AGENT INTERACTION IN CAD/CAM

David Dornfeld, Paul K. Wright, Shad Roundy, Arvind Rangarajan, and Sung-Hoon Ahn

The Berkeley Manufacturing Institute
Department of Mechanical Engineering
University of California at Berkeley
Berkeley, California 94720

Abstract

In order for a distributed agent based system to operate in an autonomous fashion, there must be a central coordination and communication framework in which the agents (or individual software applications) function. Such a framework has been developed and named the Design Consultant Shell (DCS). This framework is flexible in that it allows a wide variety of software applications to be integrated into the framework and to be remotely accessed. The functionality of the DCS has been demonstrated by integrating two process planning agents for 3-axis milling into a remotely accessed multi-agent system. One of the process planners minimizes machining time, while the other ensures edge quality by minimizing burr formation. Cost estimation, utilizing the data from these process planners is also demonstrated.

Introduction

Distributed agent systems in the field of CAD/CAM are increasing in prevalence and importance. As part of this project, a variety of process planning agents for 3-axis milling have been developed (Dornfeld, *et al* 1999). These agents, which have differing objectives, have been in use for some time. A brief description of two of these process planners, one which optimizes machining time and one which minimizes burr formation (Wright, *et al* 2000), will be given. The method in which information returned from the process planners is used to estimate manufacturing cost will also be explained. In order for these agents to function together in a distributed system there must be a central coordination and communication framework in which the individual agents can operate. The framework, which has been named the Design Consultant Shell (DCS) because it facilitates the coordination of a group of distributed software tools into a manufacturing consultant for the designer, will be described in detail. Additionally, a case study in which the two process planning agents have been added to the DCS and used to generate toolpaths for a base plate designed in a common CAD system will be described.

The Core Process Planning Agent

The first process planner, which has the goal of minimizing machining time, has been developed as part of the CyberCut system (Dornfeld, *et al* 1999). The core of this system consists of a process planner comprised of the macroplanner, microplanner and the tool path planners. The process planner is a *feature-based* system (i.e. it accepts manufacturing features as input and works almost exclusively with these features).

Feature-based systems have two means of acquiring features (Shah & Mäntylä 1995):

- features may be extracted from a geometry description of the part.
- the part can be designed using manufacturing features.

The first approach is called feature recognition while the second is called design-by-features. CyberCut supports both methods to acquire features. Thus the designer can create a description of the part using a commercial CAD system such as SDRC I-Deas or Pro-E. The feature recognition module accepts a boundary representation of the component as an ACIS file. Any CAD system that can directly output a boundary representation in ACIS format or export the geometry to STEP or SIF (Solid Interchange Format) (McMains *et al* 1998) may be used to describe the part.

Alternatively, the designer may choose to describe the part directly with manufacturing features using the WebCAD interface (Kim *et al* 1999) and export the part description using the DSG (Destructive Solid Geometry) format. The output of the feature recognition system is a feature description of the part, while the DSG file is passed through a converter that transforms the DSG file to the same description as that of the feature recognition system.

The process planner is comprised of three main components: the macroplanner, the microplanner and the tool-path planner (ElMaraghy 1993). The macroplanner has

a global view of the component and concerns itself with the selection of setups, allocating features to setups, fixturing, setup and tool sequencing, generating high-level instructions to the machinist such as orientation of the components, and determination of datum faces. The microplanner plans for one feature at a time. It is responsible for the selection of tools for the machining of the feature, cutting parameters for the tool concerned, and collision detection of the chosen tools with the part and the fixtures. The tool-path planner determines the tool motions required to machine the material designated by the macroplanner and the microplanner and outputs the machine codes (G&M codes) for the CNC machine. All steps are performed in such a way as to reduce the total machining time as much as possible.

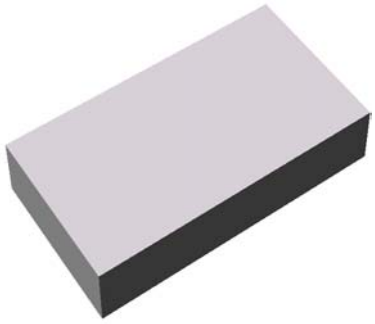
Burr Minimization Tool Path Planning Agent

Stringent requirements on having a precise edge necessitate deviation from the conventional methods of planning that are based on minimizing the total machining costs. A significant amount of research has been done on identifying and improving the parameters that affect edge finish (Hassamontr 1998, Chu and Dornfeld 1999). The edge precision planning system that has been developed (Rangarajan *et al* 2000) uses special tool path planning algorithms to avoid burr formation. The exit burr is the most critical type of burr to avoid in face milling. Exit burr formation is minimized by preventing the cutting edge of the tool from moving out of the workpiece while removing material (Chu and Dornfeld 1999).

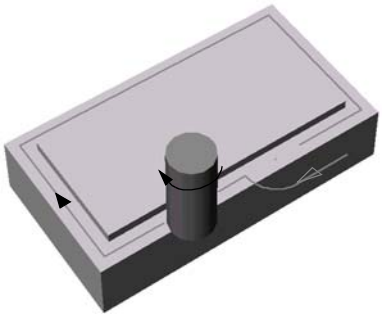
The core idea of tool path design for exit burr minimization is to mill the edges of the part first with special tool paths that remove the potential of subsequent tool paths to create exit burrs. Figure 1 illustrates how this is accomplished for a simple rectangular block. The first tool pass is along the edges of the part as shown in Figure 1 (b). Notice that the rotation of the tool is such that the leading edge enters the part (down milling) rather than exits the part (conventional milling). Figure 1 (b) also shows that the tool enters the part along an arc so that the cutting edge does not exit the stock. Additionally, the width of cut is adjusted so that the cutting edge does not exit the workpiece at the corners. For this example, this simply means that the width of cut is less than 1/2 the diameter of the tool. However, an algorithm has been developed to appropriately adjust the width of cut for more complex geometries. This first tool path leaves a small burr free step at the edges of the part. Then, the remaining material is removed from the face of the workpiece using a zig zag tool path as illustrated in Figure 1 (c).

At first glance, the total machining time required by this method appears to be more than that required using normal techniques. However because this method avoids exit burr formation an additional deburring step is unnecessary. Furthermore, since the additional tool paths do not require a setup change, there is an additional savings in the total time required for manufacturing the component.

(a):
Beginning stock.



(b):
Edge Precision Tool Path. Tool enters part along an arc and keeps width of cut less than 1/2 the tool diameter so that the cutting edge never exits the stock.



(c):
A final zig zag tool path removes the remaining material.

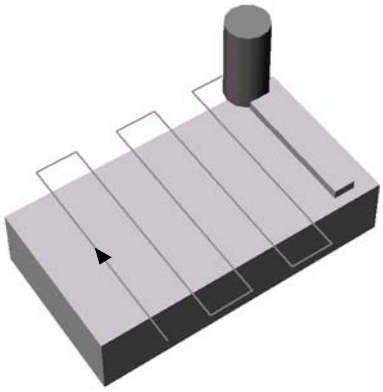


Figure 1: Illustration of Burr Minimization Tool Paths

Cost Estimation

The standard models of cost estimation for machining (Ostwald 1992; Dixon and Poli 1995) require that each machining feature be extracted from the CAD geometry, followed by estimates of the total length of the tool paths for each operation, and the calculation of the time for machining based on parameters such as feed rate and spindle speed. If no detailed process plan exists, this estimation process is very cumbersome and time consuming. However, if detailed process plans including tool paths can quickly be

generated, the cost estimation process is much easier and more accurate. Both the core process planner and burr minimization tool path planner provide tool paths and machining time information that can be used to calculate the cost of a part. The cost of a machined part can broadly be expressed by four categories. These are, (1) cost of raw material, (2) cost of tool wear for machining, (3) cost of setup/fixture, and (4) cost of machining which is converted from the time spent on machining (Dornfeld, *et al* 1999).

The cost of the part is given by Eq. (1).

$$C_{part} = C_{material} + C_{setup} + C_{tool} + C_{machining} \quad \textbf{Eq. (1)}$$

In general machining practice, there exists another component of cost, deburring cost, which consists of removing edge defects generated by primary machining. Whereas the costs of material, setup, tool, and machining are estimated for the CNC milling process, the deburring cost is based on the deburring process using deburring tools. For simple parts in a job shop environment, the cost of manual deburring is:

$$C_{deburring} = \frac{C_T}{N_P} + C_L(1 + D_o)t_{deburring} \quad \textbf{Eq. (2)}$$

where:

C_T: Cost of deburring tool including equipment and tool replacement

N_P: Number of parts deburred with the tool

C_L: Labor costs for deburring

D_0: Overhead costs for deburring

$t_{deburring}$: Time for deburring the part

The deburring cost can potentially be reduced to near zero by using the burr minimization planner, which minimizes initial burr formation.

At the same time, however, burr minimization planning requires additional CNC milling tool paths which increases the cost of the machining operation. The burr minimization cost consists of the cost of tool wear and machining for the additional tool paths.

$$C_{burr-min} = C_{burr-min-tool} + C_{burr-min-machining} \quad \textbf{Eq. (3)}$$

For complex parts, the burr minimization toolpaths may require another tool change. A term to account for this tool change could be added to Equation 3.

The cost of deburring, $C_{deburring}$, and cost of running burr minimization tool paths, $C_{burr\text{-}min}$, are compared to determine the cost of burr removal, $C_{burr\text{-}removal}$. The cost of burr removal is the minimum cost required to remove burrs either by deburring or by using the burr minimization tool paths.

$$C_{burr-removal} = minimum\{C_{deburring}, C_{burr-min}\} \quad \textbf{Eq. (4)}$$

Including the burr removal cost, the final cost becomes.

$$C_{total} = C_{part} + C_{burr-removal} \quad \textbf{Eq. (5)}$$

Overview of the Design Consultant Shell

In order for a distributed agent based system to function in an autonomous fashion, there must be a central coordination and communication framework in which the agents (or individual software applications) operate. This section deals with the structure of this framework. Agent-based DFM, manufacturability, and process planning systems have received much attention in the literature (Dabke *et al* 1998, Frost and Cutkosky 1996, Rajagopalan *et al* 1998, Karne *et al* 1998). Most systems have a rigid or closed infrastructure in that they can only incorporate agents which have been programmed expressly for use inside that particular system (Kashyap and Devries 1999). Some have incorporated sophisticated wrappers to incorporate legacy systems into the overall framework (Jha *et al* 1998). However, in general most systems in the literature are not very flexible in allowing for the addition of software tools that may not have been written specifically to integrate into the system.

It was our desire to create a central framework which would easily allow for the introduction of a variety of different kinds of software tools without requiring extensive re-coding of the framework or the individual software tools. This desire for a flexible framework led to the creation of the DCS. The discussion of the DCS will begin with an explanation of the structure, or different parts and functions, of the framework, followed by the syntax and language it uses, and finally the types of software tools that can be integrated into the framework.

Detailed Structure of the Design Consultant Shell

The DCS consists of two main parts. The Agent Coordination Center and the Agent Interface. The DCS uses JATLite (Jeon, *et al* 2000) which implements a router and router client architecture. The router can initiate contact with the router client or vice versa. (This architecture is a little different than a server client architecture where the client can initiate contact with the server, but the server does not initiate contact with the client.) Typically each agent has an associated router client. All communications from agent to

agent pass through the router which queues and routes all messages.

The Agent Coordination Center is a Java application which runs on the user's computer and implements the JATLite router. The Agent Coordination Center is devoid of any domain specific knowledge and is designed solely to administer the communication and data transfer between different agents and the user. Each agent has an associated Agent Interface, which is also a Java application. The Agent Interface runs on the computer hosting an individual software tool (or agent) that performs some domain specific function. See Figure 2. The Agent Interface receives requests and data from the Agent Coordination Center (these may originate from the user or another agent), calls upon the domain specific code to perform its function, and then returns results to the requesting agent. The Agent Interface may also send requests and data to other agents via the router.

The system is user-centric in that the Agent Coordination Center resides on the user's computer and builds and maintains a network of agents for the user. Note that in the future, the Agent Coordination Center should be converted to a Java applet which will reside on a web server and function inside a web browser, but this is not the case currently. Organizations that wish to make their domain specific software tools available would continuously run an Agent Interface application on their computer which would be open to connections from multiple users, each running their own Agent Coordination Center. In this way, each user sets up and connects to their own network of software tools. The Agent Coordination Center manages the details of this custom network.

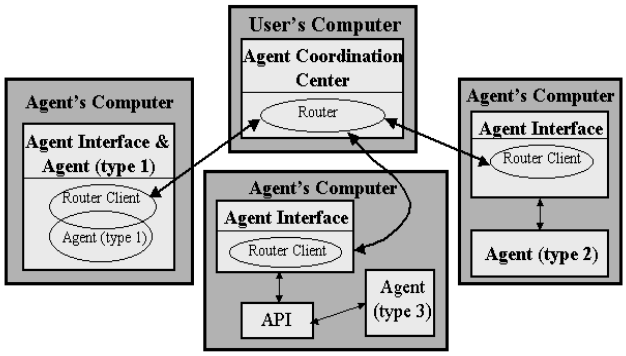


Figure 2: DCS Structure, showing connection between Agent Coordination Center and Agent Interfaces.

The long term vision is that many organizations would make their DFM, analysis, manufacturability, or process planning services available and the list of available services would be kept on a centrally located database. Each user's Agent Coordination Center would then query the database to present the user with available services. Since only a few software tools are being coordinated at present, this centrally located database is not implemented.

The Agent Coordination Center and Agent Interface applications send requests and confirmations using a text syntax based on KQML (Finin *et al* 1994) The standard format of a KQML message in the context of the DCS looks like the following:

(*performative* :sender *sending_agent_name* :receiver *receiver_agent_name* :files *list_of_files* :idNum *id_number* :content (*content of message*))

The "performative" argument tells what kind of message this is (e.g. "reply", "error", etc.) and lets the "receiver" (either an Agent Interface or Agent Coordination Center application) know how to proceed. The "list_of_files" argument notifies the receiving agent of any data files that are needed to perform the requested action. The Agent Coordination Center will then connect to the computer hosting the Agent Interface using the ftp protocol and send the files to (or receive the files from) the Agent Interface application. The remainder of the arguments are self explanatory.

As previously stated, the desire was to develop a flexible framework which could easily integrate a variety of different types of software tools. Therefore, a brief classification of the types of software tools that can be integrated into the framework will be given, followed by an explanation of how one would proceed to add them. Software applications have been grouped into four different categories which encompass most software that one could conceivably use as a remote agent. The categories are as follows:
Type 1 - Software designed or modified specifically to integrate into the DCS framework (or some other agent framework).
Type 2 - Software which can be run from a command line without user interaction.
Type 3 - Interactive software with a clearly defined Application Programming Interface (API).
Type 4 - Interactive software with no clearly defined API.

Software of *Type 1* is obviously easily integrated into the system. Software of *Type 2* would include many feature recognition and process planning routines which typically run from a command line without graphical user interaction and take simple text or file inputs and similarly return text and file outputs. Software of this type is also easily integrated into the DCS framework. About 50 lines of Java code need to be added to the Agent Interface Application which allow it to run and access the output of software of *Type 2*. Detailed instructions on how to do this along with some examples have been documented (Roundy 2000).

Software of *Type 3* can be integrated, but takes more work. An example of this type of software is the SDRC I-Deas CAD system which is highly interactive, but has a clearly defined CORBA based API. Code specific to the application's API and the functions which need to be performed must be added to the Agent Interface in order to integrate an agent based on this type of software into the system. A *Type 3* agent has been implemented in the DCS, but requires a little more programming. Finally, software of *Type 4* cannot be added to the system as there is no way to access the program's functions from another software application. A schematic showing agents incorporating software of *Types 1, 2,* and *3* integrated into the DCS is shown in Figure 2.

Case Study

The two process planning agents described above were added to the DCS and used to generate NC toolpaths for various parts. Each process planner along with its associated Agent Interface ran on a separate server. The Agent Coordination Center was run on the user's computer. It should be noted that the system was tested with the Agent Coordination Center running on a computer that was outside the Local Area Network of the process planners. Figure 3 shows the user interface for the Agent Coordination Center. As can be seen in the figure, each agent has a list of tasks that it can perform along with an explanation of the input required and output returned by each task

.

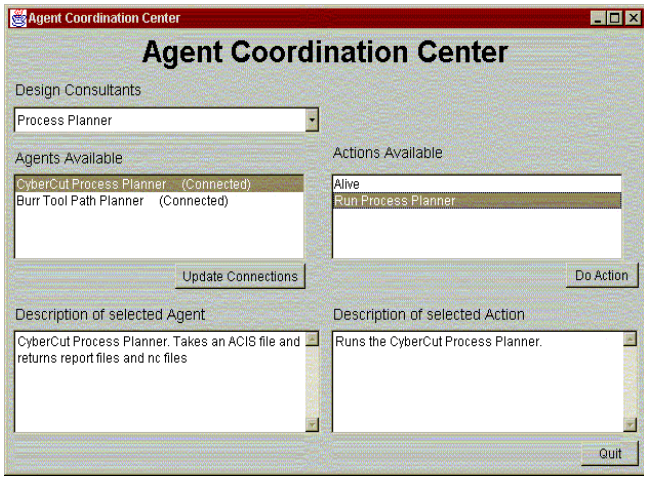


Figure 3: UI for the Agent Coordination Center showing list of agents and actions which each agent can perform.

One of the parts submitted to the process planners is the base plate shown in Figure 4.

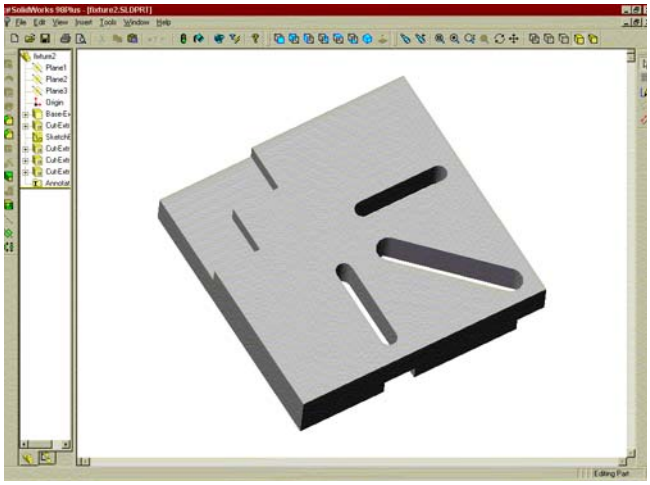


Figure 4: Fixture Part

Both process planners take an ACIS file and return NC files along with some text explaining the results and some cost analysis data. This particular part was designed in the SolidWorks CAD system which can directly output an ACIS file. However, it could have been designed in any CAD system that can output the STEP format, which the DCS can translate into ACIS.

Figure 5 shows the cost analysis data returned by the standard process planner. It should be noted that the cost model used does not include any overhead or profit, and that the footprint of this part is only 2 inches by 2 inches. A labor rate of $29 per hour was used to calculate the cost of machining time. The additional cost to deburr this part, given by Eq. (1), is $0.64.

Figure 5: Cost Analysis Results

The burr minimization process planner generates additional tool paths that will alleviate the need for deburring operations. Figure 6 shows these additional tool paths developed according to the burr minimization procedure detailed in Figure 1. Based on the time to perform these additional toolpaths, the additional cost of using burr minimization tool paths is $0.56. Therefore, it can be seen that it may be advantageous to use the burr minimization tool paths as part of the machining process.

.

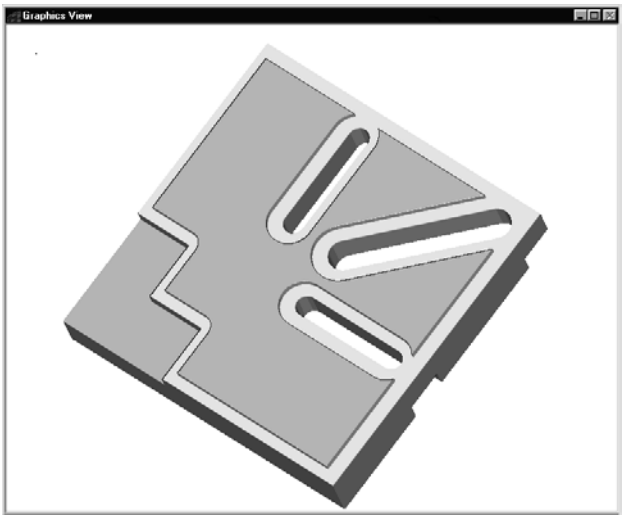


Figure 6: The light gray shows the location of burr minimization tool paths. The tool pass along the edges removes material likely to form burrs in subsequent passes as detailed in Figure 1(c).

Conclusions

1. A central framework for the integration and coordination of distributed agents has been developed. The framework has been designed to be flexible enough to easily allow the addition of a wide variety of remote agents with little or no re-coding of the communication infrastructure.

2. This framework has been demonstrated by integrating two process planners for 3-axis milling. The first process planner generates tool paths which attempt to minimize the machining time. The second process planner generates additional tool paths which will prevent the formation of burrs. The integrated system allows a user to access the process planners remotely and generate toolpaths for their part.

3. Cost related data is returned from both process planners. A cost estimation module evaluates this data and provides cost estimates to the user so that he/she can decide which tool paths are most appropriate for the given part.

Acknowledgments

The authors wish to acknowledge the National Science Foundation, NSF grant DMI-9908174, and the members of the Consortium on Deburring and Edge Finishing (CODEF) for their suggestions and their work. For more details visit http://kingkong.me.berkeley.edu and http://lma.berkeley.edu.

References

(Chu and Dornfeld 1999) Chu, C.H. and Dornfeld, D.A., "Tool Path Planning for Exit Burr Minimization by Estimating the Total Length of Primary Burrs," to appear in *International Journal of Computer Integrated Manufacturing*, 2000.

(Chu and Dornfeld 1999) Chu, C.H. and Dornfeld, D.A., "Tool Path Planning for Avoiding Exit Burr", *Journal of Manufacturing Processes*, vol. 2, No. 2, pp. 116-123, 2000

(Dabke, *et al* 1998) Dabke, P., Cox, A., and Johnson, D., "NetBuilder: An Environment for Integrating Tools And People", *Computer-Aided Design*, Vol. 30, No. 6, pp. 465-472, 1998.

(Dixon and Poli 1995) Dixon, J. and Poli, C., Engineering design and design for Manufacturability, Field Stone Publisher, 1995.

(Dornfeld, *et al* 1999) Dornfeld, D., Wright, P.K., Wang, F.C., Sheng, P., Stori, J., Sundararajan, V., Krishnan, N., and Chu, C.H., "Multi-Agent Process Planning for a Networked Machining Service", *Transactions of the NAMRI/SME*, v XXVII, pp. 191-196, 1999.

(ElMaraghy, 1993) ElMaraghy, H.A., "Evolution and Future Perspectives of CAPP", *Annals of the CIRP*, vol 42/2, pp. 739-751.

(Finin, *et al* 1994) Finin, T.; Fritzson, R.; McKay, D.; McEntire, R., "KQML as an agent communication language", *Proceedings of the Third International Conference on Information and Knowledge Management*, pp. 456-63, 1994.

(Frost and Cutkosky 1996) Frost, H.R., and Cutkosky, M.R., "Design for Manufacturability via Agent Interaction", *Proceedings of 1996 ASME DETC/DFM*, August 18-22, Irvine, California, 1996.

(Hassamontr 1998) Hassamontr, J., "Edge Finishing Planning in Milling", Ph.D. Dissertation, University of California at Berkeley, Department of Mechanical Engineering, 1998.

(Jeon, *et al* 2000) Jeon, H., Petrie, C., Cutkosky, M.R. "JATLite: a Java agent infrastructure with message routing." IEEE Internet Computing, vol.4, (no.2), IEEE, March-April 2000. p.87-96.

(Jha, *et al* 1998) Jha, K.N., Coen, G., Morris, A., Mytych, E., and Spering, J., "Agent Support for Design Manufacturability", *Proceedings of the 1998 ASME DETC/ DFM*, Atlanta, Georgia, 1998.

(Karne *et al* 1998) Karne, R.K., Dandedar, S.V., Poluri, S., Chen, G., Baras, J.S., Nau, D.S., Ball, M.O., Lin, E., Trichur, V.S., and Williams, J.T., "Web-it-Man: A Web-Based Integrated Tool for Manufacturing Environment", *Proceedings of ASME DETC/CIE*, Atlanta Georgia, 1998.

(Kashyap and DeVries 1999) Kashyap, S. and DeVries, W.R., "An Agent-Based Mechanism for Manufacturability Evaluation in the Product Realization Process", *Transactions of the NAMRI/SME*, v XXVII pp. 185-190, 1999.

(Kim, *et al* 1999) Kim, J.H., Wang, F., Sequin, C.H., and Wright, P.K., "Design for Machining over the Internet", *Proceedings of 1999 ASME Design Engineering Technical Conference*, vol 4, pp. 13 -22, 1999.

(McMains, *et al* 1998) McMains, Sara, Sequin, Carlo, and Smith, Jordan, "SIF: A Solid Interchange Format for Rapid Prototyping," *Proceedings of the 31st CIRP International Seminar on Manufacturing Systems*, May 26-28, 1998

(Ostwald 1992) Ostwald, P. F., Engineering Cost Estimating (3rd ed.), Prentice Hall, 1992.

(Rajagopalan *et al* 1998) Rajagopalan, S., Pinilla, J.M., Losleban, P., Tian, Q., and Gupta, S.K., "Integrated Design and Rapid Manufacturing Over the Internet", *Proceedings of ASME DETC/CIE*, Atlanta, Georgia, 1998.

(Rangarajan 2000) Rangarajan, A and Chu, C.H. and Dornfeld, D.A., "Avoiding Tool Exit in Planar Milling by Adjusting Width of Cut", *Proceedings of the ASME*, Manufacturing Engineering Division, ED, Vol 11, pp1017-1027, 2000

(Roundy 2000) Roundy, S., "The Design Consultant Shell: A Framework for Integrating Distributed CAD/CAM Tools into a Unified Design and Manufacturing System," MS Thesis, Department of Mechanical Engineering, University of California Berkeley, May 2000.

(Shah & Mäntylä 1995) Shah, Jami, J., and Mäntylä, Martti, 1995, Parametric and Feature-Based CAD/ CAM- Concepts, Techniques and Applications, A Wiley-Interscience Publication, John-Wiley & Sons, Inc

(Wright, *et al* 2000) Wright, P.K., Dornfeld, D., Wang, F.C., and Chu, C.H., "Decision Making in a Multi-Constraint Agent-Based Process Planning System", *Transactions of the NAMRI/SME*, v XXVIII, pp. 293-298, 2000.