This is to certify that the

thesis entitled

Internet-Based Design/Manufacturing Process Management

presented by

Hua Gu

has been accepted towards fulfillment
of the requirements for

Master's degree in Mechanics

[illegible]
Major professor

Date Nov. 9, 1999

Internet-Based Design/Manufacturing Process Management

By

Hua Gu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Material Science and Mechanics

1999

ABSTRACT

INTERNET-BASED DESIGN/MANUFACTURING PROCESS MANAGEMENT

By

Hua Gu

Competition in mechanical engineering today demands ever-faster product development, hence ever-shorter design cycles, which may be achieved by efficiently rearranging design and manufacturing processes. The necessity of such management depends on adjusting design process to accommodate constraints and frequently updating them to make full use of the new design tools and methodologies. Meanwhile, it is also a trend that design and manufacturing processes or activities are modularized and integrated in a geographically distributed environment.

In response to the growing demand for a robust infrastructure that coordinates design and manufacturing activities, this thesis presents a web-based engineering framework, which coordinates various CAD/CAM and FEM systems, provides product data relationships, and integrates with manufacturing processes. In this thesis, the current capabilities of the framework are demonstrated in conjunction with benchmark studies on functionally gradient materials (FGMs) design, and the design and manufacturing of a gearbox. In FGMs design, a design approach is represented using the framework to obtain the distribution of the reinforcing phase that minimizes the stresses in axisymmetric bodies subjected to temperature gradients and pressure differentials. In the design and manufacturing process of a gearbox, the framework autonomously performs a series of stress and fatigue analyses and searches for a set of parts that satisfy various requirements. It is also verified that the framework is capable of integrating seamlessly the design results with manufacturing operations by selecting cutting tools, machining conditions, and sequence of operations. The framework enables users to check the results ranging from graphical display of the design to the CNC code generated to machine the designed shafts.

To my wife Ying

Acknowledgements

I would like to take this opportunity to express my appreciation to a number of people, without whom my thesis could never have been completed.

First, my sincere gratitude goes to my advisor, Dr. Patrick Kwon. Dr. Kwon has been my advisor since my enrollment at Michigan State University. He has taught me many things in mechanics and manufacturing. He provided me with valuable guidance, the influence of which on my personal and technical development will be carried forward into my future endeavors. Second, I cherish very much the advice from my co-advisor, Dr. Moon-Jung Chung. Dr. Chung provided valuable advice in the computer science domain helping me with my cross-disciplinary research. I would also like to thank Dr. Ronald Averill for giving me advice and encouragement, and for serving as member of my thesis committee.

It is also a very pleasant experience to work with my fellow students, Dave Keyes, Behr Markus, Kuk-Jin Lee, and Lihua Chen. I sincerely appreciate the help, encouragement, and friendship from them.

Last, but not least, I proudly share this accomplishment with my wife, Ying Sun. I would also like to thank her for her everlasting support, patience, and love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Design

What is design? A dictionary shows a definition which stated that to design is to fashion after a plan. The essence of a design process is to create something, in one way or another, that has never been. In most cases, an engineering designer practices design following that definition[1].

From a general point of view, most design problems are typically *open-ended* and *ill-structured.* Design problems are *open-ended* because they usually have more than one acceptable solution. Unlike in many mathematical and analytical problems, the quality of uniqueness, or best solution, simply does not apply. Design problems are *ill-structured* because their solutions cannot normally be found by routinely applying a mathematical formula in a structural way[2]. Design problems possess these two characteristics because the challenge imposed on design is not only simply to create something that accurately reflects the existing domains. It is also necessary to provide for the creation of new domains and to simultaneously produce and transform objects and regularities of the world that we are concerned with[3].

Given that design has such characteristics, it is no surprise that no single universally acclaimed sequence of steps properly describes a workable design process. However, we still can see that most design processes share certain common steps in its approaches or strategies which include:

. Recognition of a need

. Definition of a problem

. Synthesis and conceptualization

. Analysis and optimization

. Evaluation

. Presentation of the optimal Design

Since we have mentioned that design problems are *ill-structured*, the above steps may be carried out either sequentially or simultaneously . Moreover, feedback of each step may lead to an iteration which is akin to most kinds of design processes.

1.2 Engineering Design

In general, we can call any design problem which is involved in any engineering activity an engineering design. It is perhaps the most important and biggest subset in the design realm.

Engineering design has some unique features. At very abstract level, we can adopt the definition articulated by Dym and Levitt:

Engineering design is the systematic, intelligent generation and evaluation of specifications for artifacts whose form and function achieve stated objectives *and satisfy specified constraints*[4].

From a practical and simplified point of view, engineering design can be properly illustrated using the basic module by M. Asimow [5]. He viewed the essential aspect of the design process as consisting of the elements shown in the figure(Figure-1) below[1]. A design problem, especially an engineering design problem, can be decomposed into basic modules shown in Figure-1. We adopt this simple model of design process because this very model makes it possible to divide a complex design process into small elements which inherit all general features of design. Thus, computer-based automated management of large design activity is possible because computers are capable of dealing with inheritance.

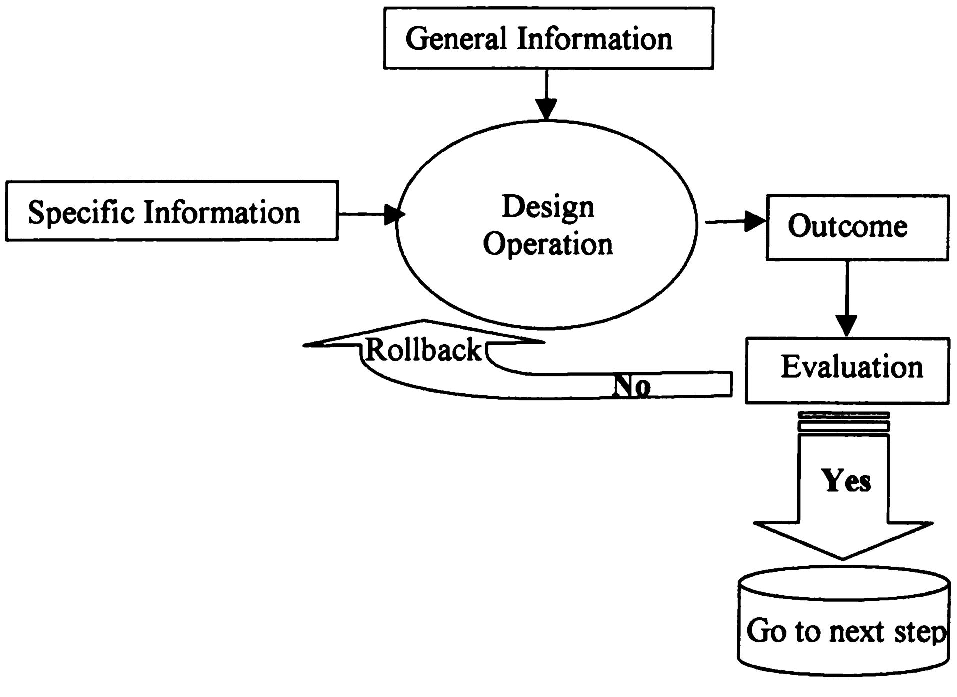

Figure-1. Basic module in the design process (*after Asomow*)

1.3 Concurrent Engineer

Today, design has become a multi-threaded process, which carries more than one child design process concurrently. This very characteristic of modern design requires teamwork approach which is called concurrent engineering(CE) in the creation or improvement of new products. Generally speaking, concurrent engineering, also called simultaneous engineering, is a design approach which incorporates product design and product manufacturing in an intimate way[6]. However, the concurrent engineering concept is more than simply to integrate design and manufacturing at the start of the design process . It can be illustrated using Dieter's Model(Figure-2) of concurrent engineering[1].

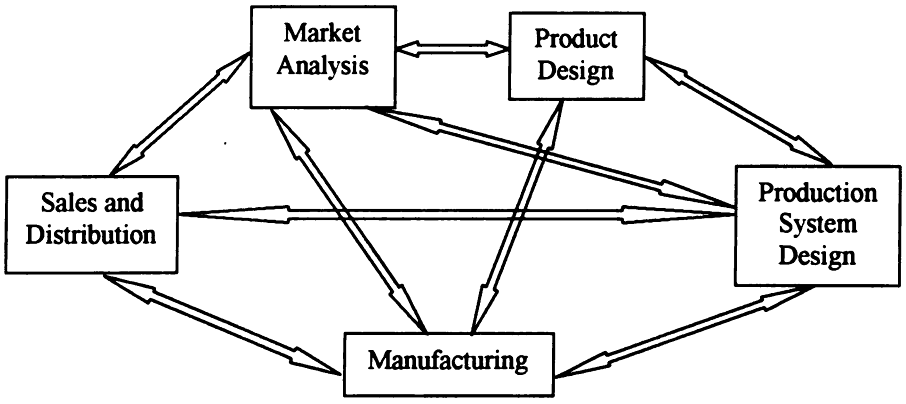

Figure-2. Model of Concurrent Engineering (after Dieter)

According to this model of concurrent engineering, product development should integrate design with other tasks such as manufacturing planning, quality, and marketing.

Products are developed more quickly and often at a lower cost with higher quality by simultaneously performing these tasks along with the use of modern design tools. This approach addresses issues pertaining to a product's entire life cycle. It also typically includes the use of computer tools like computer-aided design, shared databases, and computerized standards to help bring about efficient and accurate communications and to automate the design process as much as possible. In other words, today, concurrent engineering is usually computerized. Computer, as a machine, requires logical or mathematical modeling of a desired product. Thus concurrent engineering's success is heavily dependent upon the availability of a good standardized description of the components of a product[7].

1.4 Manufacturing Process

1.4.1 Introduction

Generally speaking, even with the use of advanced technology, the approach to product development still involves several steps, each of which must be completed before the next step in design can begin. These steps are listed as follows.

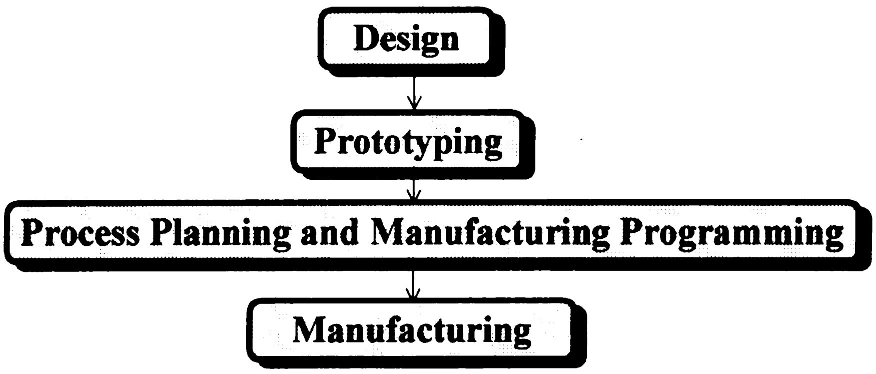

Figure-3. Product Development Steps

At the present time, using the framework implemented in this research, design can be accomplished to such a level that a 3-D solid model can be visually displayed. At the same time, it can convert the model to Stereo Lithographic (STL) format for rapid prototyping. Furthermore, after the tool path has been determined, cutter location data or even the NC codes (still premature) can be generated according to the tool path information for machining. Solid modeling is the step that unifies design and manufacturing seamlessly. Solid model data can also be used as the inputs for a finite-element analysis program. It is also used to conduct generative machining to produce numerical machining code. In this way, design and manufacturing becomes one integrated process.

1.4.2 Machining Process Automation

As one of the most commonly used processes today, machining process automation has gained its popularity. There is a strong incentive for manufacturers to increase the level of automation within their operations and by taking advantage of the available software tools. As a way to enable the operation of machines by using a series of computer-based instructions comprised of numbers, letters, and other symbols, numerical control (NC) makes machines run consistently, accurately, predictably, and essentially automatically. Based on the design information, NC programming process mostly and typically consists of five major functions:

1. Creating the geometric model (rapid prototyping can also be done using this model);
2. Establishing the machining process plan;

3. Generating the toolpath;

4. Post-processing;

5. Simulation and verification.

1.5 CAD, CAM and Integrated CAD/CAM

The last two decades have witnessed extensive infiltration of computer technologies into the design and manufacturing activities in engineering domains. Starting with simple computer-aided drafting, this process has proceeded towards various levels of sophistication in automated design and manufacturing systems.

The purpose of this section is to show readers some basic underlying concepts of CAD, CAM and integrated CAD/CAM processes. Since the framework also has strong impact on automated mechanics analysis processes, main features of mechanics analysis are presented.

1.5.1 Computer-Aided Design (CAD)

Computer-aided design is aimed at the engineering design functionality, rather than the less creative drafting function of the industrial enterprise. To a certain extent, modern computer-aided design software usually provides users with drafting utilities with a capability for engineering analysis. A modern computer-aided design system is expected to incorporate a set of analytical tools as well as extensive graphing utilities to fulfil the needs of both design and drafting functions. Implementation of such system will inevitably lead towards the full integration of the entire design-to-manufacturing

(DTM) process [8]. In an integrated system, a very direct advantage is that the geometry created in the CAD process can be used as a basis to carry out other functions in computer-aided manufacturing, which will be discussed hereafter. This helps engineers to avoid the waste of time as well as errors resulting from having to re-define the geometry from scratch.

1.5.2 Computer-Aided Manufacturing (CAM)

The result of computer-aided design or any other designing processes is a virtual product. CAM deals with the physical work done on the factory floor through which the virtual product is finally turned into a real product. Factory equipment may consist of robots, multiple-axis machine tools, and programmable controllers, which can be flexibly configured to produce different parts and interconnected to set up a computer-integrated manufacturing (CIM) system. Numerical control (NC) is one of the most mature areas of CAM[9]. The numerical control was introduced much earlier than the introduction of modern CAD and CAM systems. Computerized-numerical-control (CNC) and distribution-numerical-control (DNC) are two derivatives of the NC technology. In a CNC mode, a lathe or a milling machine is controlled by a simple computer using NC code. In DNC mode, an NC-controller receives its program via a communication line hooked up to a remote computer. Originally, the development of CAM systems were mainly conceived to produce the voluminous NC code for numerically controlled machining. Advanced CAM systems have features like assisting the task of process design and implementation of designing modifications, rather than merely generate the NC code. Now, through computer graphics, tool-path motions can be easily verified.

Computer can thus generate NC code based on the geometric data from the CAD database. In fact, many CAD/CAM systems have this capability routinely built into their software packages.

1.5.3 CAD/CAM Integration

Unfortunately, CAM systems have been evolving independently of CAD systems. The separated evolutions resulted in a formidable barrier to the further development of the concurrent systems [10]. Most of the existing CAD systems require high human interaction. Those systems are not designed to be parametric and rule-driven. Responsibility for detailing, dimensioning, tolerancing, and rule-checking is largely under the control of the computer-assisted draftspersons for whom, errors are inevitable to make. To eliminate human errors, interface between CAD and CAM systems was developed to deal with the conversion and transmission of detailed design data. In order to implement such a well-interfaced seamless design-to-manufacturing(SDTM) system [11], a preferred approach is to make the system parametric and rule-based. In parametric and rule-based design, designers start with parametric relationships in the prospective design as well as applicable rules and end up with the design as parametric-mathematical representations plus its geometrical model. The procedure of such a designing approach is ideal for computerization and automation. In our research, we build a web-based framework which enables the merger of CAD and CAM processes distributed over the internet.

1.6 Mechanics Analysis Process

Mechanics analysis is, to a certain extent, indispensable in most engineering design and manufacturing processes. It shares many common aspects with modern design and manufacturing processes. Most mechanics analyses can be represented either analytically or numerically by mathematical models, which are highly parametric. In other words, most mechanics analyses can be dealt with parametrically. A large-scale mechanics research project may be conducted in different locations using different approaches. For example, an EU (European Union) pressure vessel solar heating research project, may be carried out using analytical methods, FEM and experimental methods in different places for a number of solar conditions. That is to say, researchers in several locations should deal with various cases using different methodologies while sharing different aspects of the project. Thus, communications between different participants is important. Based on these facts, process management methodologies applicable for engineering design and manufacturing processes are suitable for most mechanics analyses and researches.

1.7 The Internet

1.7.1 Data Communication

Imagine a scene like this: a human operator goes to a machine and reads the display from its controller, and then goes to another machine to key in the readings into its controller. This typically happens where machines cannot communicate directly.

Since most controllers are computerized, point-to-point communication between them is possible and is not too expensive to implement. Such one-to-one data transmission can be accomplished using serial or parallel communication. In short, serial communication allows the 8 bits of a character to be transmitted through one single wire while in parallel communication the 8 bits of each character travel simultaneously on 8 wires. Parallel communication is faster than serial communication.

Although these two forms of data communications are inexpensive to implement, they are limited by their communication speed as well as the point-to-point communication topology. Think of a design group with its personnel as well as its CAD and CAM software distributed on different shop-floors or even on significantly distant locations, computer networking would be a wiser and better choice to handle such data communication.

1.7.2 Networking and Internet

In our research, we set up an Internet-based framework to carry out CAD/CAM process management. Before we cast our sight upon the Internet, let us take a look at the basics of the computer networking.

Local area networks (LANs), are privately-owned networks within a single building or campus of fairly limited size. LAN links computers within a limited area so that they can share files or printers. One computer, which is called server, can provide disk space for other computers. The topology of LANs, or in other words, the way in which machines are hooked up, is usually ring, star(with one of the computers at the center), or bus(all in parallel).

WAN, which stands for wide area network, spans a large geographical area, often a country or continent. Each computer on the network is called a node or end system and is frequently called on to pass along messages to other computers. WAN allows users to transfer files, mail electronic messages, and remotely log on to other computers. Each end system can either be a machine inside a LAN which is hooked up to a switching computer (or router) of the WAN or be directly attached to the router.

A internetwork is a collection of interconnected networks. It is often used to refer to a collection of LANs connected by a WAN. The Internet means a specific worldwide internetwork connecting universities, government offices, companies as well as individuals.

In other words, Internet means "network of networks". It dates back to the old ARPAnet which stands for the US Department of Defense's Advanced Research Project Agency network[7]. Now Internet is no longer limited within the military domains. Basic services of the Internet provides include: electronic mail, interactive conferences, access to information resources, network news, and the file transfer utilities.

Chapter 2

Process Management for Design/Manufacturing and Mechanics Analysis

2.1 Introduction

Complex high-tech products are designed with increasing design are manufacturing concerns by processes distributed over time, geography, participants, and functionality. Design of any complex commercial artifacts or design of a process of a complicated modern mechanics analysis may require a group of people possessing different specialties spread over several continents with a time span of several decades. Since such processes are multi-dimensional (time, geography and participants are regarded as different dimensions), integrated and interdependent, effective coordination within each dimension and between different dimensions is the essence to the success of such processes. In a traditional design and manufacturing environment, the complexity of such interdependency will result in a coordination that incurs great amount of redundant expenditures as well as delayed schedules, hence reduced quality of the product although individual productivity may be relatively high.

Modern process management often needs to have the following features: 1)Participants involved in the process should be able to share information in all dimensions. 2)All participants should update descriptions of the shared design/manufacturing process timely. 3) Ensuring that the joint actions between each party of a shared design/manufacturing process are coordinated to achieve the desired results as efficiently as anticipated.

The systems implementing the current process management technologies are listed as follows according to the capabilities of which they impose constraints on how a process can be performed [18] (please refer to Figure-4). Traditional manual systems hardly impose any constraints with regard to how a process should be carried out. Ad hoc systems are largely dependant upon the users' decisions. It is basically an email-based system used to route a work package once its destination is determined. Both systems provide tracking utilities. Some structured conversation systems are designed based upon the contract net [19] or speech act approach [20]. They have a system-imposed protocol that allows them to arrange the distribution of the tasks. Workflow systems set up a process flow using text-based or drawing tools. Once the process flow is set up, the participants in the process simply perform their respective tasks.

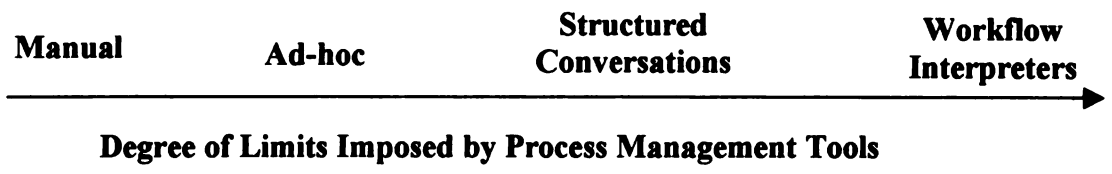

Figure-4. Types of Process Management Systems (after Klein)[7]

2.2 Automated Design/Manufacturing Process and Mechanics Analysis

In a complex process, there may be a number of tasks which are inter-dependent. These inter-dependencies can be represented by data specifications and transitions. Data specifications are usually data file stored somewhere on-line. Transition shows direction of the data flow. The whole process may be so complicated that it can be split into smaller sub-tasks. In some cases, for each task or even sub-task, there may be different approaches available for the participants to complete the task or sub-task. There are also constraints for each task which shape the process. Most manual and structured conversation systems do not support the imposition of such constraints, nor can these systems model how the tasks are sequenced. It is the human participants who make all the decisions in complex process that lead to the full execution for the whole process.

Workflow systems are able to impose constraints and we can define a process flow in a very detailed way using such systems. However, such systems model a process in a fixed or rigid manner, which means such a way of modeling would become inefficient should there be more than one approach or should any change occur. Moreover, just like the structured conversation systems, these systems are also merely electronic schedulers. All tasks are performed manually by individuals or a group or groups of people which could be located at one place or scattered all over the world. In other words, no matter what kind of existing process management system you use, the execution of the whole process flow is continuous and fully automatic.

One of the goals of the research is to develop a new approach to automate the execution of a distributed design/manufacturing process. Based on the basic element discussed in the first chapter, a simpler and more generalized elementary model was derived for both design and manufacturing. In order to automate the whole process, we need to have a task representation which is corresponding to a "design operation" or any operation inside a mechanics analysis. The task representation is connected to an executable program which is encoded to perform that task. As mentioned before, both general information and specific information is consumed by a task. Information can be interpreted as data, hence input data specifications are used to replace the general information and specific information. In an automated system, differentiating the specific information from the general information is a redundant step and it is advisable to unify these two to allow for more flexibility. Since there may be more than one information source, it is rational to have multiple data input representation for a single task. Direction of the input data is represented by a transition arrow, which shows where the data comes from and how a task converts them into other data. The "outcome" in the previous model is replaced with output data specifications. Just as the input, output may need to reside in more than one data specification. In some cases, different portions of the output data are consumed by different tasks corresponding to different programs located either on the same system or on other systems. Dividing the output into smaller data blocks will save the time spent on data transmission. It may also lessen the amount of time for the destination software to process the data, hence better performance. The "evaluation" function in the previous model is implicit here. It could be incorporated into the tool if it is fairly "rigid", which means it does not tend to be subject to change very often. On the contrary, if it is flexible, the system will provide the user with pre- or post-evaluation utilities to conduct the evaluation. It is also the pre/post-evaluation utilities that enable the user to impose constraints that shape the process.

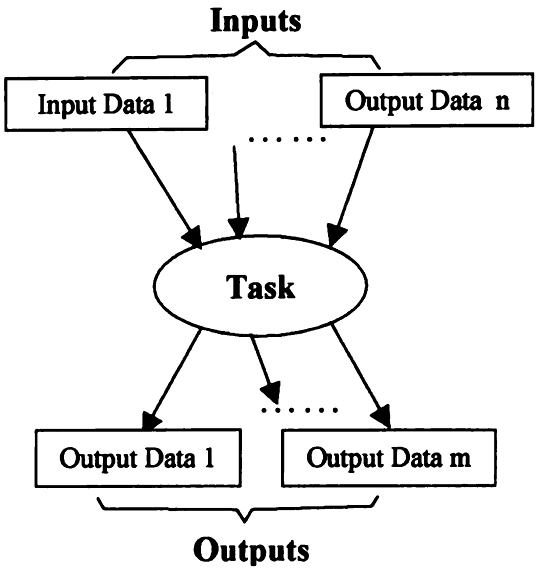

Figure -5. Automated Design/Manufacturing Element

The system leaves it to the user to determine, either at their own convenience or based on the nature of the constraints, whether it is hard-coded or imposed using the system's pre/post-evaluation functionality. As to the feedback loop and the alternatives for participants to perform a task, the system provides managerial tools to implicitly implement the feedback and choose alternatives. The major managerial tool is a linear scheduler used to organize the execution of the whole process.

Since we have identified the design process element with its manufacturing counterpart, we may easily set up a seamless design-to-manufacturing process flow diagram based on the above-mentioned model. It is also very easy to make use of this model to represent any mechanics analysis process not only because the mechanics analysis is similar to design process but also because it is discipline-independent. The system has build-in tools and graphical interface to help the user to set up and execute a process. Most of the traditional process management systems are just managerial tools or schedulers which require engineers to perform a task embedded inside a process flow. The system implements a design/manufacturing automation. Each node inside a process flow is associated with either a tool software or a data file containing information. The human participants in traditional process management has been replaced by software tools which are typically parametric. Remember, as is mentioned before, it would be ideal for us to implement an integrated CAD/CAM system parametrically. Since we are conceiving a process management system that is able to automate globally distributed design and manufacturing processes, we need to design system which is platform-independent. We will address this issue in the forthcoming section.

2.3 Why Java

In a modern a company, different departments are involved in a design process. It is rational to assume that each department corresponds to a task which can either be a decomposable task or an non-decomposable elementary task. In order to perform those tasks, each department has its own software tools running on their own systems. Based on the design/manufacturing model we have discussed before, most software tools need to have input data. Unfortunately, these input data could be output data generated by other departments or could be stored on other systems. The other departments may not necessarily be located at the same place. Meanwhile, different departments may use different operating systems. The goal of this research is to design a process managerial system capable of executing embedded tools on various kinds of operating systems. Therefore, the system itself should be able to run platform-independent. From an end-user's point of view, people tend to pick up their own machine or any machine with internet utility to run their process management. It is unnecessary to design platform-independent tool software. It is wise to set up a process management system that runs on-line and is able to invoke software tools stored on any other internet-connected systems. Based on those facts, Java provides the important characteristics to develop our system.

Initially called "Oak", Java got its name in 1995 and was published the same year. Originally, Java was designed to be a platform-independent language used to create software embedded in consumer electronic devices. By 1993 the Java design team realized that it could be adapted for internet programming and switched. It was a wise switch that led to the success of Java. Now, just as what C was to operating systems, Java has become a language for the Internet.

As a programming language, Java can generate two types of programs: applications and applets. An application is an executable program running locally. It is quite similar to those programs created by C or C++ . However, it has a very distinct feature: it works in a Java virtual machine which makes the program platform-independent. By analogy, the Java virtual machine is somewhat like an interpreter while the Java program and the operating system are two persons the interpreter is talking to. No matter what kind of language the operating system is using, this interpreter always speaks "Java" with the Java program. Just as applications, Java applets are also platform-independent. However, without applets Java would not be very different than other programming languages. An applet is a program transmitted through Internet and run by an Java-based web browser. It is a small piece of program dynamically downloaded via the Internet. Unlike image files, sound files or video clips, Java applets are executed dynamically so that they are highly user-interactive and changing dynamically in response to the users' input.

Making full use of these features of Java, our system is also designed to be platform-independent and is expected to be put on-the-fly over the Internet available to be executed on any operating system. Therefore, platform-dependent CAD/CAM software tools may be embedded regardless of the operating system. The participants are free to choose their own programming language because our Java-encoded system will invoke them on their own operating system. Java is also outstandingly capable of handling networking communication. Data consumed or generated by the software tools are delivered or directed by our Java-encoded process management system. Details of the communication will be discussed in the next chapter.

Based on what features are intended to have for the framework and what Java is capable of doing, it is chosen to implement our system.

2.4 State of the Art

Advancement in the information technology has infiltrated into many aspects of our life. Collaborations between scientists and engineers are expected to be linked by high performance Internet services such as Internet II and the Defense Research and Engineering Network (DREN). Based on the latest information technology, DOD has launched a number of programs to develop process management infrastructure: Rapid Prototyping of Application Specific Signal Processor (PASSP) [12], Manufacturing Automation and Design Engineering (MADE) [13], and RaDEO (Rapid Design Exploration and Optimization) [14]. To a certain degree, Internet has been used as a medium to manage design and manufacturing activities [15]. It serves as a pool of design information such as CAD drawing, analysis techniques or a list of off-the-shelf parts [16]. A framework designed to manage distributed business and electronic commerce on the information superhighway was developed by Andreoli et al [17].

2.5 The MIDAS System

In real practice, CAD/CAM processes arise when a given task is too large to be performed effectively by a single individual and thus must be divided into sub-tasks to achieve the top-level goal. Tasks in such a process may require some synchronization in task execution. A rational execution mechanism must be implemented to avoid contention for the use of some shared resources. Since a process is composed of a number of high-level tasks and low-level sub-tasks, it should be decomposed while it is executed.

However, most existing or proposed process management systems do not have a mathematical formalism to capture a process in a discipline independent format. Without such a formalism, task decomposition, execution synchronization, and constraint handling are difficult for most process presentations.

In this thesis, readers are presented with a new process management framework with such formalism that can be utilized to manage complex design and manufacturing processes. This framework is named as MIDAS, which stands for Manufacturing Integration and Design Automation System. With such a generalized process formalism, MIDAS provides a means to incorporate many specialized tools such CAD and FEM packages, and to specify acceptable methodologies and available tools. Its use is not limited to process management. Mechanics analysis, which shares many common features with CAD/CAM processes, can be easily captured by MIDAS. It also enables the users to adapt design changes efficiently and facilitates the exchange of information among participants using the Internet. Most importantly, MIDAS allows users to select tools and methodologies, and to optimize a process configuration within a given set of constraints.

Chapter 3

Design/Manufacturing Process Management Framework

3.1 Introduction

Based on the concepts we introduced in the previous chapters, our own process management framework was implemented mainly using Java 1.02. MIDAS is a Java applet that is running over the Internet on any Java-compatible web browser. It is a tool that is used to create, manage, and execute a process located anywhere on the Internet. In this chapter, the configuration of the framework, the components of the a process flow, and process software tool integration are addressed.

3.2 System Architecture

In this section, we will take a look at the major components of the Process Management Framework as well as their organization and functionality. Detailed descriptions of the major concepts involved in the architecture of our process management framework will also be discussed, including external tool integration, the tool invocation process, Java File System, and state properties.

3.3 System Cockpit

The MIDAS framework has an integrated interface called cockpit (see Figure-6) where all interactions between the user and the system take place. The cockpit is implemented as a Java applet which can be loaded via a Java-enabled web browser.

Besides functioning as the user interface, the system cockpit also provides the core functionality of the process management framework. Although different users may use different instances of the cockpit at the same time, direct collaboration between multiple users is currently under testing.

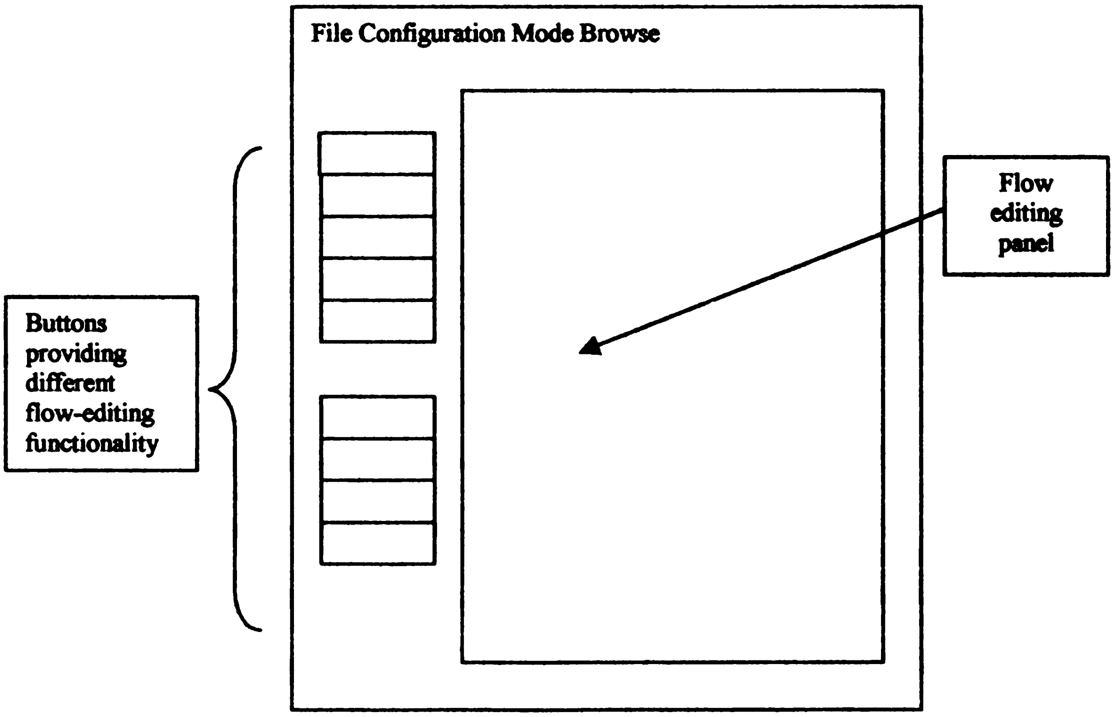
File Configuration Mode Browse
Flow editing panel
Buttons providing different flow-editing functionality

Figure-6. System Cockpit

The system cockpit provides the following functionality:

Flow editing. Users may construct and edit process flows using the flow editing panel of the cockpit. The flow editing panel or the flow editor provides the users with a graphical interface with a number of functional buttons to set up a flow. Flow layout can be optimized via services provided by a remote layout server encoded in Perl. The Cockpit manages the connection to and communication with the layout server through the Internet. This layout editing utility is associated with the "Redraw" button. By clicking on that button get the current messy layout of the flow reorganized.

Production Library Maintenance. Production library provided by the cockpit is used to help user with production maintenance. This functionality is realized by the cockpit's production editor. Users may organize productions, modify input/output sets, create or edit individual productions by using flow editor and the production browser. The production browser is especially useful when user would like to create a production which is similar to an existing one. Using the production browser and the flow editor users may easily create a new production by simply instantiating the existing production. All existing productions are stored in the production library.

Class Library Maintenance. Similar to production, libraries of task and specification classes are also provided in a general-to-specific manner. During the construction of a flow, users are allowed to instantiate a class into an actual task, specification, or database by simply dragging the appropriate class from a class browser, and dropping it onto the flow editor's canvas. The class libraries are displayed in the form of a tree structure in the cockpit. In other words, every object, no matter whether it is a task or specification, is an instance of a class on a certain level in the class tree hierarchy.

Process Simulation and Execution. Processes may be simulated or executed using the cockpit. Several scheduler modules controlling how the process configuration space will be explored are provided. Since the tools may be located externally on other servers, execution of the flow is carried out by invoking the appropriate external tools. Therefore simulation of the execution of the extern tools which are actually running on other machines is necessary. Simulation display, or in other words, flow animation, are provided on the cockpit canvas for users to monitor the execution progress graphically. The animation is realized using different colors. Yollew for unavailable, green for ready, blue for available, white for execution in process, and red for execution failure. Since the process is dynamic, user may recognize which tool is running, which specification is ready, or whether an execution of a tool is successful or has failed. There are currently two kinds of schedulers available for the users to choose. They are the manual scheduler, and the comprehensive linear scheduler. When the user is going to execute a process flow, he/she will need to choose between manual mode and automatic mode. These two modes are actually corresponding to the above-mentioned two kinds of schedulers. Users are also provided with text-based proxy monitor for both manual and automatic modes. When running automatic mode, scheduler monitor is also invoked to show users tool execution information and scheduler information. These two monitors, in conjunction with the animation simulation, provide users with a complete view of a running scenario.

Process Files. A process flow exists in the form of a file. The cockpit allows processes to be archived on a remote server using the Java File System (JFS). The cockpit is enabled by a JFS client interface to connect to a remote JFS server where process files are saved and loaded. While the JFS system has its clear advantages, it is also awkward not to allow users to save process files, libraries, and etc. on their local systems. Therefore, the current framework is configured to allow users to open or save a process flow file on the server from which the cockpit applet is loaded. Not until version 1.1 of the Java Developer's Kit(JDK), local storage by a Java applet was simply not allowed.

3.4 External Tools

As mentioned earlier, each atomic or terminal task in a process flow is corresponding and bound to an external tool. In *Run* mode, when user expands an atomic task in the cockpit, the corresponding external tool is invoked as though it was invoke under text-based command-line. Therefore, a simple negotiation is required to execute a remote external tool. In the previous chapter, our elementary process module was discussed. Inputs and outputs are two major constructs. Accordingly, an external tool uses a series of inputs and creates a series of outputs contained in files. Just as the tools, the inputs and outputs are bound to specifications in a flow. Except for initial inputs and final outputs, inputs to be consumed by one tool are generated by other tools. In previous versions of the process management framework, all tools were located on a public file system, which was called the Server File System. This file system allowed the tools to communicate with each other consuming and creating input and output files on it. Now the latest version of framework allows the tools to access and to generate their input and output files on remote sites. On each one of those remotes sites, there is a site proxy server working in conjunction with a remote file server.

These two servers automatically handle the transmission of files from one system to another during the runtime.

Users are free to choose the languages to implement their external tools. The only requirement is that the platform where a tool is located should have the capability of running a site server. External tools encoded in C, Fortran, Perl, UNIX shell script, Java, I-Deas open language, and Mathematica scripts have been tested. A simple protocol is used to execute an external tool:

1) The tool should be able to run under command-line (from a shell prompt or a DOS prompt). The *Java.lang.runtime* package provides mechanisms for the site proxy server to invoke external commands. However, the Java Virtual Machine is implemented on many platforms in such a way that it does not allow the system to set up a working directory before an external process is executed from within a Java application. In order to overcome this barrier, it was necessary to write a Perl *wrapper* that is called from the site proxy server, and then executes the external tool.

2) The tool must be capable of parsing the following command line format:

   toolname *<commandlineoptions>* INPUTS *i1 i2... iN* OUTPUTS *o1 o2 o3 ... oM*

   where *i1, i2, i3,... iN* are the names of the *N* input files that the tool must be able to open and execute, and *o1, o2, o3 ... oM* are the names of the *M* output files that the tool should produce.

   If the tool is a selector, then the command becomes:

Selector <*command line options*> INPUTS *i1 i2 ... iN* OUTPUTS *o1 o2 o3 ... oM* DB *database1 database2 ... DatabaseT* ITEM *I*

Where *database1 database2 ... DatabaseT* are names of the T databases to be consumed by the selector; The flag ITEM and the number *i* which follows it are used to indicate which record inside the temporary database is picked.

3) The input file names sent off to the proxy server for it to execute may not be necessarily sent in the same sequence for any two different invocations of the tool. That is to say the tools should be designed in a such way that it has the capability of recognizing the input file types.

The user-defined external tools are supposed to be integrated into an abstract process flow ready to execute using the process management framework. Two steps are involved in incorporating tools with a process flow: association and execution. Association includes binding an external tool or a remote file with an abstract flow object. Execution is defined to represent the steps that the framework goes through to actually run the external tools and to process their inputs and outputs.

The action of binding is actually to define a series of properties of a specific flow object. For an atomic task, the following properties must be defined in order to set up a task-tool bonding: 1) SITE. Since the framework can execute tools located at a remote site, we need to define its location. Default SITE is specified within the system so that it may not be necessary to specify the SITE property of each flow object, unless its location is different than the default site. 2)CMDLINE. The CMDLINE property specifies the command which will be executed at the remote site defined by the SITE property. The CMDLINE property should include any switches or arguments that will be sent to the external tool. 3)WORKDIR specifies the tool's working path under which the framework will actually invoke the external tool, create temporary files, etc. Usually this property is also defined in the global property defaults. 3)WRAPPERPATH. Since the 1.0.2 version of the Java Development Kit (JDK) does not allow an external program be executed in an user-specified directory, an extra layer called a *wrapper* between the tool and proxy server must be constructed to simply change directories and then execute the external tool. This layer can be a DOS/Windows NT batch file, a shell script, or a perl program. The external tool is wrapped in this simple script, and executed. Since tools may be running on different platforms, the wrapper program has to be platform-dependent.

Once the user has defined all the above properties for a task, this task is said to be bound to an external tool. In case no site, directory, or filename is specified for the outputs of the task, the framework automatically creates unique file names, and saves the files in the working directory of the corresponding tool on the system where the tool is running. If a tool uses an input that is not specified by any other task, the user must bind it to a remote static file. The framework is configured to reckon all external tools and specifications as to be remote. In other words, even if the tool is located physically on the same system where the user is running the framework cockpit, the framework will still treat it as remote tool.

After all flow objects are bound to theirs appropriate external tools or files, it is time to use the framework to perform process management or tool execution. In stead of directly linking and executing a tool from the cockpit, the process management framework actually has several layers lying in between the cockpit and the external tool that is bound to a flow. A description of each layer between the tool and the cockpit is listed below.

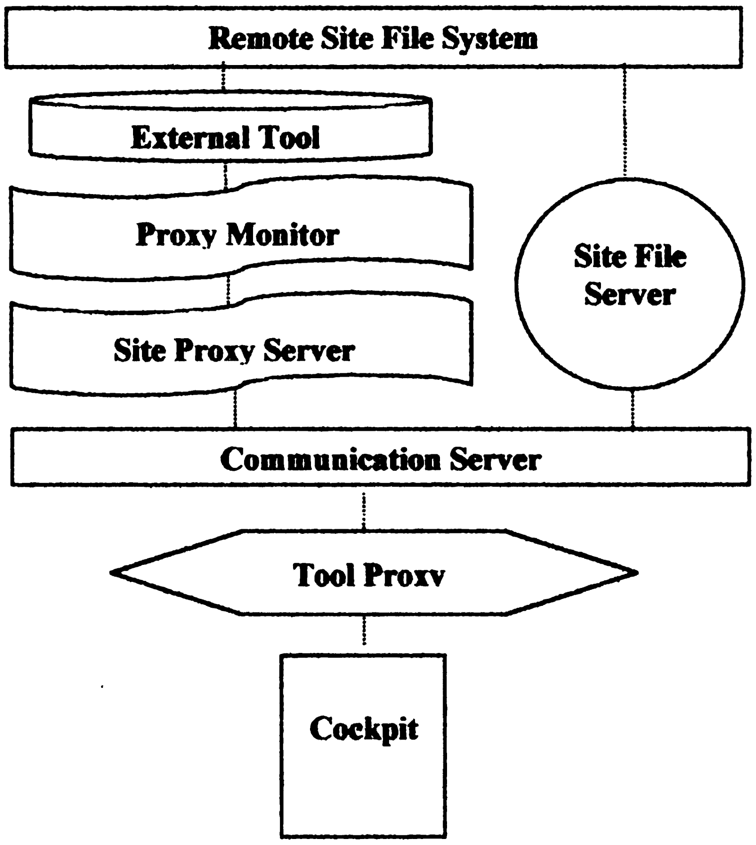

Figure-7. Communication Model

When one launches a cockpit, loads a flow file, and invokes an atomic task, the cockpit is actually talking directly to a tool proxy. The tool proxy acts as a liaison layer between the process objects loaded or defined in the cockpit and the site proxy server through TCP/IP sockets. It sends information packages from the cockpit into the form of string messages that is recognizable by the site proxy server. At the same time, it also waits for and processes messages from the site proxy server sent through the communications server and directs the information to the cockpit object where the tool proxy is originally launched.

For the sake of security, direct TCP/IP socket connections between a Java Applet and any IP address other than the address from which the Applet was loaded is forbidden. For this reason, we are running a communications server on the system where the cockpit applet is stored to listen for messages from tool proxies launched by any cockpit on the same channel.

Site Proxy Servers deal with invocation requests from tool proxies via the communications server. When an invocation request is received, the site proxy server goes through the message's syntax. It then invokes a tool monitor to manage the external tool invocation, and returns the exit status of the external tool after the tool execution is completed. It is reasonable to imagine that there may be more than one tool running on the same site. It may take a significant amount of time for each of the tools to complete its execution. Therefore, waiting for the first tools to complete its execution to launch the execution of the next one would make our process management system very inefficient. Thus the proxy server launches a tool monitor for each external tool to be invoked. Running as a separate thread, a tool monitor waits on the tool, stores its standard output and error message, moves any input or output files to their appropriate site locations, and informs its calling site proxy server when the tool has completed. This scenario allows the site proxy server to process the forthcoming invocation requests continuously without too much delay.

3.5 Representation of a Process Flow

3.5.1 Basic Elements of A Process

This framework uses a formalism called *process flow graphs* to represent design processes. A very simple *process grammar* is used to define and document a design-to-manufacturing process. As mentioned before, it is not uncommon to have alternative approaches to perform a task. The use of the formal *process grammar* makes the alternative methods within a process easily identifiable.

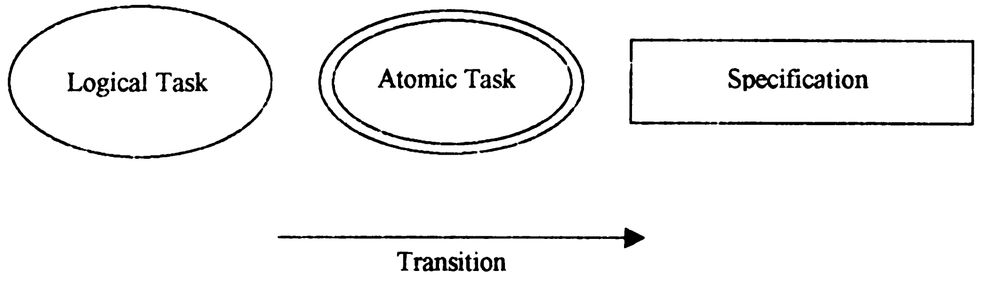

Figure-8. Graphical Representation

Based on the automated design and manufacturing process element discussed in the previous chapter, four basic graphical elements for a process flow graph are shown in Figure-8. Engineers can create design-to-manufacturing processes in a top-down fashion. In the figure above, the four fundamental constructs are listed. Oval nodes represent *Logical Tasks*; two concentric ovals represent *Atomic Tasks*; and rectangular nodes represent *Data Specifications*. The arrow is called a *transition*.

A logical task can be further decomposed into some other lower-leveled logical tasks or atomic tasks. It can be called a non-terminal task because it is decomposable. An atomic task is something that cannot be decomposed any more. It is directly hooked up to an on-line design or manufacturing software. Therefore, atomic task is also denoted as terminal task. The relation between logical tasks and atomic tasks is somewhat like the relation between folders and files. One may place small folders inside a big folder. In the meantime, he or she may also put files inside this same folder. However, unlike folders which can be empty, logical tasks cannot be empty. It should be able decomposed further and further until there are only atomic tasks.

Specification is used to represent on-line files, especially data files. The arrow, which is denoted as transition, is used to show the directions of data flows as well as to weave the stand-alone tasks and data specifications into a complete process flow.

There are two other very special derivative constructs which are used together to represent a very special kind of terminal task called selector. A selector has at least one on-line database.

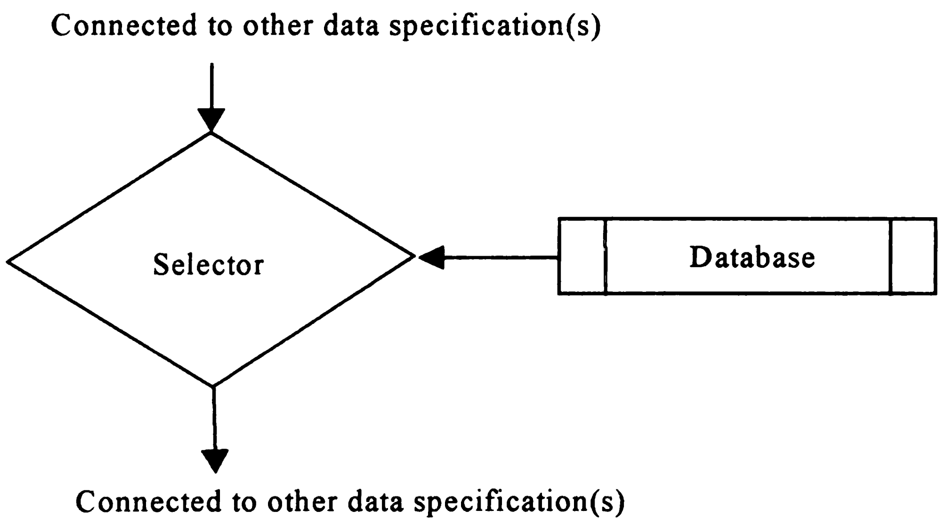
Connected to other data specification(s)
Selector
Database
Connected to other data specification(s)

Figure-9. Selector and Database

The graphical representations of these two constructs are shown in Figure-9. Just as how it is named, a selector works out its results based on the user-defined requirements or on its build-in rules. Selectors share some common features although they are implemented freely by the users. Since a selector may need to pick up its choice from a database according to the requirements imposed by the users, it has its regular inputs just like other tasks do. It also has one or more databases either hooked up to it or hard-coded. No matter how it is implemented, a selector always makes only one choice at one time. A recommended way of implementing a selector is to go through the database to pick up all possible solutions, and store them on a temporary sub-database. Sort this sub-database in an order such that the possible best solution comes the first. The selector then picks up the first record of this sorted sub-database as its output. Should this output turn out to be a bad choice, the selector would choose the next record in the sub-database until it makes a right decision or until the database is exhausted. This simple selecting algorithm is recommended because the selector only sorts the database once.

3.5.2 Process Grammar

After introducing the basic elements of a process flow, we take a look at the very simple process grammar. In order for the readers to understand the grammar, readers are presented with a very straightforward example as shown in Figure-10. In Figure-10, the language produced by

$$L \rightarrow A \mid B$$

$$A \rightarrow a$$

$$B \rightarrow bc$$

Figure-10. An example of the process grammar

this trivial grammar consists of two words: {a, bc}. An analogous *design process grammar*, which is a type of graph grammar, is represented in Figure-11. Notice that logical task *L* has two possible productions: alternative *A*, and alternative *B*. Applying either production will produce a derivation of the original graph by replacing logical task *L* with the flow graph representing that alternative.

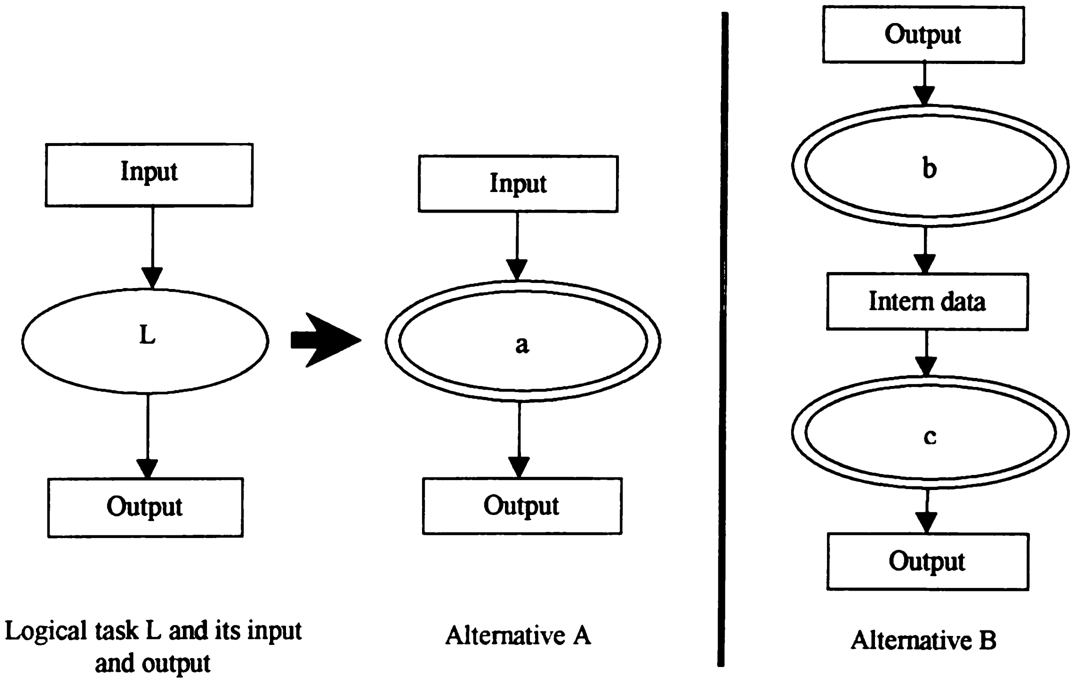

Figure-11. Graphical of a logical task

Chapter 4

Benchmark I ------ Gearbox Design and Manufacturing

4.1 Overview

Two benchmark projects, which are gearbox design and functionally gradient materials design have been developed. In this chapter, attention will be focused on the first one which is the design of a two-stage parallel shaft gearbox.

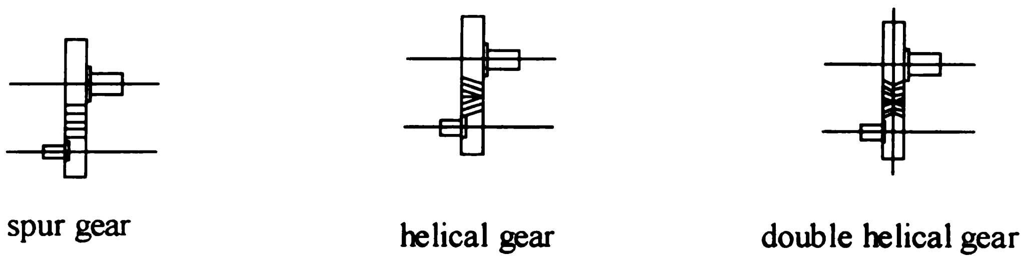

Figure-12. 2-stage parallel gear systems

For a two-stage parallel shaft gear system, there are usually three types choices of gears: spur gears, helical gears, and double helical gears(see Figure-12). With the advantage of low manufacturing cost spur gear may produce relatively high noise during operation and can only provide limited torque. Helical gear can withstand higher torque but they are more expensive and introduce thrust force during operation. In order to balance the axial force, thrust bearings are required. Double helical gear is actually a combination of a pair of helical gears which make the thrust force on a helical gear canceled off with the thrust force on the other, hence no thrust bearings are required.

In the benchmark(see Figure-13) process management, only the design and manufacturing of the shafts, the selection of gear/pinion pair and the bearings are considered. There are three major stages involved in this design and manufacturing process: pre-selection, analysis, and manufacturing.

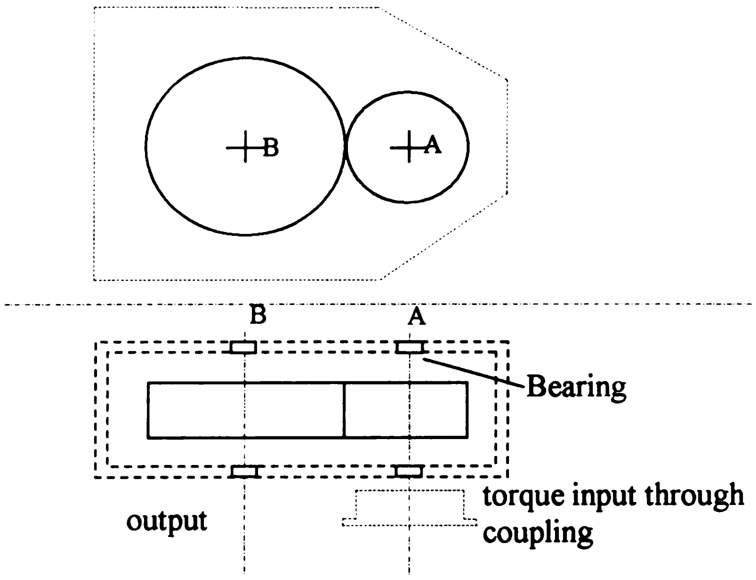

Figure-13. Two-stage parallel gearbox illustrative diagram

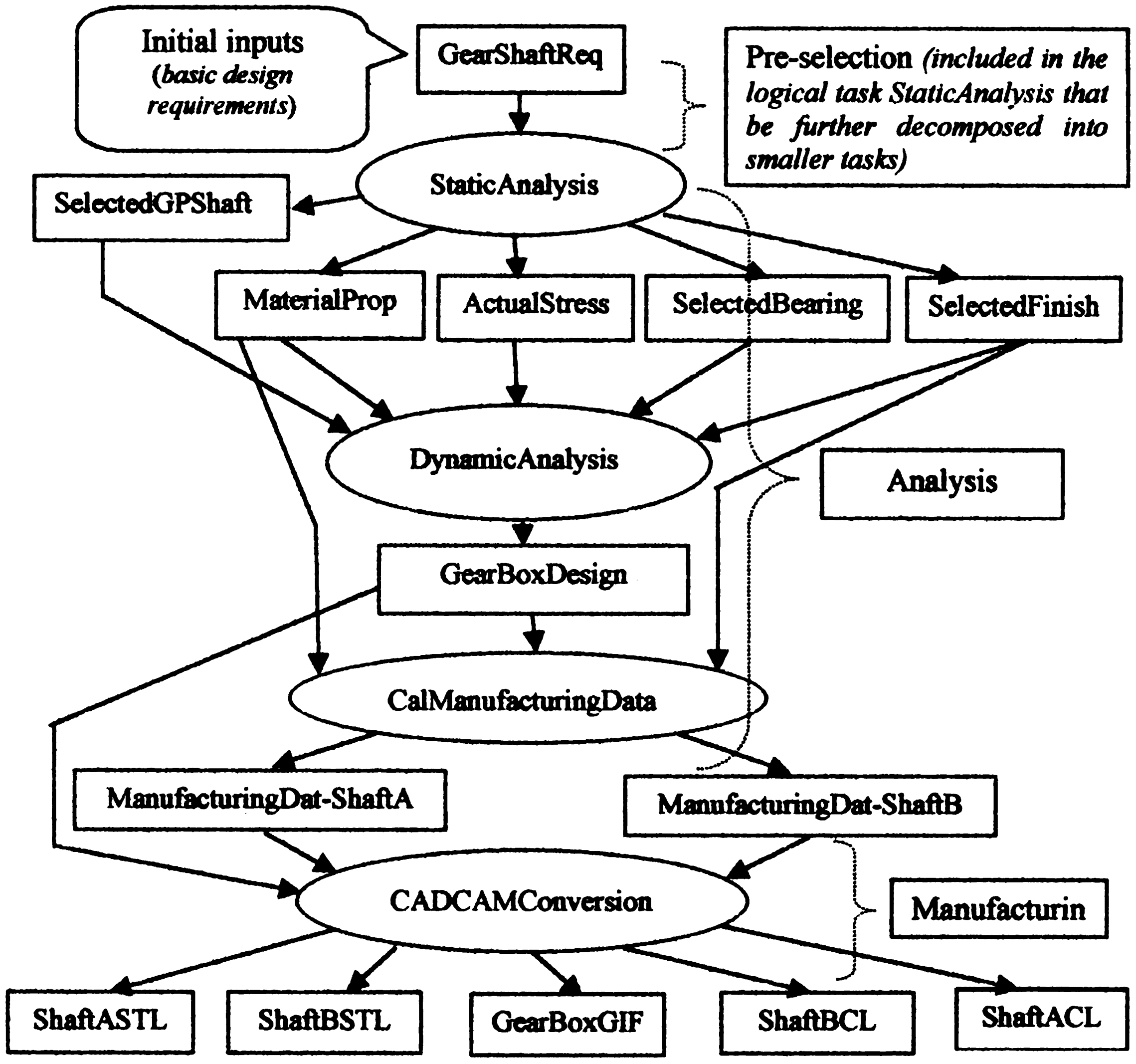

Figure-14. A overall design-manufacturing process flow diagram of gearbox

The above high-level, unexpanded flow diagram provides us with an general overview of the gearbox design and manufacturing process. The first specification *GearShaftReq* is bound to a remote data specification, which is a file containing the initial requirements defined by the users. It is to be consumed by the atomic tasked included in the *StaticAnalysis* logical task. The first few atomic tasks are designed to perform the pre-selection which provides the forthcoming tasks with possible choices of different combinations of gear, pinion, shaft as well as material. The other atomic tasks included in the *StaticAnalysis* are designed to work in conjunction with the tasks in *DynamicAnalysis* to produce the detailed dimensions for both gear, pinion and shafts. Results are stored in the data specification named *GearBoxDesign*. Based on these results, solid models of the design components are created in the logical task named *CalManufacturingData* while CNC codes are generated in the task *CADCAMCoversion* for machining the selected bar stocks into the designed shafts.

4.2 Design

4.2.1 Pre-selection

The design starts with pre-selection. The first flow object is the initial input data specification which includes the basic user-defined requirements. In this benchmark, design requirements include maximum output power, maximum rotational speed, transmission ratio, design life and some geometrical requirements such as shaft length and minimum rotational speed, transmission ratio, design life and some geometrical requirements such as shaft length and minimum and maximum gear diameter. There is a *bind* command button on the cockpit interface. By clicking on the button and then click on the initial input data specification, you will be able to enter the location of the input data file. This procedure is designed for the situation in which there are more than one tentative input data file. Once the input data file is bound to the flow, the design process can begin.

4.2.2 Gear/Pinion/Shaft Selection

A number of Java programs have been composed to handle the gear/pinion selection. As mentioned above, maximum output power, maximum speed, transmission ratio, design life plus some geometrical requirements serve as the inputs. Some pre-evaluation will be conducted to give the range of the pitch. This is to screen out any gear size whose strength is not satisfactory under the given operating conditions. The formula used to estimate the pitch range is as follows[21]:

$$W_t = F \bullet m_f \bullet \Phi_f \bullet W_{max} \tag{4.1}$$

Where, W_t = maximum suggested transmission load (lbs)

F = face width of gear(inches)

m_f = material factor(see Table-1)

Φ_f = pressure angle factor(see Table-2)

W_{max} = maximum suggested load(see Table-3)

Material	303 stainless steel	2024T4 Aluminum	Brass	Nylon	Delrin	Polyrethane
m_f	1	1.2	0.7	0.34	0.28	0.14

Table-1 Material Factors

Pressure Angle	Φ_f
20 deg.	1
14.5 deg.	0.8

Table-2 Pressure vs Φ_f

Diametral Pitch	W_{max}(lbs)
120	65
96	82
80	98
72	105
64	115
48	140
32	195
24	245
20	280
16	318
12	350

Table-3 Pitch vs maximum operational tangential tooth load

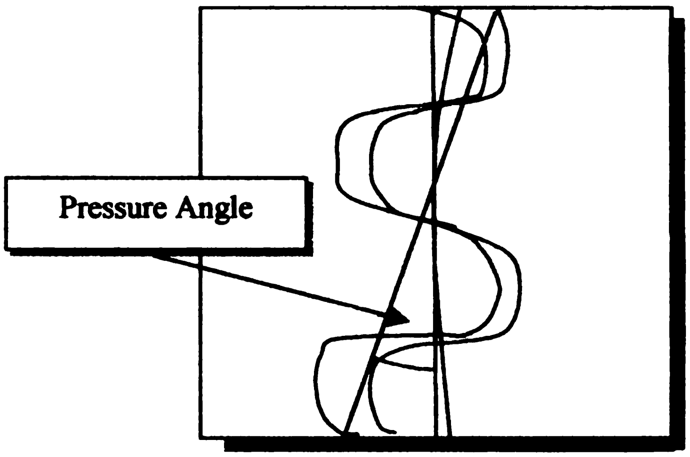

Figure-15. Pressure Angle

The formula to estimate the maximum load of the gear tooth was used. The actual strength of the gear tooth can vary drastically under different operating conditions. The standard loads listed in Table-3 are for well lubricated 303 stainless gears with 20° pressure angle, 1 inch face width, pitch line velocity below 1500 fpm, and smooth loading. The standard loads of pressure angles other than 20 degrees and 14.5 degrees are interpolated or extrapolated. The lowest designing speed and maximum output power (assume there is no power dissipated) are used to work out the possible maximum torque transmitted. Based on the torque and the relation *tangential load = 2 x torque / pitch diameter*, the maximum tangential load exerted on the tooth surface can be estimated. On the other hand, the maximum suggested transmitted load can be calculated using the formula $W_t = F \bullet m_f \bullet \Phi_f \bullet W_{max}$. Compare the maximum tangential load with the maximum suggested transmitted load W_t for each gear in the gear database to get a list of all gears whose strength is high enough to withstand the maximum required load. Since the finer gear teeth or higher pitch gives us higher precision, low backlash, and smoother operation, the computer will sort that list of the gears in terms of their pitch with the largest pitch listed first. Computer will then pick up the first record in the list as best possible choice. If this choice works out all the way to the end of the design process, there is no need to test the second one and the rest.

Once the pitch range is determined, it becomes one of the constraints in search of proper gear/pinion pairs. The simple algorithm used to search for a gear/pinion pair is as follows. The computer sorts the list of gears generated just now again in terms of the pitch diameter. Then based on the transmission ratio and starting with the smallest gear with largest pitch as a pinion, the computer begins to look for the gear simply by multiplying the pinion diameter with the transmission ratio and searching for a gear with exactly the same diameter until the list is exhausted and pick up another pinion. This process stops when the value of the product of a pinion diameter and the transmission ratio is greater than the diameter of the largest gear in the list. After going through the gear list thoroughly, all possible gear/pinion combinations are saved on a temporary file. These choices satisfy all the geometrical requirements contained in the initial input. The selector invoked will also automatically sort the temporary data pool in terms of a certain field or certain set of fields according to some built-in rules that guarantee the design will tend to be the most economical. In short, the built-in rules are defined to try the smallest gear first to make sure that it tends to be economical. As mentioned before, this built-in sorting procedure is inferior to and may be overruled by any arbitrary order issued by the framework. This is reasonable because guaranteeing that it be economical may incur sacrifices in some other respects.

Based on the previous outputs, corresponding shafts are picked up from a shaft database. These shafts are chosen according to some geometrical constraints such as bore size. Naturally, gear and pinion shafts may be different in size. The computer will also concatenate gear/pinion pairs with their selected shafts and sort and save them on a new temporary data pool. Then the selector will pick up a record on that temporary data pool and produce a one-record file as its output. The temporary data pool may be re-established if the framework issues such an order (see Figure-16).

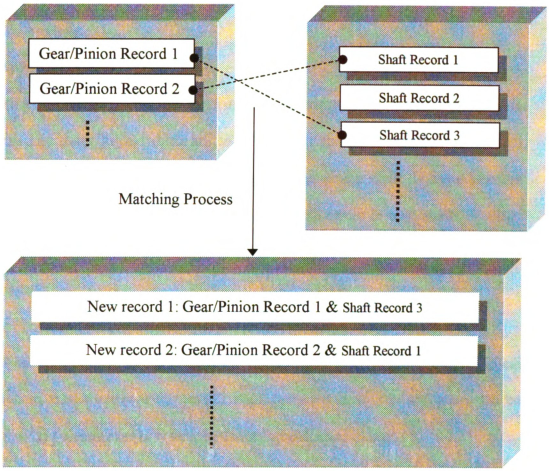
Gear/Pinion Record 1
Gear/Pinion Record 2
Shaft Record 1
Shaft Record 2
Shaft Record 3
Matching Process
New record 1: Gear/Pinion Record 1 & Shaft Record 3
New record 2: Gear/Pinion Record 2 & Shaft Record 1

Figure-16. Concatenation of gear/pinion database and the shaft database

4.2.3 Material Selection

A material selector also needs to be activated to choose the possibly satisfactory types of materials by looking up through a material database. Only one material will be chosen at a time. The material selector shares all general features with other selectors. Properties of a specific kind of material is saved on an output file and will be used by other tools.

4.2.4 Static Analysis

4.2.4.1 Static Forces

Using the formulae in solid mechanics, calculations are conducted in order to produce the key static forces which are to be used for further evaluations as well as selections. Based on the sizes of the gear shaft or the pinion shaft, a pre-evaluation function will tell whether the framework should calculate forces in both shafts or either gear shaft or pinion shaft. In order to simplify the process, most of the pre-evaluations are hard-coded.

Let us consider the forces exerted on a helical gear and its shaft. Single helical gear introduces axial force (see Figure-17). Spur gears and double helical gears can be considered as its special cases in which the thrust force is equal to zero.

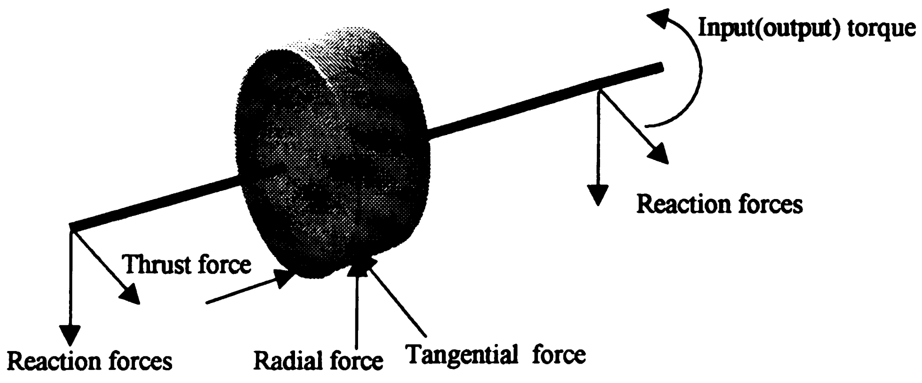

Figure-17. Forces and moment exerted on a helical gear and its shaft

The diagram below shows that the gear-shaft system can be presented by a combined loading situation which consists of a beam subjected to bending and a rod subjected to torsion (see Figure-18)

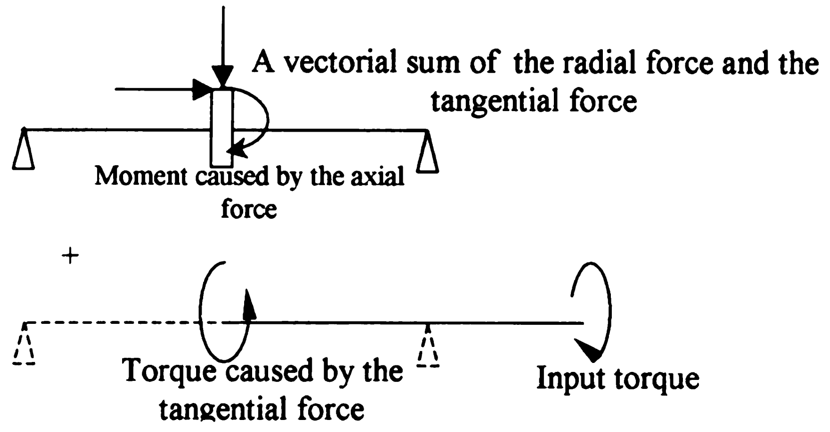
A vectorial sum of the radial force and the tangential force
Moment caused by the axial force
+
Torque caused by the tangential force
Input torque

Figure-18. Combined loading decomposition of a shaft

For a helical gear and pinion pair, the thrust forces on each of them are reaction forces so that they are equal and opposite in direction. The gear has the greater diameter, which means that the moment located at the center of the gear shaft is always greater than that of the pinion shaft. Similarly, the torque on the gear shaft is also larger than that on the pinion shaft. Since the spur gear and double helical gears do not have thrust forces, there is no concentrated moment located in the middle of the shaft. Therefore, in case of spur gear or double helical gears, the computer will first take a look at the size of the pinion and gear shafts. If the two shafts have the same diameter or the pinion shaft is thicker, then the computer will only carry out force and stress analysis for the gear shaft. If the gear shaft is thinner, then you need to check the strength of both shafts. On the other hand, in case of single helical gear and pinion pair, their dimensions will be checked. In this case, there are both moment and torque located in the middle of each shaft. Since the gear has greater radius while the thrust and tangential force on both gear and pinion teeth are reaction forces, the moment and torque on the gear shaft are always the larger. Again, if the gear shaft has the same diameter as that of the pinion shaft or it has smaller diameter, only the gear shaft's strength will be checked. Only when the pinion shaft's diameter is thinner than that of the gear shaft should the strength of both shafts be checked.

Based on the pre-evaluation result, computer will work out forces, moments and torques for the appropriate shaft or both shafts which are to be used in the forthcoming tasks. The forces are saved on a file located on a defined remote system. A flag is attached to the file to help other tools recognize the physical meanings of the data store on that file.

This scenario is a good example of the production alternative. There are three alternatives, which are to check gear shaft, to check pinion shaft and to check both. In automatic mode, the pre-evaluation function determines which alternative should be executed by the framework first. In other words, the pre-evaluation function generates a ranking of all the possible alternatives based on its input data. The framework will try the alternative ranked number one first. If it fails, the framework will try the alternative one till success or till there is no more alternative. In case no alternative is successfully executed, the framework will issue a roll-back command to select either another material or a different set of the gear/pinion pair. In manual mode, the pre-evaluation function is still called. However, the user may choose not to follow the ranking and execute any one of the alternatives.

4.2.4.2 Stress Analysis

Since the loads actually cause bending and torsion, a complex state of stresses exists in the shaft (see Figure-19). The maximum normal stress caused by bending takes place on the surface of the shaft at its center because of the relationship $\sigma_b=MY/I$, where σ_b is the bending normal stress, M is the bending moment, Y stands for the distance to the center of the intersection, and I is the cross-section's moment of inertia. The vectorial combination of tangential and radial forces in the middle of the shaft and the reaction forces produced by the bearings will also cause shear stresses on the cross-section. In case of pure shear, shear stress of this kind can be considered as evenly distributed over the cross-section. When the shear comes up in conjunction with bending, the shear stress distribution varies as described in the formula:

$$S=\left(V\int_{y}^{e} yz\,dy\right)/Iz$$

where S is the shear stress on the cross-section, V is the shear force, and I is the moment of inertia of the cross section about OX (please refer to Figure-19). From this formula, we can see that S vanishes when y equals e. This means that S reaches its maximum value along OX and decreases vertically. The shear stress S is called the transverse shear stress. Since bending stress are maximum at point A while the transverse shear stress is zero there, it is unnecessary to combine them. Conventionally, unless the shaft is very short, the value of the transverse shear stress is much smaller than stress caused by bending, hence it is ignored [22].

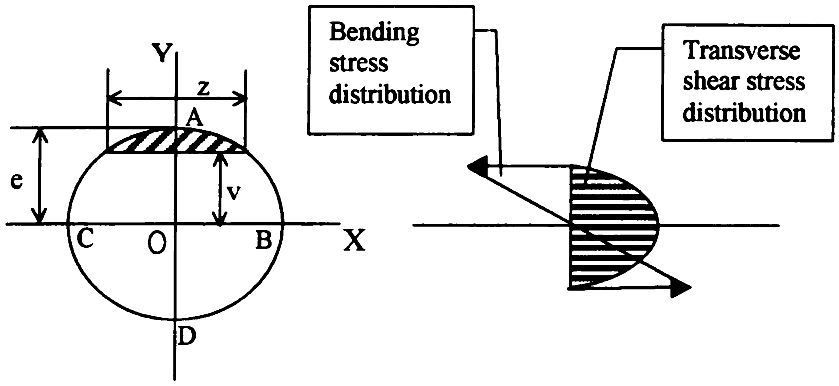
Y
z
A
e
v
C
O
B
X
D
Bending stress distribution
Transverse shear stress distribution

Figure-19. Cross section of a shaft and stress distribution on it

Since there is torque existing in the middle of a shaft, shear stress caused by the torque needs to be considered. Unlike the transverse shear stress, torsional force reaches its maximum at the surface of the shaft. Moreover, since the input power may not be trivial, it is necessary to take the torsional shear stress into account. Both stresses are at their maximum at the surface of the shaft. Therefore, element at the surface in the middle of the shaft represents the most critical spot (see Figure-20).

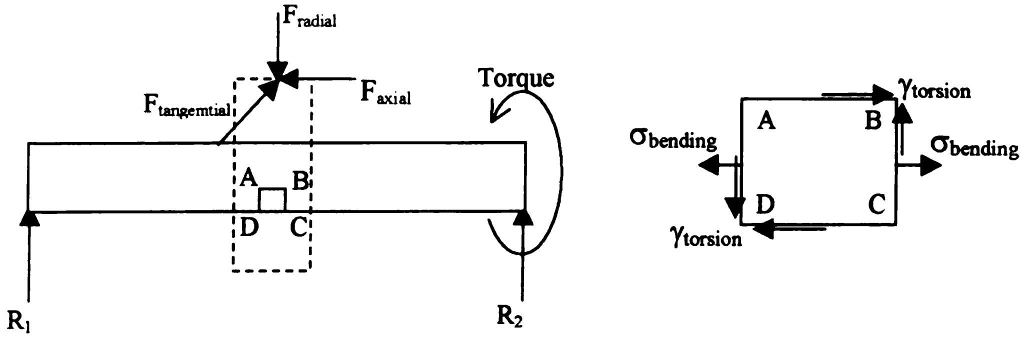

Figure-20. Stresses in the combined loading

In case single helical gears are chosen, axial force is introduced, hence the axial normal stresses. It is assumed that the axial normal stress' distribution over the shaft cross section is uniform. Also assume that the axial force is pointing to the left. Based on Figure-21 , the axial force raises the normal compressive stress in element A, which is located left of M-M. Vice versa, normal stress in element B is also increased except that the stress is tensile. Therefore, when the effective stress is calculated to verify whether the shaft has enough strength, the increase in stress should be taken into account.

When all those stress components are available, it is time to calculate the effective or equivalent stress[22]. The effective stress called the von Mises stress(σ_e) is calculated based on the Mises criterion or the maximum energy of distortion theory of failure. In a general case, in which principal stresses and directions are unknown, the following equation provide us with a way to calculate the von Mises stress:

$$2\sigma_e^2 = (\sigma_x-\sigma_y)^2+(\sigma_y-\sigma_z)^2+(\sigma_z-\sigma_x)^2+6(\tau_{xy}^2+\tau_{yz}^2+\tau_{zx}^2) \qquad 4.2$$

where x, y, and z stand for directions of three coordinates

In our benchmark, the above equation is simplified into:

$$\sigma_e^2 = (\sigma_{\text{maximum bending}} + \sigma_{\text{axial normal}})^2 + 3\,\gamma_{\text{torsion}}^2 \qquad 4.3$$

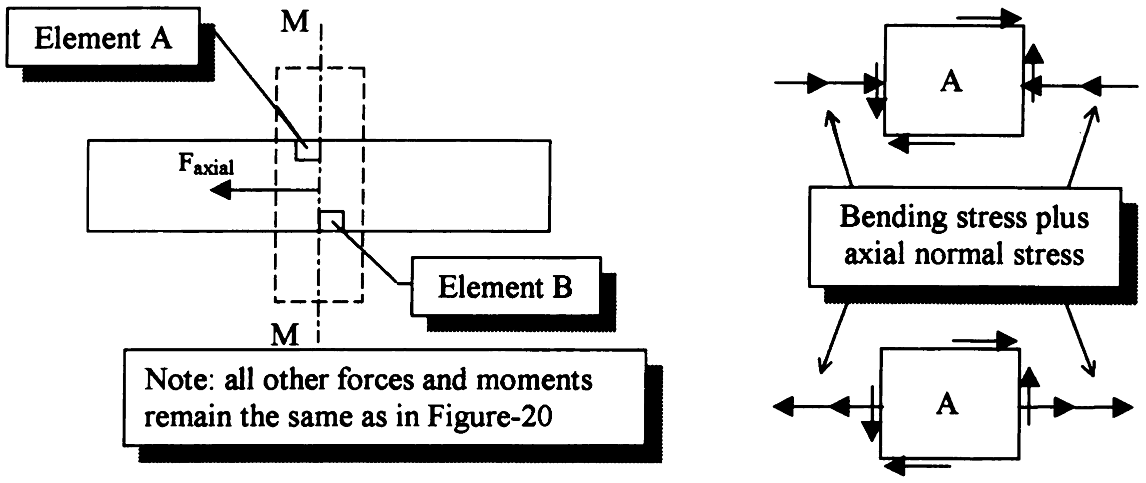
Element A
M
F_{axial}
Element B
M
Note: all other forces and moments remain the same as in Figure-20
A
Bending stress plus axial normal stress
A

Figure-21. Stresses at Dangerous Points

After the calculation of the effective stress, a comparison is carried out between the effective stress and the yield strength of the material chosen. If the effective stress is higher than the yield strength, the framework will roll back either to select another type of material with higher strength or to choose a new set of gear/pinion/shaft whose shaft size is greater. If the effective stress is lower than the yield strength, then the stress analysis is completed.

4.2.5 Bearing Selection

Bearing selection can be simultaneously carried out with the stress analysis. The bearing selection tools have the information on geometrical constraints of all connecting parts as well as information regarding the static forces calculated. Then it goes through a bearing database to seek for all possible bearings whose bore diameters are less than the shaft diameter and list all this bearings on a temporary database just as any other selectors do. The bearing with the smallest bore diameter comes first in the database. The bearing bore diameter should be less than *shaft diameter - 2 x fillet radius* (see Figure-22). In addition, the fillet radius of the bearing ring has to be larger than that on the shaft, otherwise the bearing cannot be mounted. The radial and axial forces are retreived from the inputs. The temporary bearing database is opened to get the fillet radius of the first bearing record. The a Java function is called to calculate the equivalent radial load EQRL, which equals $XR+YT$. Where R is the radial force, T stands for the thrust load, and X and Y are radial and thrust factor (see Table-4). The basic dynamic load rating C which is one of the currently selected bearing's properties stored in the memory is incorporated with EQRL and the rotational speed N to estimate the bearing life L using the formula $L=16700(C/EQRL)^k/N$ (for details, please refer to [23]). Where constant k equals 3 for ball bearings and 3.333 for roller bearings. Comparing the estimated bearing life with the expected life to see if the bearing will fail within the period of service. In case that single helical gears are chosen, thrust bearings will be selected accordingly. Therefore, when single helical gear is used, difference between the actual thrust load and the selected bearing's thrust capacity is calculated to check if it can withstand such a load. If either life requirement or thrust strength is not satisfied, the framework will roll back to choose another bearing until the requirements are met.

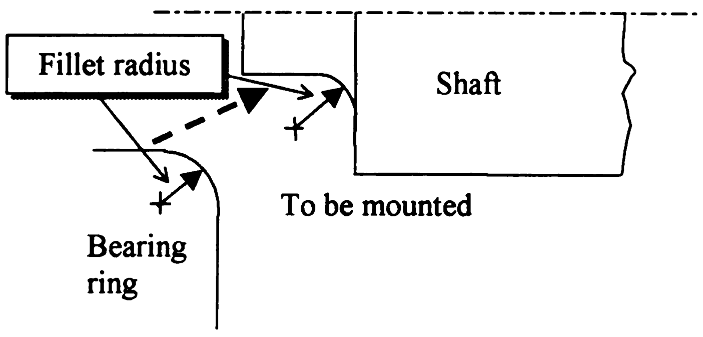

Figure-22. The fillet radius of the bearing ring

Bearing Type	*X*	*Y*
Single-row ball	1.0	0
Double-row ball	1.0	0.75
Cylindrical roller	1.0	0
Spherical roller	1.0	2.5

Table-4 Radial and Thrust Factors

4.2.6 Fatigue Analysis

Since the shafts are rotating components subjected to cyclic loading, fatigue analysis is necessary. A surface-finish-condition selector is invoked at the beginning of the fatigue analysis. Since the major factors influencing surface finish are the cutting tool outline, the fragments of the built-up edge left on the work piece during cutting and vibration, this selector works rather independently. For fatigue analysis, information produced by all previous steps except for stress analysis are used as inputs. The charts for stress concentration factors, notch sensitivity, size factor, surface factor and fatigue notch factor have been converted into Java code using interpolation. Variable stress and mean stress are calculated and the concept of effective stress is also introduced in the analysis.

The judgment of whether the material will fail is made based on Goodman line (Figure-23).

For most wrought steels the fatigue endurance limit is between 0.45 and 0.60 of the tensile strength S_u. Therefore, it is common to $0.5S_u$ as steels' fatigue endurance limit for design purposes. However, an engineering component's surface finish condition, size, operating environment would affect its fatigue endurance limit S_e very strongly. Using the equation

$$S_e=0.5\, k_s\, k_z\, k_t\, k_r\, k_m\, S_u \qquad 4.4$$

where k_s, k_z, k_t, k_r, and k_m are factors for surface, size, temperature, reliability, and other miscellaneous effects like plating or case hardening, the actual fatigue strength is reduced. For normal temperature in operation, temperature factor k_s is 1.0. Since we do not have plating, case hardening or high residual stress, we take 1.0 as the value of the factor of the miscellaneous effects.

4.2.6.1 Surface Factor

Fatigue strength varies with surface condition because surface irregularities are stress raisers which will initiate fatigue cracks. The surface factor is a positive number that is less than 1.0. Finer surface finish corresponds to higher surface factor. Surface factor is also a function of tensile strength. It decreases as tensile strength goes up. Therefore, for each finish condition there is a curve depicting the K_s - S_u relationship [22]. Read the data points off the curves. Pick more points where the curvature is large.

A polynomial with the order equivalent to the number of the data points to fit the curve is used. The material's tensile strength is then substituted into the polynomial to calculate the surface factor.

4.2.6.2 Size Factor

When the size increases, the fatigue endurance limit decreases. A larger component tends to have a weaker metallurgical defect which will cause a fatigue crack to take place. The formulas listed below determine the value of the size factor [24]:

$k_z = (d / 0.3)^{-0.068}$ $\quad 0.1 < d < 2.0$ in $\quad$ 4.5(a)

$k_z = (d / 7.6)^{-0.068}$ $\quad 5 < d < 50$ mm $\quad$ 4.5(b)

$k_z = d^{-0.19}$ $\quad 2 < d < 10$ in $\quad$ 4.5(c)

$k_z = 1.85\, d^{-0.19}$ $\quad 50 < d < 255$ mm $\quad$ 4.6(d)

Where d is the shaft diameter.

4.2.6.3 Reliability Factor

The reliability factor k_r is a function of the expected reliability R. The following table shows you the k_r - R relationship.

R(reliability,%)	90	95	99	99.9	99.99
k_r(reliability factor)	0.90	0.87	0.81	0.75	0.70

Table-5. k_r - R relationship

Based on this table, if the designer expects to have a reliability of 99%, then the reliability factor is 0.81.

4.2.6.4 Goodman line and Stress Concentration

Since the shaft is rotating and this rotation causes alternating stress, Goodman line(Figure-23), which is used to judge whether or not a specimen subjected to alternating loading can reach infinite life, is introduced. As to the theoretical details of the Goodman line, please refer to [22].

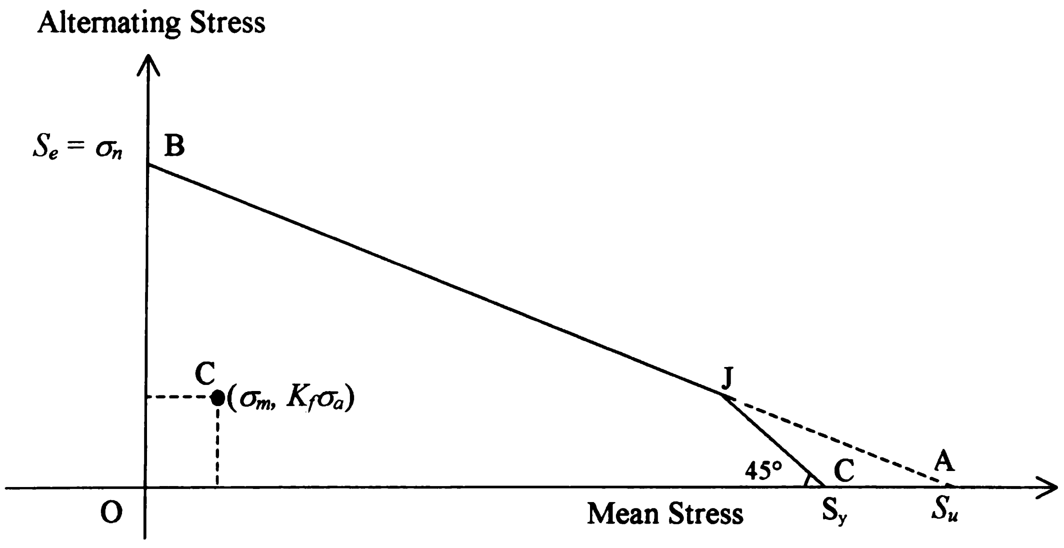

Figure-23. Goodman line

In the above chart, the abscissa corresponds to mean stress while the ordinate is the alternating stress. The value of point A corresponds to the ultimate strength S_u while point B is associated to a value of the adjusted endurance strength σ_n. Point C is corresponding to the yield stress. In order to see whether the material can withstand such an alternating load, mean and alternating stresses need to be calculated. In this case the torsional shear stress is considered to be the mean stress because it is always there. The maximum bending stress on the shaft surface in the middle is used as the alternating stress. Since the shaft has different sections with different diameters, stress concentration may take place at the change of the cross-sections. A number denoted as fatigue stress concentration factor K_f or the fatigue notch factor is calculated and is multiplied to the alternating stress σ_a to evaluate the escalated stress caused by the change of geometry. To calculate K_f, two values, which are the theoretical stress concentration factor K_t and the fatigue notch sensitivity q, should be calculated . The theoretical stress concentration factor K_t can be estimated based on the specimen's geometry and the types

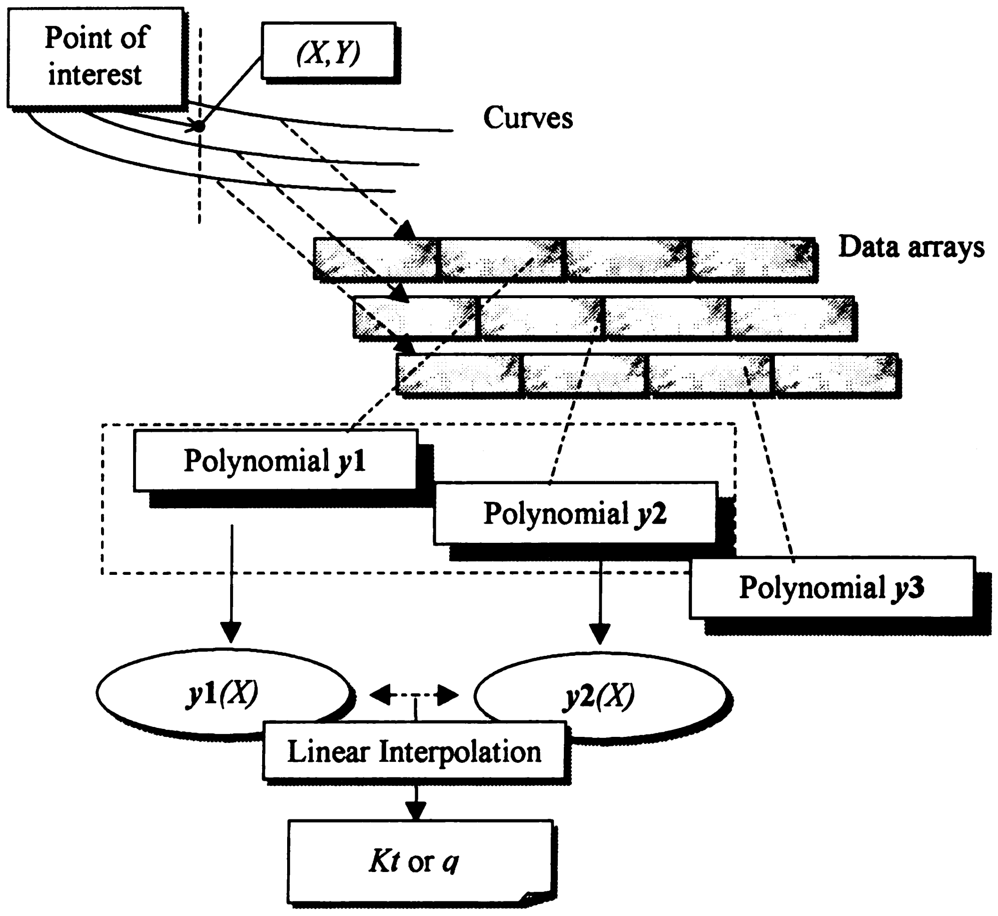

Figure-24. Interpolation process

loads exerted on it. The geometry is specified by dimensions such as radii of the different sections and the fillet radius. K_t charts are available for simple loading situations such as tension, bending and torsion. Little data exists on the effect of coincident stress raisers. Based on some photo-elastic evidence, the product of the two separate factors is used. The fatigue notch sensitivity q varies with fillet or groove radius. For details, please refer to [25].

When K_t and q are ready, K_f can be obtained using the equation $K_f = 1 + q(K_t - 1)$. K_t and q are calculated by interpolation. Since their original data is presented in a form of curves, a series of data is obtained from each curve and is stored curve by curve. When interpolation is initialized, the program will first find the curves between which the point of interest is located and their corresponding data series stored in arrays. Based on the data, two polynomials are generated to fit the two curves regionally. Since there may be many data points stored on the computer, only a few(6 to 10) points around the vicinity of the point of interest are used. Now there are two polynomials for the two curves. If a vertical line is drawn through the point of interest, it is easy to identify the coordinates of the two intersecting points on the curves. Then by simply using linear interpolation, the location of the point of interest can be identified (see Figure-24). After the point of interest has been located on the Goodman chart, it can tell whether or not the designed shaft will survive the infinite number of rotations. If the point is located below the line BJC (in Figure-23), it means that the designed shaft has enough fatigue endurance strength and the manufacturing processes can be carried out. If it is located somewhere above the line BJC (in Figure-23), it will fail within finite times of rotations. In that case, rollback will take place using one of the following scenarios in an order of the priority as listed below: 1) to select a larger fillet radius which is geometrically feasible; 2) to use larger shaft; 3) to use another material with higher ultimate strength; 4) to choose a finer surface finish.

4.3 Manufacturing

4.3.1 Solid Modeling

When the design processes have been completed, the framework invokes an I-DEAS solid modeling program to start the manufacturing processes. A transitional tool is used between the design and manufacturing processes to collect the information required for manufacturing and pass them to the manufacturing tools. In this benchmark project, this tool is not implemented as a stand-alone task. Instead, it is incorporated into the fatigue analysis. The manufacturing tools are encoded mostly in I-DEAS open language which has some special requirements on its file format. Therefore, the transitional tool incorporated in the fatigue analysis was designed not only to collect information but also to convert all information into the format that I-DEAS can recognize.

Since it is very hard for the Java-based framework to invoke I-DEAS programs interactively and graphically, in batch mode invocation is used. I-DEAS open language has a restriction that does not allow its program to take external inputs (as a parameter) under command line. This restriction makes it hard for the framework to generate output specifications with random names. Furthermore, the I-DEAS should always run with a pre-defined project file and a model file. Thus in case more than one user has logged on and is running the same process at the same time or even at different time, the consequence would be either a dead lock or that only the latest outputs exist. To circumvent these difficulties, an extra interface is written. This simple interface allows the I-DEAS open language program to run in a pre-defined environment and save output files with fixed names. It recognizes the randomly ordered inputs, constructs temporary directories with unique names, copies everything such as project file, model file and program files into those directories, invokes I-DEAS to run the open language programs, converts the outputs with unique names to the working directory and labels them with flags so that other tools can recognize them.

The solid modeling tool imports all geometrical information, builds 3-dimensional solid models of all gearbox components such as gears, shafts, bearings and gearbox housing, and stores them in a library. The tool also assembles those components and generates a postscript file for the assembled gearbox with a housing. At the same time, this I-DEAS tool also generates an IGES (Initial Graphical Exchange Specialization) file of the assembled soled model for future use. Unfortunately, most Java-enabled web browsers such as Netscape cannot display postscript images. Therefore, the interface which is lying in between the solid modeling tool and framework converts the postscript file into JPEG of GIF format image file for the users to actually view the gearbox designed. On the whole, the solid modeling process can be divided into step as described in Figure-25.

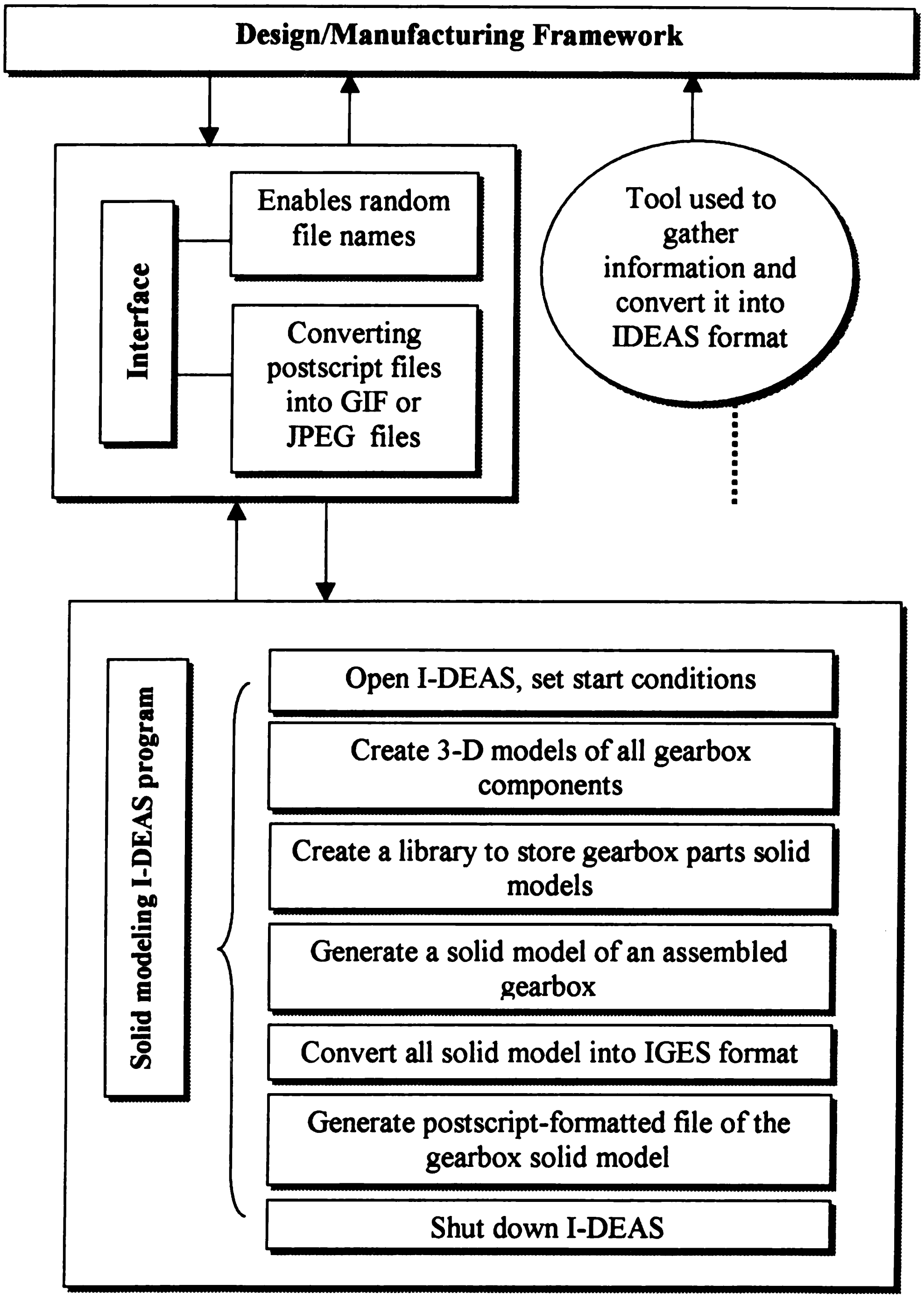
Design/Manufacturing Framework
Interface
Enables random file names
Converting postscript files into GIF or JPEG files
Tool used to gather information and convert it into IDEAS format
Solid modeling I-DEAS program
Open I-DEAS, set start conditions
Create 3-D models of all gearbox components
Create a library to store gearbox parts solid models
Generate a solid model of an assembled gearbox
Convert all solid model into IGES format
Generate postscript-formatted file of the gearbox solid model
Shut down I-DEAS

Figure-25. Solid modeling processes

4.3.2 Manufacturing

When the solid modeling is done, the framework proceeds to invoke I-DEAS again in order to run the generative machining and manufacturing program which is also written in I-DEAS open language. Up till now, using the MIDAS framework, design can be accomplished to such a level that a 3-D solid model can be displayed. At the same time, it is also converted to Stereo Lithographic (STL) format for rapid prototyping. Furthermore, if the tool path has also been ready, cutter location data and the NC codes for numerical machining can be generated according to the tool path information. A simple description of this process is listed as follows: 1) Invoke the manufacturing program encoded in I-DEAS open language and import the manufacturing data; 2) Retrieve the shaft data from the library generated in the solid modeling process; 3) Generate STL file and convert it so that the image can be posted on the Internet; 4) Create machine and stock instances; 6) Create a job setup; 7) Create tool paths according to manufacturing data; 8) Generate CL files and display the files on the world wide web; 9) Shut down I-DEAS.

4.4 Conclusion

The gearbox design/manufacturing process benchmark successfully demonstrated the framework's ability to perform a series of stress/fatigue analyses and search for a set of parts satisfying various requirements. It also verified that the framework is capable of representing and integrating design tasks and manufacturing operations seamlessly.

Chapter 5

Benchmark II ----- Design of Functionally Gradient Materials

5.1 Introduction

The second benchmark project is to use the framework to design functionally gradient materials. Functionally gradient materials (FGMs) are designed to have gradient properties resulting from continuously varying the volume fraction of a second phase in a controlled manner. By combining these approaches, materials with a variety of desired properties are designed. Recently, there has been great interest in FGM composites which are produced by a variety of schemes in which the concentration of the reinforcement phase is artificially varied.

It is challenging to determine the desired distribution of the second phase because it depends on the type of the application and the geometry of the final structure. The thermoelastic properties as well as the types of the applied loads have to be taken into account simultaneously to define the type of gradient, and the required properties of the reinforcement and matrix materials. In this benchmark project, we have studied the design of FGMs for axisysmmetic structures subjected to temperature gradients, or pressure differentials. One of the advantages of using the framework to design FGMs is the reusability of the same process flow in case studies. FGMs designers may need to investigate a variety of designs with different volume fractions of the reinforcement phase and applied loads. In that case, the framework enables the designers to feed the system with different types of initial conditions such as volume fraction and automatically executes the process flow. When the results of all cases are available, they are compared to determine which one is the most desired distribution of the second phase.

In this chapter, effective medium theory using Mori-Tanaka method is briefly outlined. Then the thermal gradient in a hollow cylinder with specified boundary temperatures is calculated. Following this, effective thermoelastic properties are determined for several second phase distributions. The thermal stresses computed show a strong dependency upon the second phase distribution. Stresses caused by internal pressure are separately considered. Using superposition, the effect of pressure and thermal gradient is combined.

5.2 Effective Properties

5.2.1 Elastic Properties

By applying the appropriate homogenous boundary conditions in conjunction with the properties of the constituent materials, the effective properties can be computed. Listed as follows are the displacement boundary condition.

$$\boldsymbol{u} = \epsilon^{o}\boldsymbol{x} \tag{5.1}$$

where $\boldsymbol{x}$ is the position vector and the ϵ^{o} is any spatially constant symmetric tensor. The effective stiffness is defined by

$$\tau = \mathbf{C}^{+}\bar{\epsilon}^{o} \tag{5.2}$$

where $\bar{\tau}$ is the average stress and the applied strain, $\in^{o}$, is equal to the average strain, $\bar{\in}$, subject to the displacement boundary condition (5.1).

Although the average stress is unknown, the assumption of Mori and Tanaka [26] may be applied to estimate the effective stiffness tensor based on the following assumption for the relation between the average strain in inclusions, $\in_1$, and the applied strain, $\in^{o}$,

$$\in_1 = \bar{\mathbf{A}} \in^{o} \qquad 5.3$$

where the strain concentrator, **A**, is defined as

$$\mathbf{A} = \mathbf{T}[(1-\nu)\mathbf{I} + \nu\mathbf{T}]^{-1} \qquad 5.4$$

where ν stands for the volume fraction of particles or fibers, **I** is the fourth rank identity tensor, and **T** is Wu's Tensor [27], defined as

$$\mathbf{T} = [\mathbf{I} + \mathbf{E}\,\mathbf{S}_0(\mathbf{C}_1 - \mathbf{C}_0)]^{-1} \qquad 5.5$$

Here **E** is the Eshelby's tensor [28], $\mathbf{S}_0$ is the compliance tensor of the matrix material and $\mathbf{C}_0$ and $\mathbf{C}_1$ are respectively the stiffness tensors of the matrix and particle materials. Under the Mori-Tanaka assumption, the effective tensor is

$$\mathbf{C}^{+} = \mathbf{C}^{mt} = \mathbf{C}_0 + \nu(\mathbf{C}_1 - \mathbf{C}_0)\mathbf{A} \qquad 5.6$$

The effective compliance tensor, $\mathbf{S}^{mt}$, may be expressed as the reciprocal of $\mathbf{C}^{mt}$.

5.2.2 Thermal Expansion Coefficient

In order to obtain the thermal gradient, the effective thermal expansion coefficient (CTE) tensor, α^{+}, should be determined. The CTE obtained using the Mori-Tanaka approach is

$$\alpha^{+} = \alpha_0 + (\alpha_1 - \alpha_0)(S_1 - S_0)^{-1}(S^{mt} - S_0) \qquad 5.7$$

where α_0 and α_1 are respectively the CTEs of the matrix and the particle materials while S_1 is the compliance tensor of the particle material.

5.2.3 Thermal Conductivity

The effective thermal conductivity properties were obtained in a similar way based on the Mori-Tanaka scheme [29]. Average temperature gradient **g**, and the average heat flux **q** are two primary variables in thermal conduction. Relation between these two variables can be expressed using Fourier's heat conduction equation

$$\mathbf{q} = - \mathbf{K}^{+}\mathbf{g} \qquad 5.8$$

where $\mathbf{K}^{+}$ is the effective conductivity tensor. $\mathbf{H}^{+}$ is defined as the resistivity tensor, which is the reciprocal of $\mathbf{K}^{+}$.

A tensor, $\mathbf{g}_1$, which is equivalent to Wu's tensor can be defined as

$$\mathbf{g}_1 = \mathbf{T}\,\mathbf{g}^0 \qquad 5.9$$

where

$$\mathbf{T} = [\mathbf{I} + \mathbf{E}\,\mathbf{H}_0(\mathbf{K}_1 - \mathbf{K}_0)]^{-1} \qquad 5.10$$

and $\mathbf{H}_0$ is the resistivity tensor of the matrix material.

Consider there is a general material with multiple second-phases. The average temperature gradient in the inclusions and the average temperature gradient in the matrix materials can be related by

$$\mathbf{g}_1 = \mathbf{T}\,\mathbf{g}^0 \qquad 5.11$$

The effective thermal conductivity tensor is expressed as

$$\mathbf{K}^+ = \mathbf{K}_0 + \nu\,(\mathbf{K}_1 - \mathbf{K}_0)\,\mathbf{A} \qquad 5.12$$

where **E** is a second order tensor which is similar to the Eshelby. **A** is defined as the temperature gradient concentration as follows

$$\mathbf{A} = \mathbf{T}\,[(1 - \nu)\,\mathbf{I} + \nu\,\mathbf{T}] \qquad 5.13$$

5.2.4 Homogenization of Inhomogeneous Properties

The expression for the effective properties are found to be functions of the constituent material properties and the volume fraction for a specific geometry of the second phases. Since the volume fraction distribution function can be expressed as

$$\nu = \nu(\boldsymbol{x}), \qquad 5.14$$

the expressions of effective properties for a heterogeneous material such as FGM can also be obtained by using the position vector $\boldsymbol{x}$. For instance, the effective stiffness tensor can be written as

$$\mathbf{C}^+ = \mathbf{C}^+(\boldsymbol{x}) \qquad 5.15$$

It is rational to use cylindrical coordinate system for axisymmetric FGMs so that the inhomogeneity is restricted to the radial direction only. If the volume fraction is a function of **r**, then the inhomogeneous properties can also be expressed using **r**. Owing to the extremely small ratio between particle size and structural dimension, these homogenization methods are applicable. Thus, the spatial distribution function of the particles becomes the key parameter in the FGMs design.

5.3 Formulation

According to thermoelasticity, the total elastic strain ϵ, equals the sum of the mechanical strain $\mathbf{e}$, and the eigenstrain ϵ^*. It is expressed as

$$\epsilon = \mathbf{e} + \epsilon^* = \mathbf{e} + \alpha^+ \, \delta T \qquad 5.16$$

where δT is the temperature variation field. The stress is therefore

$$\tau = \mathbf{C}^+ \, \mathbf{e} = \mathbf{C}^+ (\epsilon - \epsilon^*) \equiv \sigma - \sigma^* \qquad 5.17$$

Listed as follows is the equation of equilibrium with zero body force.

$$\text{div}\, \tau = 0 \qquad 5.18$$

The traction boundary condition is defined as

$$\tau \, \mathbf{n} = \mathbf{t}^0 \qquad \text{on } \partial\mathcal{B} \qquad 5{,}19$$

The polar coordinate expression of the displacement vector $\mathbf{U}$ for the axisymmetric plane strain problem can be written as

$$\mathbf{U} = \mathbf{u(r)} \, \mathbf{e_r} \qquad 5.20$$

where $\mathbf{e_r}$ is the unit vector in the direction of $\mathbf{r}$. Due to the symmetric structure, the only non-zero strains are

$$\epsilon_{rr} = du/dr \text{ and } \epsilon_{\theta\theta} = u/r \qquad 5.21$$

Applying the locally isotropic and globally inhomogeneous stiffness tensor to the strain tensor gives σ as follows.

$$\sigma = \begin{pmatrix} C_{rr} & C_{r\theta} & C_{r\theta} & 0 & 0 & 0 \\ C_{r\theta} & C_{rr} & C_{r\theta} & 0 & 0 & 0 \\ C_{r\theta} & C_{r\theta} & C_{rr} & 0 & 0 & 0 \\ 0 & 0 & 0 & (C_{rr} - C_{r\theta})/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (C_{rr} - C_{r\theta})/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (C_{rr} - C_{r\theta})/2 \end{pmatrix} \begin{pmatrix} du/dr - \alpha\delta T \\ u/r - \alpha\delta T \\ -\alpha\delta T \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad 5.22$$

In case of the FGM cylinder in consideration,

$$\mathbf{C_{rr} = C_{rr}(r) \quad \text{and} \quad C_{r\theta}(r)} \qquad 5.23$$

as with other properties such as the CTE and thermal conductivity considered.

By incorporating equation (5.22) into the equilibrium equation, we obtain the following equation in terms of u

$$[d^2u/dr^2 + (C_{rr,r}/C_{rr} + 1/r)\, du/dr + (C_{r\theta,r}/rC_{rr} - 1/r^2)u]\, C_{rr}$$

$$= d[(C_{rr} + 2\, C_{r\theta})\alpha\delta T]/dr, \qquad 5.24$$

where $C_{rr,r}$ and $C_{r\theta,\, r}$ are dC_{rr}/dr and $dC_{r\theta}/dr$ respectively. If homogeneous axisymmetric media are used, $C_{rr,r}$ and $C_{r\theta,\, r}$ are equal to zero. In other words, C_{rr}, $C_{r\theta}$ and α are constants. In that case, the above equation then transforms into the equation obtained by Timoshenko and Goodier [30] for homogeneous and isotropic materials.

In order to solve this equation, boundary conditions need to be identified. For an axisymmetric cylinder, boundary conditions at the inner and outer radii are specified as follows:

$$[\, C_{rr}\, du/dr + C_{r\theta}\, u/r - (C_{rr} + 2\, C_{r\theta})\alpha\delta T\,] \Big|_{r = r_i} = P_i \qquad 5.25$$

and

$$\left[C_{rr}\, du/dr + C_{r\theta}\, u/r - (C_{rr} + 2\, C_{r\theta})\alpha\delta T\right]\Bigg|_{r = r_0} = P_0, \qquad 5.26$$

where r_i and r_0 are the inner and outer radii respectively. P_i and P_0 stand for the internal and external pressures. In case of pure thermal loading, both P_i and P_0 are assumed to be zero.

Although in the previous part of this chapter differential equation (5.24) in terms of u is obtained, the left-hand side of the equation still contains the temperature variational field as a function of r only as $\delta T(r)$. For steady state, δT can be derived from the following heat conduction equation for cylinder

$$\frac{d}{dr}\left(2\pi r k \frac{d\delta T(r)}{dr}\right) = 0 \qquad 5.27$$

where $k = k(r)$ is defined in the same way in which the other properties were previously determined. For steady state, the two boundary conditions imposed are

$$\delta T\,(r_i) = T_i \qquad 5.28\text{ (a)}$$

and

$$\delta T\,(r_0) = T_0 \qquad 5.28\text{ (b)}$$

where T_i and T_0 are the temperature of the interior and exterior of the cylinder. For steady state, difference between the T_i and T_0 becomes the greatest, hence the most severe thermal loading on the cylindrical structure.

5.4. Materials

Due to its practical importance, aluminum alloy 6061-T6 is chosen as the matrix material. Compatible second phase materials are graphite, B_4C, glass, mica, TiB_2, ZrB_2, Al_2O_3 and SiC. Since Al_2O_3 and SiC are the best for metal matrix composites, they are chosen for the present research. Properties of these materials are shown in the table below.

	Matrix Al Alloy	Second-phase SiC	Second-phasse Al_2O_3
Modulus	70Gpa	616Gpa	379GPa
Poisson's Ratio	0.34	0.17	0.23
CTE X 10^6	23.6/°C	2.5/°C	8.3/°C
Conductivity	180W/mK	25W/mK	37.7W/mK
Strength	±0.27GPa	-8.3/+0.39GPa	-4480/+205GPa

Table - 6 Room temperature properties of matrix and reinforcement

5.5 "Multiple-version" Process Flow

Based on the inhomogeneous properties obtained in micro-mechanics analysis, the macroscopic thermoelastic analysis can be conducted on FGM cylinders. Three second-phase distribution functions were investigated : linear (type 1), concave-up (type 2), and concave-down (type 3) . In order to make comparison, several MMC cylinders whose second phase is uniformly distributed are presented as reference. For specified temperature boundary conditions, T_i and T_0 are chosen to be $0\,^{\circ}C$ and $100\,^{\circ}C$.

Type 1	$\nu = (r - r_i) / (r_0 - r_i)$
Type 2	$\nu=(r - r_i)^2 / (r_o - r_i)^2$
type 3	$\nu=(- r^2+2r_or + r_i^2 - 2r_or_i)^2 / (r_o - r_i)^2$

Table – 7. Second-phase distribution functions used

Although in this benchmark study, only three second-phase distribution functions are chosen, it is not uncommon that by far more than three cases are considered in real practice. Owing to the fact that the design is too complicated for the designer to directly come up with an ideal distribution function, multiple case study is necessary. The framework, which is designed to capture parametric processes, is very suitable for the multiple case study in the FGM design. Since the desirable design should be determined by comparing the results of each considered tentative design, the framework provides a utility called "multiple-version". The "multiple-version" enables the framework to conduct different designs, and make comparisons among the results based on certain pre-defined rules in order to determine which one is the ideal design. This functionality reflects and captures a very common feature in mechanics analysis as well as engineering designs, especially when the analysis and design have their tools scattered all over the Internet. The overall process flow chart designed using the framework is presented in Figure-26 .

In this benchmark, the whole process is divided into three steps represented by three logic tasks. These tasks are material selection, micro-mechanics analysis, and thermoelastic analysis. The material selection logic task can simply be decomposed into a material selector which selects both matrix and second-phase and their corresponding properties. In micro-mechanics analysis, Mori-Tanaka method is captured inside an atomic task to produce inhomogeneous properties such as stiffness , CTE and effective thermal conductivity in terms of the position vector. When these inhomogeneous properties are ready, temperature distribution can be determined. With the temperature distribution in conjunction with the inhomogeneous properties, stresses and displacements are obtained, hence the effective stresses (von Mises stresses). When the effective stress distribution for each second-phase volume fraction function is ready, comparisons are automatically made to determine the best design. Since the "multi-version" feature is implemented, the repetitions are implicitly made to avoid hard-coding.

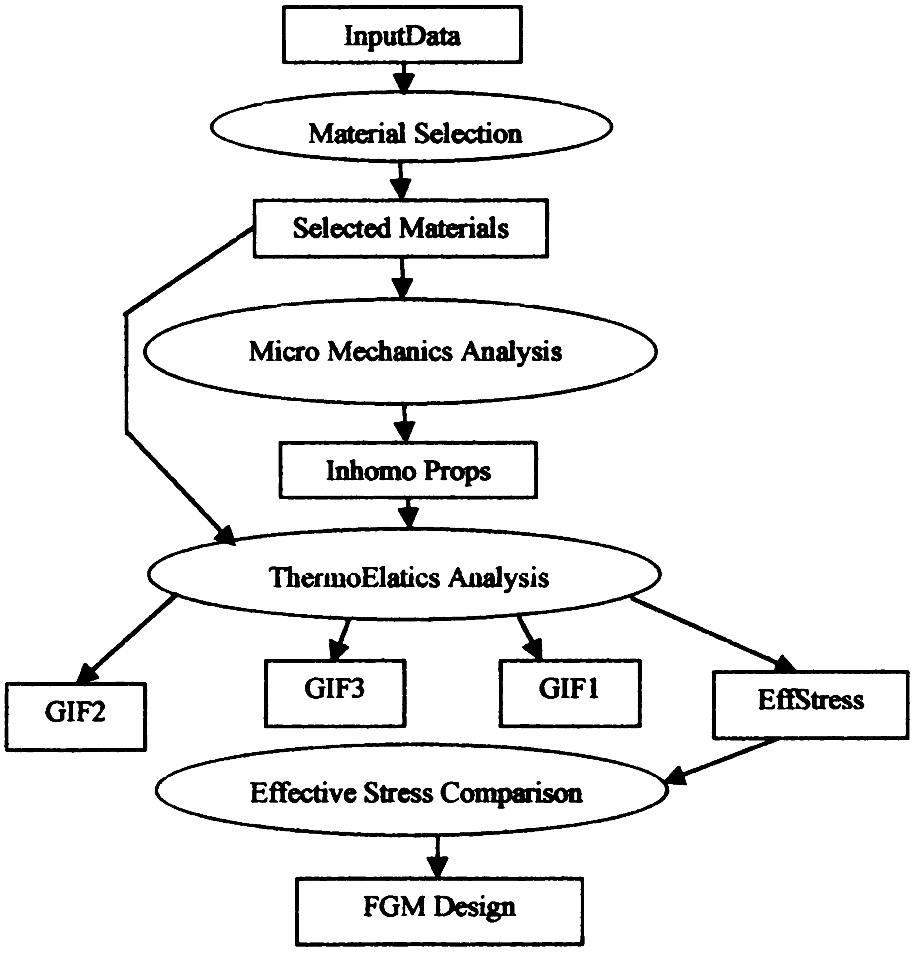

Figure - 26 FMGs design flow chart created by the framework

In computer-aided design (CAD) or analysis, modularization, which facilitates the reusing of the same process flow, is strongly desirable. This "multi-version" feature significantly reduces the complexity of a process flow while enabling the user to test as many case as possible since computer would never get tired.

5.6 Thermal Loading and Internal Pressure

5.6.1 Thermal Loading

The temperature distribution is relatively independent of the second phase profile (see Figure-27) . According to the effective thermal stress distribution reflected in Figure-28, type-1 has the lowest maxim von Mises stress which takes place at the ceramic-rich outer radius. However, considering the large margin of safety with ceramic-rich materials under compressive loading, none of these FGM cases show a clear improvement over the others. The resulting stresses of FGM type-1 and type-2 at the inner radius, the metal-rich side, are much lower than that of type-3. That is to say these two types of FGMs are safer.

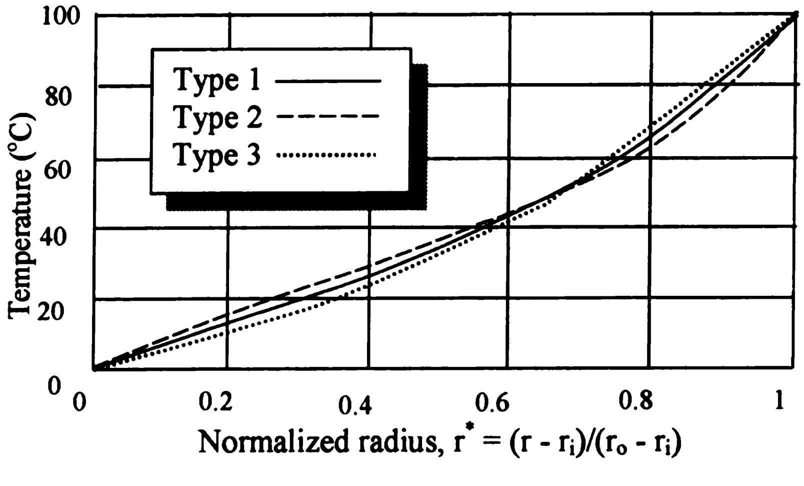

Figure - 27 Temperature fields in three type of FGMs

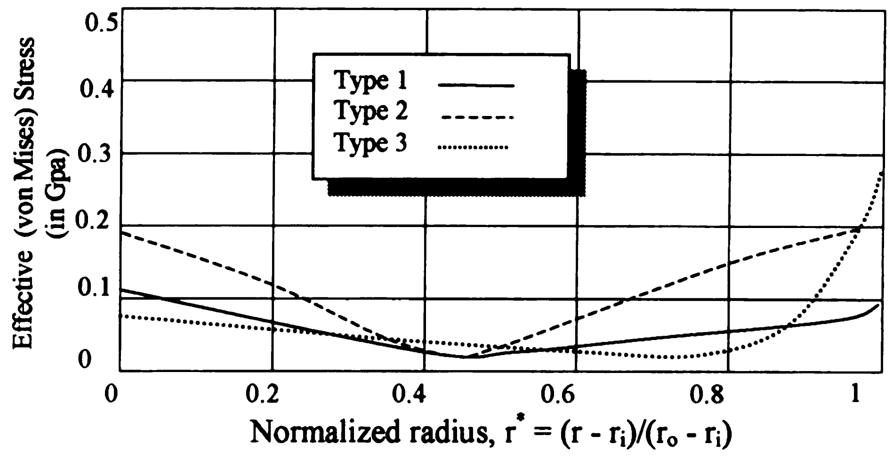

Figure - 28 von Mises stress in three types of FGM

5.6.2 Internal Pressure

The stress distribution of homogeneous thick-wall cylinders under internal and external pressures, P_i and P_o, can be determined by the following two equations.

$$\sigma_r = P_i[(r_o/r)^2-1]+P_o[(r_o/r_i)^2-(r_o/r)^2]/[\ [(r_o/r_i)^2-1] \qquad 5.29(a)$$

and

$$\sigma_\theta= P_i[(r_o/r)^2+1]+P_o[(r_o/r_i)^2-(r_o/r)^2]/[\ [(r_o/r_i)^2-1] \qquad 5.29(b)$$

where r_o and r_i are the external and internal radii, respectively. The above equations show that stress distribution is independent of materials and the maximum stress occurs at the interior of the cylinder. For stresses caused by internal pressure in FGM cylinders, the formulation derived previously can be applied without the thermal strain in Equation (5.16). Based on this formulation, the results are represented in Figure-29 with an extra stress distribution whose second phase distribution varies linearly from pure ceramic phase at the inner radius to the pure metallic phase at the outer radius. In types 1, 2 and 3, the maximum stress takes place at the outer radius, while for the linearly distributed second phase case, the maximum stress occurs inside. Therefore, FGM cylinders are useful since the service-induced defects are located outside and can be easily spotted to maintain the structural integrity.

5.7 Conclusions

Using the framework, the effectiveness of the FGMs with different second-phase distributions in reducing the stresses at critical locations has been fully explored for the case of thermal load and internal pressure of a cylinder. The use of the framework facilitates an approach toward the design of FGMs in which various distribution functions of the second phase are tried in order to reduce stress or to obtained desired stress distributions to maximize the margin-of-safety in the structure. This approach provides guidelines to the materials design engineers to reduce the wall thickness of FGM structures for a given maximum allowable stress level by appropriately designing the second phase distribution, hence low structural weight and cost for a given requirement.

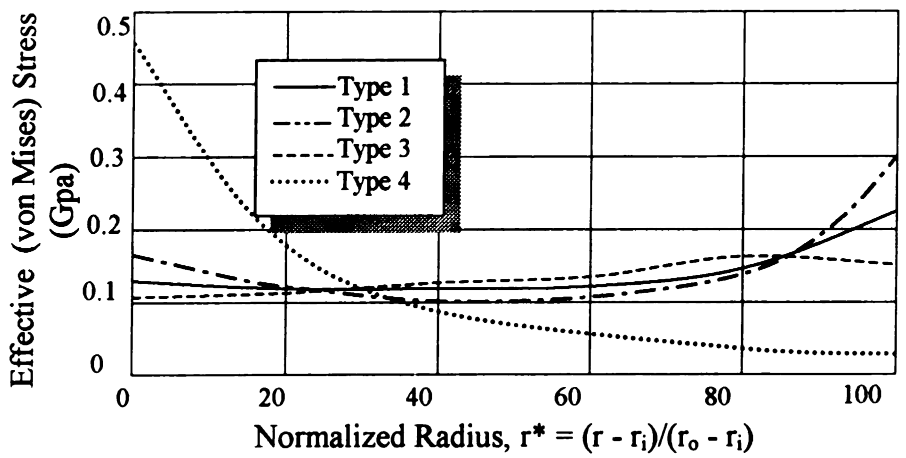

Figure - 29 von Mises stress in FGMs subjected to internal

Chapter 6

Conclusions

6.1 Contributions

The intent of this research is to design an infrastructure capable of efficiently and effectively coordinating various distributed design and manufacturing activities in either a parallel or sequential manner. An internet-based design and manufacturing process managerial framework was implemented as the prototype for the infrastructure. The main features implemented in this framework are summarized below:

Platform-independent Graphical User Interface(GUI) - The framework has a intuitive, easy-to-use graphical user interface for building, editing and executing process flows. Interaction with flow objects and operations are realized through a graphical user interface. Since the interface is encoded in Java, it has uniform appearance as well as functions for all platforms and for any Java-enabled web browsers.

Property Inheritance and Macro Substitution - Each flow object is associated with a flow object class. This flow object class allows designers to define properties that are common to all or a specific type of flow objects that are inherited from it. For the same reason, there are common system properties that are pre-defined as property macros for all flow objects. Any specific flow object may either inherit or re-define any one or all the system properties. This feature makes it more efficient to create or to maintain a process flow. All flow object classes are designed as a general-to-specific hierarchy, with children classes inheriting properties from their parents.

Outstanding Tool Software Compatibility - The implemented framework has shown its ability to effectively manage complex tasks involved in the gearbox design and manufacturing processes. Based on the available databases for bar stocks, bearings and gears, the framework brings a variety of tools into play to search for and evaluate different possible combinations until it finds the best solution. When a design is done, the framework invokes manufacturing tools to provide data for rapid prototyping and machining. Meanwhile, it also generates images for on-line simulation. During the whole process, the MIDAS framework integrated tools encoded in Java, Fortran, Unix shell scripts, Perl, Mathematica programming language, I-DEAS open language, Windows NT command scripts and HTML. It has been proven that the framework is tool software independent. In other words, an end-user do not have to worry about whether a specific tool can be captured and executed by the MIDAS framework.

Common Gateway Interface(CGI) Servers - In the current implementation of the framework, some components are interfaced with the framework's cockpit through a CGI gateway. Any remotely distributed component could be incorporated into the system through a CGI gateway, as the system's Java applet cockpit can very easily communicate with remote HTTP servers. Advantages of the CGI gateway for communicating with remote components are listed as follows: 1) *Well Defined Interface*. The web server and CGI gateway are conceptually well defined and are practically accepted; 2) *Security*

Considerations. For many organizations, system firewalls are installed. Typical firewalls allow only limited communication between internal and external networks. The *http* protocol is allowed for most systems. Access to the external tools located within the secure networks can be realized by using a web server and the CGI site proxy server.

6.2 Future Work

The present work has laid a sound foundation for many additional new features in the future. Several possibilities are presented hereafter.

New Features of User Interface. - Version 1.2 of the JDK(Java Development Kit) provides much advanced user interface utilities. Java Foundation Classes (JFC), which is among the most notable, provides advanced Graphical User Interface capabilities that have high portability. These classes provide standard new features such as scrollable canvases, drag-and-drop, and an advanced 2D-API for two-dimensional graphics support.

Remote Method Invocation (RMI). - The communication between the framework's cockpit and the external tools are currently realized via custom TCP/IP socket messaging. With all its advantages, one outstanding disadvantage is the proprietary nature of the approach. Moreover, the high overhead of maintaining, extending, and debugging custom message passing techniques could also be a heavy burden and would not be fault-proof. RMI provides an alternative way to exchange information between Java objects. The advantages include a likely standard method of communication (easy to integrate different communication components), much lower developmental overhead, hence a great way to implement collaboration functionality.

Use of JDBC. - JDBC is Commonly referred to as the Java Database Connectivity. This is a new addition to the JDK. It enables direct access to SQL-based database servers in Java code. Using this new API may enable the framework to include the storage of serialized flow objects. For instance, saving a serialized simulation run, reloading it at a later time, and continuing execution from where it was left off. It may also allow the access to remote databases for Selectors.

Common Object Request Broker Architecture (CORBA) Distributed Services. - In the current implementation of the MIDAS, all communication is implement using TCP/IP custom socket messaging that has its own disadvantages as mentioned above. One emerging industry standard, provided by the Object Management Group (OMG), is the Common Object Request Broker Architecture (CORBA). While communication Java entities can be easily implemented with RMI, a heterogeneous environment requires something more comprehensive. CORBA meets those requirements very well. *Once the framework is interfaced by CORBA, all distributed services of the system will be able to communicate through any of a number of available ORBs(Object Request Brokers).* The MIDAS programmers might be well released from the overhead of maintaining the communication utilities by allowing an ORB to execute the external tools via the standard CORBA wrappers.

Better Runtime Debugging Monitor - Although proxy monitor, scheduler monitor, flow execution graphical simulation and standard output to the Java console are already available, the lack in the means of tracing an error when it takes place is still a big issue. Especially when communication has halted or tool has its own runtime error, there is hardly any efficient way to help the end user figure out what goes wrong. In the future, a series of error code should be stipulated and a runtime error message monitor showing the users detailed error information should also be programmed. These new features will greatly help the users to resume any halted communication or to fix any corrupted tool.

APPENDICES

A. Sample HTML File Used to Invoke the Remote Servers and the MIDAS Framework's Java Applet

PrMain.html

```
<HEAD>
<title>Process Management System Demonstration</title>
</head>
<body bgcolor="#000000" text="#ffffff">
<center>
<p>
</center>
<!--#exec cgi="/cgi-user/guhua/startJFS.sh"-->
<pre>
</pre>
<tt>
<applet code=PrMain.class width=0 height=0>
<param name=username value=guest>
<param name=password value=>
<param name=debugthresh value=30>
<param name=redrawmode value=chung>
<param name=demo value=false>
<param name=platform value=unix>
<center>
<font size=4>
ERROR! Your browser is not configured to run JAVA code.
<p>
<font size=2>
If you are running Netscape v2.02+, perhaps you just have JAVA support disabled. If
you are NOT running a JAVA capable version of Netscape, you can get one <a
href="http://home.netscape.com">here</a>.
</center>
</tt>
</applet>
</body>
```

B. Sample Unix Shell Scripts Used to Invoke Remote Servers Thtough CGI

StartJFS.sh

```
#!/usr/bin/sh
MADPATH="/user/r01/chung/MAD/system"
H=`/bin/cat /etc/hostname.*`
echo "Content-type: text/html"
echo
echo "<br><font size = 2>"
echo "<center>"
echo "<h2>Executing Remote Start Hacks on <em>$H</em></h2>"
echo "Script: <strong>"$0"</strong><p></center><hr>"
echo "Attempting to start the JFS server on <strong>$H</strong>...<br>"
echo `/user/web/cgi-user/guhua/myzap 'java MultiServer' > /dev/null`
echo `/user/web/cgi-user/guhua/myzap 'java RemoteFileServer' > /dev/null`

echo `/user/web/cgi-user/guhua/myzap 'java JFSserver' > /dev/null`

cd $MADPATH/jfs-0.05
umask 0
#chmod 666 root/etc/*
echo `/soft/sparc/bin/java JFSserver ./root > /dev/null &`
echo "Attempting to start MultiServer on <strong>$H</strong>...<br>"
cd $MADPATH
echo `/soft/sparc/bin/java MultiServer > /dev/null &`
#sleep 5
echo "Attempting to start the Tool Proxy server on <strong>$H</strong>...<p>"
echo `/user/web/cgi-user/guhua/myzap 'java ProxyServer' > /dev/null`
cd $MADPATH
echo `/soft/sparc/bin/java ProxyServer CPS www > /dev/null &`
echo `/user/web/cgi-user/guhua/myzap 'java RemoteFileServer' > /dev/null`
echo "Attempting to start the RemoteFileServer on <strong>$H</strong>...<p>"
cd $MADPATH
echo `/soft/sparc/bin/java RemoteFileServer -unlimitted > /dev/null &`
echo
echo "These are the processes currently running on "$H" that are related to what we're
doing...<br><pre>"
/usr/bin/ps -ef | grep JFSserver
/usr/bin/ps -ef | grep MultiServer
/usr/bin/ps -ef | grep ProxyServer
/usr/bin/ps -ef | grep RemoteFileServer
echo "</pre>"
echo "<!--"
# -------- Run remote server ----------------------
echo `/opt/bin/lynx -dump http://www.cps.msu.edu/cgi-user/guhua/startServers.cgi |
/usr/bin/unix2dos - -`
```

C. Major Applet Class

```
/*---------------------------------
  Major Class extends Applet
 *---------------------------------*/
import java.awt.*;
import java.applet.*;
import java.util.*;
import java.io.*;
import java.net.*;


public class PrMain extends Applet implements Runnable {


  public final static String perlURL =
    new String("http://localhost/cgi-bin/drawgate.prl");


  public final static String imageName =
    new String("cool-process.jpg");


  public final static int CHUNG = 0;
  public final static int FARIBA = 1;
  public final static int DOS = 0;
  public final static int UNIX = 1;


  public static int redrawMode = CHUNG;
  public static boolean demo = false;
  public static int platform = DOS;


  static int runCount = 0;


  // public static objects that will accessed by sub-
  // modules of the system

  static StateClassDirectory taskDirectory
     = new StateClassDirectory();
  static StateClassDirectory specDirectory
     = new StateClassDirectory();
  static ProductionDatabase productionDB
```

```
   = new ProductionDatabase();
static Constraints constraints = new Constraints();
static PropertyDatabase preferences
   = new PropertyDatabase();
static String defaultSite = new String("EGR");
public static AppletContext appletContext;

Flow rootFlow = new Flow();
Cockpit cockpit;
DirectoryBrowser taskBrowser,specificationBrowser;
DatabaseBrowser productionBrowser;

static Toolkit tk;
public static URL codeBase = null;

Thread thr = null;
boolean running = true;
boolean setup = false;
long sampleInterval = 2000;

public void init(){

  codeBase = getCodeBase();
  appletContext = getAppletContext();
  tk = getToolkit();

  thr = new Thread(this);

  if(++runCount < 2){

    thr.start();
    setup=true;

  } else {

    new MessageWindow("Currently, only one copy of the
        Process Management System may be run at a time in
        a single Java Virtual Machine.  This restriction
        may be removed in the future.  If this message is
        displayed in error, PMS will not be able to start
        until your browser is restarted...",200);
    bail();

  }

  //    thr.start();
```

```
}

private void setup(){


  // instantiate the browsers
   ImageLabel im = new ImageLabel(codeBase + imageName);
   im.waitForImage(true);
   ProgressDialog pd = new
   ProgressDialog(6,300,"Initializing System
                  Cockpit...",im);
  getToolkit().sync();
  //    pd.show();

  cockpit = new Cockpit(this);

  pd.increment("Initializing Task Browser...");
  taskBrowser = new DirectoryBrowser("Task Browser",
                                     taskDirectory);
  taskBrowser.getTreeCanvas().addMouseDownHandler(new
   TaskDirHandler());

  pd.increment("Initializing Specification Browser...");
  specificationBrowser = new
                         DirectoryBrowser("Specification
                         Browser", specDirectory);

   specificationBrowser.getTreeCanvas().addMouseDownHandl
   er(new SpecDirHandler());

  pd.increment("Initializing Production Browser...");
  productionBrowser = new DatabaseBrowser("Production
                          Browser", productionDB);


  String n = getParameter("username");
  String p = getParameter("password");
  String f = getParameter("filename");
  String d = getParameter("debugthresh");
  String rdm = getParameter("redrawmode");
  String dem = getParameter("demo");
  String plat = getParameter("platform");

  pd.increment("Parsing tag parameters...");
  if(n != null && p != null){

    cockpit.usr = n;
```

```
        cockpit.pwd = p;
        cockpit.login();
      } else if(n != null || p != null){
        Debug.msg("Both a USERNAME and PASSWORD must be
                   specified.",
                   Debug.HIGHEST);
      }

      if(d != null)
        Debug.setPriorityThreshhold(Integer.parseInt(d));
      else
        Debug.setPriorityThreshhold(Debug.MEDIUM);



      if(f != null) {
        try {
       Debug.msg("filename: "+f);
       cockpit.readFile(new String("/home/guest/"+f),0); //
Consistency of the filename & version is necessary.
       cockpit.load.disable(); // Prevent from loading  &
saving the others
       cockpit.save.disable(); //
        } catch(RequestException e) {}
      }

      if(rdm != null){

        if(rdm.equalsIgnoreCase("chung"))
       redrawMode = CHUNG;
        else
       redrawMode = FARIBA;
      }

      if(dem != null){
        if(dem.equalsIgnoreCase("true"))
         demo = true;
        else
       demo = false;
      }

      if(plat != null){
        if(plat.equalsIgnoreCase("unix"))
       platform = UNIX;
        else
       platform = DOS;
      }
      pd.increment("Initializing Threads...");
      pd.increment();
      try {
        Thread.sleep(2000);
      } catch (InterruptedException e) { }

       pd.dispose();
       pd = null;



      cockpit.show();
    }



    public void run() {

      while(running){
        try {
       Thread.sleep(sampleInterval);
       if(setup){
         setup();
         setup=false;
       }
        } catch (InterruptedException e) {
       return;
        };
      }

    }

    public void showSpecBrowser(){
      if(specificationBrowser != null){
        Debug.msg("Bring up Spec Browser...");
        specificationBrowser.show();
      } else {
      specificationBrowser = new
DirectoryBrowser("Specification Browser", specDirectory);
      }
    }

    public void showTaskBrowser(){
      if(taskBrowser != null){
        Debug.msg("Bring up Task Browser...");
      taskBrowser.show();
    } else {
      taskBrowser = new DirectoryBrowser("Task Browser",
taskDirectory);
    }
  }

  public void showProductionBrowser(){
    if(productionBrowser != null){
      Debug.msg("Bring up Production Browser...");
      productionBrowser.show();
    } else {
      productionBrowser = new DatabaseBrowser("Production
Browser", productionDB);
    }

  }

  public void bail(){

    runCount--;

    if(productionBrowser != null)
      productionBrowser.dispose();

    if(taskBrowser != null)
      taskBrowser.dispose();

    if(specificationBrowser != null)
      specificationBrowser.dispose();

    // cockpit.dispose();

    // Kill the current running PrMain Thread...
    if(thr != null){
      //Debug.msg("Stopping Thread " + thr.toString());
      running = false;
    }

  }

  public static URL getStartURL(){
    return codeBase;
  }

  public Flow getRootFlow(){
    return rootFlow;
  }

  public static ProductionDatabase getProductionDatabase(){
    return productionDB;
  }

  public static StateClassDirectory
getSpecificationDirectory(){
    return specDirectory;
  }

  public static StateClassDirectory getTaskDirectory(){
    return taskDirectory;
  }

  public static AppletContext getAppContext(){
    return appletContext;
  }

}
```

D. Sample Code of the Interface Between the Framework and Tools

```
// A interface class used to invoke any
// other program in command-line

// Format:
// java call program input parameters ... output parameters


import java.util.*;
import java.io.*;
import java.awt.*;


//--------------------------------------------------------------------

public  class  call {
//       ==========
   static String trans_;
   static String str;
   int no_of_tools=200;  // subject to change for later development
   static tools tr=new tools();




/* ----------------main--------------- */

  public static void main(String argv[]) {

   tools try1 = new tools();
   tools t = new tools();
   ToIdeas T = new ToIdeas();
   rec_len l = new rec_len();
   Mat M = new Mat();
   brsel B = new brsel();
   G_P_Shaft gps = new G_P_Shaft();
   stress str_= new stress();
   fatigue fat = new fatigue();
   String[] f = new String[20];
   String[] f_= new String[20];
   String[] ff= new String[17];
   String f1="",f2="",f3="",f4="",f5="",f6="",f7="",f8="",f9="",f10="";
   String trs = "";
   float tra;
   float tra0;
   int ii=0;
   int i=0;
   int nub_Inp=0;  /* "INPUTS" location */
   int nub_Out=0;  /* "OUTPUTS" location */
   int no_of_inp=0;
int no_of_out=0;
int loop_time=0;
int bound=0;

float[] pr=new float[100];
int trai=0;
String tra_="";
int traii,b;

 trans_=argv[0];


 for (ii=0; ii<bound; ++ii){   // for starts

     if (ate("inputs")){
         nub_Inp=ii;
         no_of_inp=ii-2;
        }
     if (ate("outputs")){
         nub_Out=ii;
         no_of_out=bound-ii-1;
         bound=-1;
        }
    } // for ends here

   loop_time=nub_Out-1;

   for (ii=2; ii<loop_time+1; ++ii){   // for begins

     if(argv[0].equals("CopyF")){
       try1.CopyF(argv[ii],argv[nub_Out+ii-1]);
     }

   } // for ends here



   // ******************* Interface ********************

   if (ate("Diameter")) {
    // public float Diameter(int tn, int p)
      pr[1]=fv(argv[2],0,0);
      pr[2]=fv(argv[3],0,0);
      tra=try1.Diameter((int)(pr[1]),(int)(pr[2]));
      out(argv[5],0,0,try1.FltToStr(tra));
     }
   else if (ate("Tan_Force")){
     // float Tan_Force(float r,float pow,float Rp)
     pr[1]=fv(argv[2],0,0);
     pr[1]=fv(argv[3],0,0);
     pr[1]=fv(argv[4],0,0);
     tra=try1.Tan_Force(pr[1],pr[2],pr[3]);
     out(argv[6],0,0,try1.FltToStr(tra));
    }
```

```
else if (ate("Angle_Ft_Fr")){
  //  double Angle_Ft_Fr(float helix_angle, float pressure_angle)
  pr[1]=try1.StrToFlt(try1.goto_rec_get_field(argv[2],1,1,",","));
  pr[2]=fv(argv[3],1,2);
        tra=(float)try1.Angle_Ft_Fr(pr[1],pr[2]);
  out(argv[5],2,1,try1.FltToStr(tra));
 }
 else if (ate("ForcePlusForce")){
 // float ForcePlusForce(float f1, float f2)
  pr[1]=fv(argv[2],0,0);
  pr[2]=fv(argv[3],0,0);
  tra=try1.ForcePlusForce(pr[1],pr[2]);
  out(argv[5],0,0,try1.FltToStr(tra));
 }
 else if (ate("NormalStress")){
 // float NormalStress(float force, float nn)
  pr[1]=fv(argv[2],0,0);
  pr[2]=fv(argv[3],0,0);
  tra=try1.NormalStress(pr[1],pr[2]);
  out(argv[6],0,0,try1.FltToStr(tra));
 }
 else if (ate("Moment")) {
 // float Moment(float f,float l, float theta)
  pr[1]=fv(argv[2],0,0);
  pr[2]=fv(argv[3],0,0);
  pr[3]=fv(argv[4],0,0);
  tra=try1.Moment(pr[1],pr[2],pr[3]);
  out(argv[5],0,0,try1.FltToStr(tra));
 }
 else if (ate("Mmax")) {
 // float Mmax(float w, float m, float l)
  pr[1]=fv(argv[2],0,0);
  pr[2]=fv(argv[3],0,0);
  pr[3]=fv(argv[4],0,0);
  tra=try1.Mmax(pr[1],pr[2],pr[3]);
  out(argv[6],0,0,try1.FltToStr(tra));
 }
 else if (ate("TorqueShearStress")) {
 // float TorqueShearStress(float torque, float diameter)
  pr[1]=fv(argv[2],0,0);
  pr[2]=fv(argv[3],0,0);
  tra=try1.TorqueShearStress(pr[1],pr[2]);
  out(argv[5],0,0,try1.FltToStr(tra));
 }
 else if (ate("Von_Mis_stress")) {
 // float Von_Mis_stress(float torque, float diameter)
  pr[1]=fv(argv[2],0,0);
  pr[2]=fv(argv[3],0,0);
  tra=try1.Von_Mis_stress(pr[1],pr[2]);
  out(argv[5],0,0,try1.FltToStr(tra));
 }
 else if (ate("Max_Shear_stress")) {
 // float Max_Shear_stress(float torque, float diameter)
  pr[1]=fv(argv[2],0,0);
  pr[2]=fv(argv[3],0,0);
  tra=try1.Max_Shear_stress(pr[1],pr[2]);
```

```
   out(argv[5],0,0,try1.FltToStr(tra));
  }
  else if (ate("I_pie")){
  // float I_pie(float Diameter)
   pr[1]=fv(argv[2],0,0);
   tra=try1.I_pie(pr[1]);
   out(argv[4],0,0,try1.FltToStr(tra));
  }
 else if (ate("Max_Bend_Stress")) {
   // float Max_Bend_Stress(float M,float D, float I)
    pr[1]=fv(argv[2],0,0);
   pr[2]=fv(argv[3],0,0);
   pr[3]=fv(argv[4],0,0);
   tra=try1.Max_Bend_Stress(pr[1],pr[2],pr[3]);
   out(argv[6],0,0,try1.FltToStr(tra));
  }
 else if (ate("Mean")) {
   // float Mean(float max, float min)
   pr[1]=fv(argv[2],0,0);
   pr[2]=fv(argv[3],0,0);
   tra=try1.Mean(pr[1],pr[2]);
   out(argv[5],0,0,try1.FltToStr(tra));
  }
 else if (ate("interpolation")) {
   // float interpolation(float x1,float x2, float y1,float
   // y2,float inter_x)

   pr[1]
        =fv(argv[2],0,0);
   pr[2]
        =fv(argv[3],0,0);
   pr[3]
        =fv(argv[4],0,0);
   pr[4]
        =fv(argv[5],0,0);
   pr[5]
        =fv(argv[6],0,0);

   tra=try1.interpolation(pr[1],pr[2],pr[3],pr[4],pr[5]);
   out(argv[8],0,0,try1.FltToStr(tra));
  }
 else if (ate("out_in_GoodmanLine")) {
  // String out_in_GoodmanLine(float Sn,float Su,float Sa,float
  // Sm)

   //-------------------
   pr[1]=fv(argv[2],0,0);
   pr[2]=fv(argv[3],0,0);
   pr[3]=fv(argv[4],0,0);
   pr[4]=fv(argv[5],0,0);
   //--------------------
   str=try1.out_in_GoodmanLine(pr[1],pr[2],pr[3],pr[4]);
   out(argv[8],0,0,str);
  }
 else if (ate("Kf")) {
  // float Kf(float Kt, float q)
  // Kt = stress concentration factor;q =  average fatigue notch
  // senstivity
  // public float q(float r,float mm)
  // the input r is the notch radius (mm)
  // Note: input unit not= mm, input mm=-1

   pr[1]=fv(argv[2],0,0);

   pr[2]=fv(argv[3],0,0);

   tra=try1.Kf(pr[1],pr[2]);

   out(argv[5],0,0,try1.FltToStr(tra));
  }
 else if (ate("Kt_bending")) {

   /*
     float Kt_bending(float D,float d,float r)
   */

   pr[1]=fv(argv[2],0,0);
   pr[2]=fv(argv[3],0,0);
   pr[3]=fv(argv[4],0,0);
   tra=try1.Kt_bending(pr[1],pr[2],pr[3]);
   out(argv[6],0,0,try1.FltToStr(tra));
  }
 else if (ate("Kt_Torque")) {

   /*
     float Kt_Torque(float D,float d,float r)
   */

    pr[1]=fv(argv[2],0,0);
    pr[2]=fv(argv[3],0,0);
    pr[3]=fv(argv[4],0,0);
    tra=try1.Kt_Torque(pr[1],pr[2],pr[3]);
    out(argv[6],0,0,try1.FltToStr(tra));
  }
 else if (ate("Kt_Axial")) {
   // float Kt_Axial(float D,float d,float r)
    pr[1]=fv(argv[2],0,0);
    pr[2]=fv(argv[3],0,0);
    pr[3]=fv(argv[4],0,0);
    tra=try1.Kt_Axial(pr[1],pr[2],pr[3]);
    out(argv[6],0,0,try1.FltToStr(tra));
  }
 else if (ate("q")) {
   // float q(float r,int mm)
    pr[1]=fv(argv[2],0,0);
    pr[2]=fv(argv[3],0,0);
```

```
      tra=try1.q(pr[1],(int)pr[2]);
      out(argv[5],0,0,try1.FltToStr(tra));
     }
  else if (ate("Equal_Ra_Load")) {
    // float Equal_Rad_Load(float radial_load, float thrust_load,
    // int bearing_type)

     pr[1]=fv(argv[2],0,0);
     pr[2]=fv(argv[3],0,0);
     pr[3]=fv(argv[4],0,0);

     tra=try1.Equal_Ra_Load(pr[1],pr[2],(int)(pr[3]));
     out(argv[6],0,0,try1.FltToStr(tra));
    }

  else if (ate("Cs")) {
    /*
      float Cs(String finish, float  tensile_strength)
    */
     //pr[1]=fv(argv[2],0,0);
     pr[2]=fv(argv[3],0,0);
     tra=try1.Cs(argv[2],pr[2]);
     out(argv[5],0,0,try1.FltToStr(tra));
    }
  else if (ate("Cz")) {
    // float Cz(float speciman_diameter, int inch_or_mm)
     pr[1]=fv(argv[2],0,0);
     pr[2]=fv(argv[3],0,0);
     tra=try1.Cz(pr[1],(int)pr[2]);
     out(argv[5],0,0,try1.FltToStr(tra));
    }
  else if (ate("GPSselector")){
    /* initial file:
       1 max_rot_speed,2 max_power,3 ratio_part_hi,4
         ratio_part_low,5 Dmax,6 Dmin,

       7 no of helical, 8 life of bearing ......

       9 fraction of in/output shaft diameter in terms of the
         bearing bore size, 10 in/output shaft length,
       11  no. of teeth displayed
    */

    /* java call xxxxxx INPUTS ini_file OUTPUTS choice_file DB
       gdatabase sdatabase  ITEM   n  */


         f1=t.goto_rec_get_field(argv[2],2,1,",");
         if(!(f1.equals("ini"))){
           t.Append(argv[2],"ini");
         }

         t.dele_file("name.tmp");
         t.Append("name.tmp",argv[2]);
      f1=t.goto_rec_get_field(argv[7-1],1,1,",");

      if (!(f1.equals("gh1-1"))){
        f2=argv[7-1];
        argv[7-1]=argv[8-1];
        argv[8-1]=f2;
       }

    pr[1]=fv(argv[3-1],1,1);
    pr[2]=fv(argv[3-1],1,2);
    pr[3]=fv(argv[3-1],1,3);
    pr[4]=fv(argv[3-1],1,4);
    pr[5]=fv(argv[3-1],1,5);
    pr[6]=fv(argv[3-1],1,6);
    pr[7]=try1.StrToFlt(argv[10-1]);
    pr[8]=fv(argv[3-1],1,7);
    trai=(int)(pr[7]+0.5);
    traii=(int)(pr[8]+0.5);
    gps.gearsel_1(pr[1],
                 pr[2],
                 pr[3],
                 pr[4],
                 pr[5],
                 pr[6],
                 argv[7-1],
                 argv[8-1],
                 "gps_selected.tmp",
                 argv[5-1],
                 ",",
                 9,     // no_of_fields_gear
                 3,     // no_of_fields_shaft
                 traii, // no_of_helical
                 trai);       // item

 t.Append(argv[4],"GPS");


 /*
   choice_file(GPS):
   stock no., D_bore, pitch, teech no., pitch diameter, pressure
   angle, helical angle, hub style, face width
   1          2         3    4            5                 6
   7            8         9

  stock no., D_bore, pitch, teech no., pitch diameter, pressure
  angle, helical angle, hub style, face width
  10             11    12    13        14          15
  16 17         18

  stock no., nominal diameter,  length, stock no., nominal
  diameter,  length, cal_length

  19           20               21        22           23
  24          25      */
```

```
// Note: Pinion comes first.

// P, G, P_shaft, G_shaft

 }  // gps ends here



     else if (ate("where")){
       // java call xxx INPUTS choice_file(GPS), ini_file
          OUTPUTS go_file_name
       //                         1        2      3
       //                         4        5      6


       /* go_file

         1. pinion shaft     10 or -1
         2. gear shaft       10 or -1
         3. both             10 or -1
       */

       // java call xxx IN  choice_file(GPS) ini_file CONS
       // const.dat IOSET ioset ALTs a1  a2  a3

       //           0   1      2    3   4       5
       //           6     7     8   9  10 12


  trs=t. goto_rec_get_field(argv[2],2,1,",");


  if (trs.trim().equals("GPS")){

        pr[1]=fv(argv[3-1],1,23);
        pr[2]=fv(argv[3-1],1,20);
        pr[3]=fv(argv[4-1],1,7);

       }

  if (trs.trim().equals("ini")){

        pr[1]=fv(argv[4-1],1,23);
        pr[2]=fv(argv[4-1],1,20);
        pr[3]=fv(argv[3-1],1,7);

       }
      // java call xxx IN  choice_file(GPS) ini_file CONS
      // const.dat IOSET ioset ALTs a1  a2  a3
      //           0   1      2                3      4       5
      //           6     7     8   9  10   11
```

```
            /*    pre_where_to_go(float gear_shaft_D,

               // parameters placed here as a reminder

                  float pinion_shaft_D,
                  int no_of_helical,
                  String go_file_name
                  String ioset,
                  String alt[],
                  String const_file)
            */

        ff[1]=argv[9];
        ff[2]=argv[10];
        ff[3]=argv[11];
        ff[0]=" ";
        for (i=4; i<=16; ++i){
          ff[i]=" ";
          }

            tra_= str_.pre_where_to_go(pr[1],
                                        pr[2],
                                        trai,
                                       "/dev/null",
                                       argv[7],
                                       ff,
                                       argv[5]);
        } // where ends here

      else if(ate("g_s_force")){
         /* g_s_force( float max_rot_speed,
                            float max_power,
                            float gear_D,
                            float helix_angle,
                            float press_angle,
                            float shaft_length,
                            int helical_gear_no,
                            String outfile
                            ){
..... */
          // java call xxxxx IN choice_file(GPS) ini_file OUT
          // output_file
          //                1          2        3          4       5


          f1=t.goto_rec_get_field(argv[2],2,1,",");

          if (!(f1.equals("GPS"))){
              trs=argv[2];
              argv[2]=argv[3];
              argv[3]=trs;
             }
```

```
    pr[1]=fv(argv[3],1,1);
    pr[2]=fv(argv[3],1,2);
    pr[3]=fv(argv[2],1,14);
    pr[4]=fv(argv[2],1,7);
    pr[5]=fv(argv[2],1,6);
    pr[6]=fv(argv[2],1,25);
    pr[7]=fv(argv[3],1,7);

    trai=(int)(pr[7]+0.6);

    str_.g_s_force(pr[1],pr[2],pr[3],pr[4],
                   pr[5],pr[6],trai,argv[5]);

  } //  g_s_force

 else if(ate("p_s_force")){
  /* p_s_force( float max_rot_speed,
                    float max_power,
                    float pinion_D,
                    float helix_angle,
                    float press_angle,
                    float shaft_length,
                    int helical_gear_no,
                    String outfile
                    ){
                       .....
   */

    // java call xxxxx IN choice_file(GPS) ini_file OUT
    // output_file
    //                 1          2        3          4       5

    f1=t.goto_rec_get_field(argv[2],2,1,",");

    if (!(f1.equals("GPS"))){
        trs=argv[2];
        argv[2]=argv[3];
        argv[3]=trs;
       }

    pr[1]=fv(argv[3],1,1);
    pr[2]=fv(argv[3],1,2);
    pr[3]=fv(argv[2],1,5);
    pr[4]=fv(argv[2],1,7);
    pr[5]=fv(argv[2],1,6);
    pr[6]=fv(argv[2],1,25);
    pr[7]=fv(argv[3],1,7);
    trai=(int)(pr[7]+0.6);

    str_.p_s_force(pr[1],pr[2],pr[3],pr[4],pr[5],
                   pr[6],trai,argv[5]);
  } //  p_s_force

else if(ate("steel")){
```

```
/* steel database:

   AISI No., Condition of Steel Tensile Strength(Ksi), Yield
   Strength, Machinability, mat type  */


     /* java call xxxx IN OUt output_file DB steel_database ITEM
        n */
     //                1   2       3      4       5         6
7

     // now, no of records is 4

     //public void steels(String steel_database,
     //                   int item, int no_of_records,
     /*                   String output_file, String
                          Deli){.....
     */

     pr[1]=try1.StrToFlt(argv[7+1]);
     trai=(int)(pr[1]+0.5f);
     M.steels(argv[5+1],trai,4,argv[3+1],",");


   } // steels ends

else if (ate("bothforce")){
     /* public void gpbothforce(
                      float max_rot_speed,
                      float max_power,
                      float gear_D,
                      float pinion_D,
                      float helix_angle,
                      float press_angle,
                      float shaft_length,
                      int helical_gear_no,
                      String outfile,
                      String deli){...... */

    // java call xxxxx IN choice_file(GPS) ini_file   OUT
       output_file

    //                 1          2         3          4       5

    f1=t.goto_rec_get_field(argv[2],2,1,",");

    if (!(f1.equals("GPS"))){
        trs=argv[2];
        argv[2]=argv[3];
        argv[3]=trs;
       }
     pr[1]=fv(argv[3],1,1);
     pr[2]=fv(argv[3],1,2);
     pr[3]=fv(argv[2],1,5);
     pr[4]=fv(argv[2],1,7);
     pr[5]=fv(argv[2],1,6);
     pr[6]=fv(argv[2],1,25);
     pr[7]=fv(argv[3],1,7);
     trai=(int)(pr[7]+0.6);
     pr[8]=fv(argv[2],1,14);

     str_.gpbothforce(pr[1],pr[2],pr[8],pr[3],pr[4],
     pr[5],pr[6],trai,argv[5],",");

   }// bothforce ends here

else if (ate("stress")){

     /* void stress_analysis(float shaft_D_p,
                            float shaft_D_g,
                            String inputfile,
                            String output_file,
                            String deli,
                            float yield_stress

                           ) */

    // java call xxxx IN choice_file(GPS) inputfile(force)
    // material_file OUT outputfile
    //                 1       2             3
    //                 4          5     6


    b=4;
    for(i=2; i<=b; ++i){

        f[1]=t.goto_rec_get_field(argv[i],2,1,",");
        if(f[1].equals("GPS")){
           ii=i;
           b=-888;
          }
        }
      trs=argv[2];
      argv[2]=argv[ii];
      argv[ii]=trs;


      f[2]=t.goto_rec_get_field(argv[3],2,1,",");
      f_[2]=argv[3];
      f[3]=t.goto_rec_get_field(argv[4],2,1,",");




      f_[3]=argv[4];
      for (i=2; i<=3; ++i) {
          if (!f[i].equals("m")){
             argv[3]=f_[i];
            }
          if (f[i].equals("m")){
             argv[4]=f_[i];
            }
        }
     pr[1]=fv(argv[2],1,20);
     pr[2]=fv(argv[2],1,23);
     pr[3]=fv(argv[4],1,4);
     str_.stress_analysis(pr[1],pr[2],argv[3],argv[6],",",pr[3]);

    }//stress ends here


else if (ate("bearing")){

        //t.Append("bn_.tmp",argv[6]);
        t.dele_file("bn_.tmp");
        t.Append("bn_tmp",argv[6]);

        /* void BearingSel( //float D_bore,
                        float D_shaft,
                        String beardatabase,
                        String bearbests,
                        String Deli,
                        int no_of_fields,  // now =11
                        String choice_file,
                        int item,
                        int tot_no_of_rec,    // can let it be any
                                              //value (say 1000)
                        float axial_force,
                        float rad_force,
                        float rot_speed,
                        int lifeofbear
                        )  */

   // java call xxx IN choice_file(GPS) ini_file forcefile OUT
                    bearbests choice_file(bearing) DB
   /*                1        2            3        4       5
                     6        7                  8
                     beardatabase ITEM n
                     9           10  11 */


   // force file:

   /*   Tan_force, Angle_Ft_Fr, Rad Force, ShearForce,
        AxialForce, Torque, Central Moment,



         1          2              3           4             5
         6          7
         Moment Y, Moment Z, Total bending moment
         8            9         10
        */


       // There are three lines in the stress file. The second line
       // might be "0".
       // The first line will always consist of non-zero fields.
       // The third line contains only one field which can be "g" or
       // "p" or "gp".
       // Obviously, the 3rd line serves as an indicator indicating
       // that


          /*    g ----- first line is pertaining to gear shaft data
                p ----- first line is pertaining to pinion shaft data
                gp ---- first line  --> gear shaft data
                        second line --> pinion shaft data
          */

       // Stress file:

       /* ShearStress, TortionalShearStress, MaxBendNormalStress,
          NormalStressDueToAxialForce,
           1           2                        3
           4
           NormalStress, ShearStress, Von_Mis_Stress
           5             6              7
        */

      /*Bearing Type Codes

         1-single-row ball
         2-double-row ball
         3-cylindral roller
         4-spherical roller
      */


  // java call xxx IN choice_file(GPS) ini_file forcefile OUT
     ******** choice_file(bearing) DB
  /*                 1        2            3        4        5       6
                     7                    8
                     beardatabase ITEM n
                     9          10  11 */


          for(i=2; i<=4; ++i){
             f[1]=t.goto_rec_get_field(argv[i],2,1,",");
             if(f[1].equals("GPS")){
                ii=i;
                i=888;
               }


          }
          trs=argv[2];
          argv[2]=argv[ii];
          argv[ii]=trs;
```

```
   f[2]=t.goto_rec_get_field(argv[3],2,1,",");
   f_[2]=argv[3];
   f[3]=t.goto_rec_get_field(argv[4],2,1,",");
   f_[3]=argv[4];

   for (i=2; i<=3; ++i) {
      /* if (f[i].equals("GPS")){
          argv[2]=f[i];
         }  */
       if (f[i].equals("ini")){
          argv[3]=f_[i];
         }
       if (!(f[i].equals("ini"))){
          argv[4]=f_[i];
     }
   }


 /* void BearingSel( //float D_bore,
                     float D_shaft,
                     String beardatabase,
                     String bearbests,
                     String Deli,
                     int no_of_fields,  // now =11
                     String choice_file,
                     int item,
                     int tot_no_of_rec, // can let it be any
                                        // value (say 1000)
                     float axial_force,
                     float rad_force,
                     float rot_speed,
                     int lifeofbear)  */

// java call xxx IN choice_file(GPS) ini_file forcefile OUT
                bearbests choice_file(bearing) DB
/*               1        2            3        4       5
                 6         7                 8
                 beardatabase ITEM n
                 9          10  11 */
```

```
// bearing file format
 /* stock no, outer D, bore D, width, static radial load(lbs),
    dynamic radial load,
       1         2        3      4          5
```

```
        6
      static thrust load, length unit code, force unit code,
      fillet radius, bearing type
              7                 8                9
              10           11
   */
 // java call xxx IN choice_file(GPS) ini_file forcefile OUT
 // bearbests choice_file(bearing) DB
 /*               1        2           3        4       5
                 6-1       7-1               8-1
                 beardatabase ITEM   n
                 9-1         10-1  11-1 */


     pr[1]=fv(argv[2],1,23);
     pr[2]=tryl.StrToFlt(argv[11-1]);
     trai=(int)(pr[2]+0.5f);
     pr[3]=fv(argv[4],1,5);
     pr[4]=fv(argv[4],1,3);
     pr[5]=fv(argv[3],1,1);
     pr[6]=fv(argv[3],1,8);

     t.dele_file("bn_.tmp");
     t.Append("bn_.tmp",argv[6]);
     B.BearingSel(pr[1],argv[9-1],"bears.tmp",",",11,argv[7-
     1],trai,1000,pr[3],pr[4],pr[5],(int)(pr[6]+0.5f));


    }// end of bearing

 else if(ate("finish")){
 //  void finish_sel(String material_type,int item, String
                     mat_file)
 //  ---------------
 //  this is actually finish condition output file
 // java call xxx IN material_file OUT outfile_finish DB ITEM n
 //                1      2         3          4        4  5  6

      tra_=tryl.goto_rec_get_field(argv[2],1,6,",");
      pr[1]=tryl.StrToFlt(argv[6+1]);
      trai=(int)(pr[1]+0.5f);

      tryl.dele_file(argv[4]);

     // fat.finish_sel(tra_,trai,argv[4]);

       fat.finish_sel("steel",trai,argv[4]);


    } // finish ends here

 else if (ate("fatigue")){


   /*  String fatigue_1 (String bearing_sel,
                      int bear_bore_D_field_no,
                      int bear_fillet_field_no,
                    String shaft_sel,
                    int shaft_D_field_no,
                    String deli,      // ","
                    float max_bend_normal_stress,
                    float axial_normal_stress,
                    float torque_shear_stress,
                    String finish_condition,
                    float ultimate_stress,
                    String L_unit       // "in"
                    ) */

// java call xxx IN bearingchoicefile GPSchoicefile forcefile
//               1      2                   3          4
//                 finishselectedfile materialselectedfile
                   OUT outfile
//                               5                  6
//                 7      8


  for (i=2; i<=6; ++i){
     f_[i]=argv[i];
    }


  for (i=2; i<=6; ++i) {

      trs=t.goto_rec_get_field(f_[i],2,1,",");

      if (trs.equals("b")){
         argv[2]=f_[i];
         //f_[i]="vvv";
        }

  }

  for (i=2; i<=6; ++i) {

      trs=t.goto_rec_get_field(f_[i],2,1,",");

      if (trs.equals("m")){
         argv[6]=f_[i];
         //f_[i]="vvv";
        }

  }

  for (i=2; i<=6; ++i) {

      trs=t.goto_rec_get_field(f_[i],1,1,",");

      if (trs.equals("m")){
         argv[5]=f_[i];
       }
      if (trs.equals("p")){
         argv[5]=f_[i];
       }
      if (trs.equals("h")){
            argv[5]=f_[i];
          }
        if (trs.equals("g")){
            argv[5]=f_[i];
          }
        if (trs.equals("f")){
            argv[5]=f_[i];
          }
    }

    for (i=2; i<=6; ++i) {

        trs=t.goto_rec_get_field(f_[i],2,1,",");

        if (trs.equals("GPS")){
            argv[3]=f_[i];

           }

    }

    for (i=2; i<=6; ++i) {

        trs=t.goto_rec_get_field(f_[i],3,1,",");

        if (trs.equals("p")){
            argv[4]=f_[i];
           }
        if (trs.equals("gp")){
            argv[4]=f_[i];
           }
        if (trs.equals("g")){
            argv[4]=f_[i];
           }
    }


pr[1]=fv(argv[4],1,3);
pr[2]=fv(argv[4],1,5);
pr[3]=fv(argv[4],1,2);
tra_=try1.goto_rec_get_field(argv[5],1,1,",");
pr[4]=fv(argv[6],1,3);

str=fat.fatigue_1(argv[2],3,11-
    1,argv[3],20,",",pr[1],pr[2],pr[3],tra_,pr[4],"in");

try1.dele_file(argv[8]);

if (str.equals("in")){




// java call xxx IN bearingchoicefile GPSchoicefile forcefile
//                1     2                 3         4
//                  finishselectedfile materialselectedfile
```

```
                    OUT outfile
  //                              5                  6
  //                              7    8
  /* static void last(String f,
                  String initial,
                  String force,
                  String stress,
                  String mat,
                  String finish,
                  String gps,
                  String bearing,
                  String deli){.......
   */
    trs=t.goto_rec_get_field("name.tmp",1,1,",");
    f1=t.goto_rec_get_field("strname.tmp",1,1,",");
    f2=t.goto_rec_get_field("bn_.tmp",1,1,",");


    last(argv[8],trs,argv[4],f1,argv[6],argv[5],argv[3],
         f2,",");
    System.exit(0);
   }

 if (str.equals("out")){
    try1.Append(argv[8],"failed");
    System.exit(1);
   }

 } // fatigue ends here

else if(ate("ToIdeas")||ate("toideas")){
/*   To_Ideas(String fA_shaft,
              String fB_shaft,
              String f_bearingAa,
              String f_bearingAb,
              String f_bearingBa,
              String f_bearingBb,
              String f_pinion,
              String f_gear,
              String gear,
              String initial,
              String force,
              String stress,
              String mat,
              String finish,
              String gps,
              String bearing,
              String fatigue,
              String deli) */
```

```
// java call ToIdeas IN gear initial force stress mat
// finish gps bearing fatigue OUT  f1 f2 f3 f4 f5 f6 f7 f8
//              0    1   2    3      4       5     6    7
//              8     9      10     11   12 13 14 15 16 17
//              18 19

for (i=2; i<=10; ++i){
   f_[i]=argv[i];
  }


for (i=4-2; i<=10; ++i) {

    trs=t.goto_rec_get_field(f_[i],2,1,",");

    if (trs.equals("b")){
       argv[2]=f_[i];
       //f_[i]="vvv";
      }

}


for (i=4-2; i<=10; ++i) {

    trs=t.goto_rec_get_field(f_[i],2,1,",");

    if (trs.equals("m")){
       argv[6]=f_[i];
       //f_[i]="vvv";
      }

}

for (i=4-2; i<=10; ++i) {

    trs=t.goto_rec_get_field(f_[i],1,1,",");

    if (trs.equals("m")){
       argv[5]=f_[i];
     }
    if (trs.equals("p")){
       argv[5]=f_[i];
     }
    if (trs.equals("h")){
       argv[5]=f_[i];
     }
    if (trs.equals("g")){
       argv[5]=f_[i];
     }
    if (trs.equals("f")){
       argv[5]=f_[i];
     }
}
```

```
  for (i=4-2; i<=10; ++i) {

      trs=t.goto_rec_get_field(f_[i],2,1,",");

      if (trs.equals("GPS")){
         argv[3]=f_[i];

        }

  }

  for (i=4-2; i<=10; ++i) {

      trs=t.goto_rec_get_field(f_[i],3,1,",");

      if (trs.equals("p")){
         argv[4]=f_[i];
        }
      if (trs.equals("gp")){
         argv[4]=f_[i];
        }
      if (trs.equals("g")){
         argv[4]=f_[i];
        }
  }

   for (i=4-2; i<=10; ++i) {

      trs=t.goto_rec_get_field(f_[i],1,1,",");

      if (trs.trim().equals("45")){
         argv[8]=f_[i];
        }


  }


for (i=2; i<=10; ++i) {
      trs=t.goto_rec_get_field(f_[i],1,1,",");
      if (trs.trim().equals("gh1-1")){
         f_[2]=f_[i];
      }
}

pr[1]=fv(argv[4],1,3);
pr[2]=fv(argv[4],1,5);
pr[3]=fv(argv[4],1,2);
tra_=try1.goto_rec_get_field(argv[5],1,1,",");
pr[4]=fv(argv[6],1,3);
```

```
// java call xxx IN bearingchoicefile GPSchoicefile forcefile
//                 1       2                  3         4
//                     finishselectedfile materialselectedfile
//                      OUT outfile



//                                 5                  6
//                        7   8

/*
 As a reminder
 static void last(String f,
                  String initial,
                  String force,
                  String stress,
                  String mat,
                  String finish,
                  String gps,
                  String bearing,
                  String deli){.......
  */
   trs=t.goto_rec_get_field("name.tmp",1,1,",");
   f1=t.goto_rec_get_field("strname.tmp",1,1,",");
   f2=t.goto_rec_get_field("bn_.tmp",1,1,",");

  //

  last(argv[8],trs,argv[4],f1,argv[6],argv[5],
       argv[3],f2,",");

  // java call ToIdeas IN gear initial force stress mat
  // finish gps bearing fatigue OUT  f1 f2 f3 f4 f5 f6 f7 f8
  //              0    1  2    3      4       5    6    7
  //              8     9      10     11  12 13 14 15 16 17
  //              18 19


  for (i=12; i<=19; ++i){
      t.dele_file(argv[i]);
     }


  T.To_Ideas(argv[12],
             argv[13],
             argv[14],
             argv[15],
             argv[16],
             argv[17],
             argv[18],
             argv[19].trim(),
             f_[2],  //String gear,
             trs,    //String initial,
             argv[4],//  String force,
             f1,     //String stress,
             argv[6],//String mat,
                    argv[5],//String finish,
                    argv[3],//String gps,
                    f2,     //String bearing,
                    argv[8],//String fatigue,
                    ",");
   }


   else{}
   // do nothing here




}//main method ends here




// -------- useful methods -------


  // ------- out -------
  static void out(String file, int rec, int field,String cont){


//tools tryl=new tools();
    //tryl.repl_field(file,rec,field,",",cont);

     tr.Append(file, cont);
     return;

   }


  // ------- val -------
  static String val(String file, int r, int f){
     // file --- file name
     // r ---- record no.
     // f ---- field no.

     String hu;

     hu=tr.goto_rec_get_field(file,r,f,",");
     return hu;
    }

 // ------ fv -------

   static float fv(String file, int r, int f){

      float hu;
      hu=fl(val(file,r,f));

      return hu;

    }
```

```
// --------  est_max_pitch -----------

   static float  est_max_pitch(float speed,float p,float radius){

     //using: lb and inch.



     float[]
         pitch={120f,96f,80f,72f,64f,48f,32f,24f,20f,16f,12f};
     float[]
         f={65f,82f,98f,105f,115f,140f,195f,245f,280f,318f,350f};
     //forces


     float forc;
     float loc;
     int loc_;

      forc = tr.Tan_Force(speed,p,radius);
      loc =  tr.locate(f,11,forc);
      loc_ = (int)( loc+1);

      return pitch[loc_];
   }


// -------- ate ----------

 static  boolean ate(String method){

      String lll="0";
      lll=trans_.toLowerCase();

       if (lll.equals(method.toLowerCase())) {return true;}
       return false;

  }


// -------- in --------

/* static int in(int i){

       return (int)(tr.StrToFlt(input_v[i]));

     }*/
```

```
// ------- fl ---------
static float fl(String in__){
// convert int to float
       return tr.StrToFlt(in__);
  }



// ------- final ------

   static void last(String f,
                    String initial,
                    String force,
                    String stress,
                    String mat,
                    String finish,
                    String gps,
                    String bearing,
                    String deli){

        tools t=new tools();
        String s1,s2,s3,s4,s5,s6,s7,s8;
        String[] s=new String[30];

        String d=deli;
        int i=0;
        //String[] e1=new String[30];
        //String[] e2=new String[30];

        String[] e1={"  stock no. ", "  D_bore(in) ", "  pitch ",
                    "  teech no. ", "  pitch diameter(in) ",

                    "  pressure angle ", "  helical angle ", "
                    hub style ", "  face width(in) "};
        String[] e2={"  stock no. ", "  nominal diameter(in) ",
                    "  available max length(in) "};
        String[] e3={"  Tan_force (lb)", "  Angle_Ft_Fr", "  Rad
                    Force(lb)", "  Shear Forc(lb)", "  Axial
                    Force(lb)" ,
                    "  Torque(lb in)", "  Central Moment(lb
                    in)",
                    "  Moment Y(lb in)", "  Moment Z(lb in)", "
                    Total bending moment(lb in)"};

        String[] e4={"  ShearStress(ksi)","
                    TortionalShearStress(ksi)","
                    MaxBendNormalStress(ksi)",
                    "  NormalStressDueToAxialForce(ksi)",  "
                    NormalStress(ksi)", "  ShearStress(ksi)",
                    "  Von_Mis_Stress(ksi)"};
        String[] e5={ "  stock no", "  outer D(in)", "  bore
        D(in)", "  width(in)", "  static radial load(lbs)", "
        dynamic radial load(lbs)",
                     "  static thrust load(lbs)", "  length unit
                     code", "  force unit code", "  fillet
                     radius(in)","  bearing type code"};
   t.dele_file(f);
   s1=t.goto_rec_get_field(gps,1,7,deli);


   t.Append(f,s1);t.Append(f,s1);
   t.Append(f,"  SUCCESSFUL!");
   t.Append(f,"Fatigue Analysis Has Been Finished and This
            Design is a Well-Done");
   t.Append(f,"Gearbox Key Data Are Listed As Follows");
   t.Append(f," ");

   t.Append(f," Basic Data ");

   /* initial file:
   1 max_rot_speed,2 max_power,3 ratio_part_hi,4
   ratio_part_low,5 Dmax,6 Dmin,
   7 no of helical, 8 life of bearing ......
   */



s1=t.goto_rec_get_field(initial,1,1,deli);
t.Append(f,"  Max. rotational speed="+s1+"rpm");
s2=t.goto_rec_get_field(initial,1,2,deli);
t.Append(f,"  Max. power="+s2+"w");
s3=t.goto_rec_get_field(initial,1,3,deli);
   s4=t.goto_rec_get_field(initial,1,4,deli);
   s5=t.goto_rec_get_field(initial,1,8,deli);
   t.Append(f,"  Transmission ratio="+s3+" : "+s4);
   t.Append(f,"  Design  life="+s5+"hr");
   //t.Append(f,"  Design  life="+s5);
   t.Append(f," ");
   t.Append(f," Material Data");

   /* steel database:

     AISI No., Condition of Steel Tensile Strength(Ksi),
     Yield Strength, Machinability, mat type  */

//       1          2                    3
         4                 5            6

   s6=t.goto_rec_get_field(f,1,6,d);
   s7=t.goto_rec_get_field(f,1,1,d);
   s8=t.goto_rec_get_field(f,1,2,d);
   t.Append(f,"  AISI No.="+s7);
   t.Append(f,"  Material="+s6);
   t.Append(f,"  Condition="+s8);
   s1=t.goto_rec_get_field(f,1,3,d);
   t.Append(f,"  Tensile Strength(Ksi)="+ s1);
   s1=t.goto_rec_get_field(f,1,4,d);
   t.Append(f,"  Yield Strength(Ksi)="+ s1);
   s1=t.goto_rec_get_field(f,1,5,d);
   t.Append(f," ");
   t.Append(f,"  Machinability="+s1);
```

```
  /* choice_file(GPS):
  stock no., D_bore, pitch, teech no., pitch diameter, pressure
  angle, helical angle, hub style, face width



  1          2        3    4            5                 6
  7            8         9

  stock no., D_bore, pitch, teech no., pitch diameter, pressure
  angle, helical angle, hub style, face width

  10              11    12    13       14          15
  16      17          18

  stock no., nominal diameter,  length, stock no., nominal
  diameter,  length, cal_length

  19            20                21       22           23
  24          25      */

// Note: Pinion comes first.


      t.Append(f," ");

     for(i=0; i<25; ++i){
        s[i]=t.goto_rec_get_field(gps,1,(i+1),d);
}

     t.Append(f," Pinion Data");
     for(i=0; i<9; ++i){
       t.Append(f,e1[i]+"= "+s[i]);
      }
      t.Append(f," ");
     t.Append(f," Gear Data");
     for(i=9; i<18;  ++i){
       t.Append(f,e1[i-9]+"= "+s[i]);
      }
      t.Append(f," ");
     t.Append(f," Pinion Shaft Data");
     for(i=18; i<21;  ++i){
       t.Append(f,e2[i-18]+"= "+s[i]);
      }
      t.Append(f," ");
     t.Append(f," Gear Shaft Data");
     for(i=21; i<24;  ++i){
       t.Append(f,e2[i-21]+"= "+s[i]);
      }
     t.Append(f," ");

    s1=t.goto_rec_get_field(finish,1,1,d);
    t.Append(f, "  Finish Condition = "+s1);
```

```
   t.Append(f," ");


// force file:

/*   Tan_force, Angle_Ft_Fr, Rad Force, ShearForce,
     AxialForce, Torque, Central Moment,

     1          2             3            4             5
     6          7

     Moment Y, Moment Z, Total bending moment
     8           9         10
 */


// There are three lines in the stress file. The second line
// might be "0".

// The first line will always consist of non-zero fields.
// The third line contains only one field which can be "g" or
// "p" or "gp".

// Obviously, the 3rd line serves as an indicator indicating
// that

   /*    g ----- first line is pertaining to gear shaft data
         p ----- first line is pertaining to pinion shaft data
         gp ---- first line  --> gear shaft data
                 second line --> pinion shaft data
   */

 s1=t.goto_rec_get_field(force,3,1,d);

 if (s1.equals("g")){

    t.Append(f," Pinion Forces and Moments");
    for (i=1; i<11; ++i){
      s[i]=t.goto_rec_get_field(force,1,i,d);
      t.Append(f,e3[i-1]+"="+s[i]);
     }

   }

 if (s1.equals("g")){

    t.Append(f," Gear Forces and Moments");
    for (i=1; i<11; ++i){
      s[i]=t.goto_rec_get_field(force,1,i,d);
      t.Append(f,e3[i-1]+"="+s[i]);
     }

   }

 if (s1.equals("gp")){
     t.Append(f," Gear-Pinion Forces and Moments (gear data
             come first)");


     for (i=1; i<11; ++i){

       s[i]=t.goto_rec_get_field(force,1,i,d);
       t.Append(f,e3[i-1]+"="+s[i]);

      }

     for (i=1; i<11; ++i){

       s[i]=t.goto_rec_get_field(force,2,i,d);
       t.Append(f,e3[i-1]+"="+s[i]);

      }
    }

 t.Append(f," ");



// Stress file:
//-----------------------------------------------------------
/*
   ShearStress, TortionalShearStress, MaxBendNormalStress,
   NormalStressDueToAxialForce,

    1             2                         3
    4
   NormalStress, ShearStress, Von_Mis_Stress
    5             6            7
 */


t.Append(f," Stresses");

for(i=1; i<=7; i=i+1){


      s[i]=t.goto_rec_get_field(stress,1,i,d);
      t.Append(f,e4[i-1]+"="+s[i]);

 }
```

```
        // bearing file format:

        /* stock no, outer D, bore D, width, static radial load(lbs),
           dynamic radial load,

               1          2         3      4          5
               6
           static thrust load, length unit code, force unit code,
           fillet radius, bearing type
                  7                   8                 9
                 10              11
         */

       t.Append(f," ");
       t.Append(f," Bearing Data");

       for (i=1; i<=11; ++i){
            s[i]=t.goto_rec_get_field(bearing,1,i,d);
            t.Append(f,e5[i-1]+"="+s[i]);
         }


     t.Append(f,t.goto_rec_get_field("idf.tmp",1,1,",")
              +","+t.member[2-1]);

     return;
    }

} // class ends here
```

E.Sample Execution Procedure and the Input/Output

Gearbox Design

```
gb.tmp:  the name of the single-lined GPSselected file used for
         testing.
IN=INPUTS
OUT=OUTPUT
DB=DATABASE
item=ITEM

Initial Input File: input.dat

Database Files:
 gear_1.db
 shaft_1.db
 steel.db

                                                 -----------
1) G/P/S selector
java call GPSselector IN input.dat OUT gb.tmp DB gear_1.db shaft_1.db
item 0
java call GPSselector IN input.dat OUT gb.tmp DB gear_1.db shaft_1.db
item 1
...
..
.
java call GPS IN input.dat OUT gb.tmp DB gear_1.db shaft_1.db item 0

2)a pre-evaluation function
java call where IN gb.tmp input.dat OUT go.dat


3)Forces Calculations
any of the following three productions
 java call g_s_force IN gb.tmp input.dat OUT force.dat
 java call p_s_force IN gb.tmp input.dat OUT force.dat
 java call bothforce IN gb.tmp input.dat OUT force.dat


4) Material Selector
 java call steel IN  df OUT mat.dat DB steel.db ITEM 0



5) Stress Analysis
 java call stress IN gb.tmp force.dat mat.dat OUT stress.dat


6) "Bear" Selector
 java call bearing IN gb.tmp input.dat force.dat OUT  best_bearing.dat
DB bearing.db ITEM 0
```

7) Surface Finish Condition Selector

```
java call finish IN steel.db OUT finish.dat DB ITEM 0
```

8) Fatigue Analysis

```
java call fatigue IN best_bearing.dat gb.tmp force.dat finish.dat
           mat.dat OUT fat.dat
```

9) To I-Deas

```
java call ToIdeas in gear_1.db input.dat force.dat stress.dat mat.dat
finish.dat gb.tmp best_bearing.dat fat.dat OUT f1 f2 f3 f4 f5 f6 f7 f8
```

10) pre-eval.

```
java call where IN gb.tmp input.dat CONS const.dat IO "io gh" ALTs
PinionForce GearForce BothForce
```

FGM Design

Material Selector

```
java FGM_Mat_Sel_Kwon INPUTS OUTPUTS mat BD Materials_FGM.db item 0
```

Micro-Mechanics Calculations

```
gh_new.bat INPUTS mat OUTPUTS funct
```

Calculation of Temperature Distribution

```
Dist_new.bat INPUTS mat  fraction? OUTPUTS tempD 1.gif 2.gif
```

Calculate stresses and displacements

```
stress_fgm_new.bat INputs tempD funct fraction OUTputs  st.gif st
```

*Note: the order of the three inputs is arbitrary

Calculate effective stress and compare with yield stress

```
java StressAnalysis INPUTS  st  mat OUTPUTS effstrs
```

*Note: if the effective stress is higher than the yield stress, the program will report a failure(exit(1)) and you should go back and try another volume fraction function.

```
Effective stresses comparison
 java finalcompare INPUTS effstrs1 effstrs2 effstrs3 effstrs4...
       OUTPUTS Best

      *Note:  there can be as many input files as possible.
              Since there, at this point, are only four types
              of volume fraction functions, there are only
              four input files which contains effectives stresses
              corresponding to different types of volume fraction
              function.

          IMPORTANT, READ Carefully:
               we should NOT go further to conduct this task
               until all four or whatever number of effective
               stress files are all ready. If there are more
               volume fraction functions to try, just go back
               to calculation of temperature distribution and
               try a new input and go all the way through to get the
               effective stress file for that particular fraction
               function. Repeat this till there is not more
               volume fraction function to try.
```

References

References

1. George E. Dieter, Engineering Design, A Materials and Process Approach, McGraw-Hill, 1991

2. Clive L. Dym, Engineering Design, Cambridge University Press, 1994

3. T. Winograd and F. Flores, Understanding Computers and Cognition, Ablex publishing, Norwood. NJ, 1986

4. C. L. Dym and R. E. Levitt, Knowledge-Based Systems in Engineering, McGraw-Hill, New York, 1991

5. M. Asimow, Introduction to Design, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1962

6. J. L. Nevins and D. E. Whitney (eds.), Concurrent Design of Products and Processes, McGraw-Hill Publ. Co., New York, 1989

7. David Belson, Handbook of design, manufacturing and automation, Chapter 2, Concurrent Engineering, John Wiley & Sons, Inc., 1994

8. Y. Hazony, Towards the Factory of the Future, Perspectives in Computing(IBM), 3(4), 4-11 (Dec. 1983).

9. Carl Machover, The CAD/CAM Handbook, McGraw-Hill, 14 – 17, 1996

10. Y. Hazony, Chapter 8, Handbook of Design, Manufacturing and Automation, John Wiley & Sons, Inc., 1994

11. L. Zeidner and Y. Hazony, *Seamless Design-to-Manufacturing*, Journal of Manufacturing Systems, 11(4), 269-284, 1992

12. Chung, M. J., Carmichael, L., and Dukes, M., 1996, *Managing a RASSP Design Process*, Computers in Industry, 30, pp. 49-61

13. Cutkosky, M. R., Tenebaum, J. M., and Glicksman, J., 1996, *Madefast: Collaborative Engineering over the Internet*, Communications of the ACM, 39, 9, pp. 78-87

14. Lyons, K. *RaDEO Project Overview*, [Online] Available http://www.cs.utah.edu/projects/alpha/arpa/mind/index.html, 1997

15. Berners-Lee, T., Cailliau, R., Luotonen, A., 1994, *The World-Wide Web*, Communications of the ACM, 37, 8

16. Erkes, J. W., Kenny, K. B., Lewis, J. W., Sarachan, B. D., Sobololewsky, M. W., and Sum, R. N., *Implementing Shared Manufacturing Services on the World-Wide Web,* Communication of the ACM, 39, 2, pp. 34-45, 1996

17. Andreoli, J-M., Pacull, F., and Pareschi, R., 1998, *XPECT: A Framework for Electronic Commerce*, IEEE Internet Computer, 1, 4, pp. 40-48

18. Mark Klein, Integrated Support for Process, Conflict, and Rationale Management in Coorperative Design, Handbook of Design, Manufacturing and Automation, page 65-75, 1994

19. R.G. Smith, The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, IEEE Trans. Comput. C29(12), 1104-1113, Dec. 1980

20. T. Winograd, "A Language / Action Perspective on the Design of Cooperative Work", Proc. CSCW 86, page 203-230, 1986

21. B94 Precision Mechanical Components, W.M.Berg, Inc, 1996

22. Arthur H. Burr, and John B. Cheatham, Mechanical Analysis and Design, Prentice Hall, 1995

23. Marks' Standard Handbook for Mechanical Engineers, eighth edition, McGRAW-HILL Book Company, page 8-138 through 8-140, 1979

24. *Design of Transmission Shafting*, ANSI/ASME B 106.1M-1985, pp. 4-6, 1985

25. F. W. Paul, Jr. and T.R. Faucett, The Superposition of Stress-Concentration Factors, J. Engrg. Industry, Trans. ASME, Vol. 84, pp. 129-134, 1962

26. Mori, T. and Tanaka, K. Average Stress in Matrix and Average Elastic Energy of Materials with Mistifitting Inclusions, Acta Metallurgica et Materialia, 21, pp. 571-574, 1973

27. Wu, T. T., The Effect of Incluction Shape on the Elastic Moduli of a Two-Phase Material, International Journal of Solid and Structure, 2, pp. 1-8, 1985

28. Eshelby, J. D., The Determination of the Field of an Ellipsoidal Inclusion and Related Problems, Proceeding of the Royal Society, Lodon A, 241, pp. 376 – 396, 1957

29. Hatta, H. and Taya, M., Equivalent Inclusion Method for Steady State Heat Conduction, International Journal of Engineering Science, 24, pp. 1159 – 1172, 1989

30. Timoshenko, S. P. and Goodie, J. G., Theory of Elasticity, 3rd Edition, McGraw-Hill, N.Y., P. 442, 1970