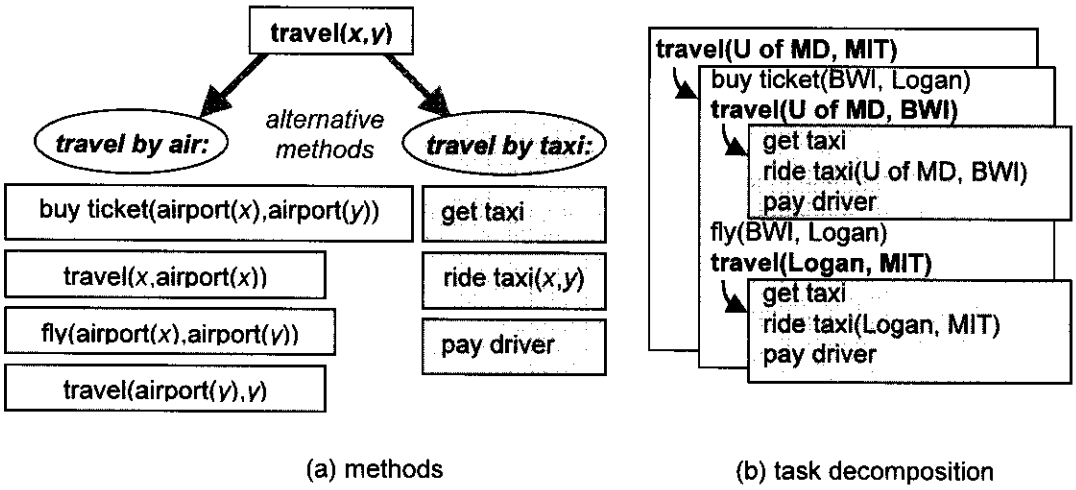


Fig. 3. Two methods for traveling from one location to another, and their application to the task of traveling from the University of Maryland to MIT.

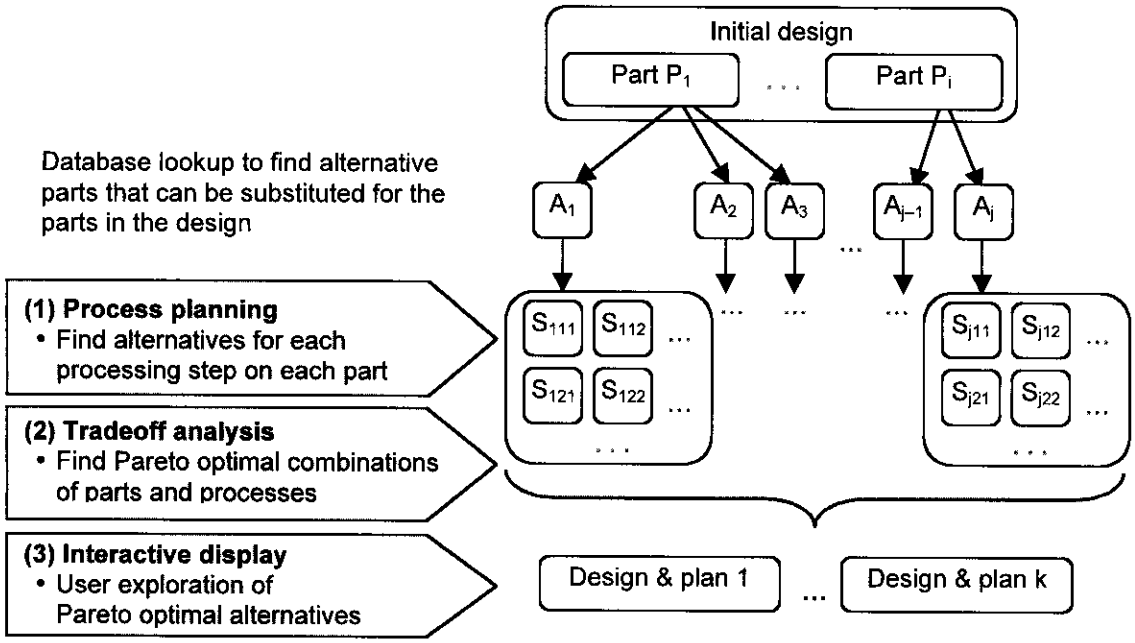


Fig. 4. Tasks that the IPPD tool helps the user to perform.

3. Select a design and a process plan from among the Pareto optimal alternatives.

The system architecture that we developed to accomplish these tasks is shown in Figure 5. The system includes the following elements:

- A *process planning module* (the "Process Planner" in Fig. 5). This module is capable of generating plan fragments for the parts of a design. For each part, it determines (1) alternatives for each of the processes required by this part (together with their yields, setup times, and run times), and (2) what processes might damage this part (and are thus precluded from being used on other parts in the design).
- A *process template editor*. This module is used to create and store the domain-specific knowledge base used by the process planning module.

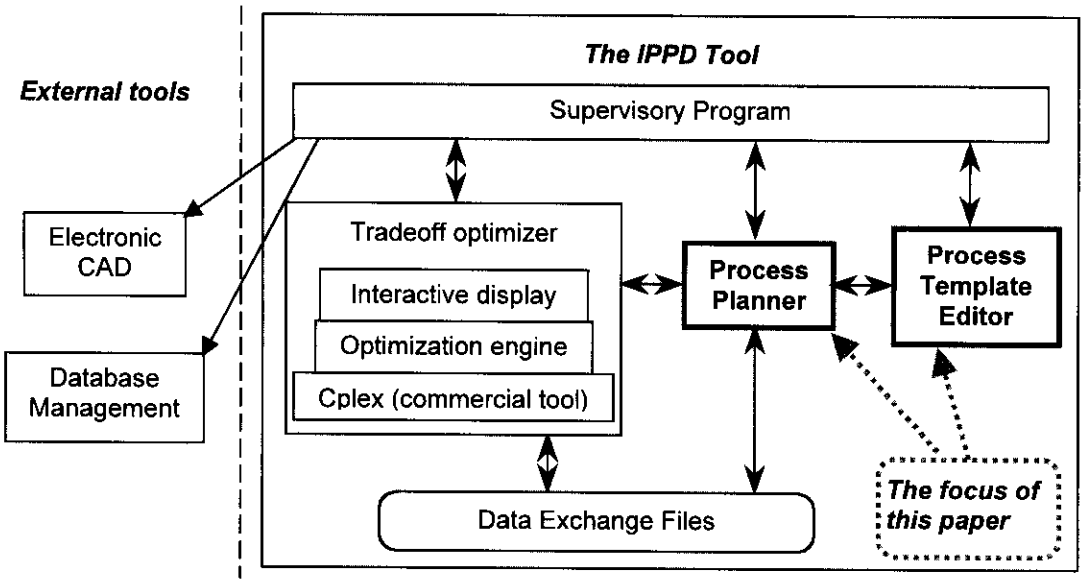


Fig. 5. Overall architecture of the IPPD tool. This paper focuses on the two modules shown in boldface: the process-planning module and the process template editor.

- A *trade-off analysis module* (the "Trade-off Optimizer" in Fig. 5). This module generates alternate realizations of the circuit schematic; all these realizations are Pareto optimal with respect to the four criteria mentioned above. Each realization is obtained by making specific choices among the available alternatives for parts and processes. This module includes a GUI whereby the user can interactively explore Pareto optimal alternatives, and an optimization "engine" written in C++, which calls optimization routines supplied by Cplex, a commercial linear and integer programming solver. To obtain the process requirements and cost estimates associated with the individual parts, the trade-off optimizer directly interfaces with the process planning module.
- A *supervisory program*. This module, written in Visual Basic, permits the designer to smoothly interact with the heterogeneous collection of modules described above. It also provides an interface between those modules and tools external to the system, such as the *data management* software written by Northrop Grumman personnel using Microsoft Access, and an electronic CAD package such as Hewlett Packard's Advanced Design System (formerly known as EEsof Series IV).
- *Data exchange files*, for exchanging information among the above modules. The data exchange is done in a manner that is transparent to the user.

The remainder of this paper concentrates primarily on the process-planning module and the process template editor, which are highlighted in Figure 5.

5. PROCESS PLANNING IN THE IPPD TOOL

The IPPD tool's process-planning module is based loosely upon the approach used in EDAPS's process-planning module (see Section 3). However, the IPPD tool's process-planning module differs from EDAPS's process-planning module in several significant respects, for the following reasons:

- *Functionality*. The IPPD tool needs to evaluate trade-offs among competing objectives (some of which are process dependent), in order to find Pareto optimal designs. Thus, a portion of its process-planning activity needs to occur in its trade-off-analysis module (see Section 5.2) rather than in the process-planning module. Consequently, unlike EDAPS's process-planning module (which generated and evaluated complete plans for complete designs), the IPPD tool's process-planning module does not need to generate complete process plans for the entire design. Instead, for each part, it needs to generate a plan fragment that describes the processes to be performed on that part. Typically, alternatives are available for each process to be performed on the part; thus, associated with each part, we might have many alternative plan fragments. This information is fed into the trade-off analysis module, which then determines combinations of parts (and plan fragments for those parts) that lead to Pareto optimal values for cost, lead time, yield, and number of suppliers.
- *Understandability and maintainability*. EDAPS's process-planning module performed quite efficiently (it could create process plans in just a few seconds). However, its knowledge base of HTN methods, which was written directly in C++, was rather complicated for nonexperts to understand. For the IPPD project, we wanted to remove the necessity of having a software engineer maintain the system after delivery, and, hence, needed to represent the process-planning knowledge in a way that would be easy for product designers and process engineers to understand and maintain.

To meet the above objectives, the IPPD tool's process-planning module uses a planning technique that could possibly be described as *nonhierarchical* task-network planning. The technique is based on the use of a *process template* that describes all of the various processes that might need to be performed to manufacture the product. For each part that is a possible candidate for use in the finished product, the process-planning module uses the process template to produce the following information:

- which manufacturing processes are applicable to the part;
- how much time it will take to run those processes, and what the yield will be;
- which processes will be precluded if this part is used in the product.

For example, in Step 2 of Figure 4, the process-planning module would produce the above information once for each of the alternatives $A_1, A_2, A_3, \ldots, A_n$. The following section describes the process-planning module in greater detail.

As shown in Step 3 of Figure 4, all of the above information (along with each part's cost, defect rate, and supplier data) is fed into the trade-off-analysis module. The purpose of the trade-off optimizer is to aid the user in selecting (from among the possible candidates for each part in the product) a combination of parts that provides Pareto optimal values for the overall delivery lead time, the total cost, the total yield, and the total number of suppliers. The trade-off optimizer employs interactive multiobjective integer programming techniques, and is summarized in Section 5.2.

5.1. Generation of plan fragments

The IPPD tool's process-planning module uses a planning architecture that could perhaps be described as nonhierarchical task-network planning. The system is based on the use of process templates that describe the various processes that each part would need to go through to be placed on a

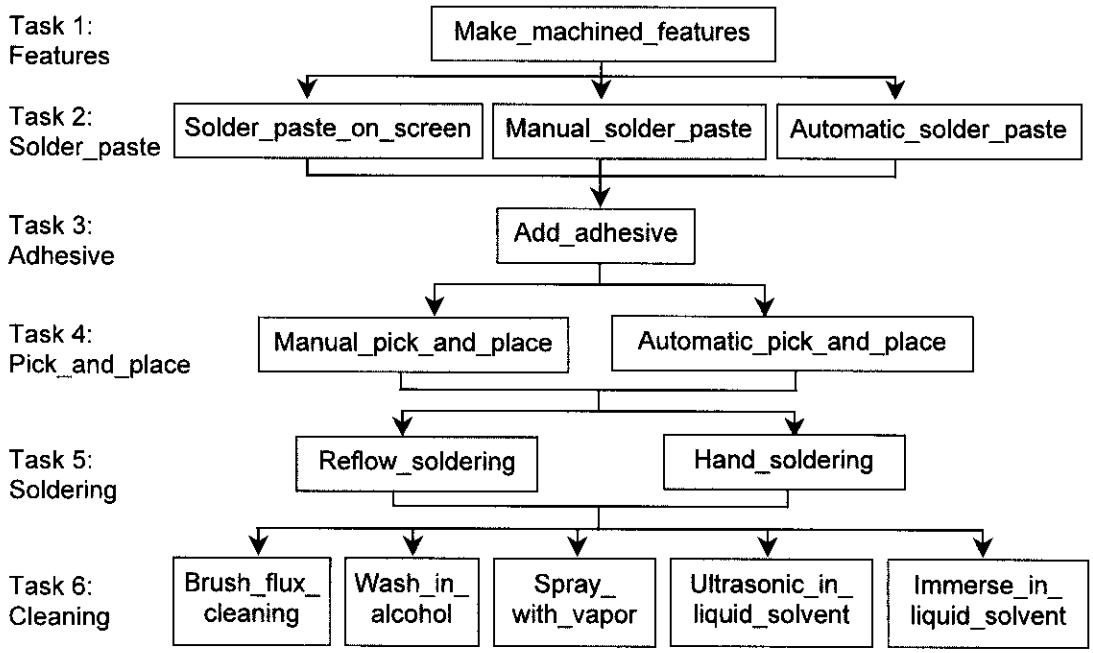


Fig. 6. A simple example of a process template.

circuit board. In general, a template should contain information about all of the possible processes that might need to be performed on any of the parts of a design to affix it to the board. This information is needed so that (as described later), the process planner can take the description of each part and run it through the template to see which processes are applicable.

Figure 6 gives a simple example of what a process template might look like. As shown in the figure, each level of the template represents a task that might need to be done on a part to incorporate it into a microwave module. Each of the processes at that level is an alternate way to perform that task. For example, consider the task of applying solder paste, shown at level 2 of the process template. According to the process template, there are three possible ways to apply solder paste: Solder_paste_on_screen, Manual_solder_paste, and Automatic_solder_paste.

Associated with each process will be formulas for computing the following information:

- *Applicability conditions*, which must be met for the process to be applicable to a part. These conditions will depend on various characteristics of the part, such as the number of pins, the size of the part, and so forth.
- Formulas for the process's *setup time* and *run time*. The setup time and run time will be used later on (in the trade-off analysis module) to compute the total time for each process plan. The difference between them is that in a process plan, the setup time will be incurred *once* for running the process (regardless of the number of parts on which the process is used) and the run time will be incurred repeatedly, once for each part on which the process is used.
- A formula for the process's *yield*. This represents the basic effectiveness of the process to accurately perform or complete its job, and can be thought of as the "probability of success" associated with the process.
- *Preclusion conditions*, which are conditions under which the process *cannot* be used on the board (regardless of whether any parts satisfy its applicability conditions). For example, the "immerse in liquid solvent" process of Figure 6 should not be used on a circuit board if the board has any nonimmersible parts mounted on it, because those parts would be damaged by the immersion process.

As an example, Table 1 shows what the above formulas might look like for the Solder_paste_on_screen process of Figure 6.

For each part P that might possibly be used in the microwave module, the process planner takes as input a description of P's attributes. Then, for each process R in the template, it evaluates R's applicability formulas and preclusion formulas using P's attributes, to determine whether R

Table 1. *Formulas associated with the Solder_paste_on_screen process of Figure 6*

Process name:	Solder_paste_on_screen
Setup time:	60.0
Run time:	num_pins * 1.5
Yield:	0.99
Applicability conditions:	num_pins > 10; length * width > 20;
Preclusion conditions:	num_pins == 0;

Table 2. *Attributes of two example parts*

Attributes of P2	Attributes of P4
immersible = 1	immersible = 1
length = 10	length = 18
width = 15	width = 24
num_pins = 12	num_pins = 2
preformed = 0	preformed = 1
adhered = 1	adhered = 0

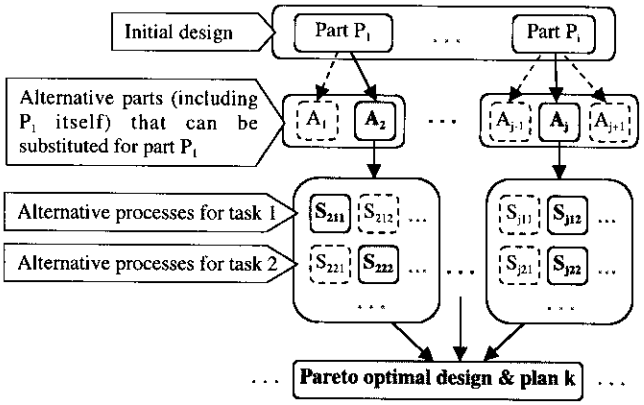


Fig. 7. The process-planning module's output consists of information about the alternatives for each part in the design, and the alternatives for each process to be performed on each part. Using this information as input, the tradeoff optimizer selects Pareto optimal combinations of parts and processes, such as the ones that are highlighted in this figure.

is applicable to P and whether R is precluded by P. For each process R that is applicable to P, the process planner computes R's setup time, R's yield, and R's run time on P.

As a simple example, suppose that there are two different candidate parts P2 and P4, and that their attributes are as shown in Table 2. Then each of the formulas in Table 1 would be evaluated twice (once for each part), to produce the following information for each part P and process R:

- Whether or not the process R is applicable to the part P. In this example, Solder_paste_on_screen is applicable to P2, but it is not applicable to P4 because P4 does not have enough pins.
- Whether or not the presence of the part P will preclude using the process R in manufacturing the product. In this example, neither of the parts precludes the use of the Solder_paste_on_screen process.
- The setup time for the process R. In this example, the setup time for Solder_paste_on_screen is 60.0.
- The process R's run time on the part P. In this example, the run time of Solder_paste_on_screen on P2 is 18.0.
- The process R's yield. In this example, the yield of Solder_paste_on_screen is 0.99.

For each part P, the above computation produces a set of alternative processes for the first task to be performed on P, a set of alternative processes for the second task to be performed on P, and so forth. Conceptually, a plan fragment (as described earlier) for the part would consist of one of the alternatives for task 1, followed by one of the alternatives for task 2, and so forth. However, we do not want to represent each plan fragment explicitly, because the total number of plan fragments for a part could be exponentially large. Instead, we take advantage of the fact that the processes for each task are largely independent of the processes for any other task. Thus, to represent the collection of all plan fragments for P, the process planning module produces a list of all of the alternatives for task 1, followed by all of the alternatives for task 2, and so forth.

Figure 7 gives an example. For the part A_2, the alternatives for the first process to be performed are $\{S_{21i}: i = 1,2,\ldots\}$; the alternatives for the second process to be performed are $\{S_{22i}: i = 1,2,\ldots\}$; and so forth. Thus, a plan fragment for A_2 would consist of one of the S_{21i} tasks, one of the S_{22i} tasks, and so forth. Section 6 gives an example of the specific information that the process planning module computes for each process.

5.2. Trade-off optimization

All of the above information is fed into the trade-off optimizer. As shown in Figure 7, the purpose of the trade-off optimizer is to help the user select 1) which parts to use in the design, and 2) for each part, which manufacturing processes to use on that part. A detailed discussion of the trade-off optimizer is beyond the scope of this paper, but we briefly summarize it below. For further details on its operation, see Trichur and Ball (1998).

The trade-off optimizer expects to receive as input a list of parts, a list of alternatives for those parts, a list of all the applicable processes for each part, and a list of processes precluded by each part. Key attributes such as material costs and defect rates, setup times, yields, and run times are assumed to be known for parts, processes, and part–process combinations. Additionally, the delivery lead time of the supplier associated with each part is assumed to be known. Using this input, the trade-off optimizer selects combinations of parts and processes to generate alternative designs and process plans. Each design and plan generated by the trade-off optimizer is *Pareto optimal* with respect to the total cost of the parts, the total delivery lead time, the total manufacturing yield, and the total number of suppliers used.

Because the concept of Pareto optimality may be new to some readers, we briefly review it here (for a more detailed description, see Gass (1985)). Suppose we have an optimization problem (such as the problem of finding an optimal design for a microwave module) in which there are several heterogeneous optimization criteria (such as cost, time, yield,

and number of suppliers). Then, in general, there may not exist a solution (i.e., a design) that optimizes all of these criteria simultaneously. However, there will always be one or more Pareto optimal solutions. A solution S is Pareto optimal if there is no other solution S' that is at least as good as S along all of the optimization criteria, and that is strictly better than S along at least one optimization criterion.

As an example, suppose we are trying to minimize two variables x and y, and suppose there are four solutions S_1, S_2, S_3, and S_4 as shown in Figure 8. Then the solution S_4 is not Pareto optimal, because S_2 has better values for both x and y, but the solutions S_1, S_2, and S_3 all are Pareto optimal.

For a complex design problem with many alternative parts and process options, there may potentially be hundreds of Pareto optimal solutions, and the problem is to find one that is satisfactory to the designer. Obviously, it would not be feasible to compute all such solutions and display them—this would overload the designer with too much information. Thus, the IPPD tool provides a GUI to help designers zero in on the particular Pareto optimal solutions that are of interest, by enabling them to control the search direction via an interactive optimization procedure.

Given the entire universe of possible parts and processes, and their associated attributes, the problem of selecting a subset of the parts (and implicitly, processes) can be formulated as an integer program (IP). The logical structure of the design (such as inclusion and preclusion conditionals for processes, etc.) gives rise to the constraints of the IP. Because we are interested in optimizing the values of four objectives, we must consider a multiobjective IP. While arbitrary integer programs can be difficult to solve (integer programming is NP-hard), the underlying structure the IP formulation of the microwave module design problem lends itself to relative fast solution by commercial off-the-shelf IP solvers. The trade-off optimizer includes two alternative solution procedures. Trichur and Ball (1998) gives details on the procedures. They also provide the results of computational experiments. Several problem test sets were generated. The number of parts ranged between 25 and 100 with 4 to 6 alternatives per part. Using a Sun Sparc 10 workstation and Cplex 4.0 as the IP solver, the time required to find an individual efficient solution ranged from a few seconds to slightly over 2 min. As, for our application, these problems are of realistic size, we feel this indicates that our IP approach provides a practical problem solving tool.

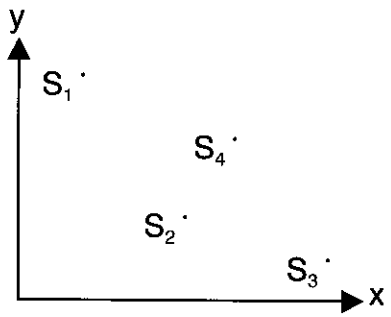


Fig. 8. An example of what Pareto optimality means. If our goal is to find a solution that minimizes both x and y, then S_1, S_2, and S_3 are Pareto optimal but S_4 is not.

Figure 9 gives an example of the trade-off optimizer in action. When it starts up, the first thing that it does is to find six Pareto optimal solutions/designs (represented by the six columns in the left and top right graphs shown in Fig. 9). The first two designs use the original set of parts specified by the designer, but with alternative process plans that optimize cost and yield, respectively. For the third, fourth, fifth, and sixth alternatives, the system chooses alternatives for the parts specified by the designer, and alternative processes for those parts, to find the best possible values for the four objectives being optimized (cost, yield, lead time, and number of suppliers). In the left-hand pane of Fig. 9, the optimizer is currently displaying the cost associated with each of these six solutions, but the user can click on the tabs to display the values for yield, lead time, and number of suppliers. In the right-hand panes, the system has normalized all four of the criteria to the interval [0,1], so that the user can compare them simultaneously.

In the boxes near the lower right-hand corner of the figure, the user can enter lower and upper bounds on the acceptable values of the objectives, and the system will produce additional Pareto optimal solutions (if any) that lie within these bounds. This provides a way for the user to zero in on solutions that provide the best balance of lead time, cost, yield, and number of suppliers.

6. IMPLEMENTATION AND USAGE

The architecture for the IPPD system is shown in Figure 5. Most of the modules run on a PC under Windows NT. However, the process-planning module runs as a server on a Sun workstation, and the rest of the system communicates with this module by exchanging files and commands over an ethernet connection.

6.1. Input to the system

As input, the IPPD system needs a list of the parts that the designer has chosen to use in a proposed design for a microwave transmit/receive module. It also needs information about the attributes of each part, including its physical attributes (dimensions, etc.), cost, defect rate, supplier information, and other parts that might be suitable to be substituted for this part. Normally, the parts list would come from a commercial CAD tool (such as Hewlett Packard's EEsof Advanced Design System, which we used during our project), and the information about the part attributes would come from a parts database (such as the one shown in Fig. 10, which was developed by our colleagues at Northrop Grumman). However, rather than tying the IPPD tool to any particular design tool or database tool, we wanted it to be compatible with a wide variety of design tools and database

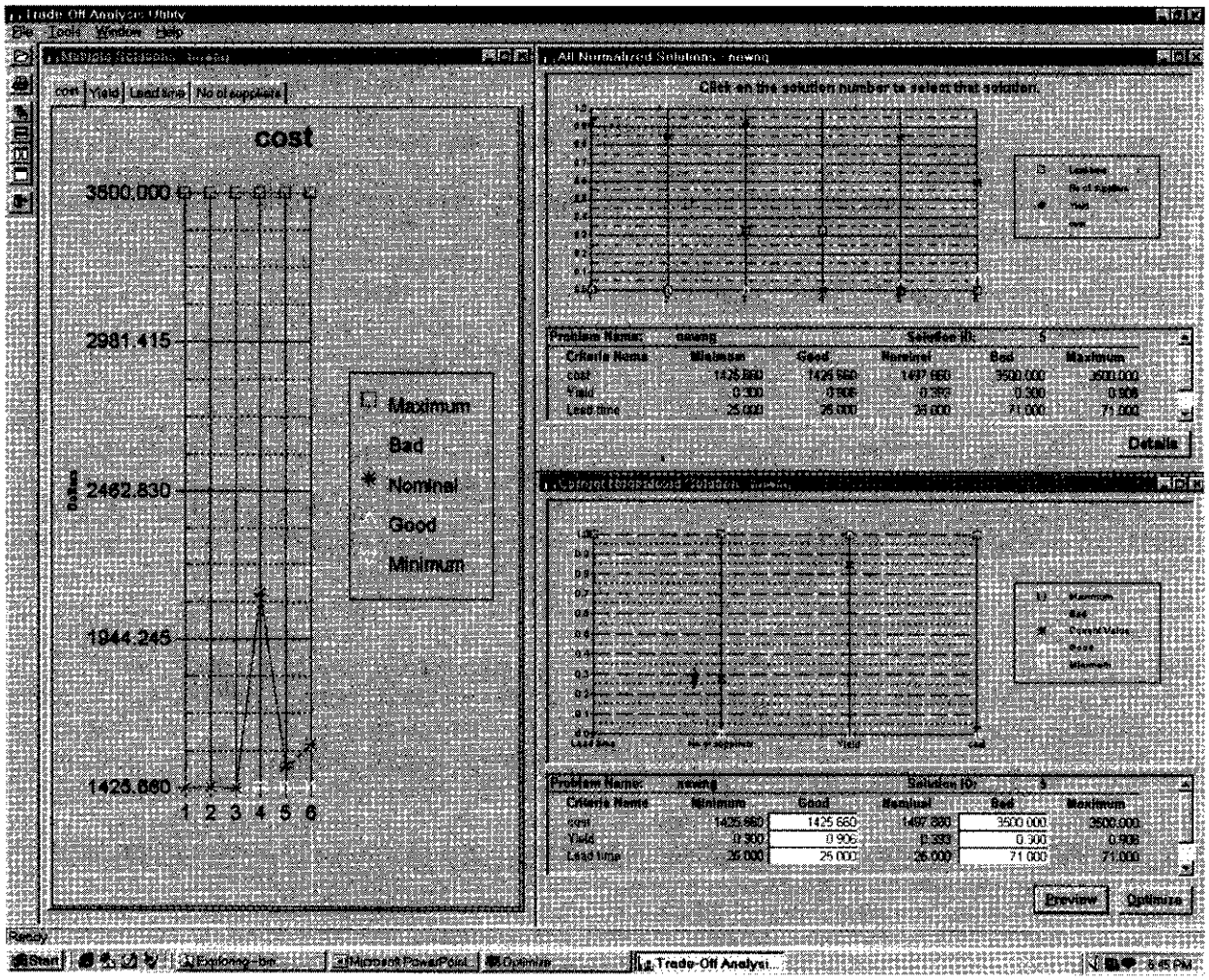


Fig. 9. An example of the tradeoff optimizer in action.

tools. For this reason, the IPPD tool reads its input from flat files rather than querying the CAD tool and database tool directly, and it is the user's responsibility to export the part information into the flat files from the CAD tool and the database tool. Once the user has done this, the IPPD tool sends this information to its process-planning module, so that the process-planning module can run the information about each part through the process template to determine which processes may be applicable and what their parameters are.

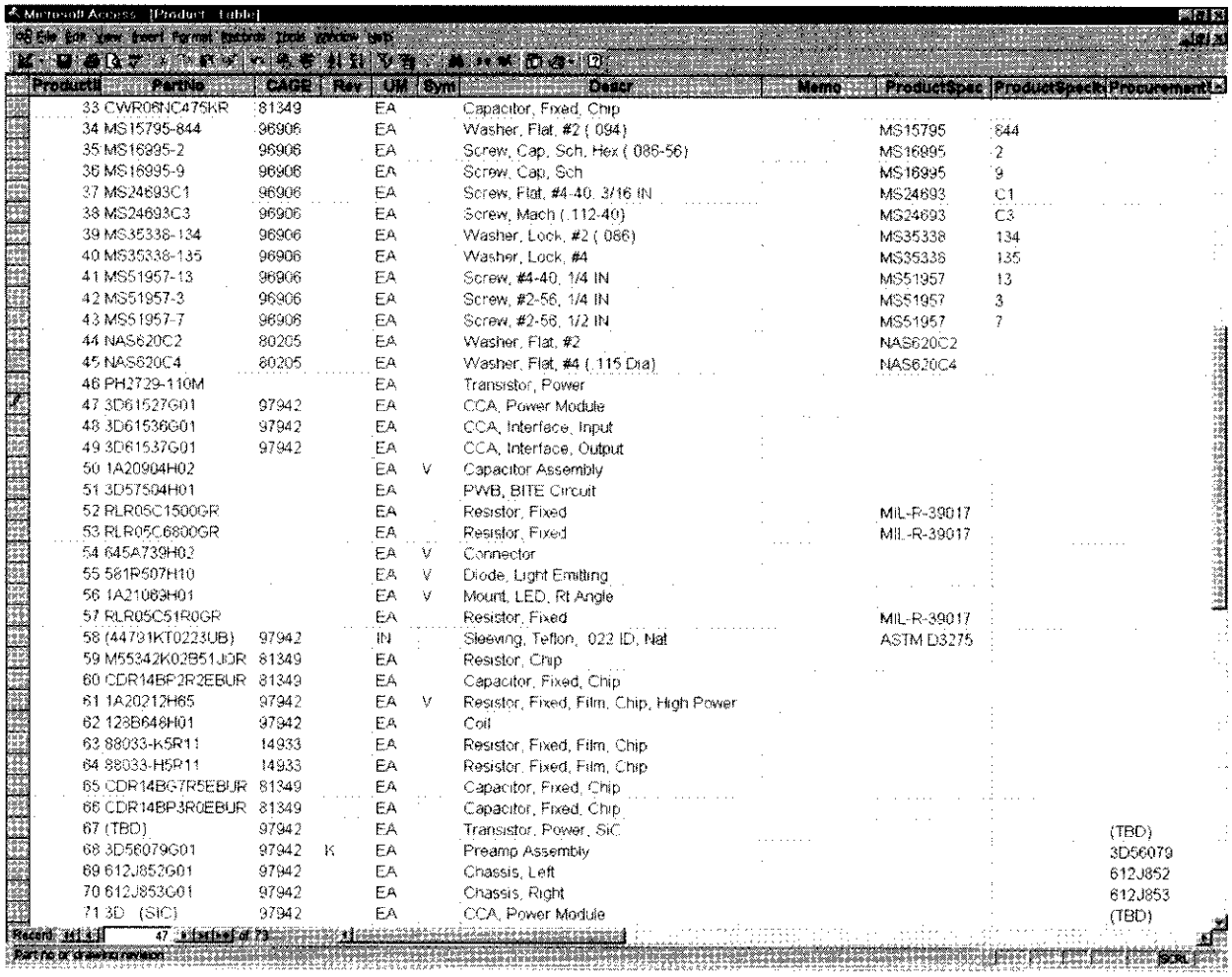
6.2. Process template editor

To provide an easy way for design and manufacturing engineers to create and maintain process templates, we used Microsoft Foundation Class (MFC) to develop a user-friendly GUI running under Windows NT. This GUI allows the user to update the planner's knowledge base without having to modify any source code (as was necessary in EDAPS's process-planning module). Using the GUI, the user can create each of the levels of the template (corresponding to tasks to be performed), create alternative processes at each level, and create the formulas for computing the process parameters (the applicability conditions, setup time, run time, yield, and preclusion conditions). After the user has created the process template, the template can be installed into the IPPD system, and used on subsequent runs of microwave module designs.

As an example, Figure 11 is a screen shot of the GUI being used to edit the process template of Figure 6. In this screen shot, the user is editing the "solder paste on screen" process, which is one of the alternatives for the "solder paste" step at level 2 of the process template. By selecting the appropriate tab of the dialog box shown in Figure 11, the user can enter each of the formulas shown in Table 1.

6.3. Process-planning module

The code for the process-planning module runs as a server on a Sun Sparc workstation—thus, after the template has



ProductID	PartNo	CAGE	Rev	UM	Sym	Descr	Memo	ProductSpec	ProductSpecR	Procurement
33	CWR08NC475KR	81349		EA		Capacitor, Fixed, Chip				
34	MS15795-844	96906		EA		Washer, Flat, #2 (.094)		MS15795	844	
35	MS16995-2	96906		EA		Screw, Cap, Sch, Hex (.086-56)		MS16995	2	
36	MS16995-9	96906		EA		Screw, Cap, Sch		MS16995	9	
37	MS24693C1	96906		EA		Screw, Flat, #4-40, 3/16 IN		MS24693	C1	
38	MS24693C3	96906		EA		Screw, Mach (.112-40)		MS24693	C3	
39	MS35338-134	96906		EA		Washer, Lock, #2 (.086)		MS35338	134	
40	MS35338-135	96906		EA		Washer, Lock, #4		MS35338	135	
41	MS51957-13	96906		EA		Screw, #4-40, 1/4 IN		MS51957	13	
42	MS51957-3	96906		EA		Screw, #2-56, 1/4 IN		MS51957	3	
43	MS51957-7	96906		EA		Screw, #2-56, 1/2 IN		MS51957	7	
44	NAS620C2	80205		EA		Washer, Flat, #2		NAS620C2		
45	NAS620C4	80205		EA		Washer, Flat, #4 (.115 Dia)		NAS620C4		
46	PH2729-110M			EA		Transistor, Power				
47	3D61527G01	97942		EA		CCA, Power Module				
48	3D61536G01	97942		EA		CCA, Interface, Input				
49	3D61537G01	97942		EA		CCA, Interface, Output				
50	1A20904H02			EA	V	Capacitor Assembly				
51	3D57504H01			EA		PWB, BITE Circuit				
52	RLR05C1500GR			EA		Resistor, Fixed		MIL-R-39017		
53	RLR05C6800GR			EA		Resistor, Fixed		MIL-R-39017		
54	645A739H02			EA	V	Connector				
55	581R507H10			EA	V	Diode, Light Emitting				
56	1A21083H01			EA	V	Mount, LED, Rt Angle				
57	RLR05C51R0GR			EA		Resistor, Fixed		MIL-R-39017		
58	(44791KT0223UB)	97942		IN		Sleeving, Teflon, .022 ID, Nat		ASTM D3275		
59	M55342K02B51J0R	81349		EA		Resistor, Chip				
60	CDR14BP2R2EBUR	81349		EA		Capacitor, Fixed, Chip				
61	1A20212H65	97942		EA	V	Resistor, Fixed, Film, Chip, High Power				
62	128B648H01	97942		EA		Coil				
63	88033-K5R11	14933		EA		Resistor, Fixed, Film, Chip				
64	88033-H5R11	14933		EA		Resistor, Fixed, Film, Chip				
65	CDR14BG7R5EBUR	81349		EA		Capacitor, Fixed, Chip				
66	CDR14BP3R0EBUR	81349		EA		Capacitor, Fixed, Chip				
67	(TBD)	97942		EA		Transistor, Power, SiC				(TBD)
68	3D56079G01	97942	K	EA		Preamp Assembly				3D56079
69	612J852G01	97942		EA		Chassis, Left				612J852
70	612J853G01	97942		EA		Chassis, Right				612J853
71	3D (SIC)	97942		EA		CCA, Power Module				(TBD)

Fig. 10. A portion of a part database created by Northrop Grumman personnel using Microsoft Access. Each part corresponds to a record in the database, and the record contains a field for each attribute that might be relevant for design and manufacturing purposes.

been created on the PC, the user needs to install the template on the Sun server for use as the process-planning module's knowledge base. To accomplish this, the GUI contains commands for creating a connection to the server, and sending the template to it. Once the template has been sent to the server, the process-planning module will validate it and then store it in a compact form. Any list of parts can then be run against that template to generate a list of alternative processes applicable to those parts.

The process planning module determines which processes are applicable to each part by evaluating each process's applicability conditions against the list of part attributes and their values. For each process that is applicable to a part, the planner instantiates the formula for the process's run time, to compute how much time the process will take on that part. The planner then determines which (if any) processes are precluded by this part (based upon the process's preclusion conditions). It then places these results in a file that is eventually passed to the trade-off analysis module. Figure 12(a) shows an example of such a file.

Next, the process-planning module creates a file that contains the entire list of possible processes, together with their respective setup times and yields. Figure 12(b) shows an example of such a file. Since the setup times and yields are not part specific, they are placed in a separate file to reduce space overhead, and speed up the preprocessing for the trade-off-analysis module. Both of these files are sent back to the PC (where the original run time calls came through the front-end software). The front-end interface forwards the files to the trade-off optimizer.

6.4. Trade-off optimizer

The trade-off optimizer first generates a file that contains the integer programming formulation of the multiobjective design problem. Next it calls optimization routines supplied by Cplex to solve the problem and generate a set of six Pareto optimal solutions, as described above. The user generates successive Pareto optimal solutions in an interactive fashion by controlling the search direction via a GUI,

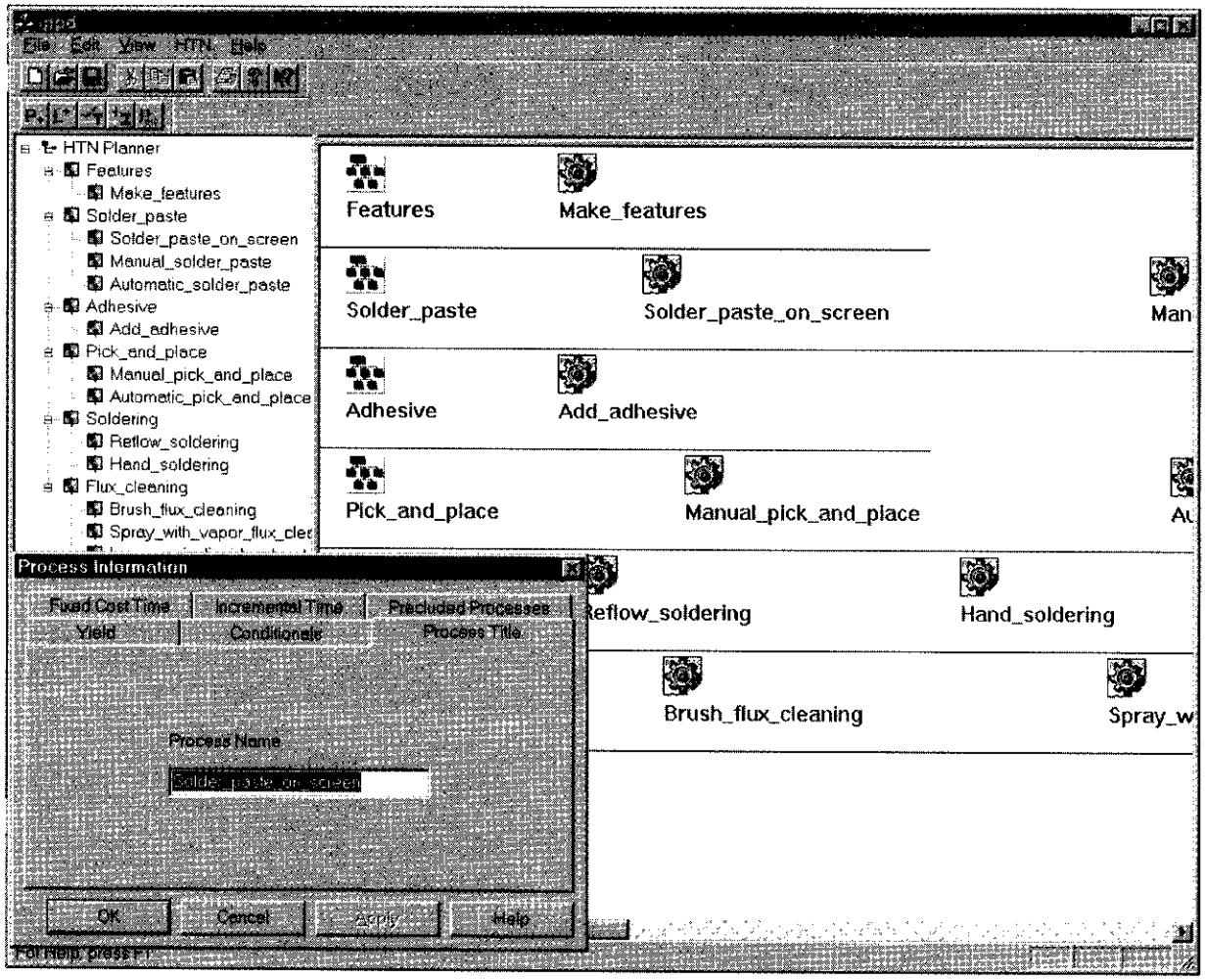


Fig. 11. The GUI for process template creation.

as described earlier. This GUI displays all the solutions generated during the optimization session, and enables the designer to keep track of the process. When the designer is satisfied with one or more solutions, the GUI recovers the relevant data (selected parts, processes, and suppliers, together with the associated values for the four objectives) corresponding to the chosen solution(s) and saves it in a file that is then passed back to the supervisory GUI of the IPPD tool.

7. SAMPLE RUNS OF THE PROCESS-PLANNING MODULE

This section describes sample runs of the process-planning module on two sets of parts, using a more detailed version of the template shown in Figure 6. Figure 11 shows a snapshot of the creation of this template using the GUI on a PC. After creating this template, the user needs to install it on the Sun server. The GUI contains commands for doing this. Once the template is stored on the server, the process-planning module will validate it and then store it in a compact form. Any list of parts can then be run against that template to generate a list of alternative processes applicable to those parts.

For the first example run, we gave the process-planning module the list of parts shown in Table 2. In response, the module created the two output files shown in Figure 12. The output file in Figure 12(a) contains the following information for each part in the design:

1. The name of the part, how many tasks to perform on it, and how many processes are precluded by it.
2. The following information for each task to be performed on the part:
   - how many alternative processes are applicable for this task on this part;
   - for each process, its name and its run time on this part.
3. The names of any processes that are precluded by this part.

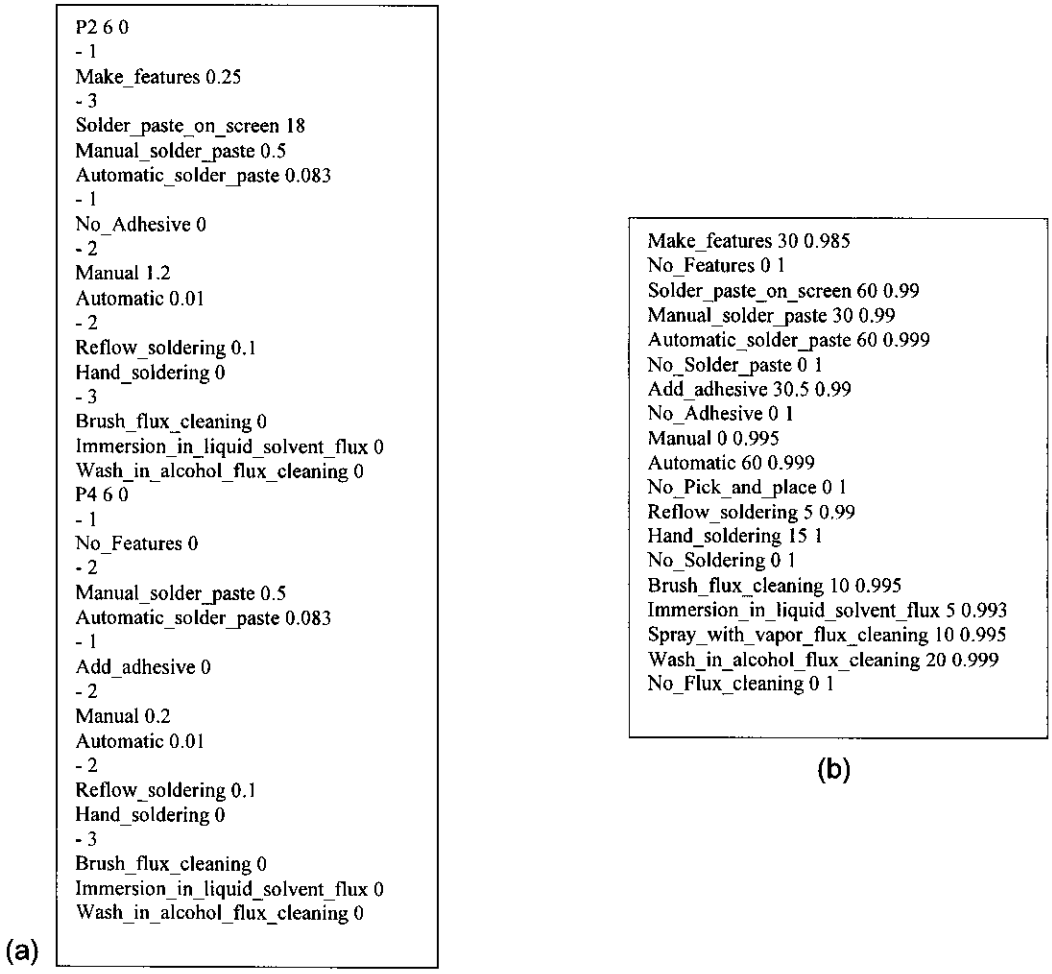


Fig. 12. Output of the first example run.

For example, the second task listed for P2 in Figure 12(a) is the "solder paste" task of Figure 6. For this task, the following three processes are applicable to P2:

- Solder_paste_on_screen, with a run time of 18,
- Manual_solder_paste, with a run time of 0.5;
- Automatic_solder_paste, with a run time of 0.083.

The second output file, shown in Figure 12(b), tells what each process's setup time and yield are. For example, the setup time for Solder_paste_on_screen is 60, and its yield is 0.99.

For the second example run, we modified the input part file shown in Table 2, by decreasing P2's num_pins field from 12 to 8, and making the part P4 nonimmersible. The results of the second run are shown in Figure 13.

In Figure 13(a), the process Solder_paste_on_screen is no longer listed as an option for the Solder_paste task on part P2, because P2 has too few pins. For the part P4, only one process is applicable for the Flux_cleaning task: Brush_flux_cleaning. Because P4 is nonimmersible, the processes Immersion_in_liquid_solvent_flux and Wash_in_alcohol_flux_cleaning appear in P4's list of precluded processes. In Figure 13(b), the file with the process names and their setup times and yields is identical to that of Figure 12(b). It does not change unless the process template is modified.

These two examples show how different process alternatives can be selected because of different part attributes.

8. CONCLUSIONS

AI planning is becoming increasingly useful in practical planning problems—but the techniques that work well in practice can significantly different from the ones used in traditional AI planning systems. For example, the process-planning module described in this paper differs from traditional approaches in several significant ways:

- The idea of process templates was inspired by HTN planning. However, by getting rid of the HTN hierarchy, we made the system significantly easier for non-researchers to understand and maintain, while retaining sufficient planning power for the problem at hand.

(a)

```
P2 6 0
- 1
Make_features 0.25
- 2
Manual_solder_paste 0.5
Automatic_solder_paste 0.083
- 1
No_Adhesive 0
- 2
Manual 0.8
Automatic 0.01
- 2
Reflow_soldering 0.1
Hand_soldering 0
- 3
Brush_flux_cleaning 0
Immersion_in_liquid_solvent_flux 0
Wash_in_alcohol_flux_cleaning 0
P4 6 2
- 1
No_Features 0
- 2
Manual_solder_paste 0.5
Automatic_solder_paste 0.083
- 1
Add_adhesive 0
- 2
Manual 0.2
Automatic 0.01
- 2
Reflow_soldering 0.1
Hand_soldering 0
- 1
Brush_flux_cleaning 0
Immersion_in_liquid_solvent_flux
Wash_in_alcohol_flux_cleaning
```

```
Make_features 30 0.985
No_Features 0 1
Solder_paste_on_screen 60 0.99
Manual_solder_paste 30 0.99
Automatic_solder_paste 60 0.999
No_Solder_paste 0 1
Add_adhesive 30.5 0.99
No_Adhesive 0 1
Manual 0 0.995
Automatic 60 0.999
No_Pick_and_place 0 1
Reflow_soldering 5 0.99
Hand_soldering 15 1
No_Soldering 0 1
Brush_flux_cleaning 10 0.995
Immersion_in_liquid_solvent_flux 5 0.993
Spray_with_vapor_flux_cleaning 10 0.995
Wash_in_alcohol_flux_cleaning 20 0.999
No_Flux_cleaning 0 1
```

(b)

Fig. 13. Output of the second example run.

- Our process-planning module evaluates the elements of its process template in a forward manner, by evaluating their preconditions against the design data, and instantiating the effects accordingly. This makes it straightforward for the elements of the process template to incorporate the mixed symbolic/numeric computations necessary for real-world process planning. In contrast, most AI planning systems evaluate their planning operators in a backward manner, by unifying their effects with the goals to be achieved, and instantiating the preconditions accordingly. Such an approach would have made it very difficult to get the expressive power we needed.
- For each part (or alternative part) in the design, the process-planning module produces a collection of alternatives for each process to be performed on the part. This makes it possible to generate Pareto optimal combinations of parts and process plans, by feeding the process-planning module's output into a trade-off optimizer.

This project illustrates some of the benefits that can be obtained by an interdisciplinary team of researchers. Our work combined the expertise of researchers from computer science, business, systems engineering, and mechanical engineering.

Furthermore, this project shows the benefits of doing research on topics motivated by complex real-world problems. The ideas described in this paper were motivated by the requirements of the practical problem at hand, but our subsequent ongoing work on ideas is leading to significant advances in the theory of planning, as described below:

- By extending our "forward evaluation" HTN planning technique we have developed a domain-independent planning shell that is capable of running orders of magnitude faster than other domain-independent planning systems (Nau et al., 1999), yet has sufficient expressive power for use in complex problems such as evacuation planning (Munoz et al., 1999*b*).
- By extending our work on combining AI planning and OR optimization, we have made significant progress on the use of Integer Programming to solve AI planning problems (Vossen et al., 1999), which was one of the challenges proposed in a prominent IJCAI-97 "challenge paper" (Selman et al., 1997).

For future work, we are continuing to make progress in the two above areas. We are also doing work on extending IPPD into earlier stages of design (i.e., conceptual design), where the decisions made during the design process can have an even bigger impact. That work is the subject of an ongoing contract with Northrop Grumman corporation.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation (NSF) grant EEC-9402384, Maryland Industrial Partnerships contract MIPS 1705.17, the Northrop Grumann ESSD design to cost program, ARL contract DAAL01-97-K0135, and DARPA contract F3060299100l3. Hewlett Packard donated the Advanced Design System 1.0 software used in this project. We would like to thank Bob Hosier and Jim Williams of Northrop Grumman for their valuable inputs to this project.

REFERENCES

Aarup, M., Arentoft, M.M., Parrod, Y., Stader, J., & Stokes, I. (1994). OPTIMUM-AIV: A knowledge-based planning and scheduling system for spacecraft AIV. In *Intelligent Scheduling*, (Fox, M. and Zweben, M., Eds.), pp. 451–469. Morgan Kaufmann, San Mateo, California.

Ball, M.O., Baras, J.S., Bashyam, S., Karne, R.K., & Trichur, V. (1995). On the selection of parts and processes during design of printed circuit board assemblies. In *Proceedings of the INRIA/IEEE Symposium on Emerging Technologies and Factory Automation*, vol. 3, pp. 241–249.

Boothroyd, G. (1992). *Assembly Automation and Product Design*. Marcel Dekker, Inc., New York.

Chang, T.C., & Terwilliger. J., Jr. (1987). PWA Planner—a rule based system for printed wiring assembly process planning. *Computers in Industrial Engineering 13(1–4)*, 34–38.

Chang, T.C., & Wysk, R.A. (1985). *An Introduction to Automated Process Planning Systems*. Prentice Hall, Englewood Cliffs, CA.

Gass, S.I. (1985). *Linear Programming, 5th ed.* International Thomson Publishing, Inc.

Hebbar, K., Smith, S.J.J., Minis, I., & Nau, D.S. (1996). Plan-based evaluation of designs for microwave modules. ASME 1996 Design Engineering Technical Conference and Computers in Engineering Conference, Irving, California.

Karne, R.K., Baras, J.S., Ball, M.O., Trichur, V., Bashyam, S., Kebede, A., Williams, J., Huang, F., Xie, H., & Karir, M. (1998). Integrated product and process design environment tool for manufacturing T/R modules. *Journal of Intelligent Manufacturing 9(1)*, 9–15.

Liau, J.S., & Young, R.E. (1993). A process planning and concurrent engineering system for PCBs. *Manufacturing Review 6(1)*, 25–39.

Maria, A., & Srihari, K. (1992). A review of knowledge-based systems in printed circuit board assembly. *The International Journal of Advanced Manufacturing Technology 7*, 368–377.

Munoz-Avila, H., Aha, D., Breslow, L., & Nau, D. (1999*a*). HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations. In *IAAI-99*, 870–875.

Munoz-Avila, H., McFarlane, D., Aha, D., Ballas, J., Breslow, L., & Nau, D. (1999*b*). Using guidelines to constrain interactive case-based HTN planning. In *ICCBR-99*.

Nau, D., Gupta, S.K., & Regli, W.C. (1995). AI planning versus manufacturing-operation planning: A case study. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 1670–1676. Morgan Kaufmann, San Mateo, California.

Nau, D., Smith, S.J., & Erol, K. (1998). Control strategies in AI planning: Theory versus practice. In *IAAI-98*, 1127–1133.

Nau, D., Cao, Y., Lotem, A., & Munoz-Avila, H. (1999). SHOP: Simple Hierarchical Ordered Planner. In *IJCAI-99*, 968–973.

Sanii, E.T., & Liau, J.S. (1993). An expert process planning system for electronics PCB assembly. *Computers in Electrical Engineering 19*(2), 113–127.

Selman, B., Kautz, H., & McAllester, D. (1997). Ten challenges in propositional reasoning and search. In *Proc. Fifteenth International Joint Conf. Artificial Intelligence (IJCAI-97)*, 50–54.

Shah, J., Mantyla, M., & Nau, D. (1994). *Advances in Feature Based Manufacturing*, Elsevier/North Holland.

Smith, S.J. (1997). Task-network planning using total-order forward search, and applications to bridge and to microwave module manufacture. PhD Dissertation, University of Maryland, College Park, MD.

Smith, S.J., Hebbar, K., Nau, D., & Minis, I. (1997). Integrating electrical and mechanical design and process planning. In *Knowledge Intensive CAD*, (Mantyla, M., Fluger, S. & Tomiyama, T., Eds.) Vol. 2, pp. 269–288. Chapman and Hall, London.

Smith, S.J., Nau, D., & Throop, T. (1998). Computer bridge: A big win for AI planning. *AI Magazine 19(2)*, 93–105.

Trichur, V., & Ball, M.O. (1998). A multiobjective integer programming framework for product design. Technical Report, The Institute for Systems Research, University of Maryland, College Park, MD.

Trinogga, L.A., Kaizhou, G., & Hunter, I.C. (1991). *Practical Microstrip Design*. Ellis Horwood, Chichester, UK.

Vossen, T., Ball, M., Lotem, A., & Nau, D. (1999). On the use of integer programming models in AI planning. In *IJCAI-99*.

Wilkins, D.E., & Desimone, R.V. (1994). *Applying an AI planner to military operations planning*. In *Intelligent Scheduling* (Fox, M. and Zweben, M., Eds.) pp. 685–709. Morgan Kaufmann, San Mateo, California.

Dana Nau is a professor at the University of Maryland, in the Department of Computer Science and the Institute for Systems Research. His research interests include AI planning, and computer-integrated manufacturing. He received his Ph.D. from Duke University in 1979, where he was an NSF graduate fellow. He has written more than 200 technical publications, and has won several best-paper awards. He is an NSF Presidential Young Investigator awardee, a Fellow of the American Association for Artificial Intelligence (AAAI), and a coauthor of the computer program that won the 1997 world championship of computer bridge. For more information, see <http://www.cs.umd.edu/users/nau>.

Michael Ball received his Ph.D. in Operations Research from Cornell University in 1977. He currently is a Professor at the University of Maryland, holding a joint appointment in the Robert H. Smith School of Business and the Institute for Systems Research. His research interests are in the areas of network and combinatorial optimization and their application within transportation, manufacturing and telecommunication systems. Dr. Ball serves as director of research for the Smith School, acting director for the Center for Knowledge and Information Management and associate director for NEXTOR, the National Center of Excellence for Aviation Operations Research.

John Baras received a Ph.D. in Applied Mathematics from Harvard University in 1973. At the University of Maryland, he is a professor in the Electrical and Computer Engineering (ECE) department, was the founding director of the Institute for Systems Research., holds the Lockheed Martin Chair in Systems Engineering, and is the Director of the Center for Satellite and Hybrid Communication Networks. His research interests include stochastic systems, signal processing and understanding, real-time architectures, symbolic computation, intelligent control systems, robust non-

linear control, distributed parameter systems, hybrid communication network modeling, and performance evaluation and management. He is a Fellow of the IEEE.

Abdur Chowdhury is the Senior Computer Science Advisor for IITRI in Rockville Maryland. He is a Ph.D. candidate in Computer Science at the Illinois Institute of Technology. He received his M.S. and B.S. in Computer Science at George Mason University in 1996 and 1994. He has authored many papers on process migration, fault tolerant routing protocols and information retrieval topics. He can be reached at abdur@cs.iit.edu.

Edward Lin is a research engineer at the computer integrated manufacturing laboratory in the Institute for Systems Research at the University of Maryland. He received his Ph.D. from the school of Industrial and Systems Engineering at the Georgia Institute of Technology in 1994. He had five years of industrial experience in automation of manufacturing and production systems. He also has several years experience in developing object-oriented database, distributed manufacturing applications, and web-based manufacturing services for government and industrial projects. His research interests include adaptable manufacturing simulation, production planning and scheduling, and web-based design. For more information, see ⟨http://www.isr.umd.edu/Labs/CIM/profiles/lin⟩.

Jeff Meyer is a software engineer with GTE/BBN Technologies. He is currently working on shallow water acoustics research and development for the Navy's Defense Advanced Research Projects Agency. He received a master's degree from the University of Maryland in 1998 under the tutelage of Dr. Nau. While at Maryland, he aided Dr. Nau in researching artificial intelligence HTN planning and knowledge representation issues.

Ravi Rajamani is a consultant at RWD Technologies. His research interests include systems modeling and analysis, object-oriented software development, and CASE tools and databases. He received his M.S. in Systems Engineering from the University of Maryland. Currently, he is working on projects in the industry to integrate and automate the different areas of business functionality using BPR techniques, intelligent algorithms and COTS packages. For more information, send e-mail to rrajamani@rwd.com.

John Splain is a lead software engineer in the Center for Information Systems at Mitretek Systems. He received his M.S. in Systems Engineering from the Institute for Systems Research at the University of Maryland in 1998. He received his B.S. in Physics with Scientific Instrumentation from the Mellon College of Science at Carnegie Mellon University in 1990. He has nearly 10 years experience in software engineering on projects ranging from satellite ground control and air traffic control to large-scale mainframe databases and latent fingerprint encoding workstations.

Vinai Trichur is a Senior Application Engineer with i2 Technologies. He received his Ph.D. from the University of Maryland, College Park, in 1999, where he was affiliated with the Robert H. Smith School of Business, and the Institute for Systems Research. His research interests include product design, supply chain management, and mathematical programming. For additional information, see ⟨http://www.isr.umd.edu/~vtrichur⟩.