INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

OPEN ARCHITECTURE CONTROL

FOR

INTELLIGENT MACHINING SYSTEMS

By

RICHARD W. TELTZ

A Thesis Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

OPEN ARCHITECTURE CONTROL

FOR

INTELLIGENT MACHINING SYSTEMS

DOCTORATE OF PHILOSOPHY (1998)

(Mechanical Engineering)

McMASTER UNIVERSITY

HAMILTON, ONTARIO

CANADA, L8S4L7

TITLE: OPEN ARCHITECTURE CONTROL FOR INTELLIGENT MACHINING SYSTEMS

AUTHOR: RICHARD W. TELTZ

B.Eng, M.Eng, (Mech. Eng) McMASTER UNIVERSITY

HAMILTON, ONTARIO,

CANADA

SUPERVISOR: Dr. M. A. Elbestawi

NUMBER OF PAGES: xii, 168

ACKNOWLEDGEMENTS

I would to acknowledge my most sincere gratitude to Dr. Elbestawi for his leadership, expertise, and unending support. I am proud and grateful to have the good fortune of his mentorship and friendship.

I would also like to thank my supervisory committee, Dr. Bone, Dr. Capson, and Dr. Poehlman, for their advice and guidance throughout the course of this work.

Special appreciation is extended to the Mechanical Engineering Department staff for their assistance and encouragement over the years. My colleagues at the IMMRC, both past and present, have made all of our endeavors here memorable and irreplaceable.

Thank you Mary, Stephanie and Emily for always being behind me and for the inspiration to complete this work. For my parents Margaret and Wolfgang and all of my family, thank you for the endless support and encouragement over the many years of my career as a student.

ABSTRACT

The purpose of this study is to examine the role that Open Architecture Control concepts have in the application of Intelligent Machining Systems. Open Architecture Control is a relatively new field whose original intent was based on the use of "Open System" computer science concepts in the development of integrated manufacturing systems. Various manifestations of Open Architecture Control systems has been published in the open literature however, in many ways the original intent of the idea has been often obfuscated by the sometimes dissimilar interests of the engineering, computer science, academic and commercial realms involved.

Intelligent Machining Systems refers to application of sensing, monitoring and control technologies to machining processes with the intent of:

- improving the economic performance of machining systems,
- controlling the processes involved in a comprehensive manner.

Consequently, such systems represent an integration of technologies which individually address only a limited part of the overall potential for economic gain. The use of "higher level" information and control functions is a common element in Intelligent Machining Systems. These may include components of Artificial Intelligence, process planning, and supervisory control technologies.

The need for a high level of integration in Intelligent Machining Systems (IMS)

has presented a difficulty for their practical application. Open Architecture Control (OAC) addresses the integration problem directly and, if the proper Open Systems issues are well considered, can enable IMS technology for commercial use.

The OAC system developed in this work does address Open Systems concepts, and has been designed with consideration for the needs of IMS's. This has been achieved through the formulation and implementation of an IMS on the designed OAC. Technologies such as process sensing, monitoring and control, in addition to planning, simulation and supervisory control have been tested and verified with the OAC for an application to turning a CNC turret lathe.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Over the past two decades, computer technology has had a profound influence on all manufacturing industries. In particular, advances in automation have contributed in areas such as:

- material handling,
- quality inspection and monitoring,
- motion and logic control,
- resource planning,
- process control.

Machining operations in discrete parts manufacturing have experienced benefits from all except the last of the above listed technologies. This is in contrast to industries such as chemical processing in which process control is in many cases a fundamental requirement.

It has been argued that automation applied to production machining operations

has reached a point at which no further benefits can be achieved from those technologies commonly applied (ie: material handling, etc). Thus, from an economic point of view, process control represents the only automation related option left for achieving a competitive advantage in machining operations.

Considerable academic research has been established on monitoring, adaptive and optimization control of the machining process. However, few of these systems have involved the control of more than one [process] variable (Koren, 1983). Limited work has pursued the development of 'intelligent' machining systems, ie: systems that control or optimize multiple aspects of the machining process. While such systems do address the fact that industrial applications require the consideration of many process aspects, the definition of a general framework for Intelligent Machining Systems (IMS) has been an elusive goal.

Industrial implementation of machining process control has also been impeded by the practical limitations of the equipment used to realize these processes. In particular, commercial machine tool controllers are not designed to be integrated into the larger process control system. They are typically of a 'closed' nature in that no mechanisms are provided with which the controller's internal command and data resources can be accessed and integrated with external control system components. Open architecture control (OAC) for machine tools is an area of current academic and commercial research that aims to create machine controller systems that support integration with end user systems.

This thesis describes the design and development of an Open Architecture machine tool controller for Intelligent Machining Systems. Using established, industry standard open systems design concepts, the controller is developed to demonstrate capabilities required for integration with established machining process technologies such as real time monitoring and control, process and model based planning, and supervisory control.

The intelligent machining system is based on a turning application. The comprehensive control of multiple part production is formulated according to activities occurring while cutting (in pass), between individual cuts (per pass), and over multiple cuts or discrete parts. The functional components of the system are developed and demonstrated and moreover, are used to guide the design and implementation of the open architecture controller (OAC).

The thesis is organized into 6 chapters. Chapter 2 is a survey of related research in both Open Architecture Control and Intelligent Machining Systems. Chapter 3 describes the design basis and implementation of the open architecture machine tool controller system. In this chapter an open systems controller is defined in terms of its functional model, hardware topology, and its applications interface and communications. The Machine Server component of the open systems controller is described in detail regarding its service model, interaction with client applications, and specific support for client timing and data requirements for process and machine control.

The machining test bed developed for the Intelligent Machining application is

detailed in chapter 4. The application is for a CNC turret lathe. The implemented sensing sub systems described in this chapter address the key machining process variables:

- cutting forces,
- machine vibration,
- surface roughness,
- tool wear.

Chapter 5 describes the formulation and implementation of the Intelligent Machining System. Control activities are defined according to three time frames: in pass, per pass, and multi pass. Individual control sub systems at each frame are described, and include established technologies for process:

- monitoring and control,
- planning and simulation,
- supervisory control.

Experimental evaluations of these sub systems with the Open Architecture Controller are performed to evaluate the controller design, in particular with regards to time, performance, and integration criteria.

Chapter 6 concludes the thesis, with summarizing commentary on possible directions for further research.

Chapter 2

LITERATURE SURVEY

2.1 Open Architecture Machine Tool Control

Open architecture machine tool control has been identified as an 'enabling' technology for the application of advanced controls in manufacturing. The desire for such systems has been in response to the frustration experienced by practitioners and researchers alike when advanced control techniques were attempted on current 'closed' architecture controllers. The term 'open' originates from the PC and workstation industry, where 'open systems' technologies have greatly enhanced the use and development of complex computer systems.

Several commercial and academic programs that have focused on realizing some form of open architecture for machine tool controllers, and manufacturing automation systems in general. Fundamentally, each approach has been concerned with one or more of the following:

1) the integration of physical hardware devices,

2) communication interfaces between devices,

3) models of interaction between computing resources and motion control and I/O devices.

These items provide a useful point of reference with which the different OAC systems can be compared. The following survey outlines the major accomplishments in this area.

Manufacturing Automation Protocol (MAP)

The MAP program represented the first 'rebellion' against the closed architecture of proprietary manufacturing systems, being motivated by "...the need for compatibility of communications to integrate factory floor devices" (North American MAP/TOP Users Group, 1988). In order to standardize such an effort, an abstracted model of the typical manufacturing device was required. The result (MAP v3.0) was a uniform application interface to data exchange, file system access and remote program invocation between peer entities referred to as Virtual Manufacturing Devices (VMD's). The VMD abstraction was derived explicitly from the requirements of numerical control, robot control, PLC's, and process control systems. MAP was developed compliant with the Open Systems Integration standard (ISO-OSI).

Although MAP is a specification for communications (ie: not machine control), it does relate strongly to the OAC effort. Firstly, MAP represents a large scale

standardization program directed at manufacturing devices. Secondly, the success of MAP as envisioned required compliance on the part of the end point devices to the openly specified interface, including the concept of the VMD. In this context, the MAP effort can be viewed as an early relative of the current development of OAC.

Machine tool Open System Architecture for Intelligent Control (MOSAIC)

Perhaps the most established research related to open architecture machine tool control centers around the MOSIAC system developed at New York University (Greenfeld and Wright, 1989). The Machine Open Systems Architecture for Intelligent Control was originally developed to facilitate research related to expert planning systems and quality related sensor data. Conventional controllers at that time were unable to support the information flow required for these tasks. The MOSAIC architecture was based on a memory mapped backplane architecture (VMEbus) with specialized hardware for axis control, machine input / output, and a general purpose processor running a real time POSIX compliant operating system. Access to the controller functions was provided by a library of C language routines. These routines formed part of a larger 'machine' operating system that, being POSIX compliant, was able to freely exchange information with workstation based software for expert system based process planning and part probing.

Later work related to the MOSAIC system focused on the development of these higher level functions. Several published works have included a wide range of

applications including adaptive control strategies (Wright et al, 1991), and an Internet based machining center (Hansen et al, 1993).

The MOSAIC system has also been considered in an "agent based" formulation of machining system architecture (Dornfeld and Wright, 1995). In this paper the authors described an Integrated Manufacturing and Design Environment (IMADE) in which a group of planning agents for tooling, fixturing, and process planning communicated with the MOSAIC system (a fabrication agent) using a simple scripting language. The main idea was that the planning agents would provide the "global" knowledge required for part realization, while the MOSAIC system would provide the "local" knowledge through its use of sensors for process evaluation. Over the production of multiple parts, it was proposed that the global scripts could be modified (or "validated") by the locally generated information. The concept of the IMADE approach was demonstrated for the problem of burr formation in face milling.

In a recent summary paper (Wright, 1995) on the history of the MOSAIC system the author stated the principles of open architecture manufacturing as:

1) the use of open architecture hardware platforms at all levels of the machine and factory,

2) the use of a common operating system at all levels of the machine and factory,

3) the use of a standardized, object oriented format for information and knowledge exchange.

Next Generation Machine / Workstation Controller (NGC)

The Next Generation Machine/Workstation Controller (NGC) program was initiated by the U.S. government with the goal of restructuring and revitalizing of the US machine tool industry through the enabling of technologies in the areas of computer hardware and software, and process controls. Extensive consultation with industry experts resulted in a specification of 175 requirements for the NGC. These requirements addressed a wide range of issues including hardware and software architectures, networking capabilities, and provisions for task planning, process control, fault management, and sensor interfaces.

The proposed controller system consisted of separate 'workstation' and 'real time' computing facilities (NCMS, 1990). Figure 1 depicts the envisioned system.

As a modular architecture, the NGC was intended to provide well defined interfaces and guidelines to direct scaleability and third party product development. While hardware interfaces were to be chosen from existing standards, software interfaces were to be built around a syntax referred to as the Neutral Manufacturing Language (NML). The purpose of NML, being similar to the MAP VMD concept, was to provide a

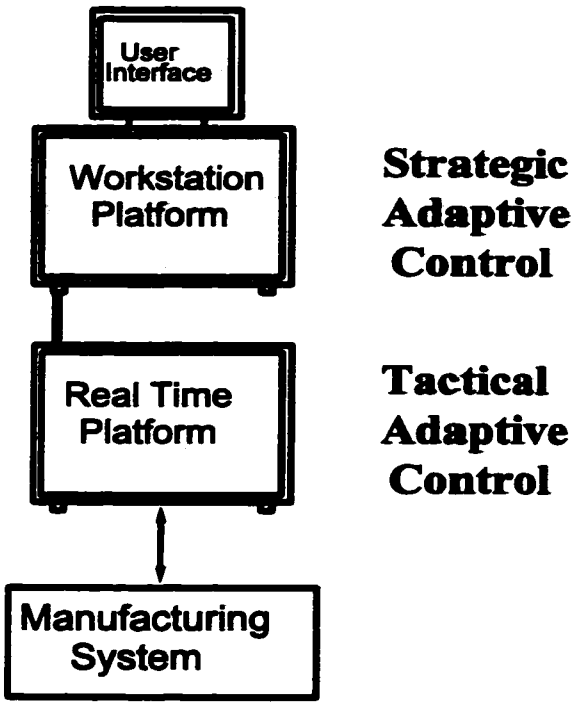


Figure 1 Next Generation Controller (NGC) architecture.

standardized software interface to the services of the virtual machine through which end users could expand and enhance the software base of their systems

Regarding process control, the NGC requirements specified provision for 'tactical' adaptive control (real time), and 'strategic' adaptive control (non real time). The tactical control element was aimed at providing on line compensation for volumetric and surface finish errors, to monitor for tool breakage and chatter, and to allow for the

implementation of optimization strategies. Technologies such as neural networks, multi-variate statistical process control, adaptive control, and sensor fusion were identified. The strategic control element was aimed at providing "...the future strategic approach to the manufacturing process..." (NCMS, 1990) through longer term analysis of the processed results of sensor data. Technologies such as rule based and knowledge based decision making systems, and process modeling and planning systems were identified.

The NGC specification is relatively comprehensive in comparison to previous efforts such as MAP. Consequently, due to the broad range of 'enabling technologies' it addresses, it arguably lacks the focus of the MAP effort. Within its broad scope however, the NGC specification does contain conceivably all that OAC can and will be. In this respect, NGC can be considered to be a goal definition for OAC development efforts.

Later work reported by the NCMS involved a "low end controller" (LEC) implementation of the NGC paradigm. In this case, the term "low end" referred to that part of the proposed NGC system without the higher end functions such as process planning.

The LEC system proposed a further definition of the original NGC architecture into non real time, real time, and hard real time components. It is suspected that this further division was intended to recognize the fact that motion control activities, being the most time critical, are almost exclusively implemented in commercial systems using dedicated processing and input / output hardware.

Outside of this definition, the LEC system advocated no specific hardware

implementation of these three elements. It was proposed that the logical decomposition of controller functions could be formulated and realized using an agent based paradigm. In this form, processing modules (agents) were defined for such components as command interpretation, trajectory generation, user interface, and machine drives control. Each "agent" represented "... a collection of software organized for a specific purpose and either encapsulating or capable of obtaining all the data and functionality required to fufil its responsibility" (NCMS, 1994).

Key to the realization of the agent based approach was something referred to as the Common Execution Enviroment (CEE). The CEE was described as the "glue that held the whole project together". It was envisioned as an application programming interface specifying:

- message semantics and syntax,
- timing services for real time operation.

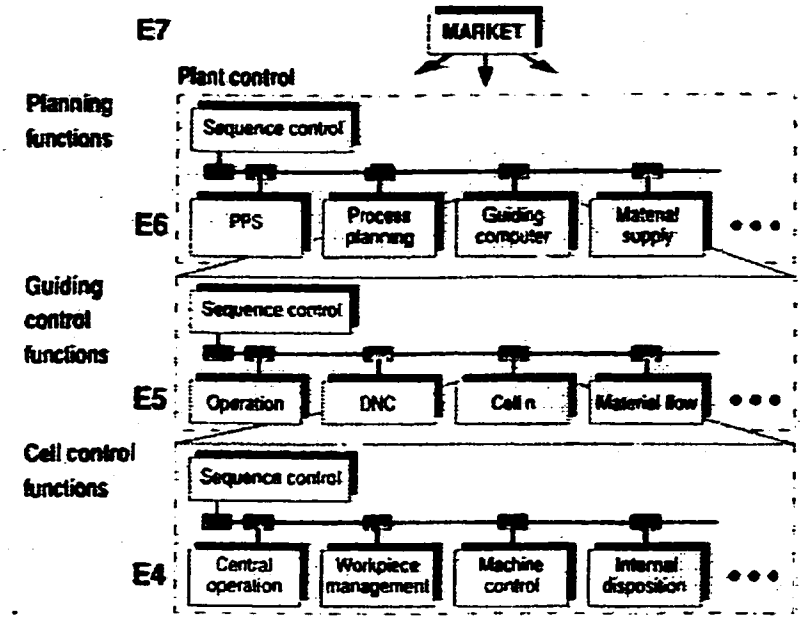
The researchers stated that "the technology for CEE is today a missing building block" (NCMS, 1994).

Open Systems Architecture for Controls within Automation Systems (OSACA)

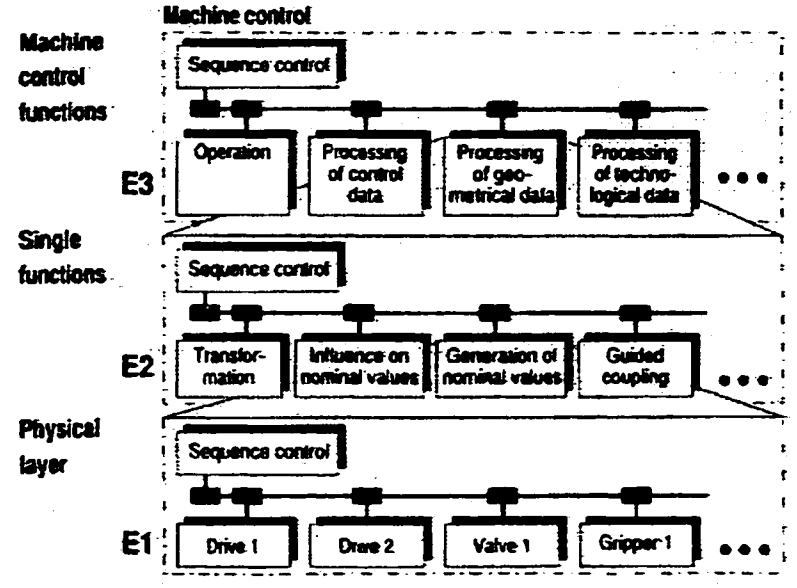
Prompted by similar concerns experienced by their North American counterparts, the European manufacturing community has embarked on research initiatives geared

towards the definition and development of OAC systems (Pritchow, 1990). This research has involved a detailed 'open system' definition of the entire manufacturing process, using a seven layer, hierarchical model adopted from the OSI-ISO communications standard. Figure 2 shows a reprint of this model of the manufacturing process. Layer E4, the Cell Function Layer, corresponds to the machine to machine communication layer occupied by communications standards such as MAP. Layers E3 to E1 represent the functionality concerned with machine control, and hence OAC. In light of this breakdown of OAC functionality, and the existing standards for communications protocol definitions, Pritchow presented a 'cube model' for the OAC system. Figure 3 is a reprint of this cube model. Viewed in this perspective, the realization of OAC involved the provision of standardized ordering of functionality, similar to the OSI communications standard, at the machine controls level.

More specifically, the objective would be to provide layered protocols and functional decomposition for the program processing, coordination, and drives level of the machine control. It is worth noting that recent European standards such as SERCOS (described later in this survey) - satisfies in principle this concept for the lowest layer of the machine control (E1).



Hierarchical control concept E4–E7.



Hierarchical control concept E1–E3.

Figure 2. Hierarchical model of the manufacturing process (Pritchow, 1990).

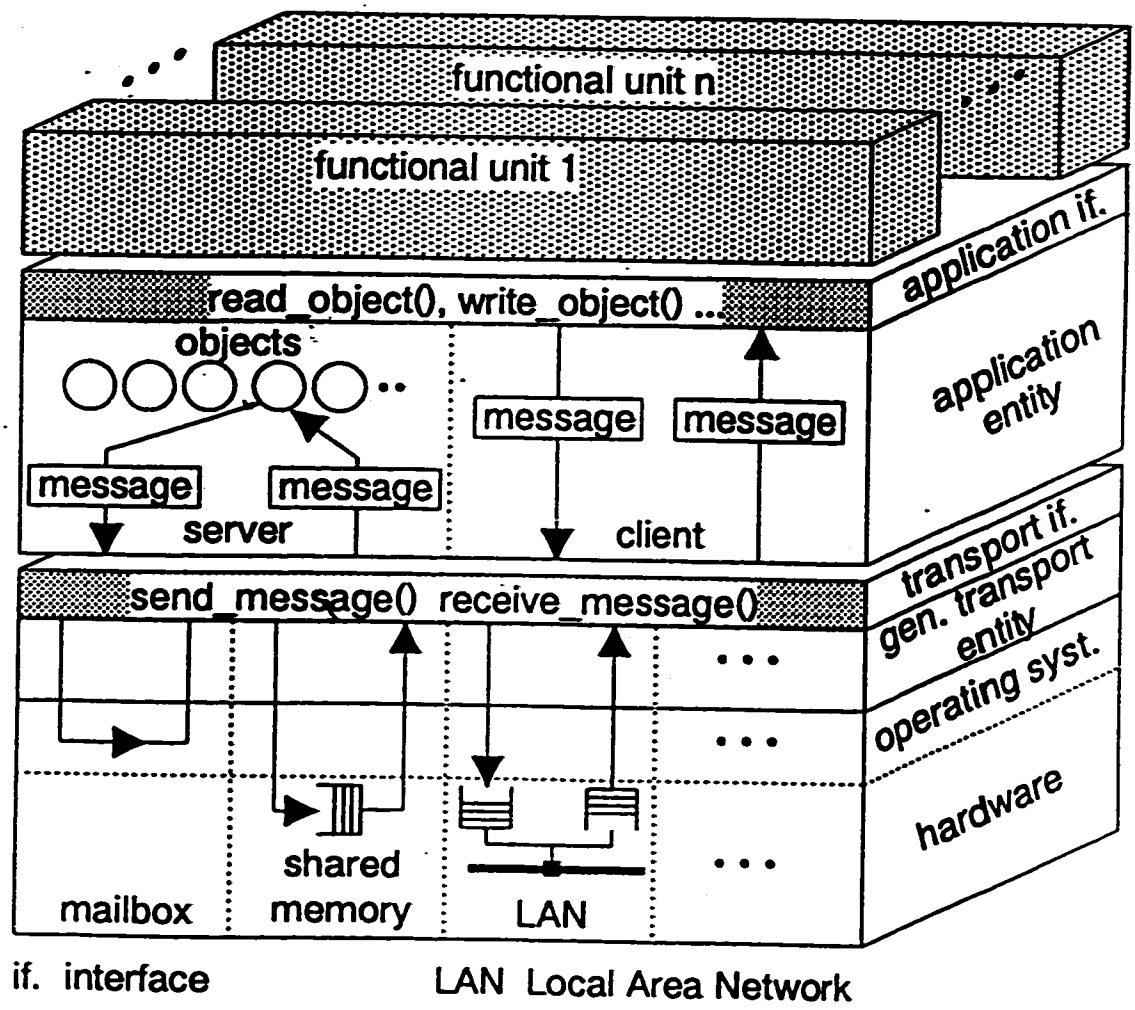


Figure 3. Reprint of OSACA Cube Model (Pritchow et al., 1993)

A later publication (Pritchow et al., 1993), presented an alternative, more functionally oriented view of OAC, addressing specifically the needs of numerical control (NC). An application program interface (API) provided the configuration abilities that allow the system designer to arbitrarily combine functional units (FU's) - such as part program processing or monitoring functions - with database and user interface functionalities. An additional, lower layer provided uniform communications services through which FU's could interact transparently with the operating system and hardware levels of the machine control. As a standard interface to the machine and between FU's of a given implementation, the API and communication levels of this model of OAC are similar in concept to the MAP VMD. In fact, Pritchow et al noted that "although the MMS [ie: the MAP programming interface which utilizes the VMD concept] fulfills this [concept], it is too complex and does not offer all of the features needed for communication within a control."

More recent descriptions of the OSACA project (Pritchow and Junghans, 1994) reiterated these design paradigms, in particular emphasizing the formulation of open communication / messaging services within the OAC. The proposed system was designed based on the Open Systems Interconnection (OSI) reference model. A message transport system (MTS) was proposed to implement the first four layers of the OSI model (ie: the Physical, Data Link, Network and Transport layers). The MTS acted as an adaption layer in that it was itself intended to be built upon a wide \ variety of existing communications protocols and media including:

- shared memory,
- POSIX Inter Processor Communication (IPC),
- Internet Protocol,
- Controller Area Network, Profibus,
- Integrated Services Digital Network.

It is in this manner that the OSACA systems addresses interconnection with devices and sub systems that use existing commercial and *de facto* standards.

The remaining four OSI layers were to be provided by an application services system. This level of communications supported client / server type transactions, flow control, and other tasks normally associated with these levels of the OSI model.

At this level in the system description, the OSACA architecture provided only for an OSI compliant communications system. Issues related to machine control were not explicitly addressed by these layers. However, on top of the application layer (top level of the OSI model), the OSACA system proposed an elaborate object oriented interface to model the interactions and provide the services of low level devices to end user "architecture objects". This interface was to be based on an object manager and a set of object oriented information models describing internal data and functionality related to:

- program execution,
- operator station management,

- synchronization and event management,
- access to internal data,
- file system management.

It was proposed that the association of data objects describing machine attributes could be used to create a software configured machine control. This would in turn facilitate the enhancement, replacement or reuse / reconfiguration of standard controller sub systems.

At the top level of the OSACA model were architecture objects that used these information models to implement actual high end functions such as:

- human / machine control,
- logic control,
- motion control,
- axis control,
- process control.

Architecture objects represented the system components that would most likely be implemented by third party vendors. Programming libraries providing the templates of the underlying object oriented information models would be used by the developers of architecture objects. Seen from this perspective, the goal of the entire OSACA architecture was to provide these architecture objects with the faithful ability to control

the low level functions of the machine with no knowledge required of the actual hardware used, the communications medium, or the overall hardware topology.

Sperling and Lutz, 1997, presented the most recent developments regarding the OSACA project. In their paper they reiterated this OSACA design approach and documented an example of integrating a human / machine interface (HMI) architecture object with OSACA infrastructure. In their example, Internet Protocol communications were used to link the hardware platforms for the HMI and motion control sub systems.

National Institute for Standards and Technology (NIST)

Researchers at the U.S. National Institute for Standards and Technology (NIST) have a long established reference model for intelligent machine systems (Albus, 1989). This model, originally developed for robotic systems, has been applied to machine tool and manufacturing controllers. This detailed formulation (Albus, 1994) identifies the elements of machine intelligence as:

- Task Decomposition,
- World Modeling,
- Sensory Processing,
- Value Judgement.

The program for machine tool controllers evaluated several architectures using various commercial motion control and discrete event control sub systems. Software running on a workstation platform - the host machine executive - provided various application programming interfaces that tied the different systems together.

University of Michigan Open Architecture Controller (UMOAC)

Birla et al (1995) described an open architecture test bed developed at the University of Michigan (UMOAC) using a variety of platforms:

- PC based
- VME bus (backplane)
- CANbus (distributed)

Custom software libraries allowed the development of control and monitoring systems within the environment of a real time operating system. An elaborate software hierarchy provided an object oriented application interface with abstractions of the machine functions.

The authors proposed a supervisory control / process control structure for the test bed in which adaptive control and monitoring functions developed on the platform were able to integrate directly with low level motion and input / output functions. The supervisory level was also given the responsibility to "integrate and coordinate the

control modules to complete the operation" (Birla et al, 1995).

Later work (Koren et al, 1996) related to the UMOAC test bed examined distributed control architectures and compared their requirements to that of more traditional control architectures. These systems connect a collection of self controlled drive units using a digital network. A central processor interpolates motion. The author's research focused on quantifying the real time requirements for the distributed and conventional architectures. They conclude that such systems are flexible and promote user enhancement, but must be designed carefully with respect to effective network bandwidth if real time requirements are to be satisfied.

Serial Realtime Communication System (SERCOS)

It is worth at this point mentioning developments aimed at providing open architecture interfaces at the device level. Machine control systems based on such distributed architectures are becoming increasingly used and have been declared official "requirements" for installations in major automotive manufacturers like General Motors.

There are several vendor specific protocols that support discrete input / output and motion control systems. The former application has been adopted by industry for some time as "field busses" have the obvious advantages of greatly reducing the amount of wiring for Programmable Logic applications such as transfer lines.

Perhaps more challenging is the use of distributed open architectures for motion control systems. Generally, it has been at the level of motion control that most

commercial systems have proven to be "closed". This is perhaps understandable, due to the time - and part precision - critical nature of motion control activities. However, the increasing use of embedded micro controllers at the machine drives level and the improvements in the bandwidth of digital data links have allowed the technology and practical benefits of distributed open architecture systems to be realized. These benefits include:

- the construction of the CNC system from a heterogenous mix of drive sub systems (eg: AC, DC, stepper, etc),
- minimized wiring between sub systems,
- improved noise immunity for low level command signals between interpolation and loop closure processors,
- the ability to scale system configuration.

The Serial Realtime Communication System (SERCOS) standardization effort (German Machine Tool Manuf., 1988) in Europe was one of the first "field bus" approaches to CNC system construction. The standard outlined the protocols and physical specifications of a communication network for interconnection of servo drive sub systems and other processor components.

The physical layer of the SERCOS system is a 2 Mbits/sec fibre optic link. A ring topology transfers data packets with a 32 bit data field and an 8 bit addressing field, for a

total allowable number of 255 "slave" drive units. Each slave may service one or more actual drive units. A master computer on the ring is intended to perform interpolation and other CNC duties such as user program processing. The original standard, at 2 Mbits/sec, allows for 8 slaves to updated at a rate of 500 Hz. The SERCOS standard was created by a consortium of supporting vendors including Bosch, Siemens, and Indramat - all major European automation controls vendors.

Hierarchical Open Architecture Multiprocessor Computer Numerical Controller (HOAM-CNC) Controller

Altintas et al (1993) presented a hierarchical, open architecture multi processor computer numerical controller (HOAM-CNC) based on multiple processors on a PC bus. Individual axis control processors performed loop closure while a master processor executed functions for axis interpolation, adaptive control, and \ process monitoring. A job manager on the master processor coordinated task scheduling in a time sliced manner.

More recent publications (Erol and Altintas, 1997), described a new formulation of the HOAM-CNC system in which the job manager functions were extended to an Open Real Time Operating System (ORTS). The ORTS executes on both the main master processor (PC motherboard), and on the single board processors of the PC's ISA bus. The described system includes the host processor and two single board computers for motion control and process monitoring and control. The latter item is referred to as an Intelligent Machining Module (IMM), and is designed to interface to commercial

"semi-open" controller systems or with the HOAM-CNC system. A script like language allows the master processor (PC) to control the execution of activities on the IMM, such as adaptive force control and tool breakage monitoring. The IMM used a special form of the ORTS that allowed a collection of processor modules tc be serially connected to form a signal processing network. In this case, special ORTS functions were required to support the interactions between modules.

University of British Columbia Open Architecture Controller

Yellowley and Pottier (1994) presented an OAC system based on a modular backplane architecture. A combination of processing modules was used to implement axis control, trajectory generation, and process monitoring and control. An interpreted, FORTH based programming interface allowed high level macros to be called from each motion program line. In this manner it was proposed that monitoring and control functions to be called from within the motion program, thus associating the process and geometry with the machine control.

A novel hardware signaling method allowed the custom programmed servo cards to have feed rate override from multiple sources. The technique involved a 'state line' which when asserted caused the servo processors skip a control cycle. When asserted periodically and with a controlled rate, it allowed other system processors to effectively reduce feed rate. The state line logically OR's the assertions and as a result the feed rate at any given time becomes the lowest of those being commanded by any collection of

other processors. The idea was that the actual feed rate is always that of the worst case constraint source. The system was demonstrated for milling aluminum with constant cutting force control.

GM / FORD / CHRYSLER Survey of the Requirements of Open Modular Architecture Controllers

Balio et al (1994) summarized the requirements of Open, Modular Architecture Controllers (OMAC) for the automotive industry. Identified needs were addressed such as safety, cost, flexibility, etc. A loose decription of an example architecture was given in which the overall functionality was decomposed into modules for:

- human interface,
- motion control,
- sensing interface,
- discrete event control,
- information base,
- network connection.

The needs of each of these modules were discussed in detail. Issues such as the use of standards, access to low level motion control functions and the the ability to scale systems was addressed. More significantly perhaps was a discussion of the possible

difficulties in having open architecture systems accepted in automotive industry. It was suggested that a change in business practices may be required regarding retraining, inventories, and maintenance. Moreover, it was clearly stated that "using OMACs also requires users to be more involved in system integration and assume responsibilities for keeping systems operational. Users cannot rely on technology providers to take care of all of their problems anymore." (Balio et al, 1994).

2.2 Intelligent Machining Systems

The principal elements of Intelligent Machining Systems (IMS) are:

- sensing,
- monitoring,
- control.

Common throughout the literature, discussions of IMS often further define the last element - control - as consisting of:

1) "traditional" feedback control, based on classical control theory,

2) "supervisory" type control, often based on Artificial Intelligence (AI) technologies.

In this regard, the second attribute of control in IMS relates to the larger, emerging discipline of Intelligent Control, whose own definition describes it as being "task oriented", and applied specifically to physical systems whose "... dependancies are generally too complex to admit an analytical representation" (Zadeh, 1996).

In the case of machining systems such complexities can include controlling the multiple constraints and process variables involved, and managing operation in the event of contingencies such as tool over load or breakage. Advanced interpretations include accommodations for other elements of the overall production process such as planning and design. The following survey examines the development of research in Intelligent Machining.

Machining Process Sensing and Monitoring

The subject of machining process sensing and modeling is an extensive one, and a comprehensive overview of the techniques and principles involved is beyond the scope of this survey. This section is included here for completeness in the discussion of Intelligent Machining Systems.

Excellent reviews have been given by Micheletti et al. (1976), Tlusty and Andrews (1983), Tonshoff et al. (1988), Dan and Matthews (1990) and Dornfeld (1992). Suffice it to summarize here that the principal sensing and monitoring issues in machining involve:

- process constraints: cutting forces and temperatures,
- tool condition: wear and failure,
- part: dimension and surface quality.

There are obvious difficulties measuring some of these quantities, and accordingly indirect methods are used in which alternate variables are sensed and signal processing and / or process models are used to infer the value of the desired information. It is in this manner that sensing and monitoring issues are closely related.

Machining Process Control

Traditional approaches to optimization of machining processes have concentrated on systems referred to as "Adaptive control optimization" (ACO) and "Adaptive Control of Constraint" (ACC).

ACO systems formulate the manipulation one or more process variables in order to maximize a predefined performance criterion. This criterion is usually economically motivated, and is manifested in terms of indices related to cost, maximum productivity, etc. Wu and Ermer (1966) describes a typical approach to the ACO philosophy, being based on the following techniques:

1) formulation of equations for machining unit cost,

2) the use of empirical or semi empirical relationships to describe tool life and cutting power in 1),

3) constraint descriptions of the operational variables (feed, speed, depth of cut),

4) solution for the operational variables at points of minima in 1) subject to 3).

While this approach is both rigorous and comprehensive it has not proven to be effective in practice due to the inadequacy of the models for tool life, the inability to measure the required variables to give an accurate indication of performance.

Modified approaches to ACO have attempted to address these problems directly. For example, Onetsu et al. (1978) used on line estimation of tool wear and tool life equation parameters in the cost formulation (Onetsu et al., 1978). Carlsson and Strand (1992) used stochastic models of variability in tool wear predictions to address these limitations. In both cases however, limited success was achieved and / or no experimental verification was presented.

Yen and Wright (1983) suggested an alternate interpretation of the ACO approach. As opposed to concentrating on performance indices, they suggested that optimum process cost could be achieved by choosing an operating point having maximum metal removal rate from within the process parameter space bounded by the multiple constraints of the dominant tool failure mechanisms. The mechanisms considered included:

- edge breakage,
- plastic deformation,
- adhesion / oxidation.

The key to the solution of this operating point involved the mapping of the constraints to the parameter space of the operational variables. The authors demonstrated the concept using classical mechanistic and semi empirical modeling of cutting process temperatures and stresses, and tool failure mechanisms. In this regard however, the approach was still subject to the inadequacies of these models and as such did not avoid the problems that indices based ACO systems had encountered.

In ACC systems a performance index is not directly evaluated. Instead, one or more of the process conditions are maximized within known constraints. These constraints are typically related to the physical limitations of the system. Applications to machining most often involve the manipulation of tool feed to achieve cutting force regulation at the constraint level of tool breakage. The majority of adaptive control applications in machining systems are of the ACC type.

The first ACC systems in machining were simply feedback control systems (Stute and Kapajiotidis, 1965). However, as static cutting force is dependant on operational variables - in particular feed and depth of cut - the fixed parameter controller approach would only provide stable and effective behavior for a limited range of these variables.

Later contributions applied ACC as true "adaptive" control systems; as a

controller whose parameters adapt to changing process behavior. Koren and Masory (1981) presented an ACC system for turning on a CNC lathe with constant force constraint. Their observations from earlier work with fixed gain controllers was that as the feed forward gain of the control loop varied due to changes in the depth of cut or spindle speed, the controller performance became either too sluggish (small gain) or unstable (large gain). They concluded and demonstrated successfully that universal stability could only be achieved if the open loop gain was kept constant by continuously estimating the process gain and recalculating of the controller parameters accordingly.

A comprehensive study of several ACC strategies in milling was presented by Elbestawi et al (1991). The parameter adaptive technique was applied to milling and evaluated for several controller systems. Robust operation of the AC system was found to be enhanced by disabling the parameter estimation algorithms once the estimates converged. This was done to avoid unpredictable controller behavior due to parameter 'drifting' during periods of low signal excitation. It was found that the system reaction time to sudden increases in cutting force could be significantly reduced by resetting the covariance matrix recursive least squares estimation algorithm to its starting valve of identity. The evaluation concluded that a dead beat controller of increased order, combined with the estimation disable / reset functions, was found to result in the best overall response.

Lundholm et al (1992) described an optimization system that employed three 'levels' of control: monitoring, ACC, and ACO. The proposed system consisted of

monitoring and ACC levels to act on line for the monitoring of tool breakage, collision, and chatter, and to regulate cutting force and suppress chatter. The ACO level acted between passes to select cutting conditions based on an economic optimization of the process with consideration of cutting force and tool wear. Intermittent tool wear measurements were to be provided by a vision system.

Several researchers have emphasized the role that computer aided process planning (CAPP) can play in cutting process optimization. Van Houten and Tiemersma (1989) proposed an integrated system involving off line CAPP, and on line monitoring modules. The CAPP system was intended to select optimal cutting conditions and acceptable levels of cutting force expected during cutting. On line monitoring used the error between measured and predicted cutting forces to detect breakage or collision conditions. This system was referred to as 'Model Based Monitoring'. The system proposed to use the data acquired in pass to re- calibrate the models used in the CAPP. An additional 'learning control' loop was proposed to monitor even longer term variations in behavior. The system was not demonstrated experimentally.

Yamazaki et al (1991) developed an integrated CAM / monitoring system for three dimensional sculptured surface machining. The proposed system attempted to perform tool path and feed planning in real-time. A CAM system was developed using surface modeling techniques and implemented as a real-time tool-path generator. A dynamic machining simulator was used to simulate the geometrical interaction between the workpiece and the tool and to evaluate the performance of the machining process in

terms of material removal. A final module, designed as a type of off line model reference adaptive controller, performed simulation of the cutting process for the generated tool path from 1) and at incremental steps along the path determined material removal for different feed values. The applied feed was selected to represent a safe and productive level of cutting load.

Takata (1993) proposed a model based CAM / monitoring system. The proposed system, although similar in concept to Yamazaki's work, had the following additional characteristics:

- explicit models for cutting forces, torques, machining error,
- the use of sensory feedback data to verify the planned operation,
- the use of solid modeling for tool / work piece intersection geometries.

Solid models of the tool and work piece were used to generate simulated cutting force to infer machining torque and surface errors. This "machining scenario" is a record of a simulation run for a particular operation and consists of:

- process plan data (tool positions, NC commands, tool types, etc),
- machining conditions (feeds and speeds),
- geometric data (tool-workpiece contact faces)
- physical data (expected cutting force, torque, and machining error)

Takata proposed two uses of the machining scenario: control of the machining process, and monitoring and diagnosis of a machining process. The control application of the machining scenario is divided into pre-process scheduling of feed rates etc., and on-line adaptive control. The monitoring and diagnosis application uses the generated process parameters as reference data to compare with sensory feedback, diagnose the cutting performance, and initiate the required corrective actions..

Intelligent Machining Systems

At this point it is clear that the concept of a comprehensive control strategy for machining processes requires a complex collection of activities, information, and actions. During the 1980's, artificial intelligence research had reached a level of maturity that made it an almost obvious - and fascinating - approach to the problem of machining process control and optimization.

Chryssolouris et al. (1988) presented a "decision making" approach to the optimization of metal cutting processes. In a similar manner to ACO systems the proposed controller also used semi empirical tool life and cutting force models. However, the basic operation of the controller involved, at each control interval:

- a rule based evaluation of both process models and real time sensory data to generate a set of performance indices,

- the selection of control actions by mapping the calculated indices into a "decision matrix",
- the application of control actions.

Figure 4 shows reprint from the paper depicting the decision making approach.

The approach demonstrated several significant benefits related to the use of artificial intelligence techniques such as the ability to handle virtually any number of constraints, process models, and sensory inputs, and the ability to include heuristic type reasoning into the overall control solution. While the work was a novel alternative to previous ACO approaches, it also introduced the idea of using "machine intelligence" as a control strategy for machining processes. Simulation studies of the decision system indicated that the technique performed better than the traditional ACO methods.

O'hare (1990) suggested that ideas from the field of distributed AI had direct applications in manufacturing systems. In particular, the use of an agent based paradigm for the decomposition of computer integrated manufacturing systems. Under this approach, individual agents maintained certain "responsibilities" and interacted with other agents to achieve an overall goal. For the machining application the author suggested agents for:

- planning and scheduling,
- computer aided design,
- material handling systems,
- machining systems.

In the paper the individual agents were each modeled as a rule based system however, it was noted that this was not a necessity for the concept.

Teltz and Elbestawi (1993) presented a hierarchical, knowledge based controller for turning. The system consisted of three distinct sub systems, implemented as independent tasks in a multi threaded application:

- a parameter adaptive force controller,
- a chatter detection and control system,
- a knowledge based supervisory system.

The latter of these elements coordinated overall operation, monitored and controlled the performance of the parameter adaptive algorithms, and sequenced the systems actions to suppress chatter in an event driven strategy to sub divide the cutting depth. The knowledge based and numerical algorithms were implemented in object oriented C in a multi threaded operating system. The knowledge based supervisory system coordinated the monitoring and control sub systems through calls to the operating system task

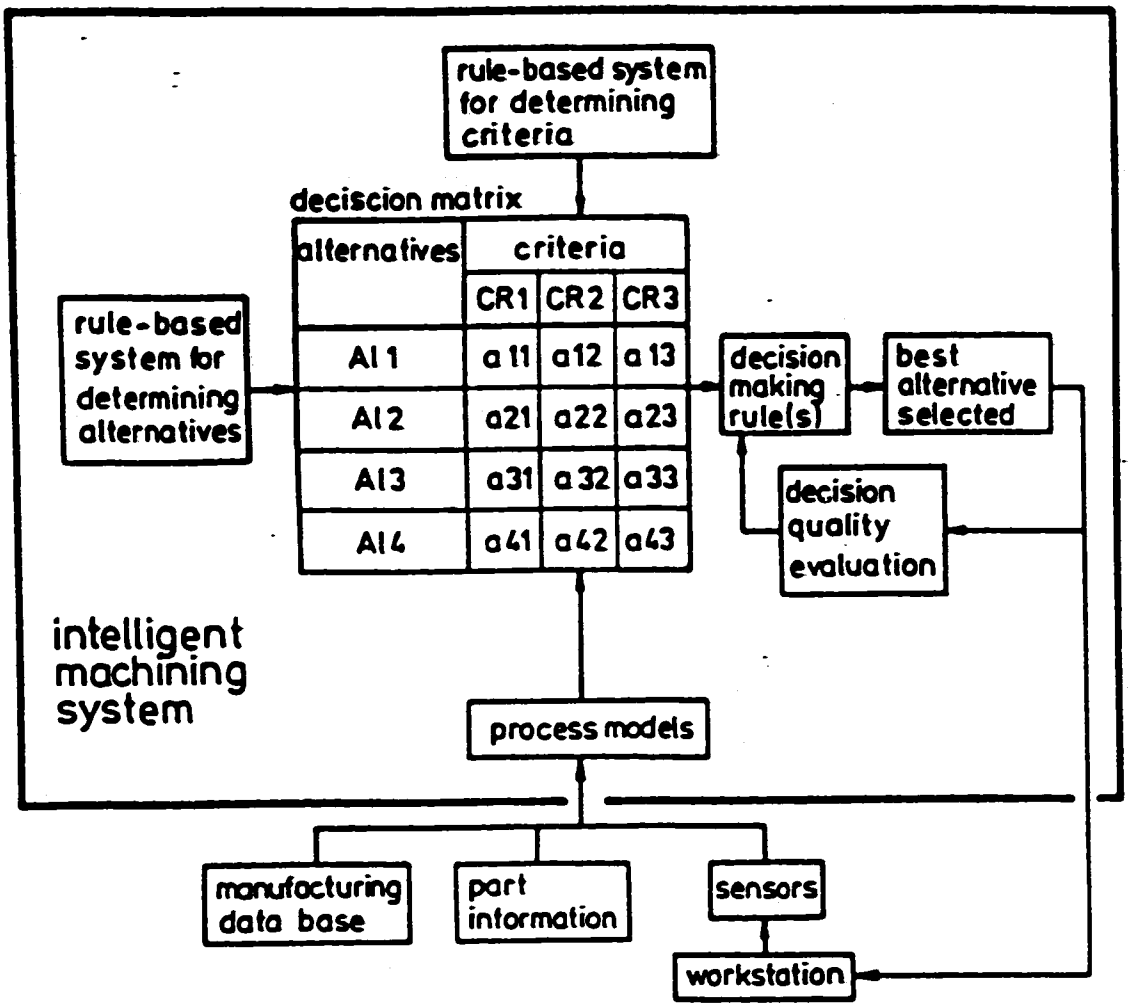


Figure 4. Decision Making Approach to Machining Process Control (Chryssolouris, 1988).

scheduler.

Hatamura et al (1995) detailed a design process for Intelligent Manufacturing Systems. In their application, the objective was to minimize geometric error in milling. The authors proposed a fundamental structure for IMS in which modeling and control elements were implemented as knowledge based systems. The application was interesting in that they used thermal and load actuators to apply control actions to the body of the machine tool to counter distortion due to cutting loads. The authors summarized six design principals for IMS:

1) all phenomena in a manufacturing system emit information,
2) the relationship between the phenomena and the information is predetermined and gives rise to models,
3) using 1) and 2) control can be achieved,
4) Intelligent Machining control represents an *active* approach to process improvement (as opposed to passive control from the machine design),
5) the main elements of IMS are: sensors, knowledge, and actuators,
6) the ability of the system to "conceptualize" (ie: reason over time, consider heuristics) is a key element of the IMS.

While the majority of reported applications of AI used in machining control are rule based systems (also referred to as "knowledge based", "production" or "expert

systems"), there has been application of other AI techniques such as fuzzy, neural network, and combined neural network and fuzzy logic. For example, Tarng et al. (1993) created an adaptive fuzzy controller which basically replaced the control block in a classical PID control loop. A gain on the control action from the fuzzy controller was adaptively modified according to measured process behavior.

Chiang et al. (1995) presented a neural network control strategy for end milling. The approach used two back propagation neural networks:

- a process model (observer),
- a controller.

The process neural network model was trained using simulated data from semi empirical models of cutting forces, power and surface roughness. The controller neural network was trained from data generated by a traditional optimization strategy (ACO) for maximum metal removal. The ACO system was based on the same models for process neural network training. The on line system was found to execute very fast, allowing the addition of other constraints and process variables without a significant effect on computational load.

Chapter 3

DESIGN BASIS AND IMPLEMENTATION OF AN OPEN ARCHITECTURE MACHINE TOOL CONTROLLER

There are generally four aspects of machine control systems that are considered in discussions of OAC:

1) physical hardware, electrical interfaces,

2) motion control and I/O sequence control,

3) operator interface and system management,

4) factory interface (or 'enterprise' level).

It is generally agreed that existing standards satisfy the requirements for 1) (eg: Controller Area Network, SERCOS, Interbus-S). These approaches however, only support user interests for new or newly rebuilt machines. Most systems in industry today are not based on the technology required to supported distributed 'open' hardware

solutions (eg: 'smart' servo drives). It must be assumed then, that from the perspective of the majority of machine tool users, items 2), 3), and 4) above represent the areas most in need of open interfaces. The following discusses the technical alternatives for these areas.

3.1 Functional Model

The functional model describes a decomposition of the overall activities of the machine controller into elements, and the interactions between these elements. Figure 5 depicts a functional break down of a typical CNC machine tool controller.

There are two predominant functional models that have been advocated in published OAC designs:

1) master / slave,

2) peer to peer.

Master / slave type systems are characterized by the dependance of system components on some central administrative entity. Examples of such approaches

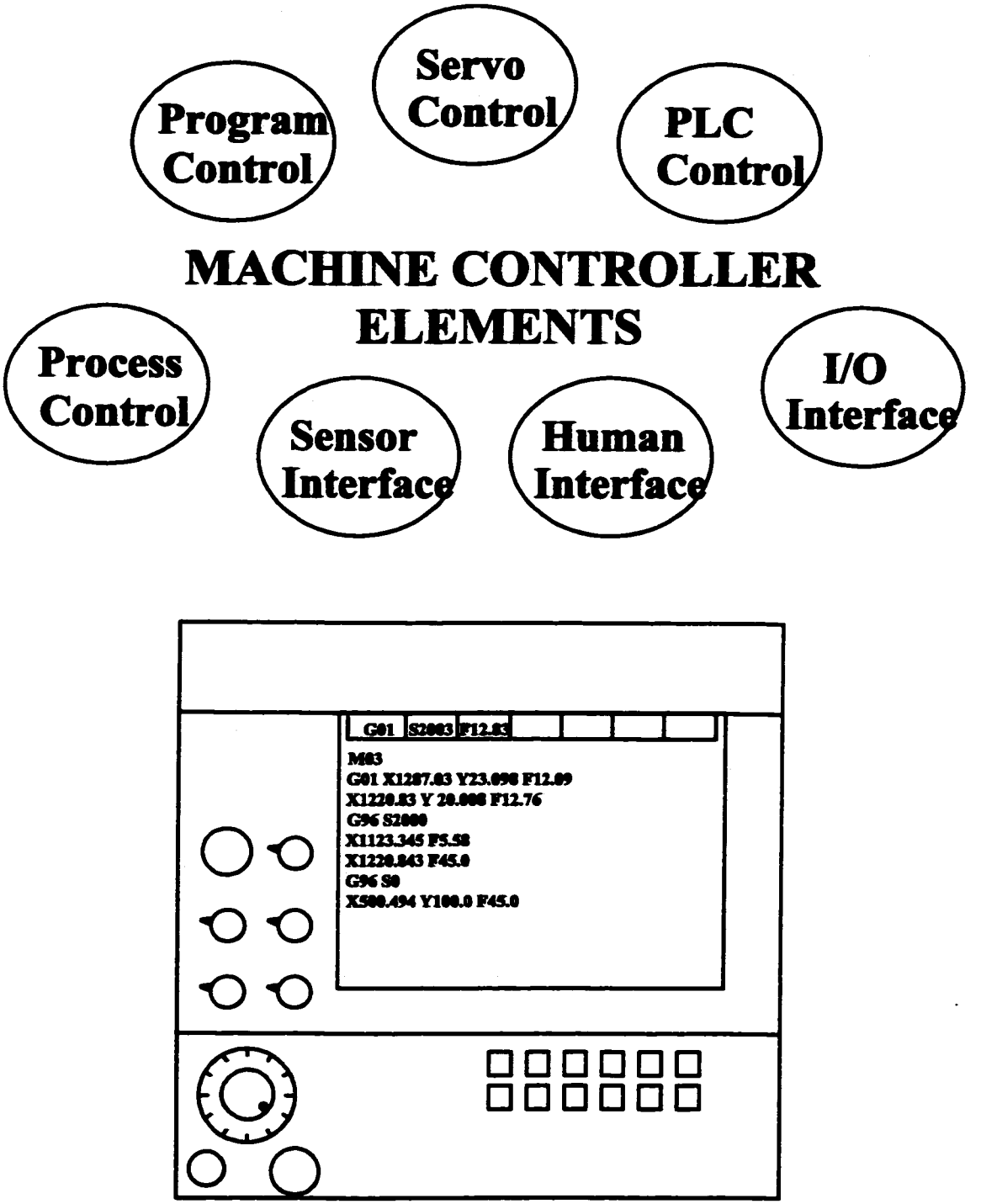


Figure 5. A decomposition of machine controller functions.

include (Altintas, 1994), and (Park et al, 1996). In theory, such systems are prone to performance problems if high demands are placed on the master. This may be manifested as a limit on system scaleability (ie: the master as a performance 'bottleneck'), or on a reliance of the entire system integrity on the master. Note that for the machine control application, this last point may be argued as being desirable.

Peer to peer type systems are very much the opposite of master / slave type systems. In this case, system modules are considered as independent 'agents', and are driven by individual 'responsibilities'. Peer modules can act as both service providers or consumers. Examples of such approaches include (Haynes, 1994), and (Pritchow and Junghans, 1994). In fact, the peer to peer concept is the predominant model used in established open computing systems.

While the peer to peer approach promotes system scaleability by removing dependance on a master, it places a greater emphasis on mechanisms for communication and interaction among modules. Alternatively, master / slave type systems can simplify the communication issue by insulating system modules from each other's hardware specifics. This does however, only concentrate hardware dependancies within the overall system.

3.2 Hardware Topology

There are three basic topologies under which OAC system modules are integrated:

1) within a processor (or an operating system),

2) between processors on a backplane,

3) between processors on a network (or similar).

Most discussions regarding hardware topology in OAC systems are based on time requirements for different machine control modules. This was the basis of the real time / non real time topology proposed by the Next Generation Controller project (NCMS, 1990).

However, due largely to advances in computer technology, it can be argued that definitions of OAC hardware topology should be discouraged. For example, the widely advocated NGC 'Workstation / Controller' topology has since been replaced by single PC systems.

While it is clear that different topology options should be made available to the user of OAC systems, in order to guarantee the scalability and inter-operability of existing systems, the mechanisms through which OAC modules are integrated must be topology independent. In order to address the timing requirements of machine control,

methods should be available through which system modules can evaluate timing capabilities.

3.3 Application Interface and Communications

The application interface is a particularly important aspect of any system that claims to be 'open'. It can ultimately decide how well third party or user developers can integrate their applications with the base system. A poorly designed or overly complex application interface can arguably decide the viability of any standardization effort.

Application interfaces that have been reported in the literature for OAC systems tend to be based on:

- application programming libraries,
- message passing techniques.

The former of the approaches includes, for example, traditional C language function libraries (Wright et al, 1991), and object oriented programming (OOP) libraries (Park et al, 1996). Message passing schemes have been reported by the NGC (Haynes, 1994 and NCMS, 1990) and by the OSACA project (Pritchow and Junghans, 1994). In the case of the OSACA project, the message passing approach is used as the underlying mechanism for an OOP interface.

Application libraries in most cases have a performance advantage over message passing techniques. This is due to their closer association with underlying hardware, and the additional overhead required by most message passing schemes. However, due to similar reasons, application libraries are less portable, and cannot readily adapt to changes in hardware topology.

Messaging techniques are usually able to avoid these limitations as they are based on mechanisms that are independent of the application. These mechanisms simply support the communication of messages between modules and, if they are chosen from existing standards, can be expected to be available for a variety of hardware platforms. Apart from the communication software, no specialized software is required to realize the message based interface.

There are several communications techniques reported for use between OAC system modules, including:

- shared memory,
- bus communications (eg: device ports),
- standard communications protocols.

The method used has typically been associated with the topology of the communicating modules; the choice in any given situation largely due to the time requirements of the particular modules involved.

In the case of shared memory and bus based communications, modules wishing to interface with other modules must use hardware addresses. These addresses are typically 'hard coded' into module source code, and if modified due to changes in hardware configuration, require at best minor source code modifications and re compilation.

Alternatively, most standard communications protocols use 'soft' addressing schemes. In this case, the underlying system resolves communication addresses at run time. Changes to hardware configuration require only modifications to addressing data bases. Application code remains unchanged.

3.4 Examples of Open Systems Concepts

The UNIX operating system generalizes all device I/O as file I/O. (SUN Microsystems, 1988). Operating system services map the low level drivers for a device to a generic 'file descriptor' handle that is used to reference the device via a programming interface. This allows applications to be developed without dependancies on:

- the hardware configuration (eg: local or network),
- the actual type of the device (eg: disks, tape drives and the console are treated the same).

The Network File System (NFS) (SUN Microsystems, 1988) allows the collective file resources of a network to appear as a single directory structure. This is achieved by 'client' and 'server' processes running on the application computer and the computer having the disk resource, respectively. These processes intercept and manage disk accesses in a manner that is transparent to the application.

The X Windows graphics system was originally designed to allow mainframe computers to support multiple graphics terminals of different types. The system is based on the 'X server': a program running on the displaying computer (terminal) that controls it's particular graphics hardware. A client application (mainframe) wanting to draw to a terminal does so by sending service requests to the terminal's X server. The service requests are defined in a generic sense; the receiving X server translates the generic requests into the specific actions for the graphics device. The X server and the client application may reside on the same computer or on different computers across a network.

User code developed for the X window system can be used with no modification among dissimilar graphics devices, thus enhancing the user base and the long term viability of developed software. Moreover, as the application is independent of the hardware topology the user can modify the topology without affecting developed software.

The X Windows system is an excellent model of open systems technology that can be applied to open architecture machine control. It demonstrates the concepts of a hardware independent software interface for both the device and the system topology.

3.5 OAC Design

3.5.1 Functional Model

The IMMRC OAC is based on a distributed, peer to peer functional model of the entire machine controller system. Currently developed modules address:

- machine motion and I/O,
- operator interface,
- tool inspection,
- tool breakage monitoring,
- adaptive control.
- CAD based feed planning,
- supervisory control.

The last five of these modules are part of the Intelligent Machining System, and are the subject of Chapter 5. The operator interface is described in Chapter 4.

Machine motion and I/O functionality is provided by a module referred to as the 'Machine Server'. This module provides a generalized set of services that support 'client' applications. Command, data and timing mechanisms are provided to facilitate control of the machine's resources. The machine server is described in section 3.6.

Practically, the OAC design presented here defines:

1) the mechanism through which system modules communicate,

2) the services provided by the Machine Server module.

The design does not define (or restrict) the development of other system modules.

3.5.2 Hardware Topology

The hardware topology chosen for the current implementation of the IMMRC OAC is shown in figure 6. Note that the topology presented here is only one example of many possible implementations. The design explicitly does not define or put limitations on hardware topology.

3.5.3 Application Interface and Communications

Applications access and control the machine device through a message based interface that defines the set of services that model motion and discrete I/O functions. These services emulate industry standard approaches to machine control; for example, the program interface is based on G codes and a DNC type 'drip feed' mechanism. Particular effort was made to present these services in as simple a manner as possible.

Modules in the OAC present commands to the machine server in the form of text messages. Machine server responses, if applicable, are also text based. While this may not be the most efficient method of transferring data, text has the advantages of being processor independent and an international standard. Note that only the Machine Server

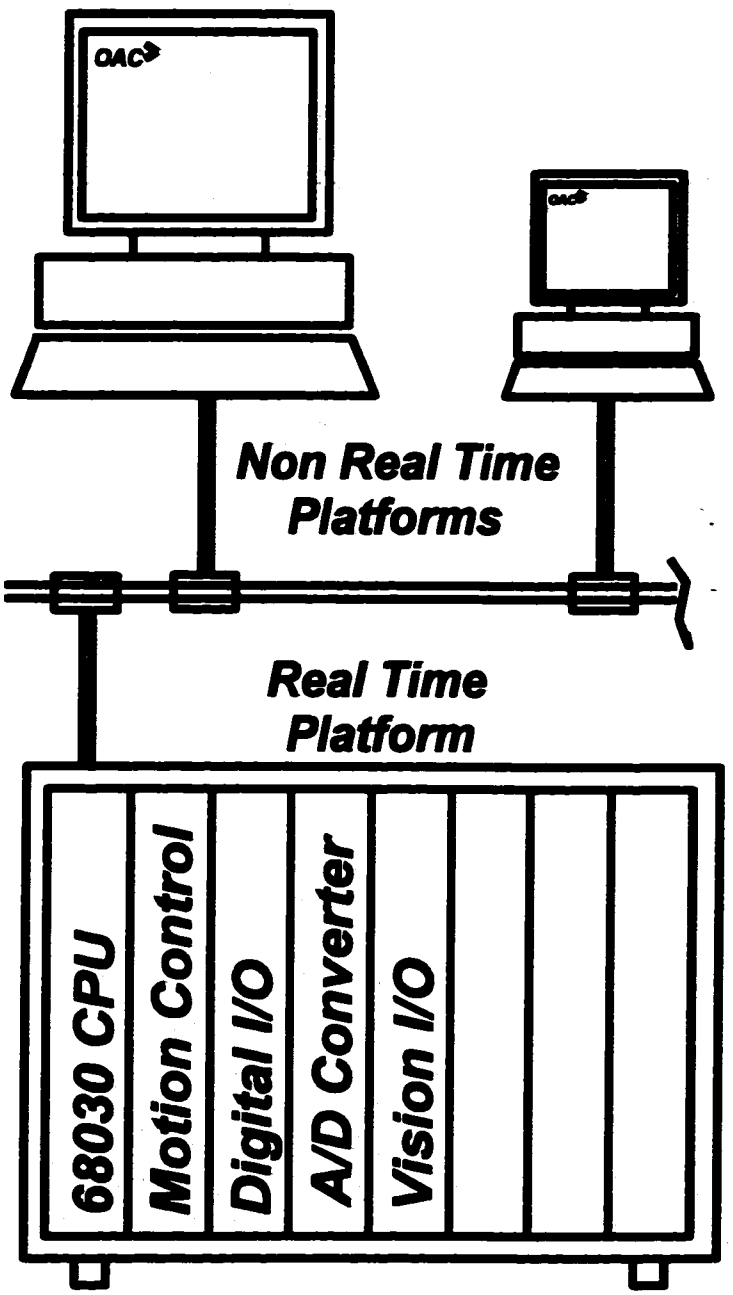


Figure 6. IMMRC OAC Hardware Topology.

is limited to text based commands. There are no restrictions on the format of data exchange between other modules in the OAC.

Internet Protocol (IP) is used as the communications mechanism between modules in the IMMRC OAC. This protocol was chosen because it:

i) is a widely accepted standard (IEEE 802.3),

ii) is supported by the UNIX (POSIX) I/O system,

iii) is supported between modules under all three possible topologies:

- within an operating system (OS),
- across a backplane,
- across a network.

While reasons i) and ii) recognize the potential value of including current standards and open systems technology in the OAC design, the last reason addresses the practical requirements of system designers for a comprehensive range of module coupling options. In many ways, this approach satisfies the concept of a "Common Execution Environment" (CEE), as envisioned by the NCMS LEC project (Haynes, 1994).

The Internet protocol (IP) is not deterministic in time. That is to say that the time taken for a message to be sent over an IP link cannot be strictly guaranteed. This does

influence the real time aspects of a system designed using this protocol however, the following comments can be made:

1) no deterministic protocol currently exists that supports the three reasons listed above for the choice of IP,

2) mechanisms can be built into the application level software to monitor transmission times and manage error conditions (refer for example to Castellote and Schnieder, 1994),

3) in most applications, modules requiring 'real time' response from the Machine Server will not be using a high traffic connection (ie: most likely within an OS, across a backplane, or at worst within a small, dedicated sub network).

Regarding the last point, it should be noted that the design proposed here does not constrain the user from applying poor judgement in the design of their application. This lack of constraint is accepted in order to avoid the definition of hardware configuration and to promote a more 'open' system.

3.6 Machine Server Design

The Machine Server's main function is to isolate other modules in the overall controller system from the hardware dependant aspects of the machine tool "device" - ie: motion and input / output hardware. It achieves this through:

- a defined model of machine "services", ie: actuation and configuration of motion and input / output functions,
- a common interface method (standard communications).

Figure 7 depicts this relationship between the Machine Server and other system modules. The Machine Server encapsulates both the most performance critical and the most hardware specific aspects of the controller system. In this respect it is the focal point of the designed and implemented system.

3.6.1 Software Architecture

The Machine Server is a collection of software processes that provide the various 'services' of the machine device:

- actuation: motion and discrete outputs,
- configuration and state data.

Figure 8 depicts the Machine Server and its various processes.

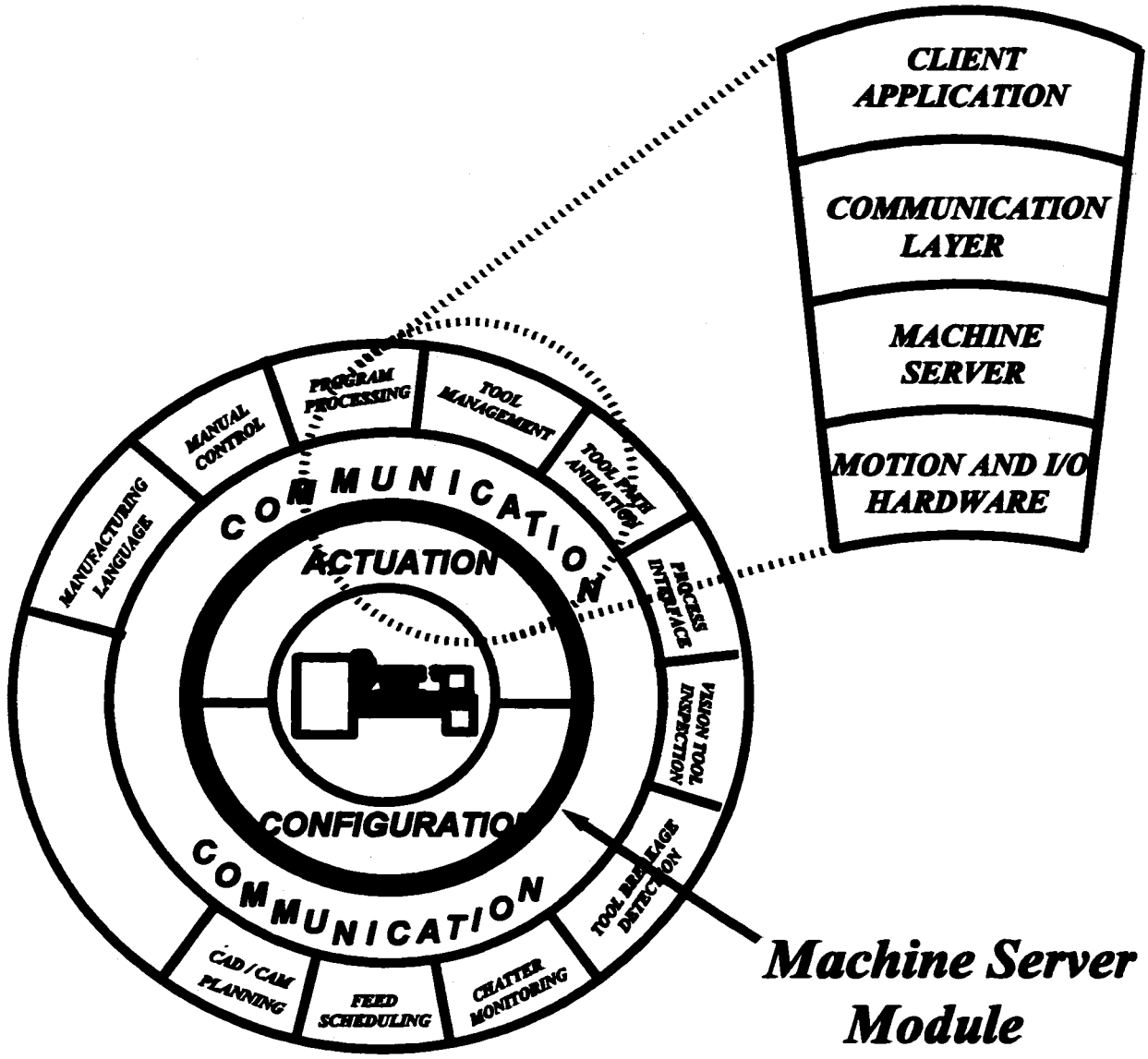


Figure 7. The OAC Machine Server and its Relationship to Other System Modules.

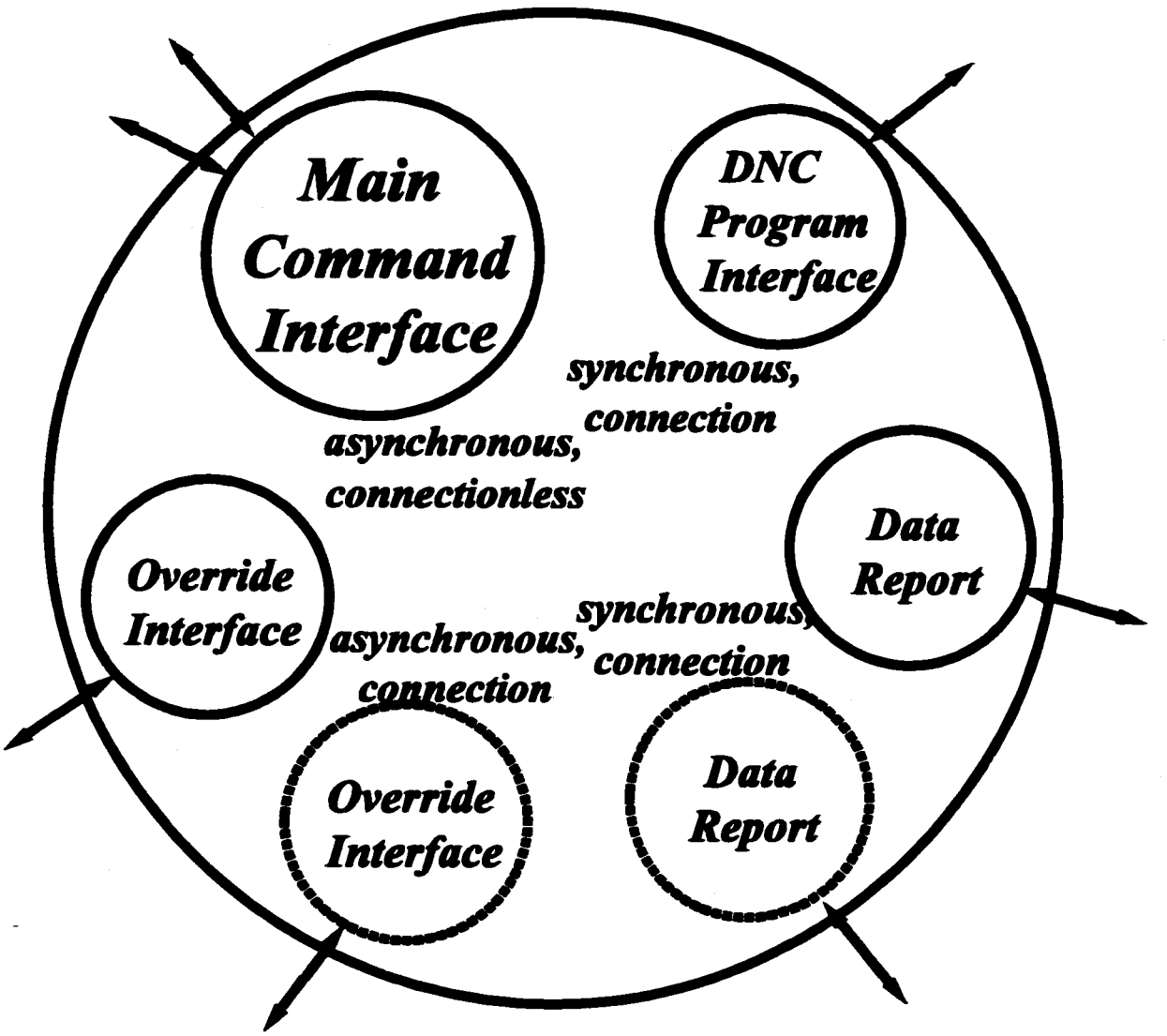


Figure 8. OAC Machine Server.

Primary communication with the Machine Server occurs through it's command interface. The IP address (or name) of the command interface is all that the client application requires to use the Machine Server.

All service requests are issued through the command interface. The command interface is:

- connectionless: able to respond to multiple clients arbitrarily,
- asynchronous: able to respond asynchronously to client requests.

Certain machine services characteristically require dedicated access by the requesting client (eg: motion control). In these cases, the service is initially requested through the command interface, and the Machine Server starts a service specific process dedicated to that client. Once established, the client will communicate directly with the dedicated server process in which some form of data is exchanged. The processes for program control and data reporting address synchronous client / server interactions. The details of these services are outlined in the following section.

3.6.2 Service Model

The functionality of the machine device is presented through the Machine Server as a set of service requests. The service requests are divided into five service classes:

1) Machine: basic machine device operations

2) Program: RS274 (G code) based program control

3) Data: machine device configuration and data access

4) Override: machine device override control

5) System: OAC system commands

These services attempt to present the machine functionality in a manner that is comprehensive, yet as simple and concise as possible.

Machine Services

Machine services are related to the basic operation of the machine. This includes services related to manual control of motion and discrete I/O. Table 1 below summarizes the machine service commands.

Program Services

Program services involve:

- transfer and execution of G code based motion program data,
- client synchronization to motion execution.

COMMAND	FUNCTION
machine powerup	*turn on all machine power systems*
machine powerdown	*turn off all machine power systems*
machine status	*report status information*
machine home AXIS_ID	*home specified axis*
machine move abs AXIS_ID VALUE	*move axis to absolute location*
machine move inc AXIS_ID VALUE	*move axis by incremental amount*
machine jog AXIS_ID DIR	*jog axis in specified direction (+/-)*
machine hold AXIS_ID	*hold motion for specified axis*
machine io test IO_NUM STATE	*test the state of a discrete I/O point*
machine io set IO_NUM STATE	*set the state of a discrete output*

Table 1. Machine Service commands.

Currently, the Machine Server supports only one motion 'device' (ie: program coordinated set of axes). Clients wishing to gain program control issue a 'program open' request to the server. The server will, if program control is not in use, start a dedicated process and communication channel for client / server program data transfer.

This dedicated process, referred to as the 'DNC interface', accepts client program data and buffers it internally for execution. Figure 9 depicts the program buffer model. During execution of the program data, the DNC interface will monitor execution and signal the client to allow continuous program execution. Additional server commands are provided for client control of program execution (start, stop, and reset).

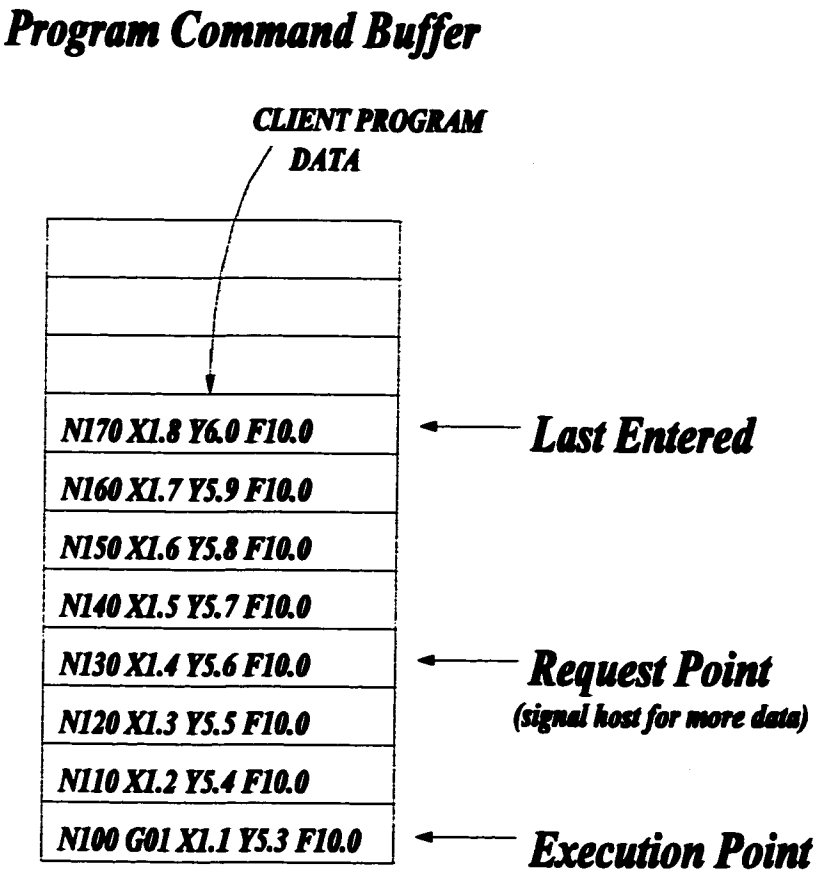


Figure 9. OAC Program Service Program Buffer Model.

There are several instances in which client applications may want to be synchronized with the motion program or the motion itself. For example, the acquisition of monitoring or inspection data, or the use of adaptive control may only be required for certain features of a part being produced. The Machine Server provides a 'program sync' service, specified for either a program line or axes position, that will internally monitor the motion as it proceeds and signal the client application when that line or position is reached. Table 2 below summarizes the program service commands.

COMMAND	FUNCTION
program open	*open the DNC program interface*
program close	*close the DNC program interface*
program status	*report status of the program interface*
program start	*begin program execution*
program stop	*stop program execution*
program reset	*stop execution / clear program buffer*
program sync position ...	*set a 'sync' to specified position*
program sync line ...	*set a 'sync' to specified program line*

Table 2. Program Service Commands.

Data Services

The OAC maintains an internal database consisting of motion and I/O data for configuration and current state. All system data is referred to by logical name (eg: position_x, velocity_y, etc). Services are provide for reading and writing of data.

A data reporting service is provided to regularly send clients data updates. Upon receiving a 'data report' service request, the Machine Server starts a dedicated process and communication channel to send the data item(s) to the client at the specified rate. The rate has a resolution of milliseconds, and can be used by the client to update displays, record sensor data, or time external feedback loops. Table 3 below summarizes the program service commands.

COMMAND	FUNCTION
data get NAME, ...	get value(s) of specified data items
data set NAME VALUE	set value(s) of specified data items
data report RATE NAME, ...	start data report at specified rate
data cancel ID	cancel data report

Table 3. Data Service Commands.

Override Services

Client applications are able to override motion at any time using the override services. The following variables can be affected:

- feed (vector velocity),
- spindle speed,
- axes position.

Note that override can apply even when not running a motion program. In the case of position override, a 'dynamic' offset is superimposed on the program generated trajectory.

Override values have a resolution of unsigned 16 bit. This implies for example, a resolution for feed commands of one thousandths of an in per revolution for a programmed maximum (100%) feed of 65 inches per minute.

When an 'open override' request is received by the Machine Server, the server starts a dedicated override process and communication channel for the requesting client. Client override data is then sent through this channel using a simple data oriented format (eg: F 123.45, feed percent override).

Override services apply to the single motion device (ie: set of axes). Multiple clients may request and receive access to device override simultaneously. Currently no priority mechanism is implemented to resolve conflicting client override commands, although this may be implemented at a future date.

COMMAND	FUNCTION
override open	open an override interface
override close	close an override interface
override status	report override status and settings

Table 4. Override Service Commands.

System Services

System services are used for basic maintenance of the OAC system Currently only two services are supported; one for system shutdown, and one for client / server communications testing. The latter of these can be used by client applications to test request transmission timing.

COMMAND	FUNCTION
system shutdown	disable Machine Server software
system ping	return communications acknowledge

Table 5. System Service Commands.

3.7 Performance Results

Several tests were performed to evaluate the communication performance of the machine server with client OAC modules. The time required for a client application to send a command and receive an acknowledgment from the machine server was measured under different conditions. The test data represent communication timing only. The results obtained are shown in the following figures as histograms summarizing 160 tests of 1000 send / receive cycles each. Datagram (UDP) packets were used to send and receive short text strings typical of the machine server commands.

Tests were performed for different cases of loading on the server (in the form of active data report tasks), and for different client / server topologies. Network based tests were done on a lab sub network (10 Mbits/sec Ethernet) during normal daytime loading. Figure 10 shows a sample result of the testing. Other test results can be found in the appendices of this report.

The results indicate that the server response under reasonable loading is adequate

to support client applications requiring data / override at a rate of 50-100 cycles per second or less. This will readily support feedback based process control and monitoring of low bandwidth process variables (eg: spindle power). Higher bandwidth monitoring data, such as that used for tool breakage and chatter, must be achieved using more direct interfaces to signal sources (eg: memory mapped) as opposed to using the OAC server interface (IP based) This does not however, alter the functional view of the OAC system as a whole consisting of loosely coupled, heterogenous processing agents. It simply treats such monitoring systems as having independent signal sources.

From a systems concern, the key factor for any monitoring activities remains the latency between detect and react times, in which the machine server must be commanded to alter its actions (usually motion). In this context however, the communication capabilities of the OAC appear to be adequate for both tool breakage and chatter monitoring in which the reaction time is still one or two orders of magnitude less than the time constants of the machine drives involved.

It is interesting to note that the communications performance under the topology in which the client and server are on the same CPU is worse than that of the case in which they are separated over the network. It is suspected that this behavior results from the fact that the implementation of the communication protocol for network media is hardware based (Ethernet 'chip'), while the backplane and intra-CPU configurations are software based. Performance for these latter cases will consequently depend on processor speed and loading - a result that appears to be evident in for those test cases.

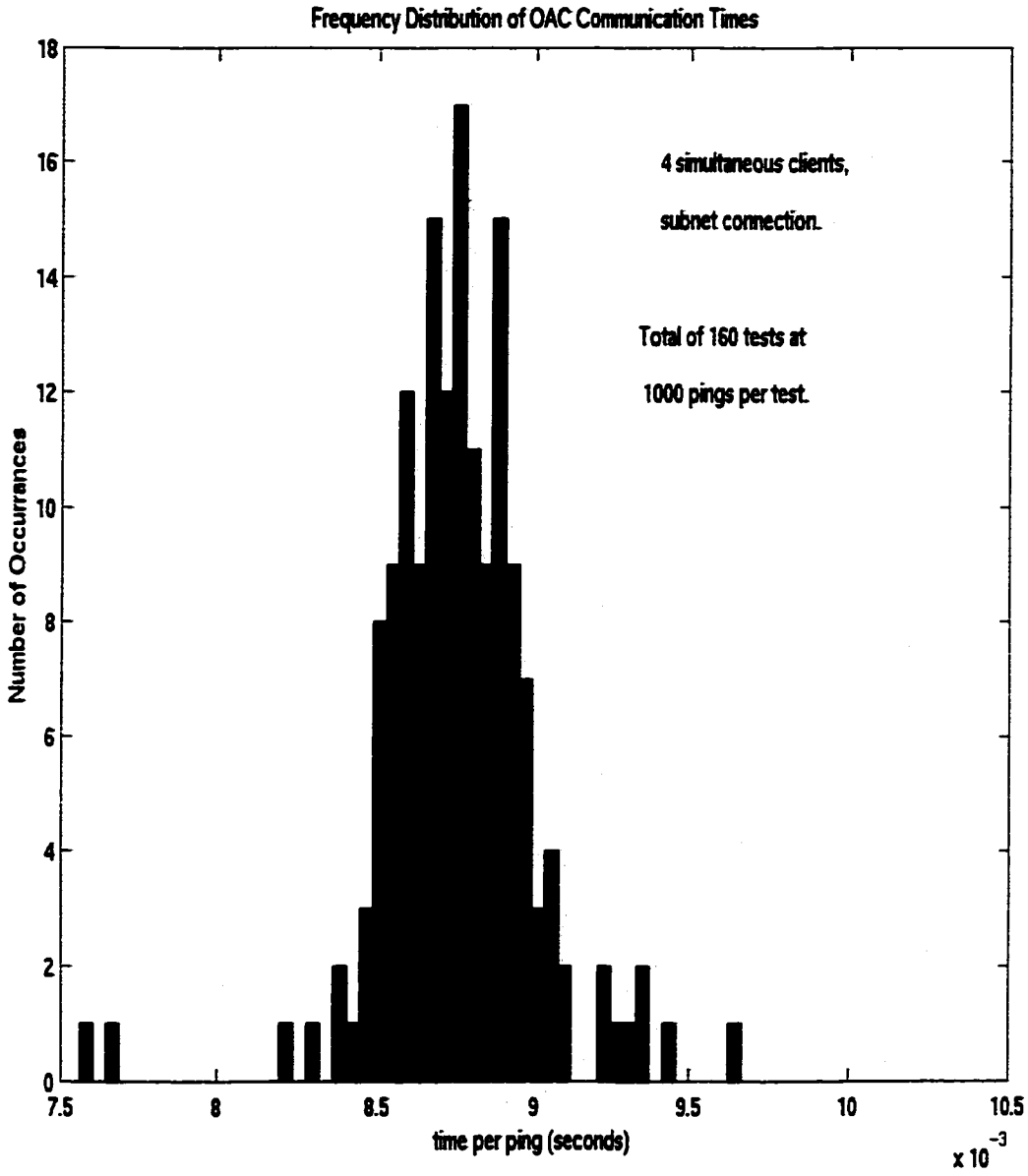


Figure 10. Typical results of OAC communications tests.

Chapter 4

DEVELOPMENT OF AN INTELLIGENT MACHINING TEST BED

4.1 Machine Tool

The machine tool is a LeBlond Knight turret lathe. The spindle has a 20 horsepower drive with continuously variable speed to 2200 rpm. Figure 11 shows the machine.

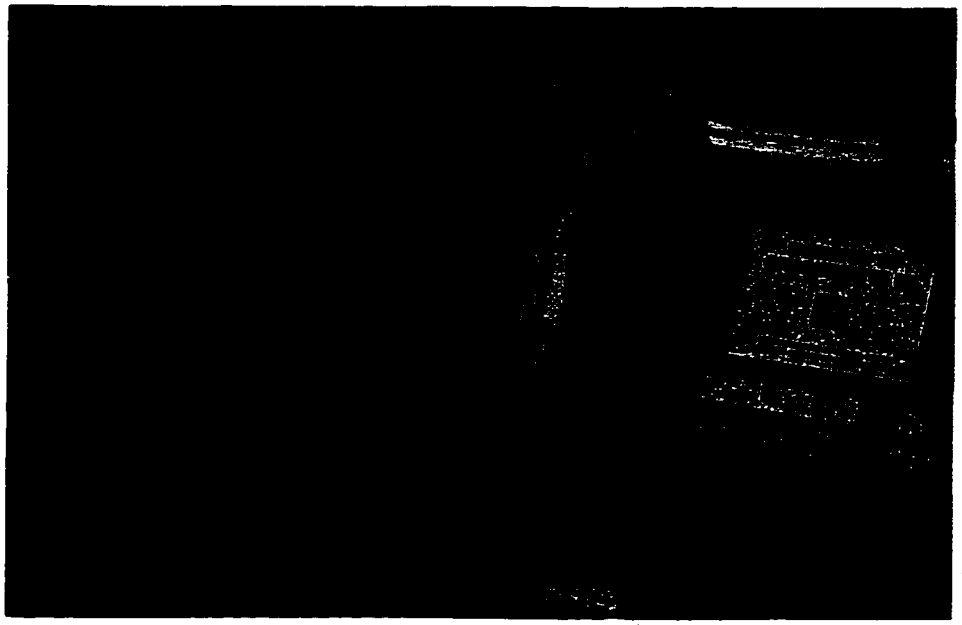


Figure 11. Instrumented LeBlond knight 20 Machine Tool.

4.2 Cutting Force Sensing System

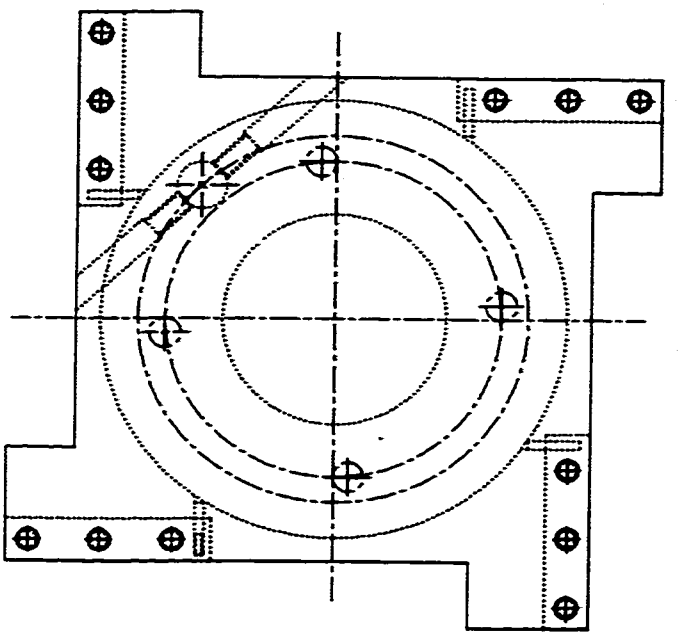
A three axis plate dynamometer was designed and installed on the turret lathe. The dynamometer is mounted between the four station turret and the lathe carriage as shown in figure 12.

The dynamometer uses four piezo electric load cells, each measuring three axes of load. The charge current produced under loading by the piezo electric elements is summed between load cells for each of the directional components. This results in a cancellation of induced moments under load, and correspondingly negligible cross sensitivity between measurement axes.

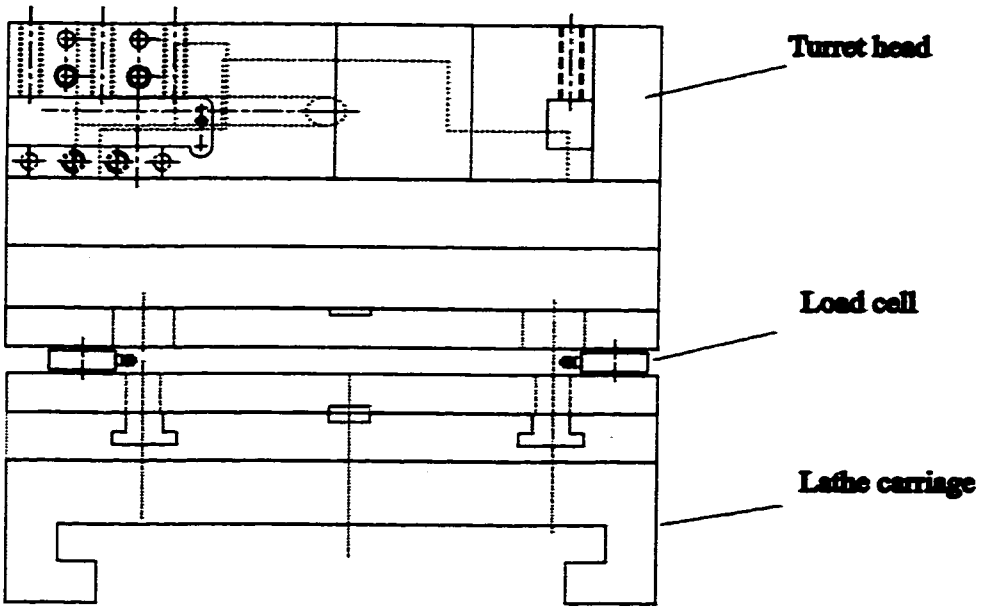
The charge signals are routed to appropriate signal conditioning hardware and then connected to the OAC analog input terminals. Software configuration of the OAC machine server database allows client applications to access this data through requests to the machine server (data services). Other analog signals, such as that from accelerometers mounted on the turret, are handled in a similar manner.

4.3 Vibration Sensing for Machining Chatter

In turning, the distinction between stable and unstable cutting is rather clear. Vibration signals measured at the tool post during stable cutting are essentially Gaussian noise. When chatter occurs, the same signal will exhibit string sinusoidal behavior, with a dominant frequency in the range of 100 to 600 Hz.



TOP VIEW



FRONT VIEW

Figure 12. Cutting Force Dynamometer Designed for Turret Lathe.

In this system, an algorithm is used that counts the vibration signal mean crossing rate to detect the transition from random to sinusoidal behavior. The analog signal is band pass filtered from 50 - 500 Hz to 1) remove rotational frequencies, and 2) to prevent aliasing during digitization at 2 kHz. Every ten samples the algorithm updates the rate calculation which, if it remains within a narrow band for a specified period of time, will trigger a chatter alarm.

4.4 Tool Wear Measurement

4.4.1 Camera and Lighting System

Automated tool wear measurement is realized using a machine vision system (Sexton et al, 1996). Figure 13 depicts the camera and lighting setup, illustrating the location of the system in the turret lathe application. A vertical camera provides measurement of flank wear, while a second, inclined camera provides measurement of crater and nose wear, and tool edge profile. Both cameras use fixed focus magnifying lenses.

Two high intensity light emitting diodes (LED's) are used to illuminate the flank surface and tool profile. A miniature red laser with a diffracting head is used to project light stripes on the rake face. The crater depth can be estimated from the distortion of the stripes as they lie across the crater. This is discussed more in the following sections.

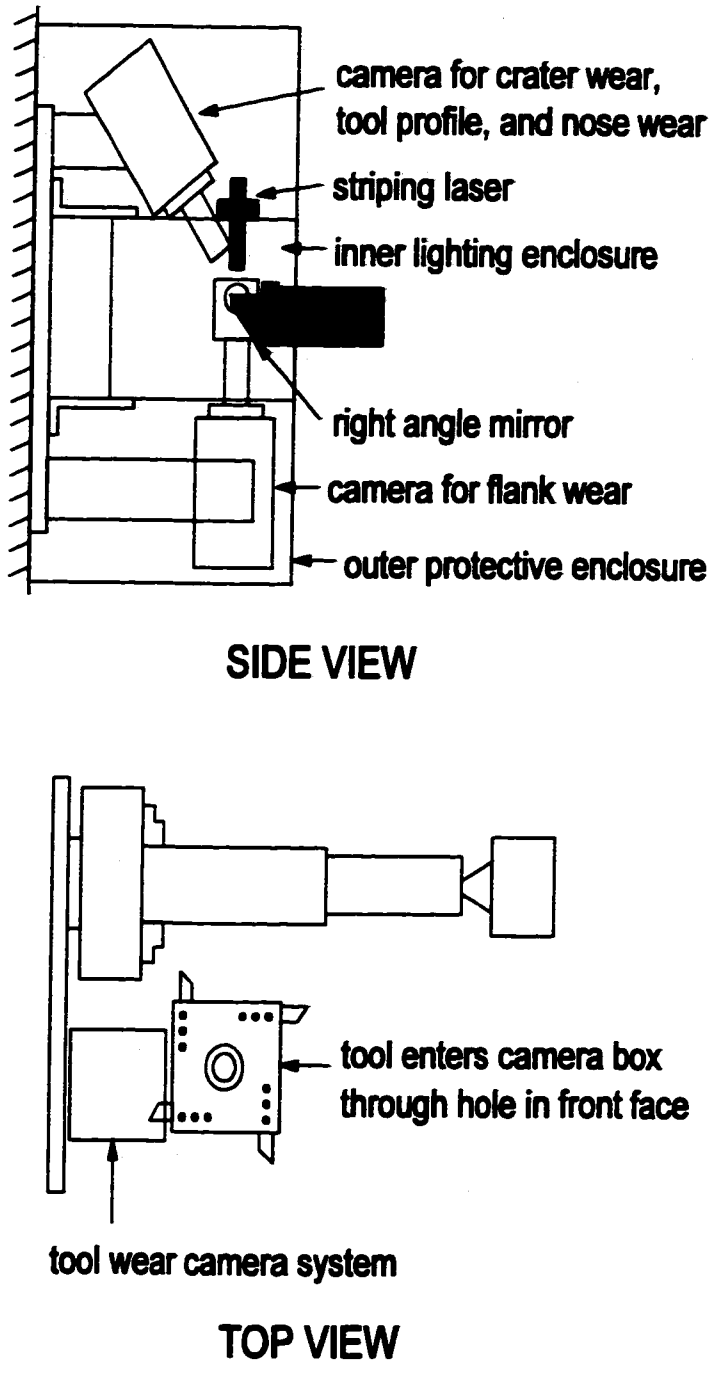


Figure 13. Machine vision system for tool wear measurement.

4.4.2 Machine Vision Algorithms

Flank Wear

The width of the wear land is calculated at several points along the cutting edge, providing maximum (VB_{max}) and average values (VB_{ave}). Figure 14 shows an example of an original and processed image.

Flank wear estimation relies on the difference in reflectiveness of worn and unworn regions on the tool insert. Repeatable lighting conditions are critical for accurate estimation from the acquired image. This is largely provided for by the camera system enclosure and the highly repeatable positioning of the lathes CNC system.

The image processing algorithm performs the following steps to generate the wear land estimate:

1) separate the worn region from all other image data,
2) rotate the worn region to align the cutting tool edge perpendicular to the image pixel columns,
3) at multiple points across the wear land, search vertically to find bottom and top coordinates of the edges of the worn region (for width),
4) calculate maximum and average value, calibrate to distance units.

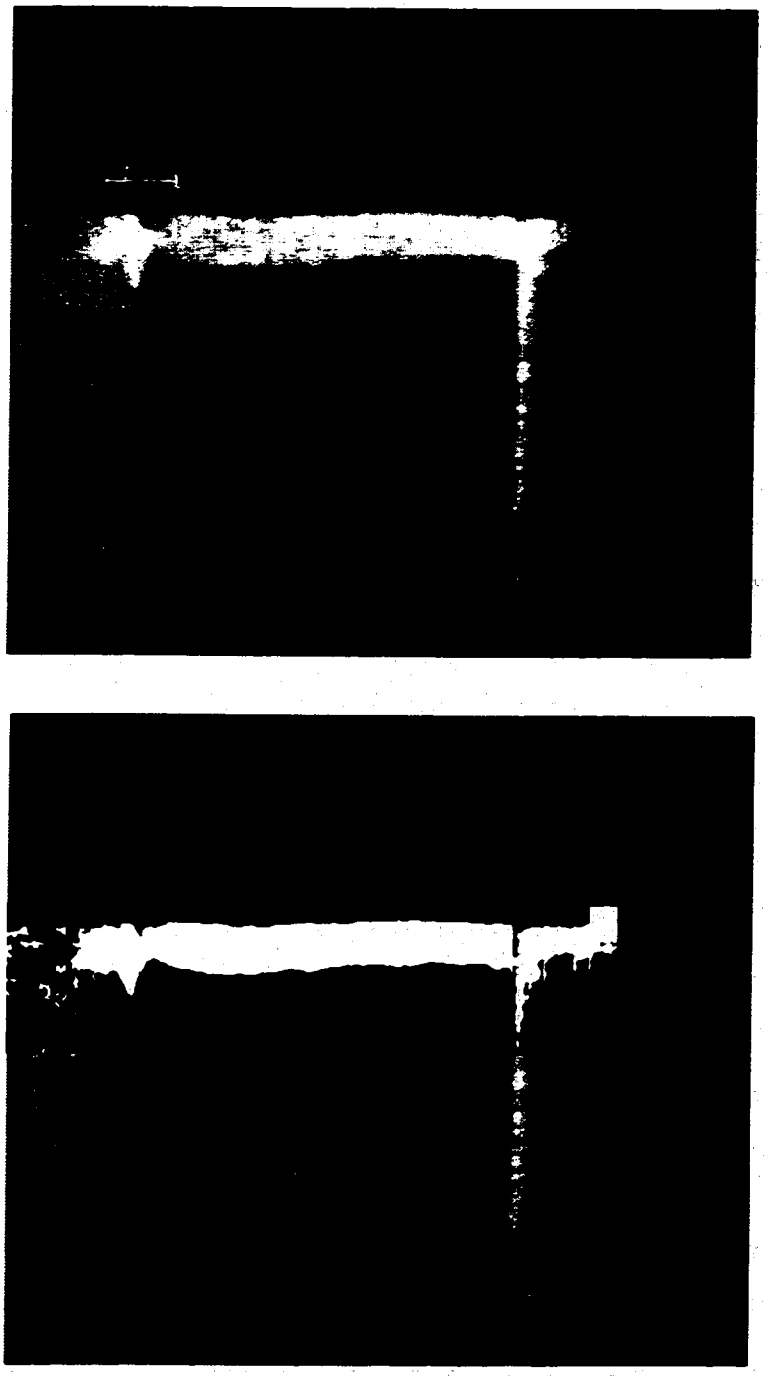


Figure 14. Examples of raw and processed flank wear images.

Separation is performed by applying a threshold on the image data resulting in a binary image consisting of a contiguous white worn region and black everywhere else. The threshold level is calculated from a fixed fraction of the average pixel intensity. The operation of calculating and applying this threshold is repeated for sub sections of the entire image. This localized approach increases the overall robustness of the separation in so far as allowing for low frequency variations in contrast across the image.

The geometric center of the worn region is calculated and the worn region is translated to the center of the image data array. This step is done to avoid losing image data during rotation (as results when the worn region is close to an edge of the image and the corresponding data gets rotated 'off' the image). Starting at the top of the image, the pixel columns are searched downwards to find points at the top edge (cutting edge) of the worn region. Several of these points are used to fit a least squares estimate of the equation of a straight line to the top edge of the worn region. The calculated slope is used to determine the rotation angle required to align the top edge of the worn region perpendicular to the direction of image pixel columns. The image is then rotated about its geometric center, resulting in the ability to search along pixel columns to measure the width of the worn region perpendicular to the cutting edge. The rotation is performed only if the calculated slope is greater than two degrees (back rake for carbide turning tools is typically + / - 6 degrees).

To calibrate the flank wear measurement a precision gauge block is used an artifact and positioned by the CNC system at the tool measurement location. The

calibration factor of pixels per millimeter is taken from the acquired image. This value is then used to scale width values from the processed image. Using a toolmaker's microscope, actual tools from wear tests were compared with the data from the vision system. It was found that the values compared very well, particularly for the maximum wear land values. It should be noted that average flank measurement using the microscope is somewhat inconsistent as the "average" is based on a much coarser approximation than the vision based system (by an order of magnitude in the number of data points used). Figure 15 below shows a comparison plot of maximum flank wear measurements from a toolmaker's microscope and the implemented vision system.

Tool Profile and Nose Wear

The tool profile image is captured by the same camera used to measure crater wear. In this case, a small light source over the tool is used to illuminate the tool face. The overall image is divided into three regions:

1) the tool nose radius,

2) the tool leading edge,

3) the tool trailing edge.

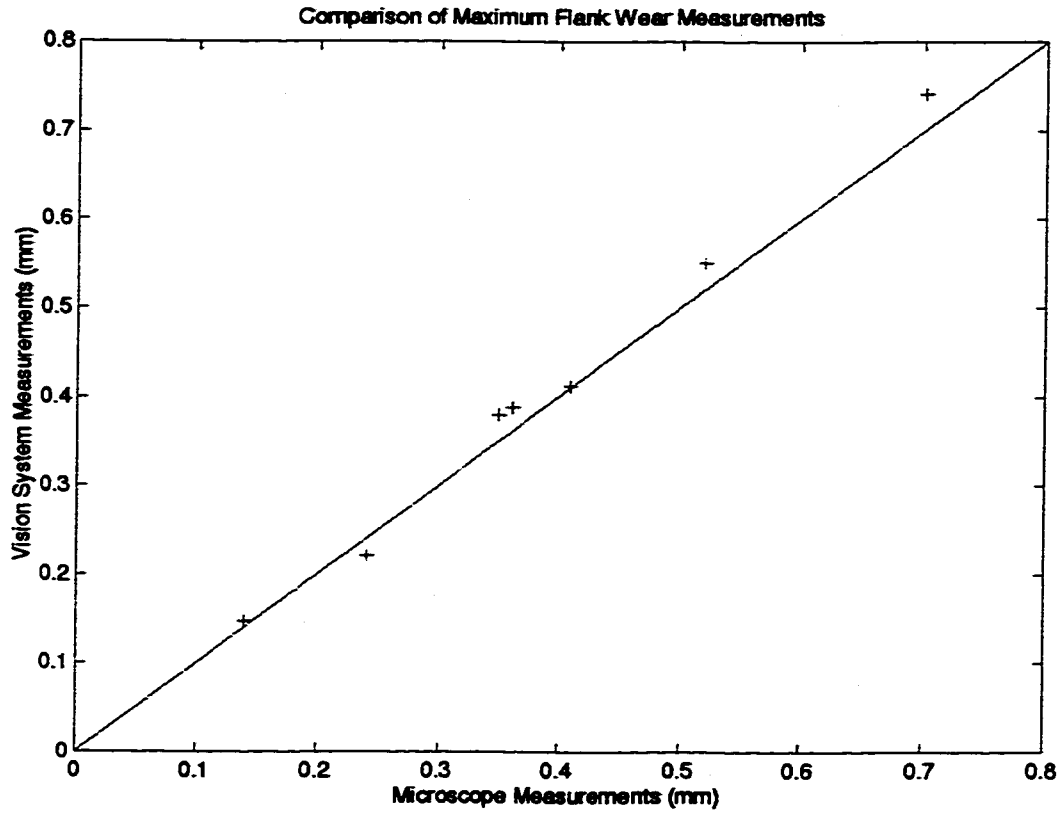


Figure 15. Comparison of Maximum Flank Wear Measurements Obtained from a Tool Maker's Microscope and the Implemented Vision System.

The coordinates of these regions are determined beforehand, and remain valid for a given tool positioned with the CNC

The profile image is thresholded in a manner similar to that described for the flank wear measurement. Points along the leading and trailing edges are used to determine line equations describing their orientation. Points taken about the tool nose radius are used to estimate tool form in this region by comparison with a predefined circle describing the manufacturer specified sharp tool nose radius. For each point taken,

the distance between that point and the nose radius center is calculated. The difference between this value and the original nose radius gives an estimate of the tool nose wear. Gross values of this or the leading and trailing edge line descriptions can also be used to identify a broken tool. Figure 16 shows raw and processed tool profile images.

Crater Wear

Crater wear is most typically quantified by the maximum crater depth (Kt), and the distance of the maximum depth to the cutting edge (Kr). In order to measure the depth from the two dimensional picture, a technique used by Giusti et al (1987) is employed. A diffraction pattern is projected onto the rake face of the tool, and the camera views the image at an angle approximately 35 degrees from the perpendicular. Portions of the laser stripe entering the crater will show up on the resulting image as a deviation from the otherwise straight line. For a fixed tool geometry the deviation is calibrated to crater depth. Similarly, a discrete outline of the perimeter of the crater can be determined, and consequently, an estimate of the parameter Kr. Figure 17 shows a typical picture of a crater worn tool with the stripe laser projected on it and the illustrated results of the processing algorithm.

The region of the image about the tool tip is thresholded in a similar manner to that described for flank wear measurement. A search along the top pixel row in this

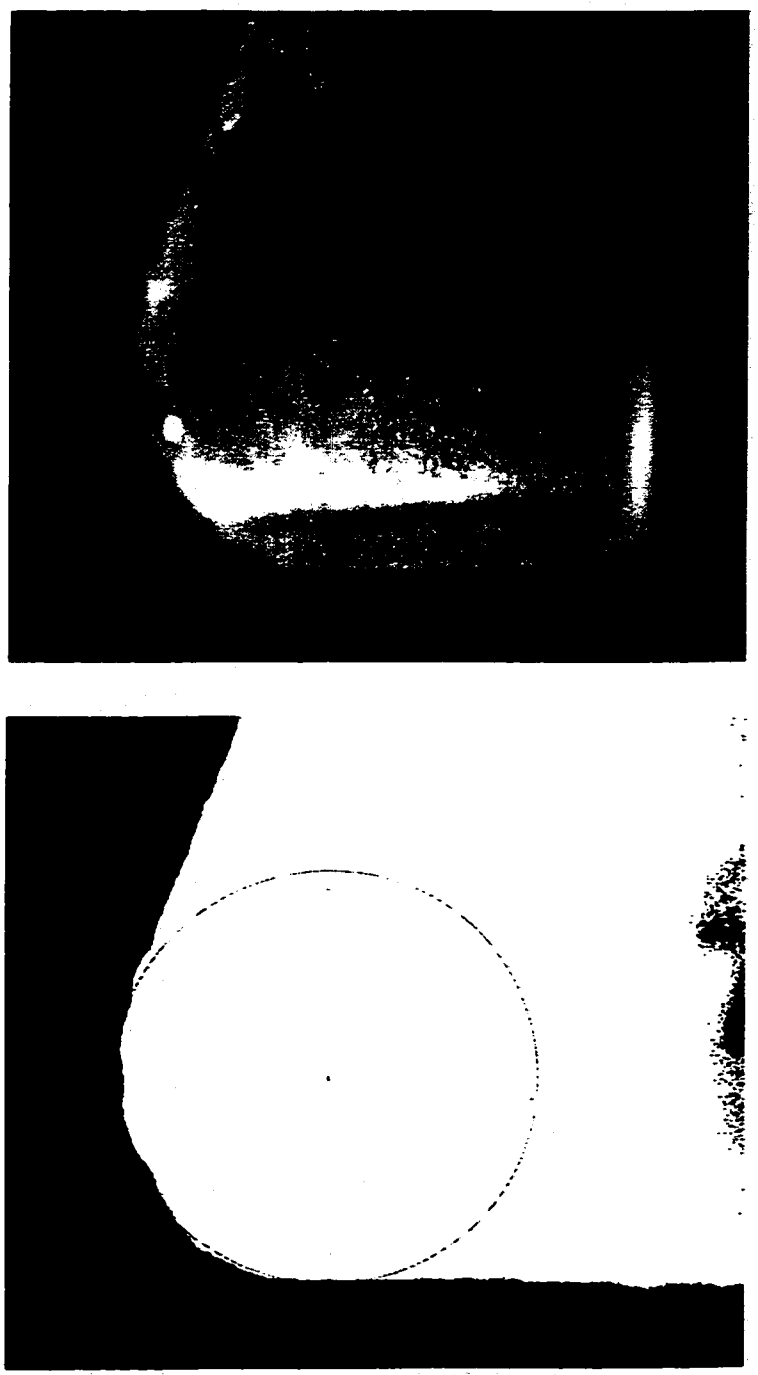


Figure 16. Examples of Raw and Processed Tool Profile Images.

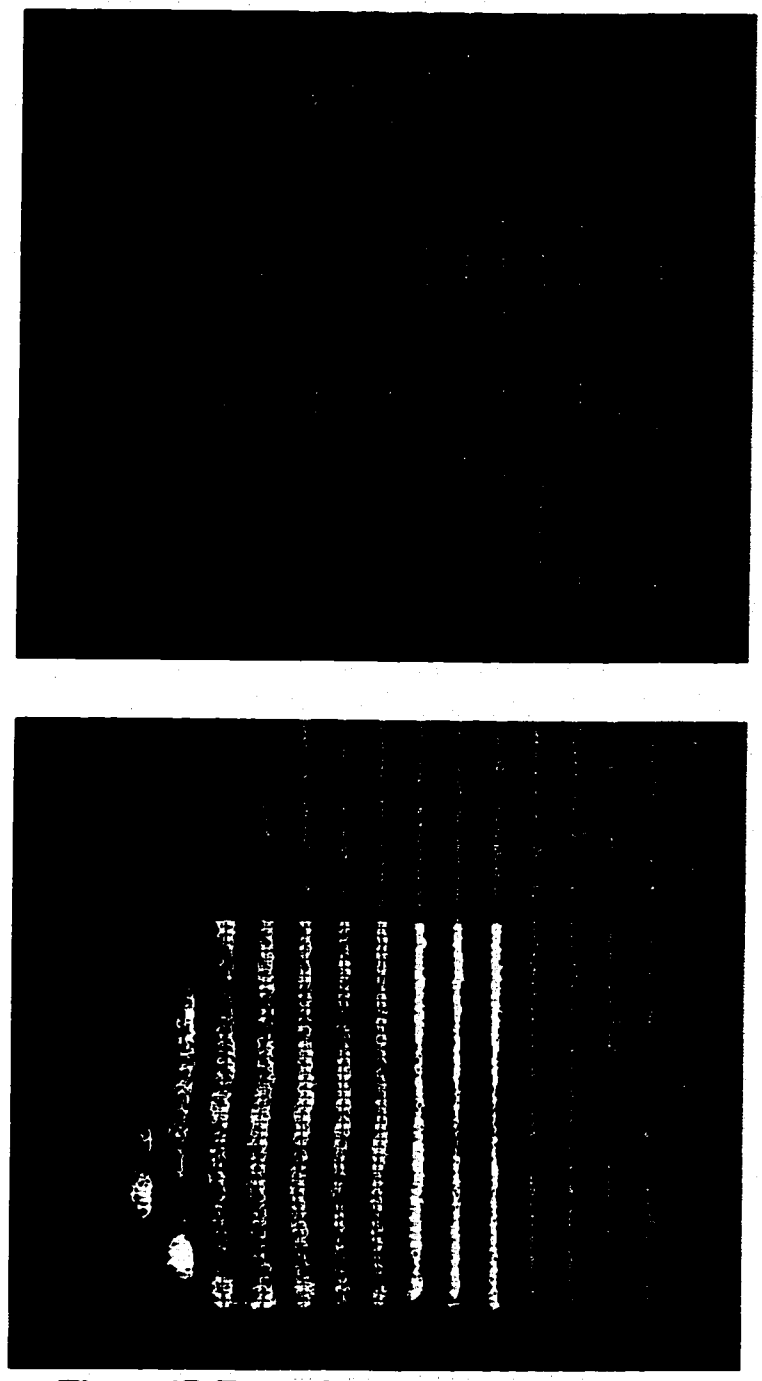


Figure 17. Examples of Raw and Processed Crater Wear Images.

region determines the coordinates of points on the individual laser stripes. Within a sub region about each stripe, the center line of the stripe 'tracked' by performing a medial axis transformation (MAT) row by row down the sub region. The processed image in figure 17 shows the located positions marked by '+' symbols. The maximum deviation of the tracked stripe from its straight line value found in the upper portion of the rake face provides the measurand for maximum crater depth (K_t). An equation describing the cutting edge in the pixel coordinate system is determined from the tool profile measurement in a manner similar to that described in the flank wear section. The value of K_t, measured perpendicular to cutting edge, is determined by:

1) finding the equation of a line passing through the pixel coordinates of the location of maximum crater depth,
2) finding the intersection of the line found in 1) with the line describing the cutting edge,
3) calculating the distance in pixels between the point found in 2) and the point of maximum crater depth, and calibrating the value to distance units.

To calibrate the depth measurements a Mitutoya surface profilometer was used. This device uses a mechanical stylus to measure surface profiles. By scanning over the crater region of a tool, the maximum displacement of the stylus provides the K_m

measurement. Similar to the flank wear measurement, a calibration factor between the measured straight line deviation and the actual crater depth is determined in units of pixels per mm. Figure 18 shows a comparison between K_m obtained with the vision systems and with the surface profilometer. Note that the calibration is only valid for a fixed value of tool back or side rake.

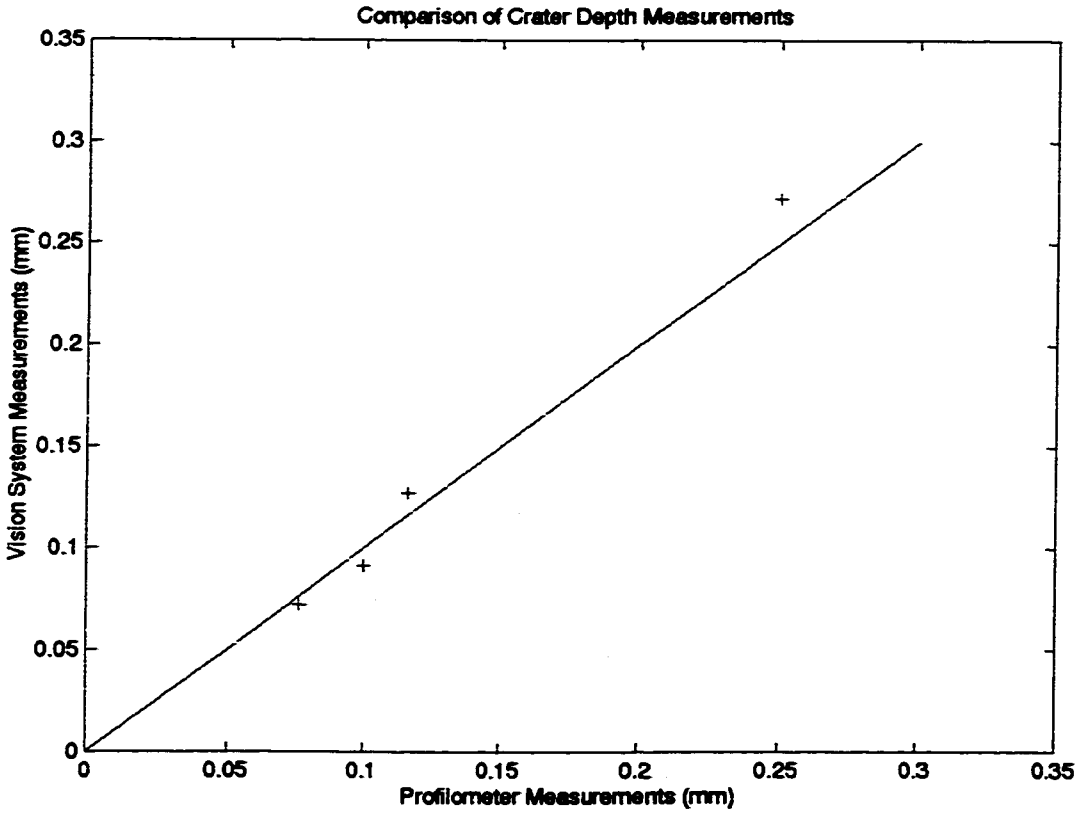


Figure 18. Comparison of Crater Depth Measurements Obtained from a Surface Profilometer and the Implemented Vision System.

4.5 Surface Roughness Measurement System

4.5.1 Ultrasonic Transducer and Electronics

An ultrasonic system is used to measure surface roughness between part cycles. A highly focused, non contact transducer is used as an ultrasonic transmitter and receiver. Sensor to part surface displacement is estimated from the measured pulse / echo time of flight. The focused ultrasonic wave is scanned along the workpiece in the feed direction, resulting in a stream of data representing the surface topography.

The ultrasonic wave, being highly attenuated in air, is coupled to the part surface using a stream of cutting fluid from a specially designed nozzle that houses the transducer element. The sensor is mounted on the far end of the cross slide of the lathe, and moved under the part to scan along the part length. The 33 MHz ultrasonic signal is coupled to the part using cutting fluid with a 'dribbler' end. Figure 19 depicts the setup. While this arrangement is limited to a fixed part diameter, it is suitable for multiple parts production, in which only the final pass of each part is measured. Figure 20 shows a typical scan from a turned workpiece in which several patches have been cut with different feeds. The peaks at the end of each patch are the results of cusps left on the work surface from the tool's radial motion programmed between cuts.

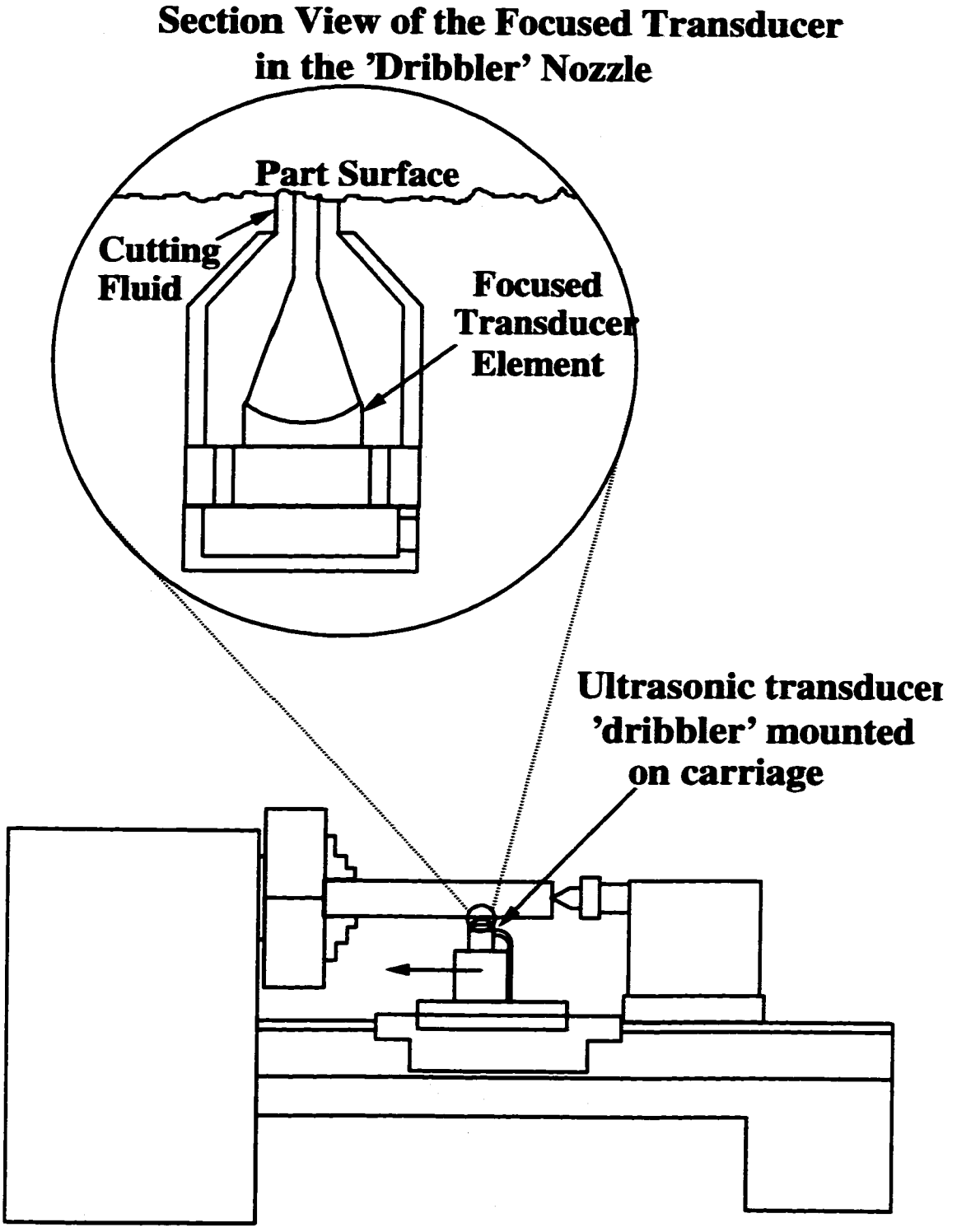


Figure 19. Surface Roughness Measurement using an Ultrasonic Transducer Mounted on the Radial Axis Table.

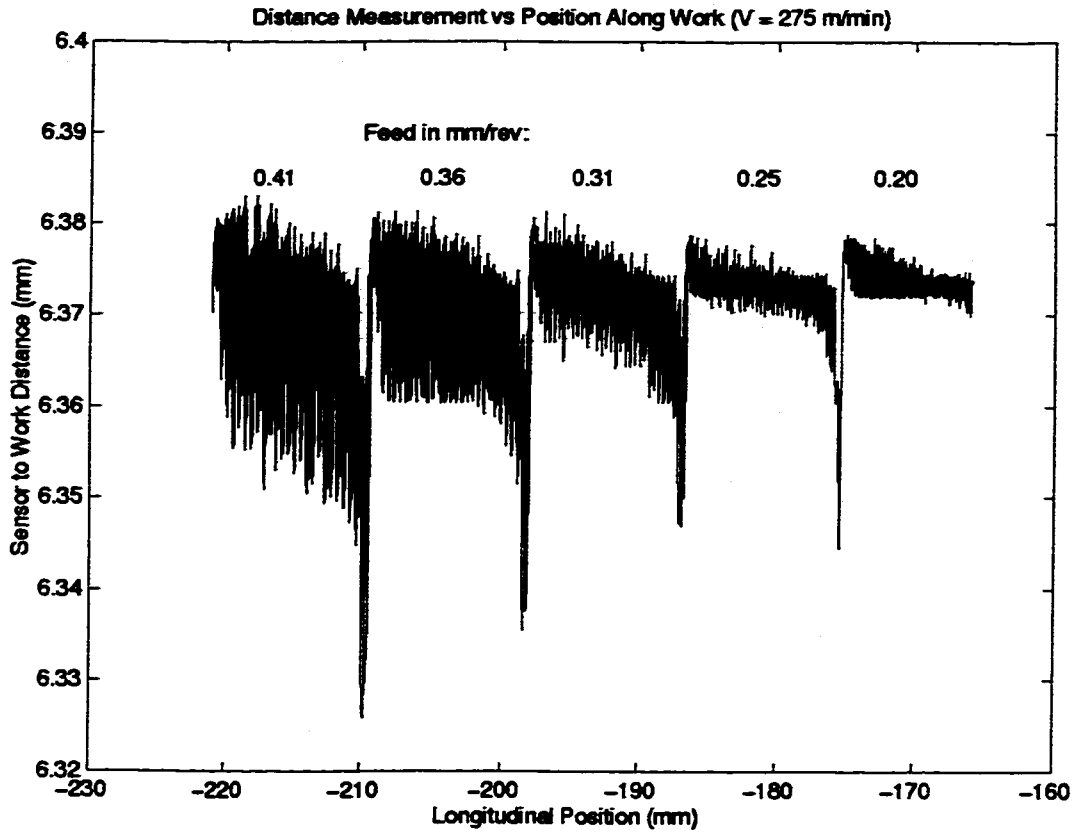


Figure 20. Surface Scan of Workpiece with Sections cut at different feeds.

4.5.2 Signal Processing

Figure 21 show a sample plot of the raw surface data from a machined mild steel part. The raw surface data is processed to:

- remove gross drift in the mean level due to spindle or part holding error,
- filter outliers from the data due to phase inversion of the ultrasonic signal.

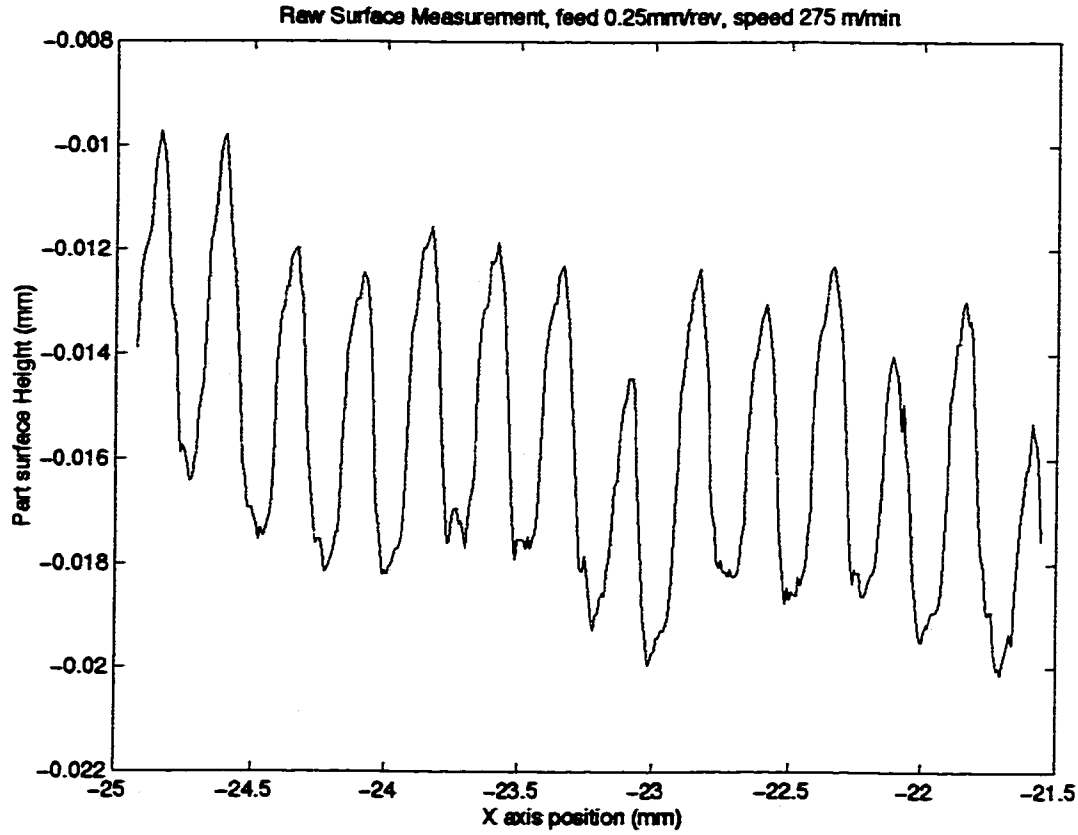


Figure 21. Example Plot of Raw Surface Data.

The first of these operations is achieved by performing a recursive estimation of the signal mean level, and subtracting this mean from the signal at each sample:

$$m(i) = m(i-1) + K(i-1)*(s(i) - m(i-1)) \qquad (1)$$

$$K(i) = \frac{1}{\lambda}(K(i-1) - \frac{K(i-1)^2}{\lambda + K(i-1)}) \qquad (2)$$

$$S(i) = s(i) - m(i) \qquad (3)$$

where,

i - data sample indices,

s - part surface to sensor distance (raw data),

m - estimated mean value,

S - processed part surface data,

λ - 'forgetting factor' for recursive calculations,

K - the recursive gain,

This is in effect a low pass filtering operation that specifies the amount of filtering by an effective window size, W in the mean level calculation:

$$W = \frac{1}{1 - \lambda} \qquad (4)$$

The geometric average surface roughness, Ra, is calculated by numerical integration:

$$R_a = \frac{1}{N*dx} \sum_{i=1}^{N} |S(i)| * dx \qquad (5)$$

where,

N - total number of data samples

dx - distance between sampled points

Filtering for outlier data is performed by simply checking the data serially for large shifts in value and removing the detected shift. The highly focused ultrasonic wave is sensitive to sharp changes in the surface topography that result in an inversion of phase in the returning pulse. The effect of the phase inversion is a shift in the detected return time of the transmitted pulse. Because this shift physically occurs between 'samples' of the time data, it is easily identified as it is the largest shift seen between any two adjacent samples (ie: all other valid data represents lower frequency changes in the surface topography that will occur over several samples).

Figure 22 shows the processed data from figure 21. The ultrasonic pulser / receiver / timing system is connected to the OAC through a serial link. The OAC data base is configured to perform acquisition cycles on client request.

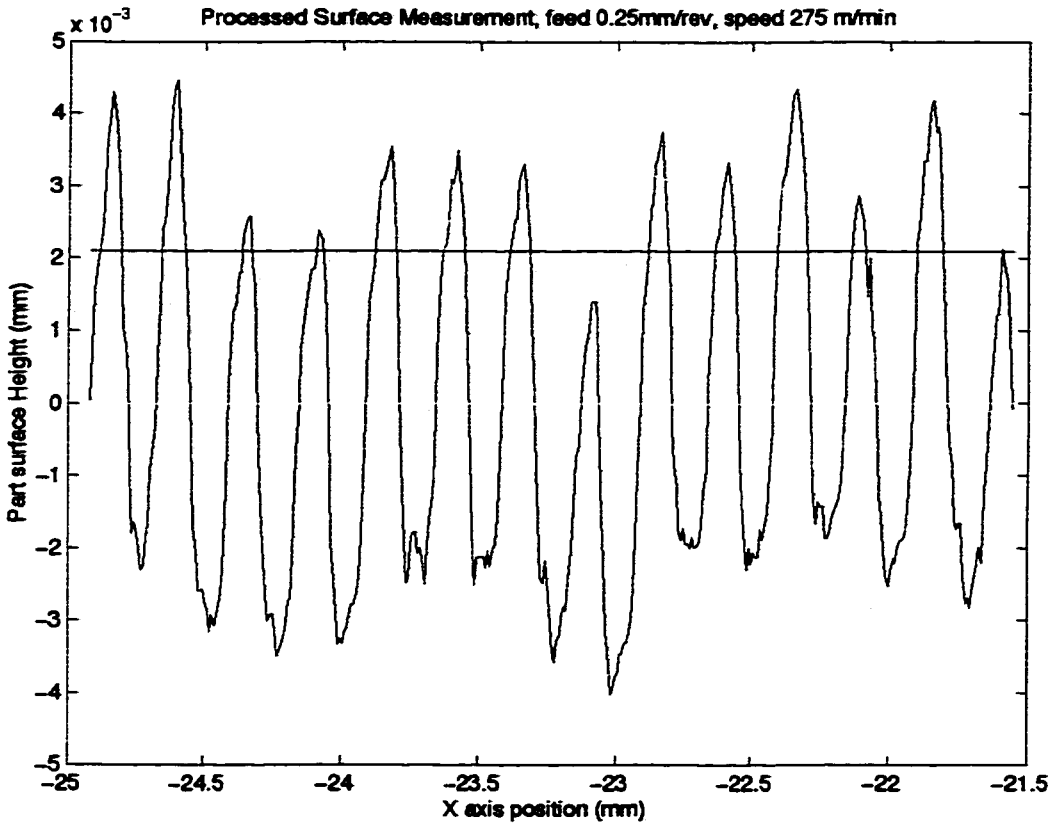


Figure 22. Processed Data, with Solid Line Showing the Calculated Average Surface Roughness, R_a.

Figure 23 shows a calibration plot of the ultrasonic surface roughness measurement system. The reference measurement is taken using a stylus type Mitutoya 211 "Surftest" profilometer. The results show good agreement over the tested range, with overall fit improving for higher values of roughness. This observation is consistent with the known behavior of the ultrasonic system in which the averaging effect of the non contact measurement (ie: the effect of a finite "spot size" of the ultrasonic beam).

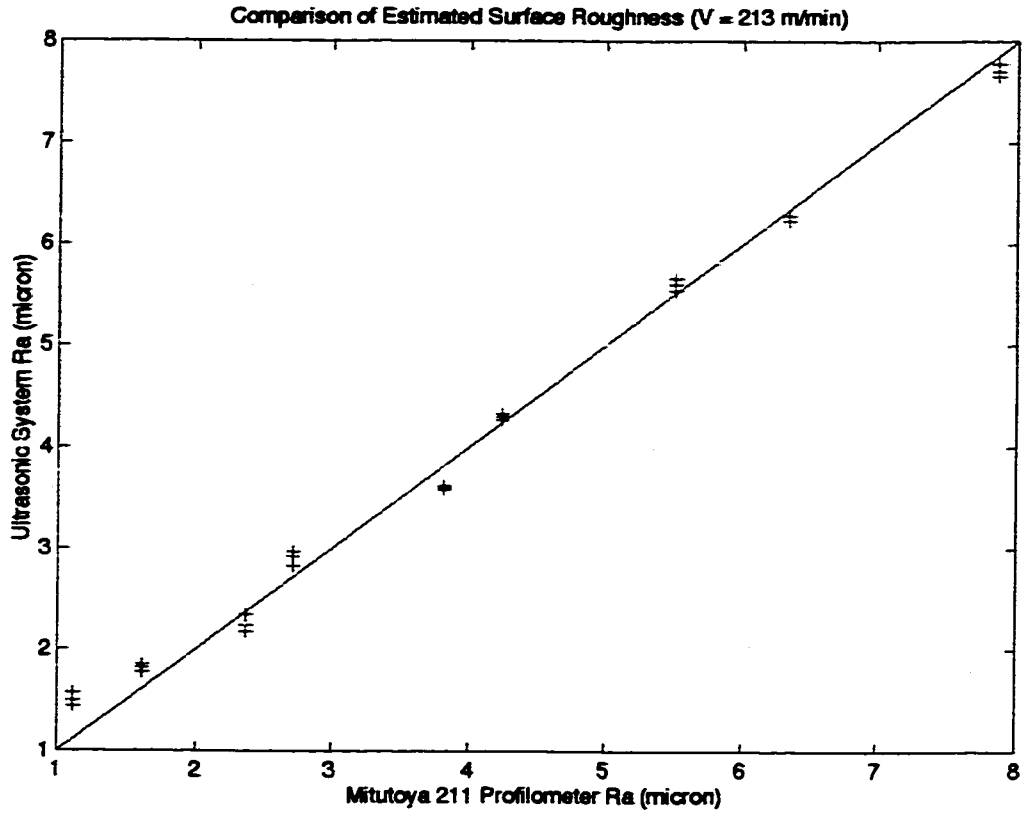


Figure 23. Comparison of Surface Roughness Measurement with Ultrasonic System vs Conventional Stylus Type Device.

4.6 OAC Implementation

The lathe application of the OAC is shown in figure 24.

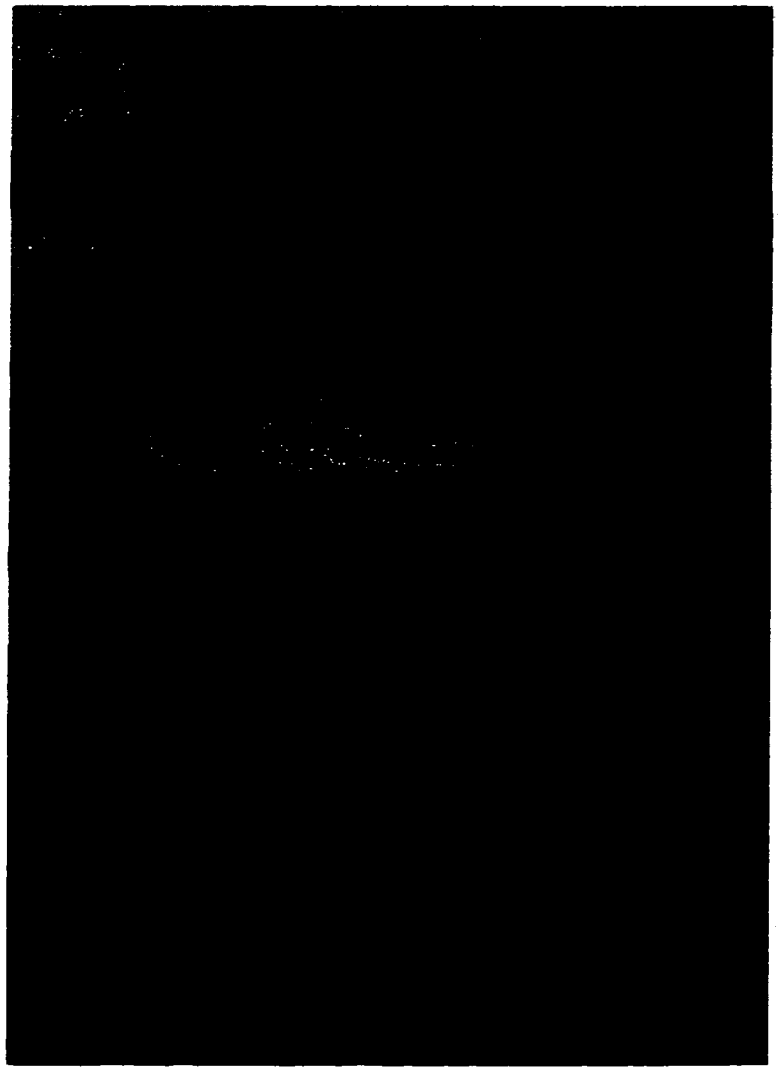


Figure 24. Implementation of the OAC System.

The OAC machine server is run on a general purpose CPU on the real time platform. A real time operating system (VxWorks) hosts the server software and supports Internet Protocol communications for clients across network, backplane, and interprocess interfaces. The server software controls motion and input / output hardware through memory mapped interfaces.

As an example client application, an operator interface has been developed for the lathe. Figure 25 shows the interface screen.

Figure 25. LeBlond Lathe OAC Operator Interface Application.

Each window of the interface is implemented as an independent client application, having it's own communication interface. In this manner operator interface components can be:

- distributed among multiple computers,
- run simultaneously from multiple sites,
- run remotely from the machine.

The interface consists of typical manual machine control elements: coordinate displays, manual axes and spindle controls, program processing control, feed and speed override, and tool data base. Other applications have been developed to demonstrate remote access to 'on line' machine data; for example a process interface that logs and displays engineering data from the running machine: spindle power, cutting force, metal removal rate, tool state, etc.

The operator interface application has demonstrated many of the features of the OAC design, and in particular the design of the OAC Machine Server. Figure 26 depicts how a typical component of the interface is implemented using the server.

Figure 27 depicts how an adaptive control module is implemented with the Machine Server. In this case, the data report facility supplies data over the communication interface at a rate specified by the client application (with a resolution of milliseconds). The client application can, for example, use the incoming data as an event

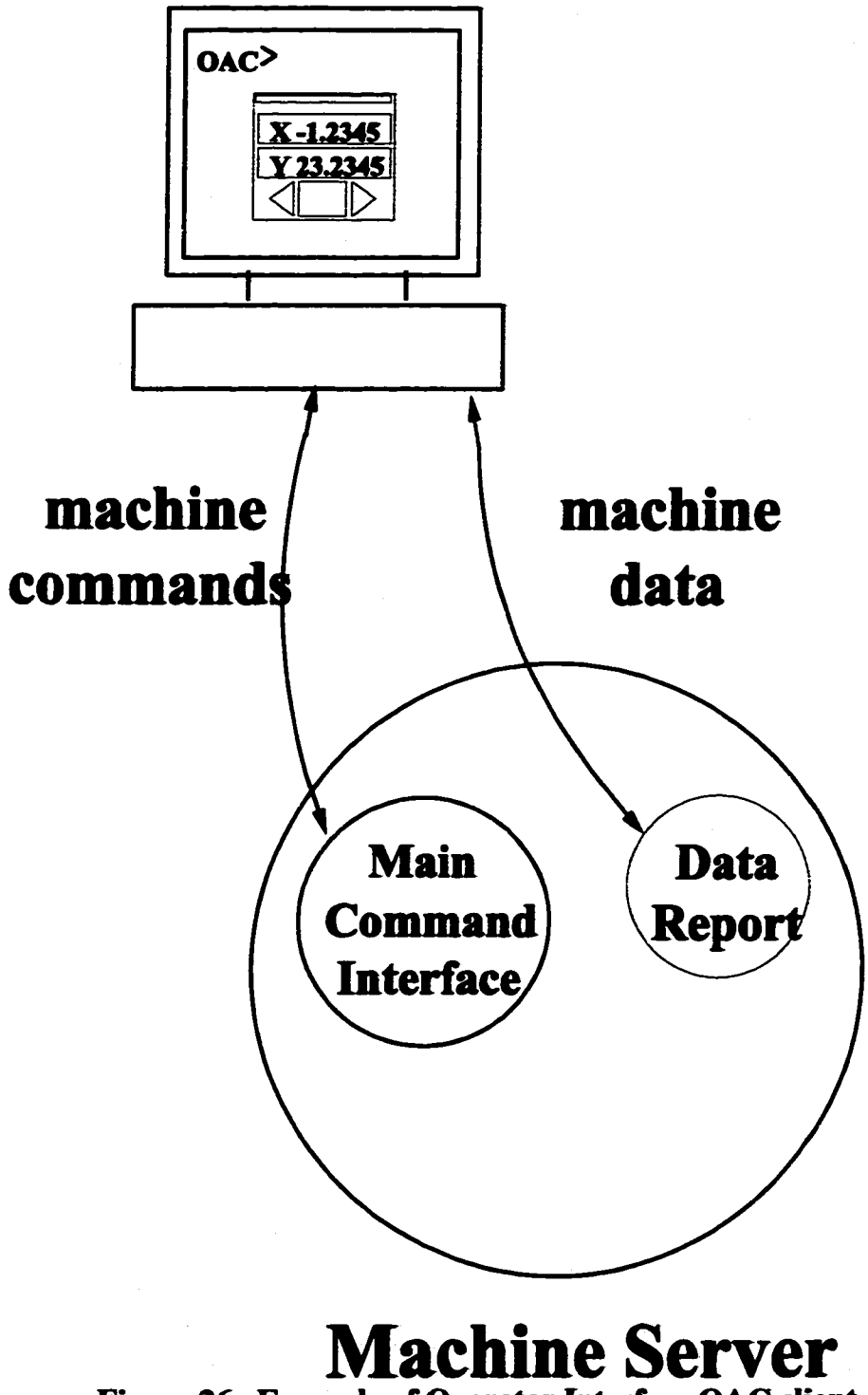


Figure 26. Example of Operator Interface OAC client.

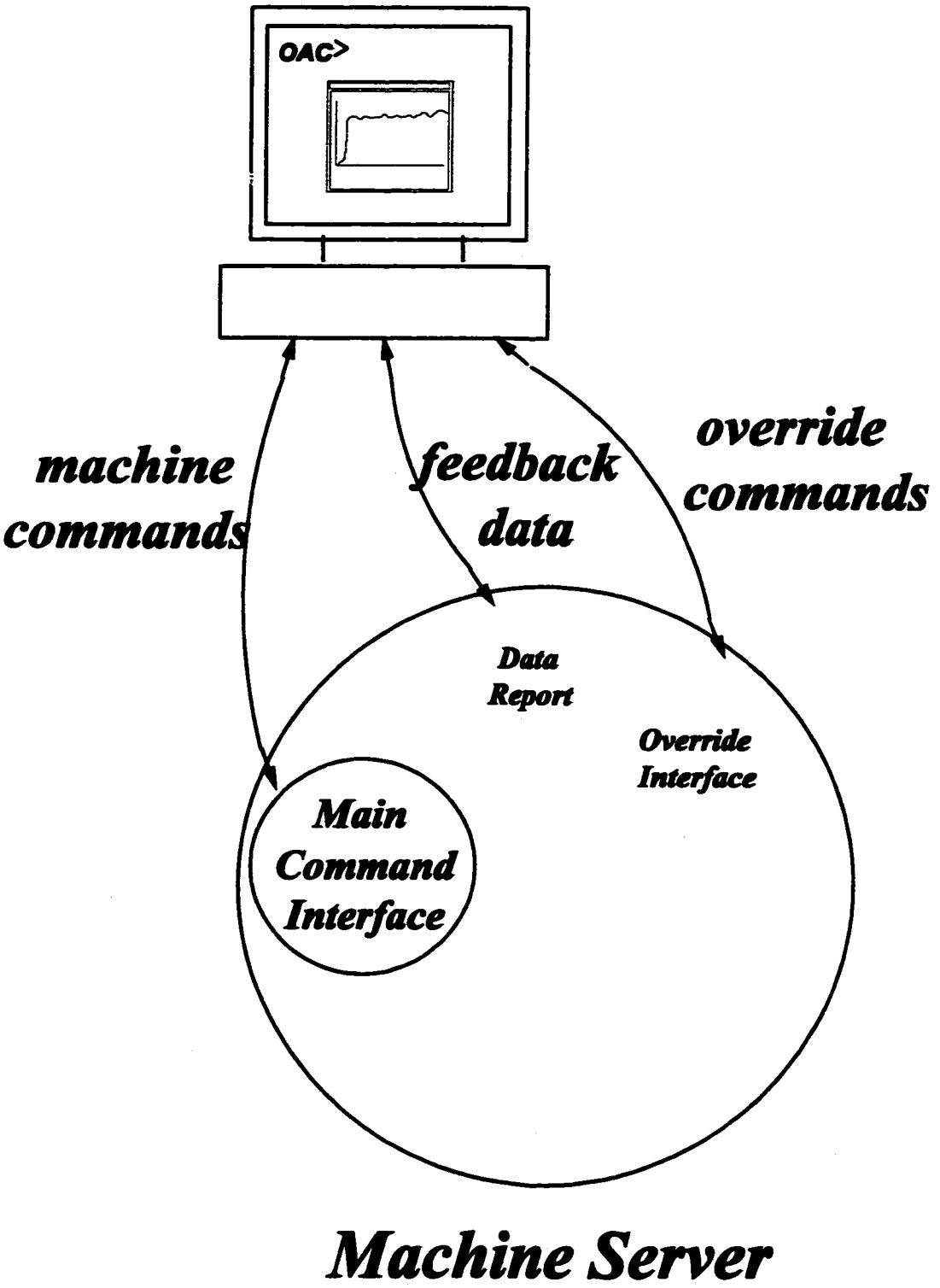


Figure 27. Example Adaptive Control OAC Client.

source to provide timing for a digital control loop. Calculations performed on the data - PID control for example - result in a desired control action that is applied through a Machine Server override interface. In this scenario, the client requires no timing mechanism, and no knowledge of the hardware interfaces required to realize the acquisition of sensor data and actuation of control actions. The example illustrates the high degree to which the Machine Server facilitates realizing the client application.

Chapter 5

INTELLIGENT MACHINING SYSTEM

5.1 System Structure

Multi pass or discrete parts machining processes have many conditions that affect their output. These conditions can include both short term and long term process behavior. Figure 28 depicts several of these factors and the approximate time frames associated with them.

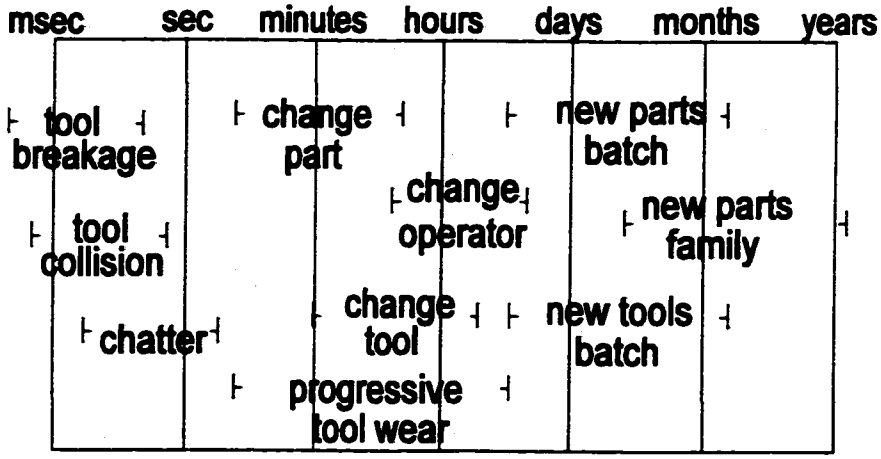


Figure 28. Factors affecting typical production machining systems and their time frames.

Accordingly, it is considered here that an 'intelligent' machining system would address the control of:

1) multiple aspects of the machining process,

2) behavior that occurs over various time intervals.

Figure 29 depicts the proposed Intelligent Machining System. Three feedback loops each address different time frames for the multi pass control problem.

In pass control ensures the safety and integrity of the current pass by addressing activities associated with time critical events: tool breakage, chatter, and cutting force control.

Per pass control combines knowledge of the current part geometry and tool state with a detailed process simulation to provide a process plan for the next pass. The process plan is generated to provide an optimized feed based on either a constant force constraint (roughing passes) or a constant surface roughness constraint (finishing passes).

Multi pass control provides a supervisory level of control for:

- coordinating lower level control tasks,
- controlling long term process behavior.

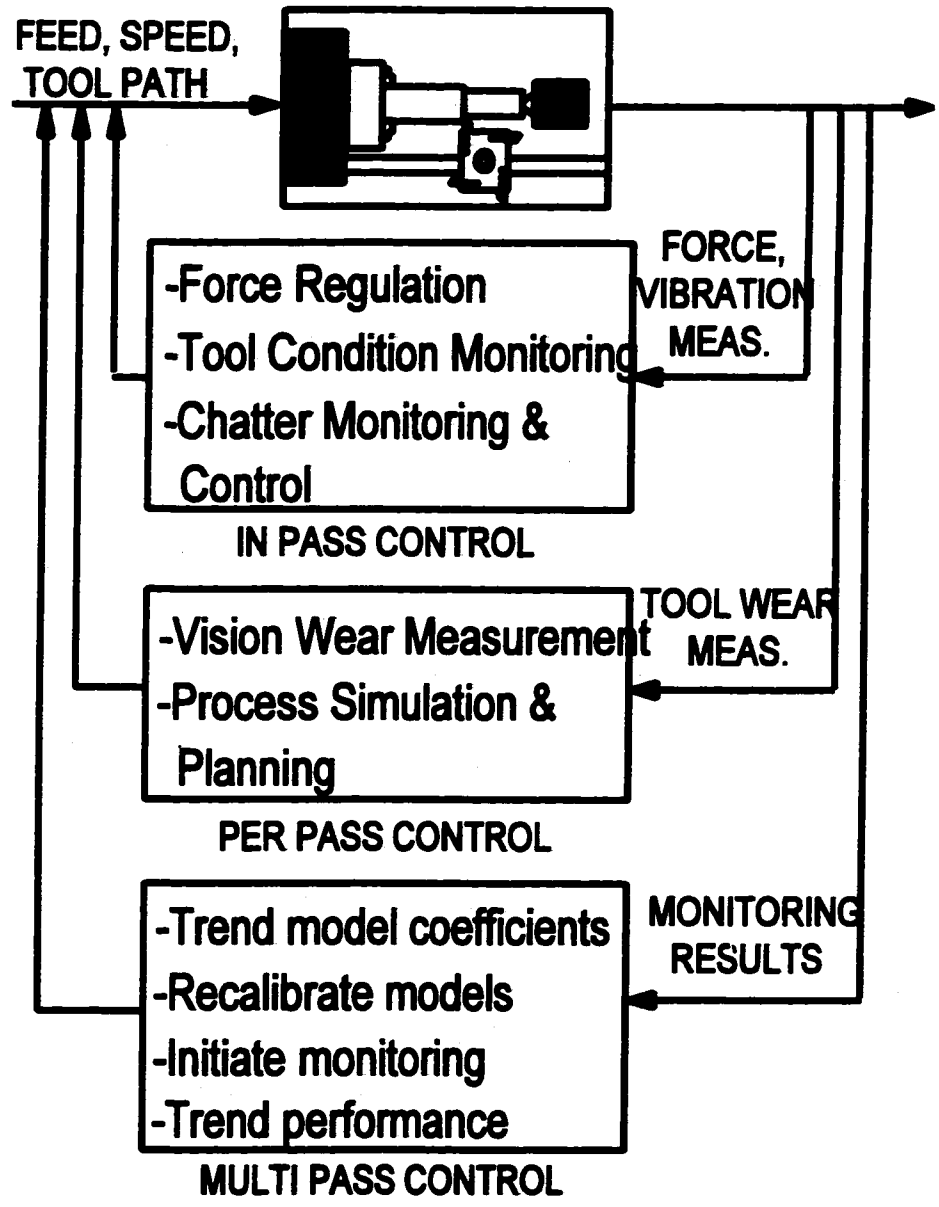


Figure 29. The proposed Intelligent Machining System

Coordination duties will involve selecting active control and monitoring tasks, managing their interactions, and defining the values of regulated process constraints. Long term control will address tool wear rate regulation and process model adaptation over slow changes in material and tool characteristics.

Each of these control levels and their associated subsystems are further described in the following sections.

5.2 In Pass Control

In pass control manages real time contingencies such as tool breakage, chatter detection and suppression, and force regulation.

5.2.1 Tool Breakage Monitoring

Tool breakage monitoring employs a technique that detects the transient in cutting force associated with breakage. The approach relies on detecting the excursion of the force signal from an envelope that represents the range of signal variation under normal cutting conditions. The technique has been reported in the literature in several forms (refer for example to Jemielniak, 1992) however, the approach used here is unique in that the envelope width is calculated recursively from the incoming force data as the 95%

confidence interval on the assumed Gaussian force variation:

$$\tilde{F}(t) = \tilde{F}(t-1) + K(F(t) - \tilde{F}(t-1)) \qquad (6)$$

$$\tilde{\sigma}(t)^2 = \tilde{\sigma}(t-1)^2 + K((F(t) - \tilde{F}(t-1))^2 - \tilde{\sigma}(t-1)^2) \qquad (7)$$

$$K(t) = \frac{1}{\lambda}(K(t-1) - \frac{K(t-1)^2}{\lambda + K(t-1)}) \qquad (8)$$

and hence the envelope values are:

$$\tilde{e}_{1,2}(t) \pm 2\sqrt{\tilde{\sigma}(t)^2} \qquad (9)$$

where,

t - time step,

F(t) - is the measured force at time t,

$\tilde{F}(t)$ - is the estimated mean value of the force,

$\tilde{\sigma}^2$ - is the estimated variance of the force signal,

K(t) - is the recursive gain of the estimation,

λ - is the 'forgetting factor' of the recursive calculations,

$\tilde{e}(t)_{1,2}$ - are the calculated envelopes about the force signal.

This approach is particularly appealing in that no a priori specification of force levels, envelope widths or cutting conditions is required. The only parameters required are a recursive forgetting factor, which is related to the effective time window length of the recursive calculations, and the time duration that the algorithm allows the excursion of the signal from its envelope before breakage is considered to have occurred. The latter of these acts as a single parameter determining the sensitivity of the algorithm. Figure 30 shows the operation of this algorithm.

5.2.2 Chatter Control

The phenomena of chatter in machining involves complex, nonlinear mechanisms that are difficult to model in most practical situations. This is due to the inherent dependance of chatter on the dynamics of the machining system, which in turning is effected by many factors that are practically difficult to control; for example work clamping forces. Consequently, many of the chatter suppression systems proposed in the literature are event driven (Teltz and Elbestawi, 1993, Hoshi et al., 1977, Smith and Delio, 1989).

If it can be assumed that the primary goal of a chatter suppression system is to allow the safe completion of an otherwise unstable process, it can be argued that the optimal strategy will be that which allows the current pass to complete safely in minimal

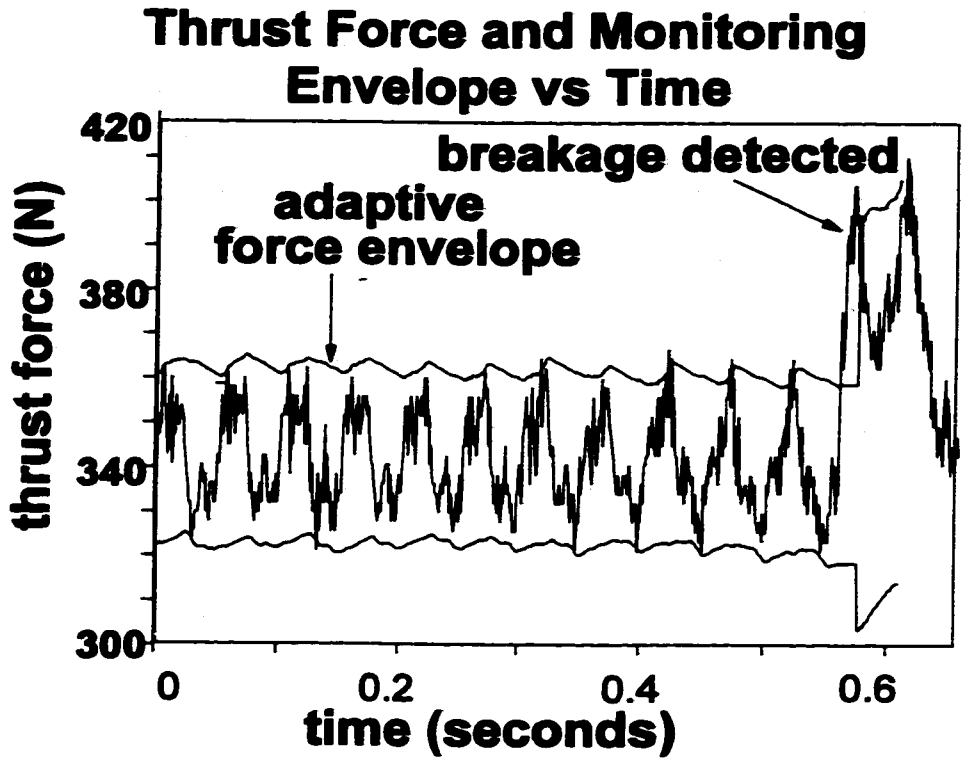


Figure 30. Operation of the Cutting Force Based Tool Breakage Algorithm.

time. It is then assumed that the process inputs are adjusted such that the input having the worst effect on machining time is modified off line. In either case, once the current pass is complete, the tool path plan is regenerated for following passes to ensure chatter free operation by planning a lower depth of cut.

For the implementation here, depth of cut is used as the primary variable to

suppress chatter once it is detected. This variable is chosen because it:

- has the greatest influence on the occurrence of chatter,
- leads directly to identifying the stable level of depth for subsequent passes.

This latter point arises from the case where multiple chatter alarms lead to multiple reductions in depth of cut before the pass proceeds. In this manner, the approach can be related to "Automatic Cut Distribution" techniques (Weck et al, 1975). Figure 31 depicts the concept.

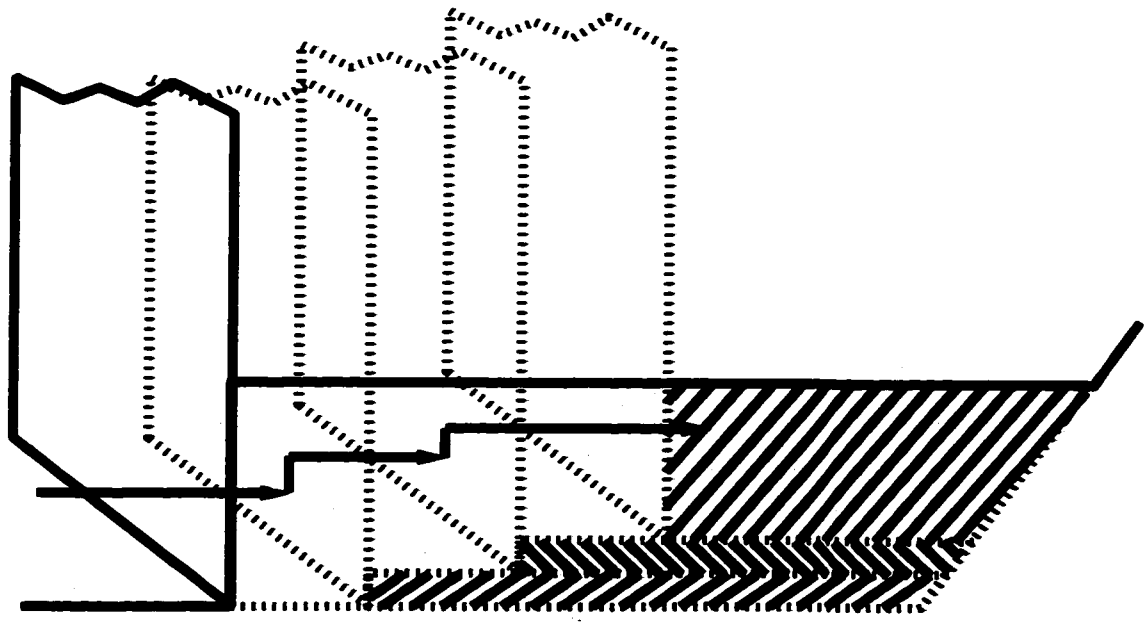


Figure 31. The Use of Depth of Cut Modification to Suppress Chatter.

The chatter control strategy was examined in a series of tests where slender work pieces were machined. Typical slenderness ratios of the work pieces used was 12:1 (length to diameter). Figure 32 shows a typical test result for a mild steel work piece with cutting conditions of speed 50 m/min and feed 0.17 mm/rev. Depth of cut reduction for this test was 50%, with an initial depth of cut of 2.5 mm. The tests demonstrated that reducing the depth of cut effectively suppressed the chatter. The test case shown in figure 32 resulted in a reduction of overall vibration level of greater than 75%.

Figure 33 shows the system responding to multiple chatter detections. Depth of cut reduction of the single pass was set for 25% of the initial depth of cut. Test conditions were similar to that shown in figure 32, with the depths of cut being 2.5 mm initially, 1.9 mm and 1.3 mm for the final stable cut.

In some cases it was observed that, at very low depths of cut, chatter occurred under conditions identical to those where chatter had not occurred at a significantly higher depth of cut. This behavior was attributed to the effects of the tool nose radius; at low depths of cut the chip load is distributed over the nose of the tool instead of the cutting edge. This can reorient the resultant cutting force in the direction of the weak structural mode of the work piece (ie: radially), thus stimulating the regenerative chatter. In practice however, the chatter suppression system will be applied to roughing cuts in which the depth of cut will generally be significantly higher than the tool nose radius.

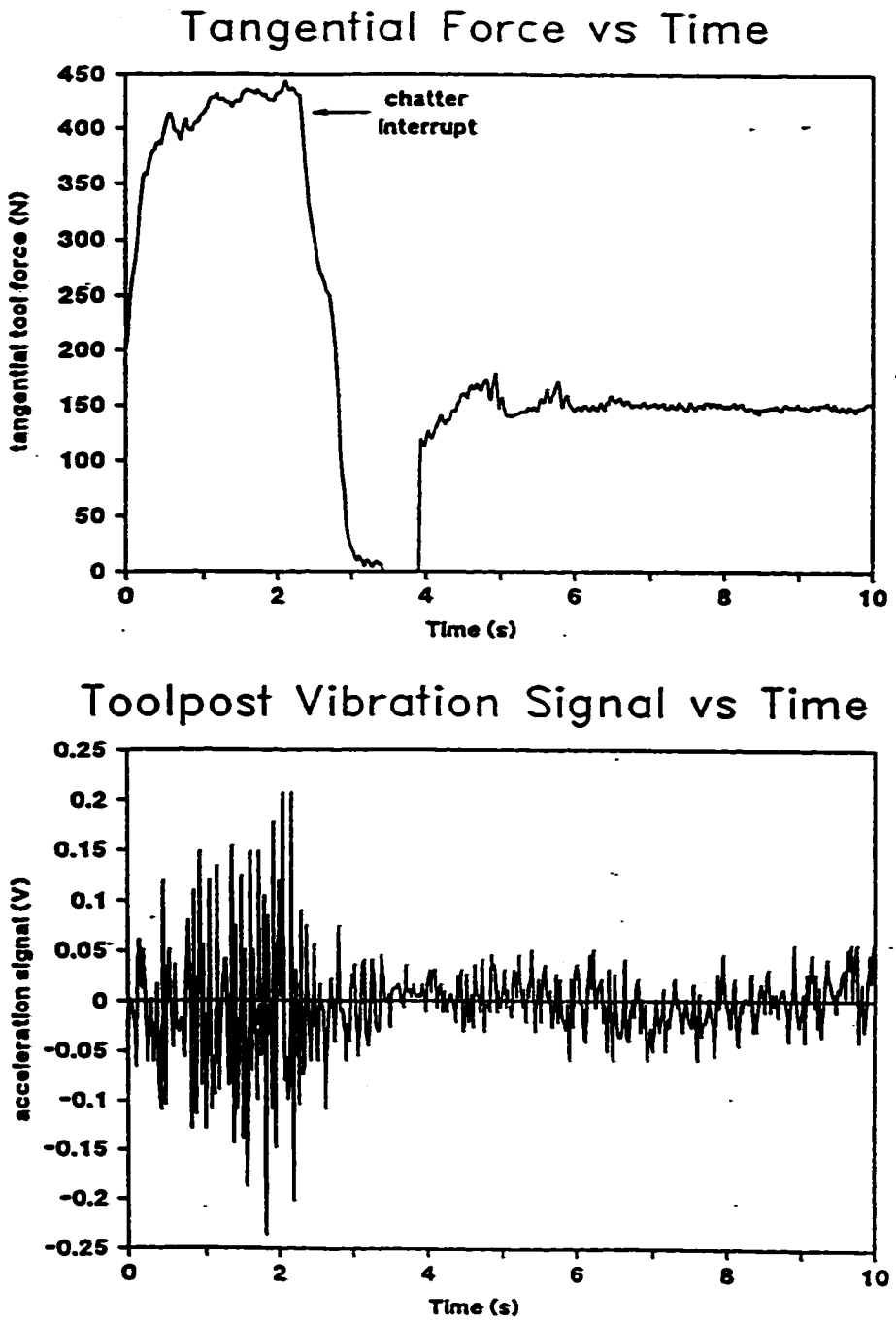


Figure 32. Chatter Suppression via On Line Reduction in Depth of Cut.

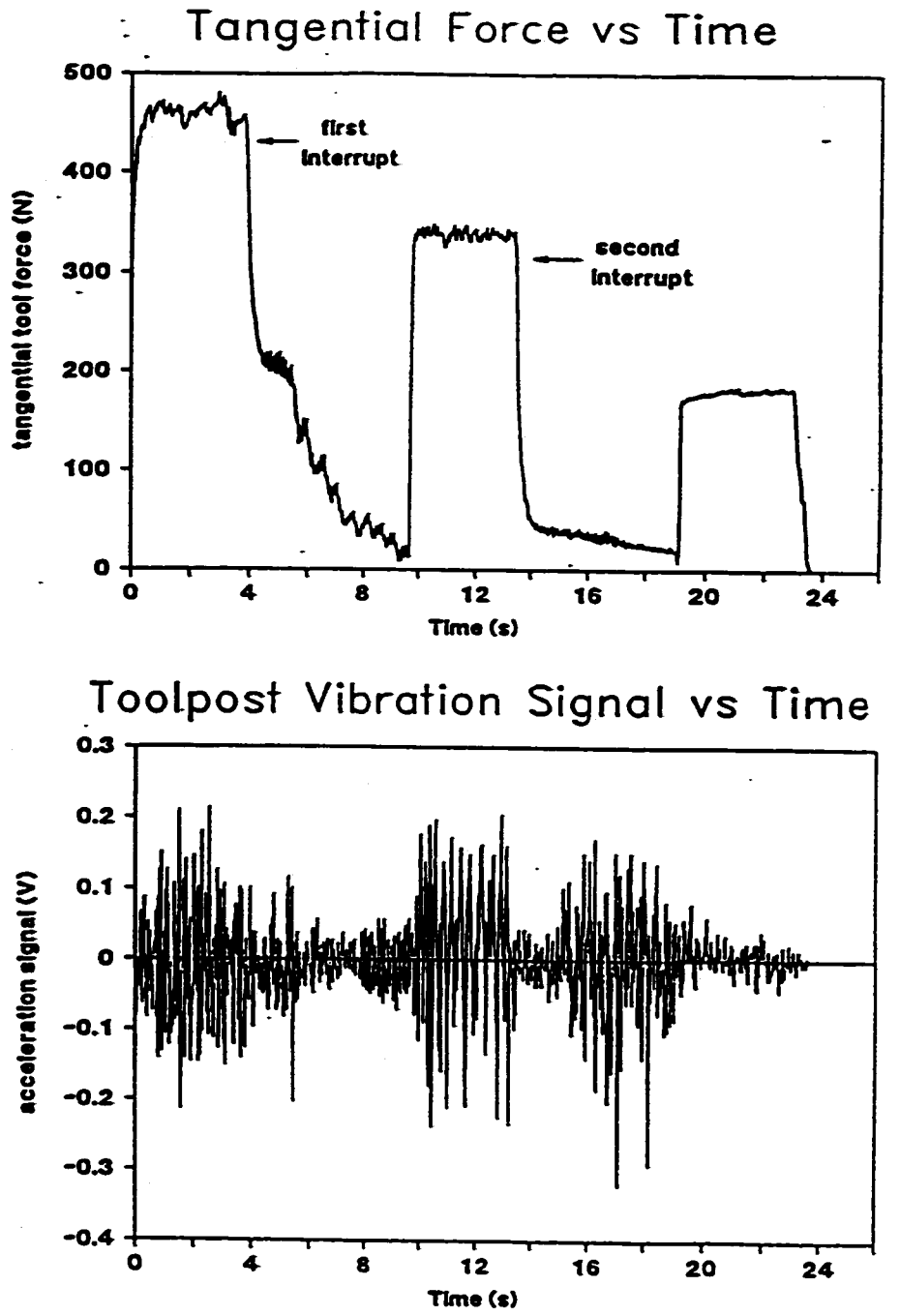


Figure 33. Chatter Suppression via Multiple Reductions in the Depth of Cut.

5.2.3 Force Regulation

Force regulation used at the in pass control level is only applied selectively to correct the planned feed for excessive modeling error. As such, the force control system can be considered to act as a supervising monitor for the planned feed, being applied only in situations where the measured force resulting from the planned feed sequence differs from the target force by a predefined margin. For this purpose, a digital low gain integral controller is used:

$$\frac{f_c(z)}{F_{err}(z)} = \frac{(K_p(z-1)+K_i z)}{(z-1)} \qquad (10)$$

where,

Ferr - target force error; tangential force, (N),

fc - is the feed command (mm per revolution),

K_p - is the proportional gain,

K_i - is the integral gain,

z - is the Z transform operator.

The cutting process dynamics are modeled as a first order system:

$$\frac{F(s)}{f_c(s)} = \frac{K_s b}{1+\tau s} \qquad (11)$$

where,

F - tangential cutting force, (N),

K_s - specific cutting force (N/mm^2),

b - is the chip width (mm),

τ - first order time constant,

s - Laplace transform operator.

For cutting mild steel with P40 grade carbide tools over a range of feeds and speeds of 1.0 - 4.0 mm and 300 - 450 m/min respectively, Ks was found to vary from 1.2 - 1.8 MN/mm2. Choosing Kp and Ki corresponding to roughly the middle of this range of Ks resulted in a robust, non aggressive control.

Figure 34 depicts the operation of such a controller. It is important to note that force regulation, at this level of control, does not attempt to regulate cutting force during

abrupt changes in part geometry. Regulation of cutting force in these conditions results from a change in the feed schedule, as planned by the per pass control level. The limited use of continuous on line force regulation in the system proposed here is made in recognition that such systems cannot adequately deal with the fast transients in force that arise due to abrupt changes in part geometry. The cut entry depicted in figure 34 illustrates this problem. The use of parameter adaptive systems, even in the best conditions, do not eliminate such transients, which are clearly unacceptable from a breakage perspective. Limited, on line feedback control is used here to address slow transients associated with, for example, part material variation. Off line planning of the feed sequences for constant force cutting addresses the fast transients that threaten the integrity of the tool and the machining system.

5.3 Per Pass Control

The per pass level of control is concerned with providing an optimized plan of cutting feed for the upcoming pass, with consideration given to the current tool state and cut geometry. The plan consists of a schedule of feeds required to realize either a constant cutting force (roughing passes), or constant surface roughness (finishing passes). Calculated feed plan data is associated to the coordinate space of the machine tool for

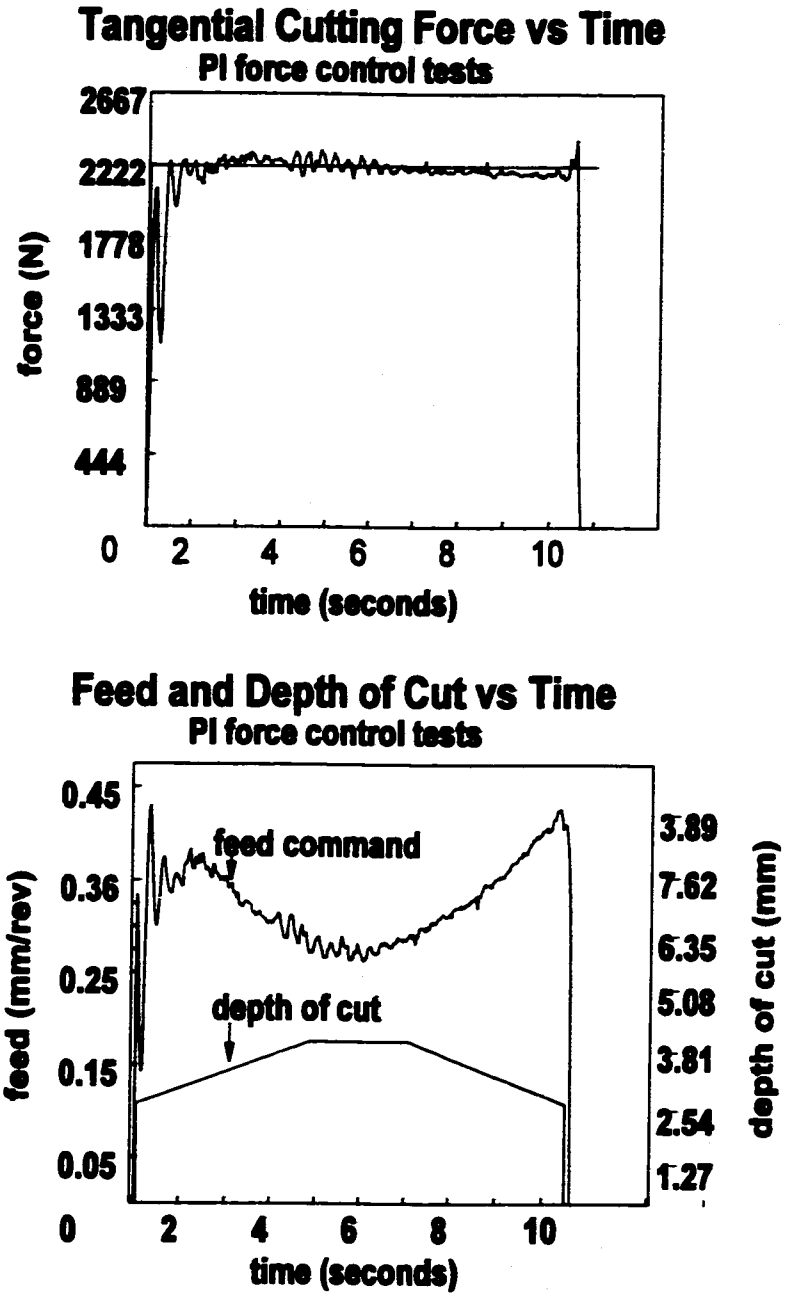


Figure 34. Operation of Proportional-Integral cutting force controller.

direct use as CNC command data.

Planning is realized through the use of process constraint models and measurements of the actual tool state. The cutting force model incorporate tool wear measurements taken by the machine vision based measurement system. The surface roughness predictions are verified by an on-line measurement system. The models are used in a simulation of the cutting pass that is solved to determine the feed required to satisfy a specified constraint value. The following describes each of these steps in more detail.

5.3.1 Cutting Force Modeling

There are several well established techniques for cutting force modeling. Semi empirical approaches have proven to be the best suited for industrial application. Such models are easily calibrated over a wide range of cutting conditions, and can be used to account for complex tool and work geometries.

For sharp tools, the three dimensional oblique cutting process is viewed in terms of an equivalent two-dimensional cutting force system in the plane normal to the rake face. The cutting forces are treated as being proportional to the area of tool / work intersection (the 'chip load' area). The tangential and normal cutting force components are given by the equations,

$$\begin{aligned} F_T &= K_T * A_c \\ F_N &= K_N * A_c \end{aligned} \qquad (11)$$

where A_c is the chip load area. K_t and K_n are semi empirical, being dependant on tool geometry, cutting conditions and tool / work material combination. They are accordingly modeled as:

$$\ln K = C_0 + C_1 \ln t_c + C_2 \ln V \qquad (12)$$

where, t_c is he effective chip thickness and V is the cutting velocity. $C_0 \ldots C_2$ are determined by multi variate linear regression using force measurements. The longitudinal and radial force components F_L and F_R are determined by resolving the normal force component F_N using the effective lead angle la_e as follows,

$$\begin{aligned} F_L &= F_N \cos la_e \\ F_R &= F_N \sin la_e \end{aligned} \qquad (13)$$

where, the effective lead angle is used to compensate for the effect of the tool nose radius on the nominal lead angle.

The chip load is taken as the projection of the chip cross-sectional area on the

plane normal to the cutting velocity. It is calculated using a geometric model of tool / workpiece intersection. Section 5.3.4 describes the implementation of these calculations.

It is well understood that tool flank wear can significantly increase cutting forces. It has been shown that the worn tool cutting force Fw can be well predicted as:

$$F_w = F_s + K V_B \qquad (14)$$

where, Fs is the sharp tool cutting force and K is an empirical constant, (Koren, 1973, 1978). This model has also been used for wear estimation under varying cutting conditions (Koren et al., 1991).

In the application here, the flank wear estimates are obtained intermittently from the machine vision measurement system, and are used to improve the overall cutting force prediction. The cutting force / flank wear relationship has been calibrated for a range of average flank wear lands using tangential cutting force measurements.

Figure 35 depicts test results for the calibrated model, shown here for a step change in depth of cut. The model has been calibrated for cutting mild steel with P20 grade carbide tools over a range of feeds, depths of cut and speeds of 0.127 - 0.508 mm/rev, 2.5 - 4.8 mm and 91 - 137 m/min, respectively. Results for worn tools correspond to flank wear lands over the range of 0.2 mm $< VB_{max} <$ 0.45 mm.

5.3.2 Surface Finish Modeling

Models for surface finish in machining generally fall into two categories. The first is concerned entirely with tool / work intersection geometry. Sometimes referred to as "theoretical surface roughness", these models consider the generation of surface on the work due to the secondary cutting edge of the tool. In this case the nose radius, effective lead angle, and feed per revolution play the largest roles in the generation of "feed marks". Figure 36 depicts the concept.

Determining theoretical surface finish involves the evaluation of a set of cases which define the particular class of geometric calculations used (Samaranayake, 1996). More sophisticated approaches use solid modeling techniques to handle the geometric calculations and can include the contributions of tool / workpiece vibration and complex tool and work geometries (eg: Imani, 1997).

The second type of modeling approach recognizes that practically the theoretical predictions for surface roughness are rarely achieved in practice. This recognizes the presence of various other aspects that can influence surface finish in machining, such as:

- work material fracture,
- material side flow,
- tool wear.

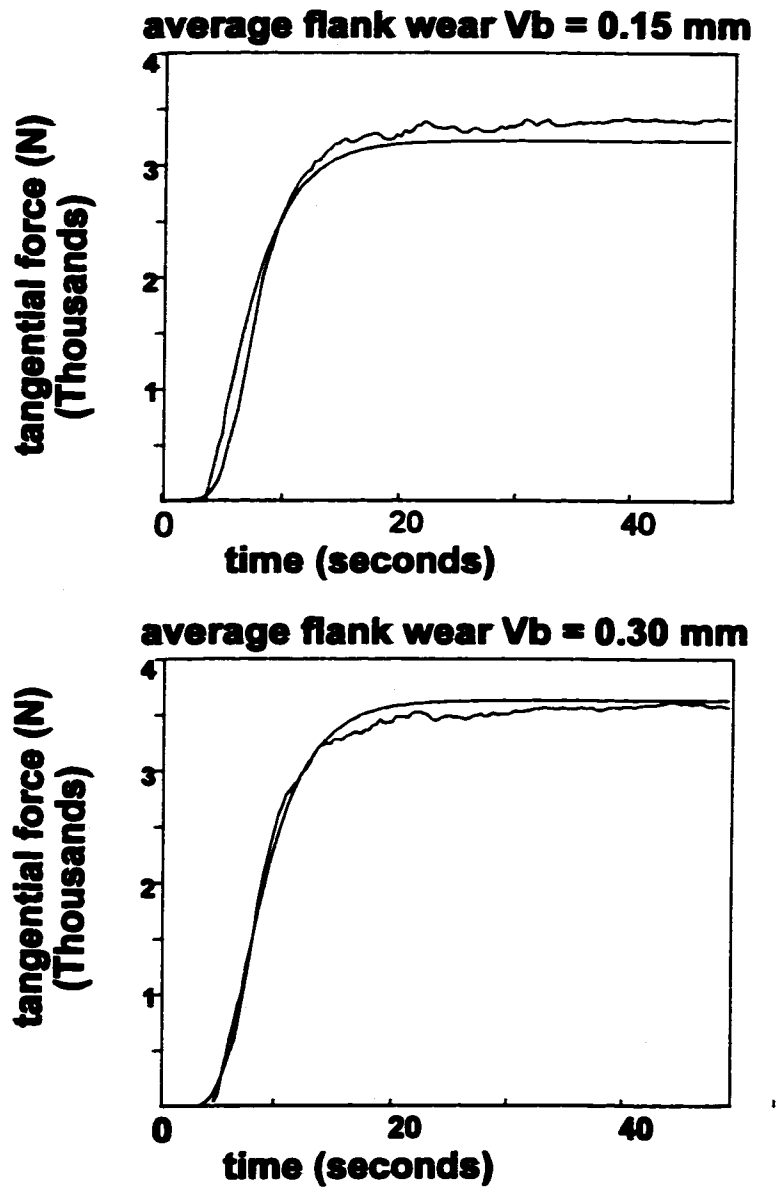


Figure 35. Test Results for the Calibrated Cutting Force Model.

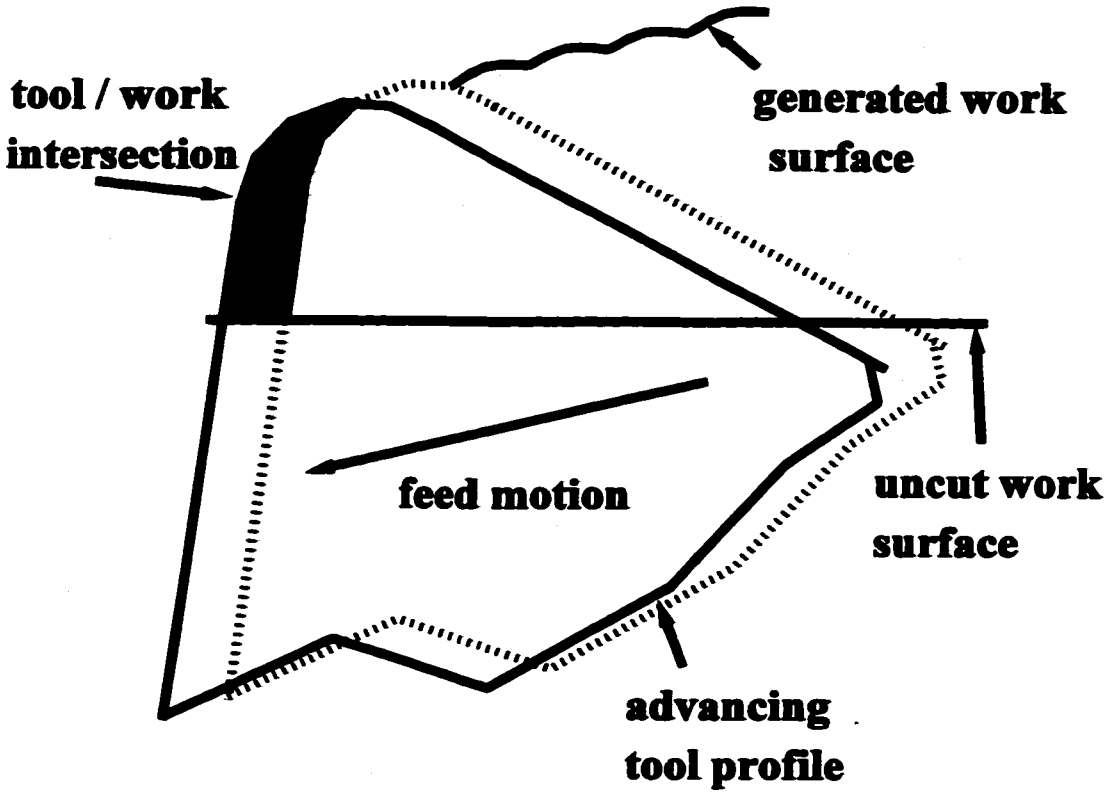


Figure 36. Generation of "theoretical" Surface Finish in Machining.

In this case, semi empirical models are predominantly used because of the inherent difficulty in quantifying a mechanistic representation of these phenomenon. Most commonly, a feature of the surface topography (eg: average or peak value) is

characterized as a function of the cutting conditions. For example, for a sharp tool with a fixed geometry, the following relationship has been used by many authors:

$$R_a = K_1 * f^a * V^b \qquad (15)$$

where,

R_a - geometric average surface roughness,

f, V - machining feed and speed,

K_1, a,b,c - are exponents, determined experimentally.

For the purpose of planning finishing feed in the Intelligent Machining System, the latter modeling approach is used. A series of cutting tests are performed for mild steel work pieces using P40 carbide tools over a range of cutting conditions: feed: 0.2 mm/rev $< f <$ 0.45 mm/rev, speed: 210 m/min $< V <$ 350 m/min. Sharp tool conditions were considered. The results of these tests are used to form a data base, in the form of a set of empirical equations, that are used by the planning system to selected the feed and speed required to generate a specified surface roughness. Figure 37 depicts typical test curves used for the prediction model. Appendix 3 contains the test results used for these models.

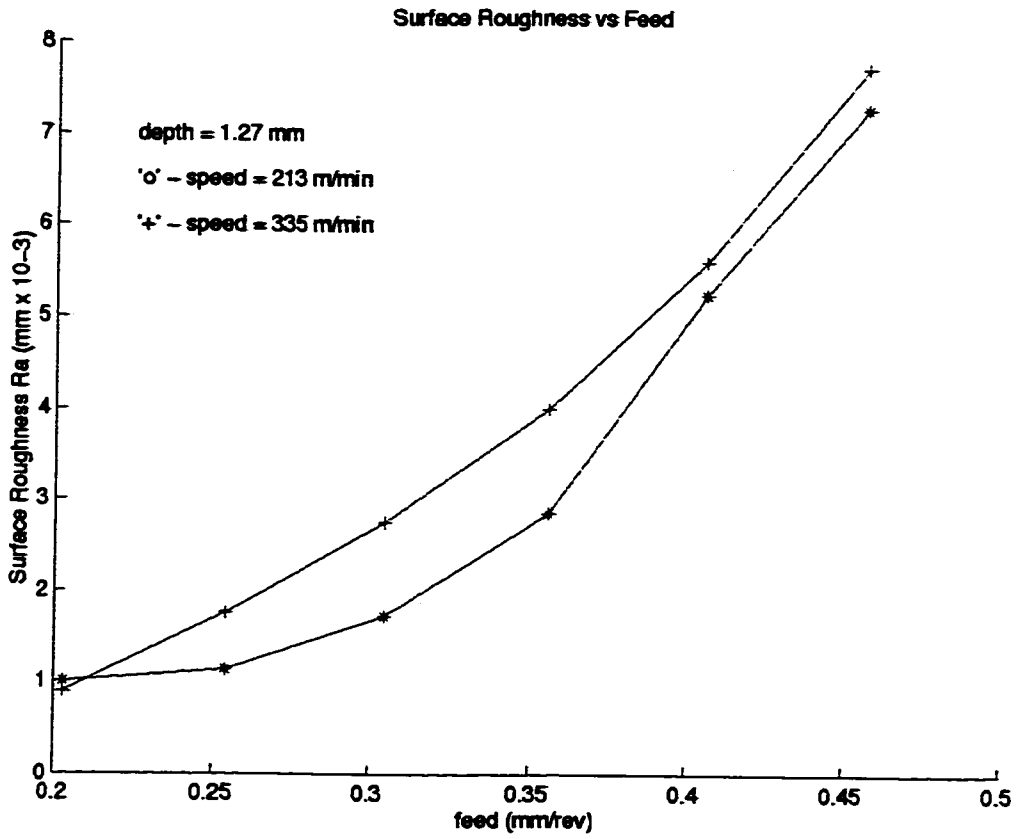


Figure 37. Example calibration curves for surface roughness model.

Samaranayake (1996) has found that the theoretical surface roughness does provide a more accurate prediction for coated tooling than for uncoated tools. It is proposed here that for such cases in which theoretical roughness applies, the empirical models may be substituted for calculated theoretical roughness values generated by a geometric simulation of the pass. Feed planning for roughing constraint will, as a by product of the chip load calculation, automatically generate such geometric surface data. In such cases the same planning software can be used for both roughing and finishing.

5.3.3 Feed Planning Solution

The objective of feed planning is to pre-determine the feed sequences required to maintain a constant value of a process constraint during cutting. Tool motion at nominal feed is specified by input G code commands. Geometric models of the part and the tool are used to determine intersection areas. Feed planning effectively replaces the input G code block with one or more output blocks having the feed adjusted to satisfy the process constraint.

For a surface roughness constraint, the calibrated surface roughness model is used to select the feed required to satisfy a specified value of average roughness R_a. This feed is held constant for the current pass.

For cutting force constraint, the feed planning solution is somewhat more complex, requiring a time scheduled solution. The feed solution is determined iteratively by simulating the cutting force (tangential component) for a set of input conditions and adjusting the feed until the predicted force matches the desired force. This iterative solution is required due to the non linearity of the cutting force model: the chip area is a non-linear function of the feed, and the specific cutting resistance Kt is a function of feed and cutting speed as described by equations 11, 12, 13. A simple secant method search is used.

For a G code block in which the depth of cut and velocity are constant (ie: bar turning), a single G code is produced with the adjusted feed. However, during contouring operations in which depth of cut and velocity are changing continuously, the

initial block must be subdivided into smaller blocks within which the depth of cut and velocity can be approximated as being constant. The feed solution is performed for each of these 'sub blocks', and a series of G codes blocks result. The selection of sub block size is determined from the geometric description of the cut such that the variation in depth of cut within the sub block is less than five percent of the total width of cut.

Figures 38 and 39 demonstrate the results obtained with feed scheduling for linear and circular interpolated cutting, at a cutting speed of 350 m/min. The results indicate that the feed schedules produced are reliable within the accuracy of the cutting force model (within approximately 10%). Other factors that influence the performance of the feed scheduling systems include:

- the accuracy of the workpiece geometric model,
- the amount of variation in feed between scheduled 'sub blocks'.

The impact of the latter factor will depend on the motion control system's ability to change vector feed rate between commanded motion blocks. This in turn is a manifestation of the acceleration abilities of the drive system and the axes configuration during the move in question (ie: during contoured motion).

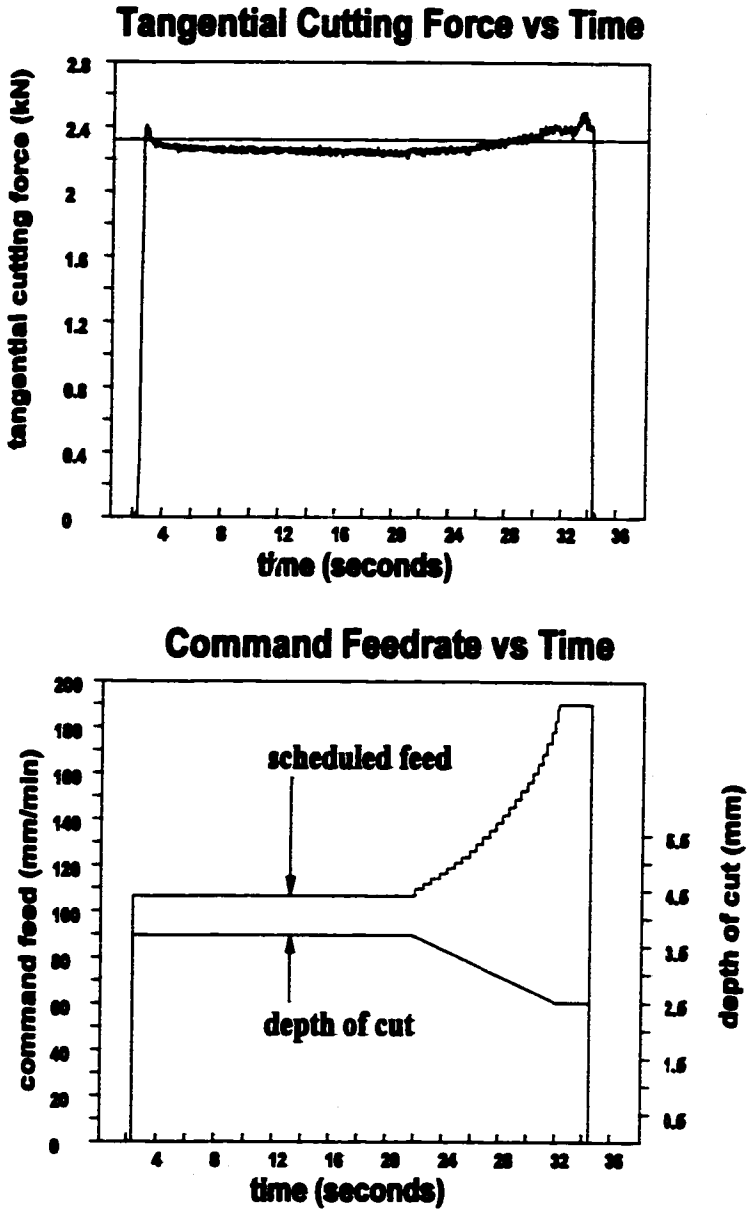


Figure 38. Feed Scheduling Results for Linearly varying Depth of Cut.

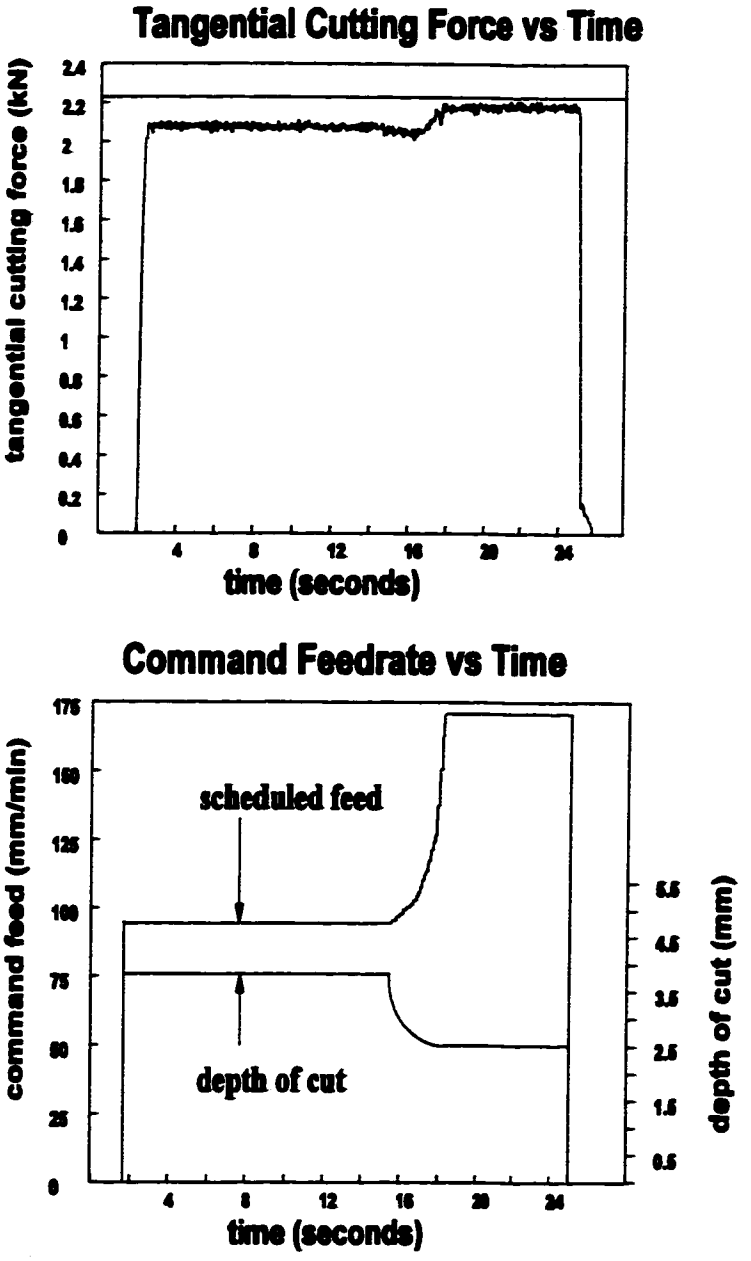


Figure 39. Feed Scheduling Results for Circular Variation in Depth of Cut.

5.3.4 Implementation of the Per Pass Control Module

The feed planning system used for the Per Pass control is implemented as an application on a UNIX workstation. The software consists of C language algorithms for the numerical constraint model (cutting force and surface roughness), and geometric modeling algorithms for the chip load area calculations. The latter of these is supplied by commercial solid modeling libraries (ACIS, by Spatial Technology Inc.). The software is implemented using X Windows for the user interface. Figure 40 shows the user interface.

The Per Pass control module includes code that implements a "server" interface using IP. This allows other applications to drive the feed planning system by sending commands to the planning system specifying:

- part, tool, and constraint data files,
- active constraints (ie: force or surface roughness),
- motion code blocks to process.

The output of the planning system consists of modified G code text that is sent directly to the OAC Machine Server for execution. The integration of the feed planning module in the overall Intelligent Machining System is described in more detail in section 5.4.4.

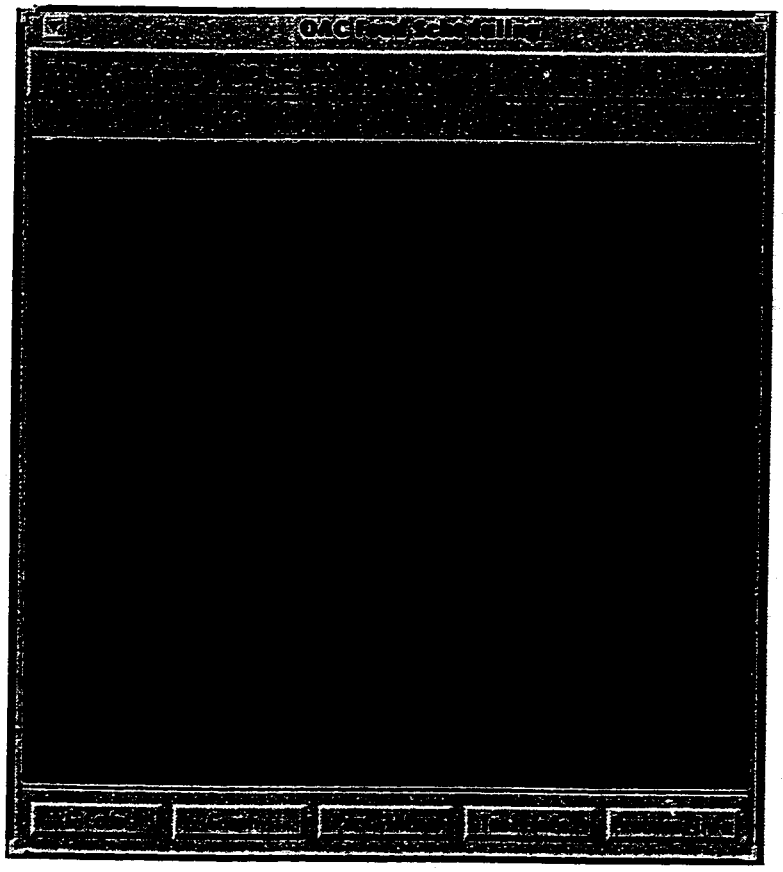


Figure 40. Implementation of the Feed Planning System.

5.4 Multi Pass Control

Multiple pass control addresses two objectives:

- coordination of control tasks,
- modeling and control of long term processes.

To achieve these, the controller must be able to assess and react to the state of the system as a whole. Numerical data and logical data must be processed to determine actions. The proposed approach is to implement the multi pass controller as a rule based system with integrated numerical processing.

5.4.1 Coordination of Control Tasks

Activities occurring at the per pass and in pass levels of the proposed system involve:

- tool wear and surface roughness measurement cycles,
- feed planning calculations,
- monitoring and control tasks,
- tool breakage or chatter detection events.

These activities can strongly interact, depending on the particular circumstances under which they occur. In addition, the selection of active constraints and related controls and monitoring tasks will depend on the particular operations being performed at the time (ie: roughing or finishing). Coordinating the potential interactions of this collection of subsystems can become relatively complex.

As an example, consider a part requiring one roughing pass and one finishing pass. For the roughing pass, the corresponding tool is selected and it's wear land

measured. The active feed constraint will be cutting force, and the feed plan is performed accordingly, with consideration for the latest tool wear measurement. During cutting, tool breakage, chatter and cutting force monitoring will be activated. If for example, the maximum allowable cutting force is exceeded during operation, the feed plan is suspended and force regulation is enabled to complete the pass.

The multi pass controller requires a mechanism to regularly acquire system data, evaluate system state and apply corrective actions. System data will include sensor data, monitoring status data, tool motion and machine state data. Data may require numerical processing to allow it to be expressed logically to the rule based system. At each interval, the system re evaluates its rule base against the new data. Corrective actions are encoded in rule structures and are applied to the machine control, planning, and monitoring tasks of the Intelligent Machining System.

5.4.2 Control of Long Term Process Variability

Multi pass control also attempts to maintain the long term validity of process models used at the lower levels of control, and to control relatively slow processes such as tool wear rate. These tasks both require long term trending of process and tool performance, and the inference of actions from such data.

The process models used by the per pass control level for feed planning have several empirical constants. These constants will be dependant on the tool / workpiece material and may vary significantly between material batches.

Over long term operation the process model predictions can be expected to degrade. The detection of such conditions can be used to initiate the re calibration of these empirical constants using data collected during the operation of the system. For this purpose a general least squares parameter estimation can be applied to the model equations for cutting force and surface finish models used in the Per Pass control system. For surface roughness, equation 15 can be rewritten in a linear form as:

$$\ln(R_a) = \ln(K_1) + a*\ln(f) + b*\ln(V) \qquad (16)$$

In many cutting operations noticeable changes in tool wear will occur only after several cutting passes. The rate of wear is primarily affected by the cutting speeds. Using models describing the behavior of tool wear, the Multi Pass controller can select speed to achieve a desired tool wear rate. For a fixed tool and work combination, empirical curves such as those shown in figure 41 can be used. Accurate measurements of current tool wear state are supplied on demand by the vision system. They are used with the empirical models to predict the incremental wear expected under each of the cutting speeds the model data was collected under. The cutting speed having the most desired incremental wear amount is chosen and implemented (for example to achieve a specified tool life).

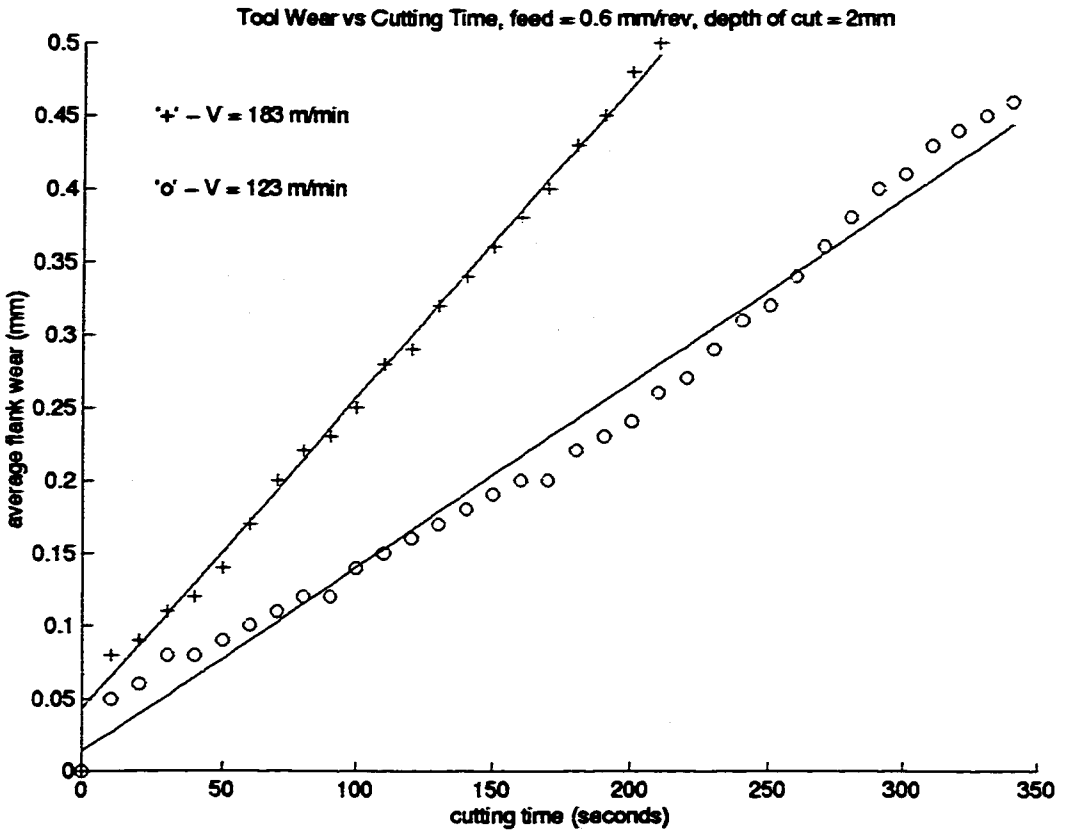


Figure 41. Example Tool Wear Curves used for Control of Wear Rate.

To practically achieve such a strategy, an extensive data base of tool wear data is required. Simple, semi empirical Taylor equations are employed of the form:

$$VB_{ave} = K * V^{a} * f^{b} \qquad (17)$$

where K, a, b are empirical constants. A series of these equations are generated for various operational conditions that are known to affect tool wear. Such data was generated in a series of cutting tests for mild steel with P40 carbide tooling. Over a range of feeds and speeds, the empirical equations were generated for different durations of cut. Figure 42 shows an example of wear plots for the case of 10 second and 5 second duration cuts. In this case, the differences in wear rate described by this data can be used to estimate the increase in wear expected for a given length of cut under specified cutting conditions.

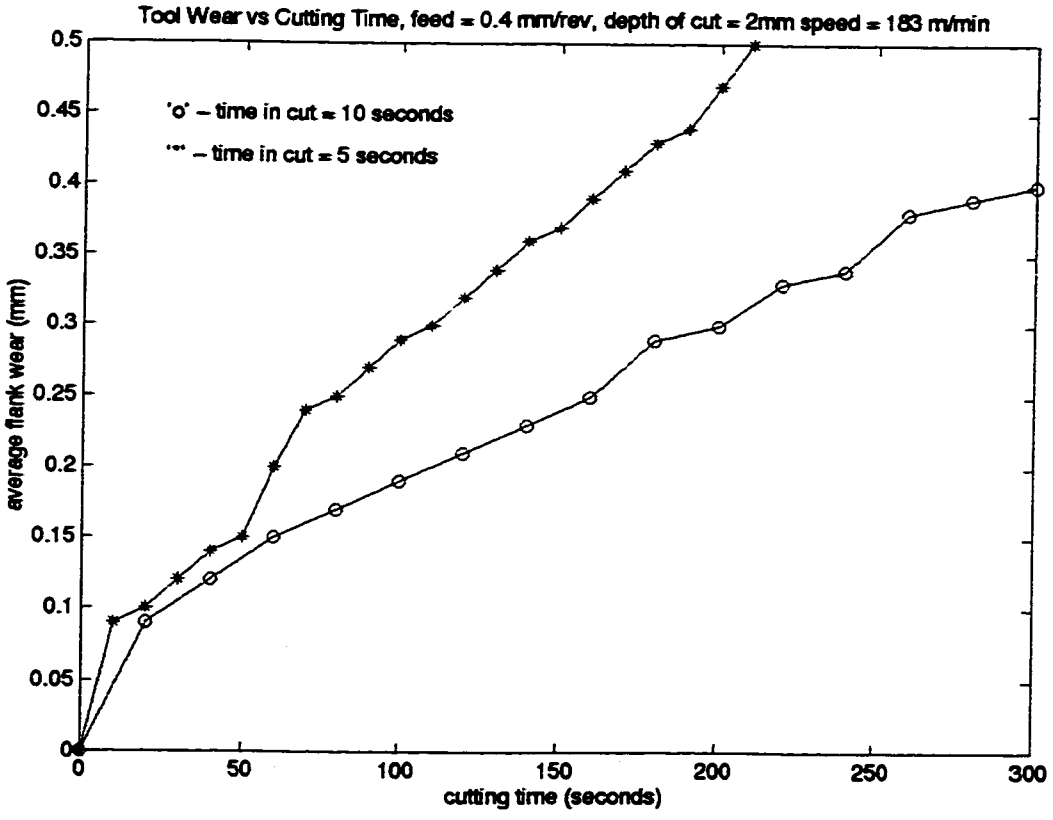


Figure 42. Example Plots of Wear for Different Durations of Cut.

5.4.3 Implementation of the Multi Pass Control Module

The Multi Pass control module is implemented using the CLIPS expert system (Giarratano, 1993). The C language based implementation allows for user written code to be added to the existing functionality of the rule based processing engine. The following functionality was added to the standard program distribution:

1) IP communication to other Intelligent Machining System modules,
2) a background update function of the CLIPS fact data base,
3) numerical algorithms to implement statistical and model functions.

The first of these items is described in the following section. The second item addresses the Multi Pass control's ability to react to changes in overall system state. A Machine Server data report function regularly sends data to an IP interface added to the CLIPS executable code. This handler for this input decodes and resets "fact" data in the CLIPS system, and triggers a re evaluation of the programmed rule base. The "firing" of programmed rules can call other user added functions that send commands over the IP interface to other Intelligent Machining System modules, or perform numerical processing of data. The latter of these allows for example, statistical features of data to be tracked by the rule based system.

5.4.4 Overall Implementation of the Intelligent Machining System

The Intelligent Machining System software is implemented as follows. The Multi Pass and Per Pass control software exist as two applications running on a UNIX workstation. The workstation is connected by a network line the CPU on the real time platform. The OAC Machine Server software executes on this CPU, in addition to modules for monitoring and control applications (tool breakage. force and chatter control), and measurement tasks (tool wear vision).

All modules communicate with one another using the IP interface. Figure 43 shows an overview of the information flow. The Multi pass controller receives data from and sends commands to the Machine Server. It enables and disables monitoring tasks and initiates measurement cycles. The Multi Pass controller sends commands to the Per Pass controller specifying tool, part, and constraint data, and the motion segments to be scheduled. The Per Pass controller feeds the Machine Server's DNC interface with the feed scheduled motion code. OAC Machine Server functions are used as follows:

- **data report** for the Multi Pass control module at 2 Hz,
- **data report** for force control task at 40 Hz,
- **override interface** for force control,
- **DNC interface** for Per Pass control feed schedule G code.
- **main command interface** for general machine control.

Monitoring and measurement tasks acquire data from memory mapped devices.

The implementation of the Intelligent Machining System illustrates some of the advantages of the OAC design. Firstly, it is clear that the provision of timing and synchronization mechanisms by the Machine Server reduces the facilitates required by clients for implementing control functions. This allows for example, a generic workstation or PC computer to be used for control applications without any additional hardware.

Secondly, from the perspective of any individual module, the interface to all other modules is identical and is hardware neutral. This provides maximum flexibility for adding new modules to the system. In addition, the distribution of processing hardware used to create the system can be altered without affecting the functionality of the system. For example, the Multi Pass and Per Pass applications could be run on individual workstations, and the monitoring, control or measurement modules could be executed on multiple PC computers. Module source code would remain the same as it is currently and moreover, the un altered modules would not be influenced by the change. This allows enhancement of existing systems without compromising existing development efforts. It also directly addresses the problem of integrating new computing hardware with older controller systems.

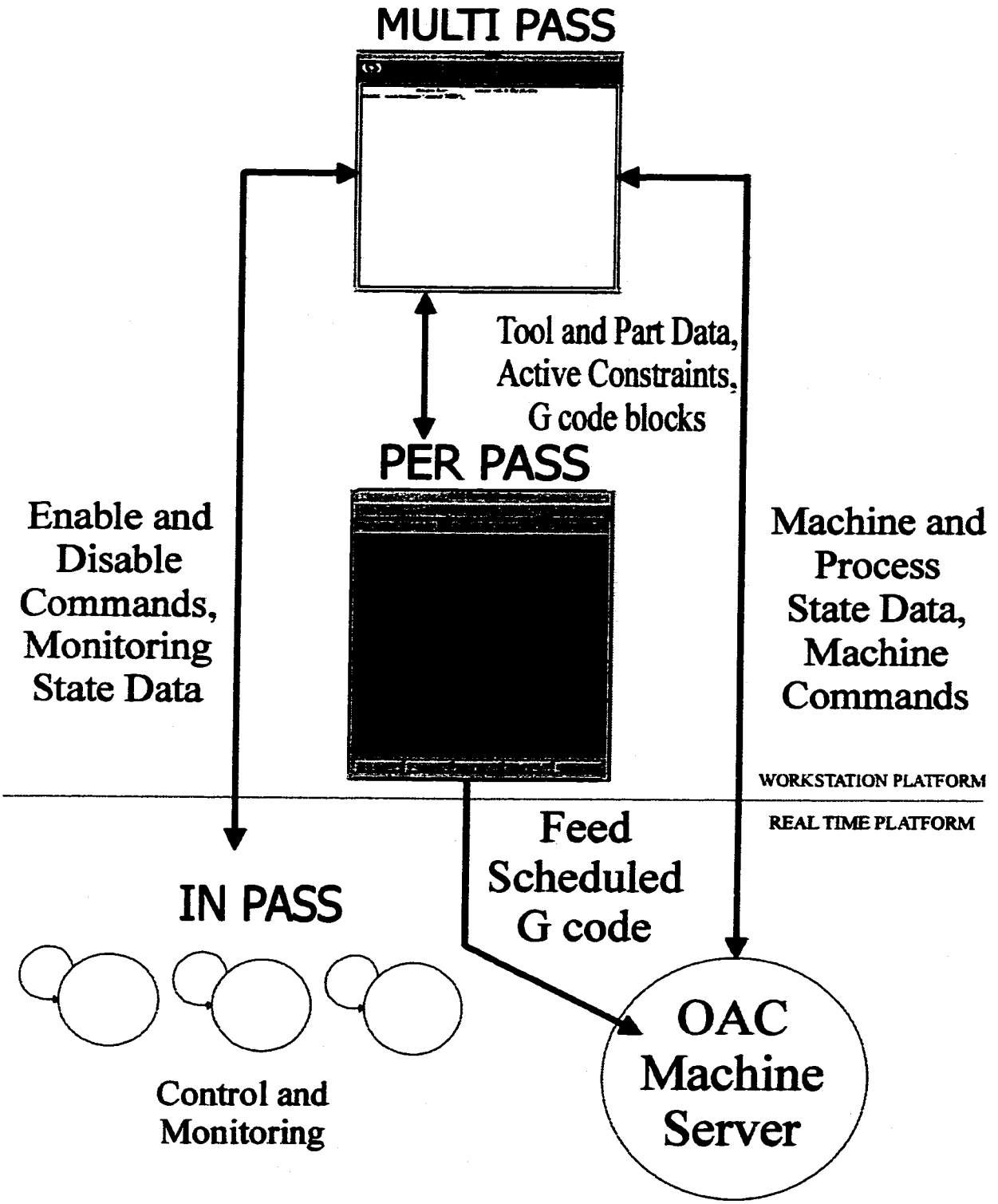


Figure 43. Overview of Information Flow among Modules of the Intelligent Machining System.

Chapter 6

CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

The research addressed the design and development a machine tool controller with the goal of facilitating the realization of Intelligent Machining Systems. As a fundamental component of achieving this goal, the use of open architecture design concepts was formulated, applied, and tested with a proposed Intelligent Machining System structure. The proposed Intelligent Machining System represented a comprehensive application of existing manufacturing technologies.

The Open Architecture Control design satisfied the requirements of the Intelligent Machining System. Moreover, its design is in no way specific to the application, and thus it will support a virtually limitless combination of system components. Its design and implementation is the main tangible outcome of this research.

6.1 Contributions of the Research

This research addressed the development and definition of a general framework for automated, technologically optimized machining systems. The formulation of an open architecture machine controller is the fundamental contribution towards realizing this goal.

Previous research and commercial implementations of open architecture machine tool control has lacked focus in two particular areas:

1) the use of open systems concepts,

2) the application of these concepts, in a general sense, to the needs of advanced machining and manufacturing systems.

The approach taken in this research has attempted to unify these two concerns in a manner not demonstrated yet by either of industry or academia. The work developed has in the authors opinion, achieved this goal and moreover provided the clearest demonstration of the enabling potential of Open Architecture Machine Tool Control technology.

Another major contribution of the research is the formulation of the Intelligent Machining System. The system proposed demonstrates the integration of key elements of advanced manufacturing sciences:

- process monitoring and control,
- process simulation and planning,
- supervisory control.

These elements were used in a framework for automated control of multiple aspects of the machining process. The structure of this framework considered several critical issues including:

- economic improvement of roughing and finishing metal cutting processes,
- the relative impact of different process factors,
- the time frames and variability of different process factors,
- available measurement technologies.

The research demonstrated and confirmed the ability to realize such a system. Furthermore, by exploring the interactions and requirements for realization in a combined form, the design requirements of the OAC system were better defined.

. The OAC system proposed by this research is unique in that it constructs the manufacturing control system using a relatively loosely coupled model. It is in this manner that a high degree of hardware independence - and hence "openness" - is achieved. Previous researchers (for example Altintas, 1994) have demonstrated Intelligent Machining technologies using "Open Architecture" Controllers. These have

generally been more tightly coupled systems, based on a particular hardware and / or operating system platform.. The research described in this thesis makes a significant contribution by realizing and examining the combination of a loosely coupled, highly open system and a process control system in which performance is critical. The results demonstrate that the demands of Intelligent Machining applications can be satisfied under the more open system formulation. This important result, along with continuing evolution of computing and Internet technologies, confirms that true "Information Revolution" benefits can be achieved in the manufacturing realm.

6.2 Direction for Further Work

There are several areas of further research that can follow from this work. Firstly, regarding the OAC design it is apparent that the communications system used in his work has limitations in overall bandwidth and moreover, it's suitability for real time applications. This issue is recognized as the key technological aspect of OAC systems requiring attention at this time. Research currently underway in Europe (OSACA) has focused a great deal of effort on developing a deterministic communications interface for OAC modules that incorporates the multitude of existing de facto standards used in factory systems integration.

Another fundamental issue requiring research is the definition of a uniform - and widely accepted- interface to machine motion and input / output devices. This issue is

important because it is these aspects of the overall manufacturing system the represent the most closed (or hardware / vendor specific) areas.

In the research presented here, the Machine Server module defined this interface through its "service model". The definition of a service model in a general, yet concise manner for a wide range of machining systems remains a difficult task. Ultimately, the adoption of any such description by hardware and software vendors will determine its overall impact in industry.

There are also issues related to security that require some resolution with the existing OAC system. Some form of access control and fault tolerance mechanisms will be required for practical implementations of the OAC. These are largely development - and not research - problems.

Regarding the Intelligent Machining System, there are many possibilities for application development in other machining and / or manufacturing systems: milling, robotics, flexible manufacturing applications. It is expected that for most of these applications, the majority of work will involve the selection of proper process models, measurement technologies, and knowledge formulation for the task at hand. The framework presented here for intelligent control may not apply readily to non machining systems. It is expected however, that the proposed OAC system will support a broader range of applications, not defined by the proposed intelligent system structure.

REFERENCES

Albus, J. , H.G. McCain, R. Lumia, 1989, "NASA / NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), Technical Report 1235, National Bureau of Standards, Gaithersburg, MD, USA.

Albus, 1994, "A Reference Model Architecture for Intelligent Machine Systems", Proceedings of the International Workshop on Open Architecture Controllers for Automation, Ann Arbor MI, April, 12, 1994, pp 115-132.

Altintas, Y., N. Newel, M. Ito, 1993, "A Hierarchical Open Architecture Multi-Processor CNC System for Motion and Process Control", ASME International Mechanical Engineering Congress, PED-Vol 64, pp 195-201.

Altintas, Y., W.K. Munasinghe, "A Hierarchical Open Architecture CNC System for Machine Tools", (1994), Annals of the CIRP, Vol.43., No.1, pp 349-354.

Balio, C., G. Alderson, J. Yen, 1994, "Requirements of Open, Modular Architecture Controllers for Applications in the Automotive Industry, revision 1.1", General Motors Corporation, Presented at the International Workshop on Open Architecture Controllers for Automation, Ann Arbor MI, April, 12, 1994.

Birla, S., J. Park, Z. Pasek, A.G. Ulsoy, Y. Shan, Y. Koren, "An Open Architecture Testbed for Real Time Monitoring and Control of Machining Processes", 1995, Proceedings of the American Control Conference, Seattle Washington, June 1996, pp 200-204.

Carlsson, T.E., Strand, F., 1992, "A Statistical Model for Tool Life as a Basis for Economic Optimization", Annals of the CIRP, Vol.41, No.1, pp 79-82.

Catellote, G.P., S. Schnieder, 1994, "The Network Data Delivery System: A Real Time Connectivity System", Proceedings of the IEEE International Conference on Robotics and Automation, IEEE Press, pp 342-355.

Chiang, S.T., D.I. Liu, A.C. lee, W.H. Chang, 1995, "Adaptive Control Optimization in End Milling using Neural Networks", International Journal of Machine Tools and Manufacturing, Vol 35, No 4, pp 637-660.

Chryssolouris G., Guillot, M., Domroese, M., 1988, "A Decision Making Approach to Machine Control", Trans. ASME Journal of Eng. for Ind., Vol 110, No 4, pp 397-398.

Dan, L., J. Matthews, 1990, "Tool Wear and Failure Monitoring Techniques for Turning - A Review", International Journal of Manufacturing Systems, Vol 30, No 4, pp 579-598.

Dornfeld, D.A., 1992, "Monitoring of Machining Processes - Literature Review", presented at the CIRP STC "C" meeting, Paris, January, 1992.

Dornfeld, D.A., P.K. Wright, 1995, "Intelligent Machining: Global Models, Local Scripts and Validations", Transactions of the north American Manufacturing Research Institution of the SME, Vol. XXIII, 351-356.

Elbestawi, M.A., Y. Mohammed, M. Lui, 1991, "Application of Some Adaptive Control Algorithms in Machining", J. Dyn. Sys. Meas. and Control, Trans. ASME, Vol 112, pp 611-617.

German Machine Tool Manufacturers (VDW), 1991, "SERCOS Interface: Digital Interface for Communication Between Control and Drives for NC Machinery".

Giarratano, J, 1993, "CLIPS 6.0 Programming Manual", NASA Software Technology Branch.

Greenfeld, I., P.K. Wright, 1989, "A Generic User-Level Specification for Open System Machine Controllers", ASME Winter Annual Meeting, PED-Vol 37, pp 63-76.

Greenfeld I., Hansen F., Fehlinger J., Pavlakos E., 1989, "MOSAIC, System Description, Specification and Planning", Technical Report No.201, New York University, Dept. of Computer Science, Courrant Inst. of Mathematical Sciences.

Hansen, F., E. Pavlakos, E. Haffman, T. Kanade, R. Reddy, P. Wright, 1993, "PARES: A Prototyping and Reverse Engineering System for Mechanical Parts on Demand on the National Networks", Journal of Manufacturing systems,

Vol 12, No 4, pp 269-281.

Hatamura, Y., T. Nagao, M. Mitsuishi, M. Nakao, 1995, "Actual Concept Design Process for Intelligent Machining Centers", Annals of the CIRP, Vol 44, No 1, pp 123-128.

Haynes, T., (1994), "The NCMS Low End Controller Project", Proc. of the International Workshop on Open Architecture Controllers for Automation, Univ. of Michigan, pp 143-160.

Imani, B.M., 1997, "Model Based Die Cavity Machining Simulation Methodology", PhD Dissertation, Department of Mechanical Engineering, McMaster University.

Jemielniak, K., 1992, "Detection of Cutting Edge Breakage in Turning", Annals of the CIRP, Vol 41, No 1, 1992, pp 97-100.

Koren, Y., Z. Pasek, A.G. Ulsoy, P.K. Wright, 1996, "Timing and Performance of Real Time Open Architecture Controllers", ASME International Mechanical Engineering Congress, DSC-Vol 58, pp 283-290.

Koren, Y., and Masory, O., 1981, "Adaptive Control with Process Estimation", Annals of the CIRP, Vol.30, No.1, pp 373-376.

Koren, Y., Ko, T., Ulsoy, G., Danai, K., 1991, "Flank Wear Estimation Under Varying Cutting Conditions", Trans. of the ASME Journal of Eng. for Ind., Vol.113, No.2, pp 300-307.

Lundholm, T., Bergstrom, D.E., Harder, L., Lindstrom, M.N., Nilsson, B.,

1992, "New Techniques Applied to Adaptive Controlled Machining", Robotics and CIM, Vol.9, No.4/5, pp 383- 389.

Micheletti, G.F., W. Konig, H.R. Victor, 1976, "In Process Tool Wear Sensors for Cutting Operations", Annals of the CIRP, Vol 25, No 1, pp 483-496.

National Centre for Manufacturing Sciences, 1990, "Next Generation Workstation / Machine Controller (NGC) Requirements Definition Document", NCMS.

North American MAP/TOP Users Group, 1988, "Manufacturing Automation Protocol (MAP) Specification v3.0", Ann Arbor MI.

O'hare, G.M., 1990, "Distributed Artificial Intelligence: An Invaluable Technique for Development of Intelligent Manufacturing Systems", Annals of the CIRP, Vol 39, No 1, pp 485-488.

Onetsu, S., Inasaki, I., Kijima, T., 1978, "Optimisation of Turning Operations", Proc. of the Sixth NAMRC, pp 17-23.

Park, J., S. Birla, K.G.Shin, Z.J. Pasek, G. Ulsoy, Y. Koren, (1996), "An open architecture testbed for real time monitoring and control of machining processes", Proc. of the American Control Conference, pp 200-204.

Pritchow, G., 1990, "Automation Technology - On the way to an Open System Architecture", Robotics and CIM, Vol 7, No.1-2, pp 103-111.

Pritchow G., Daniel Ch., Junghans G., Sperling W., 1993, "Open System Controllers - A Challenge for the Future of the Machine Tool Industry", Annals of

the CIRP.

Pritchow, G., G. Junghans, "General Overview of the OSACA Project and The OSACA Architecture", Proceedings of the International Workshop on Open Architecture Controllers for Automation, Ann Arbor MI, April, 12, 1994, pp 15-70.

Rossol, L., "NOMAD Open Architecture Motion Control Software", 1993, Proc. Int. Robots and Vision Automation Show and Conf., pp 11-13.

Samaranayake, P., 1996, Phd. Dissertation, University of Melbourne, Austrailia.

Sexton, D., D. Capson, R. Teltz, 1996, "Automatic Tool Wear Monitoring Using Machine Vision", Proceedings of the 13th Symposium on Engineering Applications of Mechanics, Manufacturing Science, and Engineering, Canadian Society of Mechanical Engineering, McMaster University, May 1996, pp 161-167.

Shawky, A., M.A.Elbestawi, (1996), "Model -Based Predictive Control of Workpiece Accuracy in Bar Turning", ASME Journal of Manufacturing Science and Engineering, Vol 120, No 1, pp 57-67.

Sperling, W., and P. Lutz, 1997, "Designing Applications for an OSACA Control", ASME International Mechanical Engineering Congress, MED Vol 6-1, pp 91-95.

Stute, G., N. Kapajiotidis, 1965, "Integration of Adaptive Control Constraint (ACC) into CNC", Annals of the CIRP, Vol 24, No 1, pp 411-415.

SUN Microsystems, (1988), "Sun Microsystems Network Programming Manual", rev.A.

Takata, S., 1993, "Generation of a Machining Scenario and its Applications to Intelligent Machining", Annals of the CIRP, Vol.42, No.1, pp 531-534.

Tarng, Y.S., Y.S.Wang, 1993, "An Adaptive Fuzzy Controller for Turning Operations", International Journal of Machine Tools and Manufacturing, Vol 33, No 6, pp 761-772.

Teltz, R., and Elbestawi, M.A., 1993, "Hierarchical, Knowledge Based Control in Turning", Trans. ASME Journal of Dyn. Sys. Meas. Control, Vol.122, pp 122-131.

Tlusty, J., G. Andrews, 1983, "A Critical Review of Sensors for Unmanned Machining", Annals of the CIRP, Vol 32, No 2, pp 563-573.

Tonshoff, H.K, J.P. Wulfsberg, H.J. Kals, W. Konig, 1998, "Developments and Trends in Monitoring and Controls of Machining Processes", Annals of the CIRP, Vol 37, No 2, pp 611-621.

Tsai, S.M., Eman, K.F., Wu, S.M., 1984, "Chatter Suppression in Turning", Proc. of the 12th NAMRC, pp 399-402.

Van Houten, F.J., and Tiemersma, J.J., 1989, "An Adaptive Control Module for Round - Improvement of Cutting Force Modelling by the use of Process Monitoring Data", Manuf. Systems, Vol.18, No.1, pp 53-66.

Weck, M., E. Verhaag, M. Gather, 1975, "Adaptive Control for Face Milling with Strategies for Avoiding Chatter Vibrations and for Automatic Cut Distribution", Annals of the CIRP, Vol 24, No 1, pp 405-409.

Wright, P.K., E. Pavlakos, F. Hansen, (1991), "Controlling the Physics of Machining on a New Open Architecture Manufacturing System", ASME PED-Vol.53, pp 93-100.

Wright, P.K., 1995, "Principles of Open Architecture Manufacturing", Journal of Manufacturing Systems, Vol 14, No 3, pp 187-202.

Wu, S.M., D.S. Ermer, 1966, "Maximum profit as the Criterion in the Determination of Optimum Cutting Conditions", ASME Journal of Engineering for Industry, Vol 88, No 4, pp 234-244.

Yamazaki, K., Kojima, N., Sakamoto, C., Saito, T., 1991, "Real Time Model Reference Adaptive Control of 3D Sculptured Surface Machining", Annals of the CIRP, pp 479-482.

Yellowley, I. P. Pottier, 1994, "The Integration of Process and Geometry Within an Open Architecture Machine Tool Control System", International Journal of Machine Tools and Manufacture, Vol 34, No 2, pp 277-293.

Yen, D.W., Wright, P.K., 1983, "Adaptive Control in Machining - A New Approach to Based on the Physical Constraints of Tool Wear", Trans. ASME Journal of Eng. for Ind., Vol 105, pp 31- 38.

Zadeh, L., 1996, In "Intelligent Control Systems, Theory and Applications", edited by M.M. Gupta and N.K. Sinha, IEEE Press, New York, pp xix-xx.

APPENDIX 1
OAC CLIENT DEVELOPMENT

INTRODUCTION

Developing client applications for the IMMRC Open Architecture Controller requires the following:

- a computing platform with support for TCP/IP communication protocols,
- a suitable compiler with TCP/IP sockets libraries,
- physical connection between the client computing platform and the OAC Machine Server (via Ethernet network cable, backplane interface, or inter processor).

The widespread use of Internet Protocol has resulted in almost guaranteed support for this protocol on any platform. Certainly, these requirements are commonly present for UNIX systems. VxWorks, a real time operating system for embedded platforms, supports the TCP/IP interface for network computers for targets integrated on standard backplane systems (eg: VMEbus, MULTI-Bus, PCI bus, etc). Several other real time operating system vendors have similar TCP/IP support.

On personal computers (PC's) running Microsoft Operating Systems, the network components are widely available, and the Microsoft WIN32 application programming interface supports Berkley TCP/IP sockets under the WinSock libraries.

SOCKET PROGRAMMING

The OAC uses sockets to pass data between programs representing the various modules of the overall controller system. The Machine Server is one of these modules. As a module, the Machine Server can be thought of as a "wrapper" that presents the "services" of the machine tool hardware to other modules in a soft manner. Most likely, as a programmer interested in doing something with the machine tool, your primary interest will be concerned with creating and using one or more sockets for communication between your program(s) and the Machine Server.

A complete discussion of socket programming is beyond the scope of this document, but excellent discussions on the subject can be found in several books such as "UNIX Network Programming", by W.R. Stevens, (Prentice Hall, 1990), and "Visual C++ 4.0 Unleashed", by V. Toth, (SAMS, 1996). A study of the various UNIX on line manual pages for commands such as *socket, select, sendto, recvfrom* will of course yield first hand knowledge for real programming issues.

In socket communications, two programs communicate across a virtual line in which each end of the line is a socket. The socket is referred to as an end point of communication, "belonging" to one of the communicating programs. A socket interface is defined by two pieces of information:

1) a machine IP address (or name),

2) a port number.

To set up a socket, the user program defines this information by encoding the fields of a

data structure of type *struct sockaddr_in*. The exact form of this data structure is defined in a C include file *socket.h*. The first item is the standard Internet address of the form such as *130.113.217.074*. Most computer systems support a naming system which will map this number into a machine name, such as *immrc1*. The routine *gethostbyname()* will perform this conversion for you.

The Machine Server uses IP Datagram packets. This form of IP communications is, as the name says, based on packets of information and is a connectionless protocol. Typical packets used to communicate with the Machine Server are text commands such as *machine powerup* (machine services), or text strings of data sent from the Machine Server to your program.

The Machine Server is assigned a fixed port number. For the VxWorks implementation (described in this thesis), **the port number is 2000.** To set up communication between your program and the OAC Machine Server perform the steps listed in the following section (described in the C programming language).

More complex applications may want to modify blocking behavior on sockets, or maintain multiple sockets between the application and different Machine Server entities (such as data reporting, override, and program interfaces). They may also involve socket connections t other OAC modules such as CAD / CAM systems, process control modules, etc.

EXAMPLE OF SIMPLE INTERFACE TO OAC MACHINE SERVER

1) Create and initialize a sockaddr_in structure for the Machine Server:

```
struct sockaddr_in msAddr;

struct hostent *hostaddr;

hostaddr = gethostbyname("immrc08");

msAddr.sin_family = AF_INET;

msAddr.sin_port = htons(2000);

msAddr.sin_addr.s_addr = hostaddr->h_addr;
```

2) Create a local socket:

```
int mySock;

mySock = socket(AF_INET, SOCK_DGRAM, 0);
```

3) Send a message:

```
char msg[32];

strcpy(msg, "data get position x");

sendto(mySock, msg, strlen(msg)+1, 0, &msAddr, sizeof(struct sockaddr_in));
```

4) Receive a message:

char buffer[32];

int aSize = sizeof(struct sockaddr_in);

recvfrom(mySock, &buffer, 32, 0, &msAddr, &aSize);

Note that the recvfrom call will normally block until data is available. This means that the operating system will effectively put your program to sleep until the data arrives. There are several ways to avoid this if required. To remove blocking altogether for the socket use the *ioctl()* call in UNIX. Using Microsoft WinSock libraries, special functions are provided to control blocking. To block, but be able to respond to several sockets (or file descriptor in UNIX), use the *select()* call.

The following is a complete program listing encapsulating the steps described above

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<netinet/in.h>

 main()
{
char msg[32];
char buffer[32];
struct sockaddr_in msAddr;
struct hostent *hostaddr;
int mySock;
int aSize = sizeof(struct sockaddr_in);
```

```
/* get the host IP address by name (OAC machine server is immrc08) */
hostaddr = gethostbyname("immrc08");

/* sockaddr_in fields for OAC machine server address */
msAddr.sin_family = AF_INET;
msAddr.sin_port = htons(2000);
msAddr.sin_addr.s_addr = hostaddr->h_addr;
mySock = socket(AF_INET, SOCK_DGRAM, 0);

/* create a command string for the OAC machine server */
strcpy(msg, "data get position x");
sendto(mySock, msg, strlen(msg)+1, 0, &msAddr, sizeof(struct sockaddr_in));

/* wait for response (this will BLOCK) */
recvfrom(mySock, &buffer, 32, 0, &msAddr, &aSize);

/* print the data */
printf("machine X position is %s\n", buffer);

/* close the socket */
close(mySock);
} /* end of main */
```

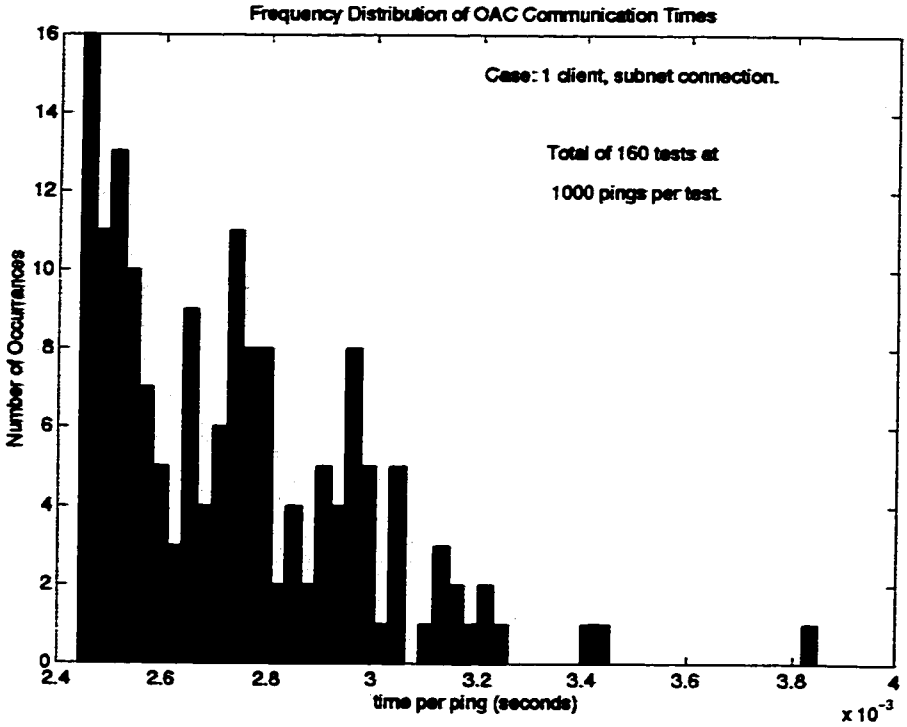
APPENDIX 2
OAC COMMUNICATIONS TIMING RESULTS

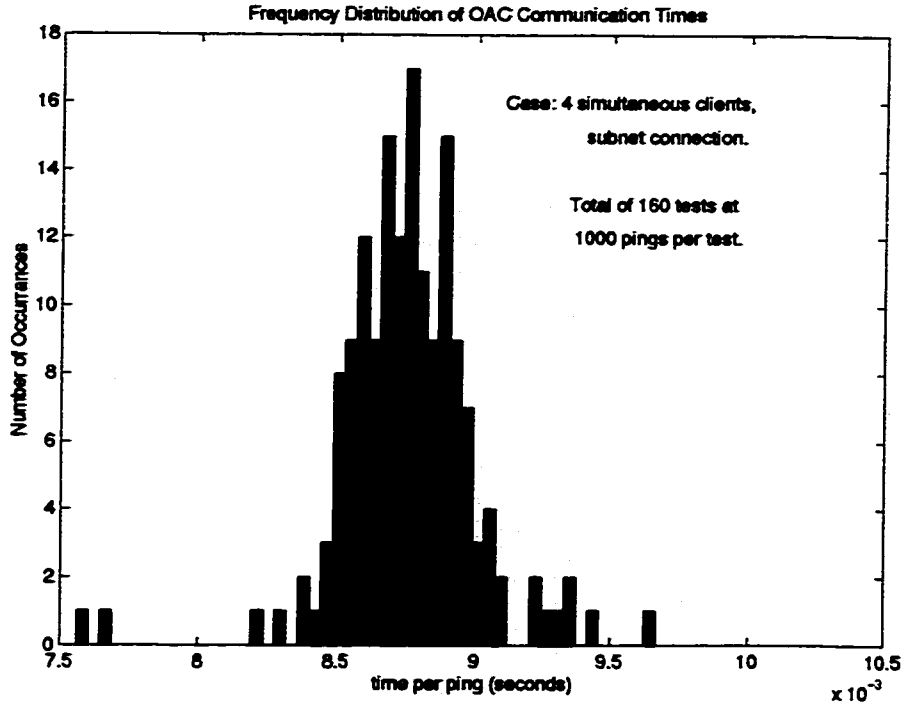
OAC Communications Performance

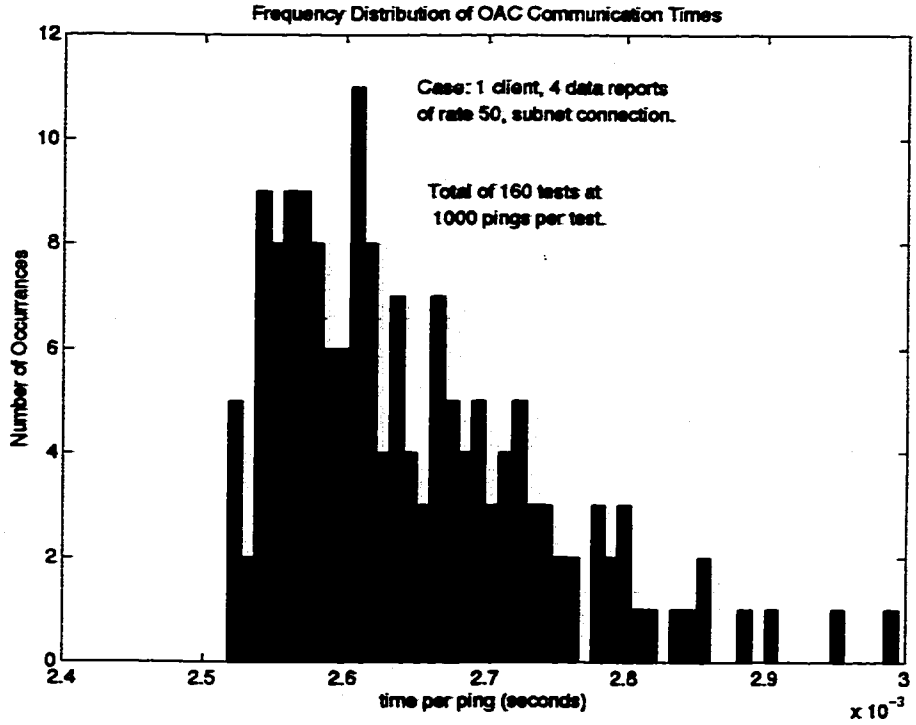
The following graphs are the results of an evaluation of the OAC Machine Server performance under different loading conditions. The tests were performed to evaluate the communications response only.

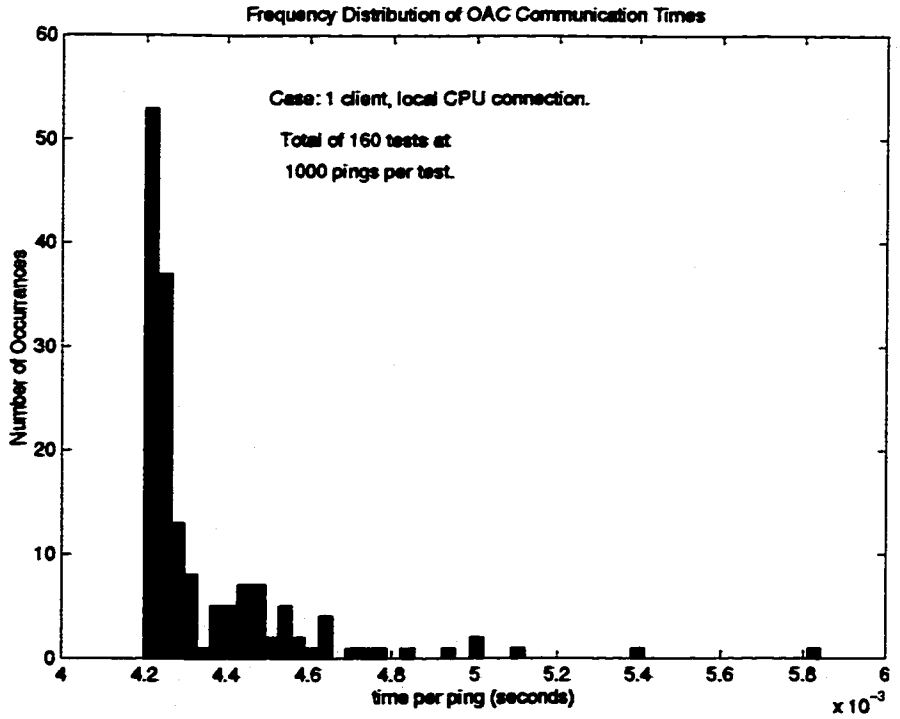
Each histogram shown involve 160 tests of a 1000 send / receive cycles each. The different load cases involve the Machine Server "servicing" different clients requiring data reporting functions. Six cases are shown:

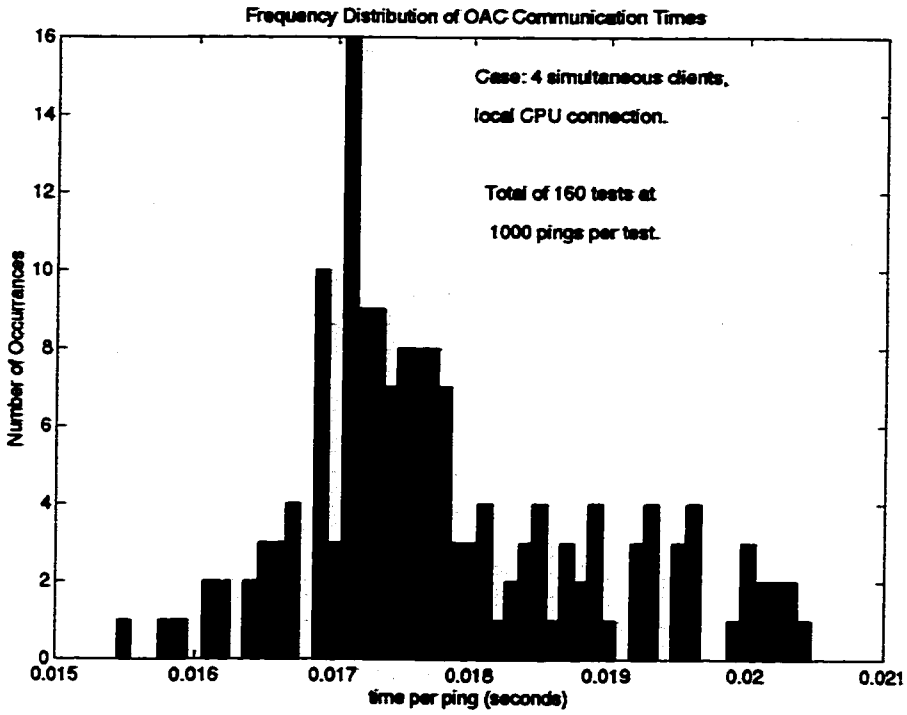
1) 1 client connected over the network, no server loading,
2) 4 clients connected over the network, no server loading,
3) 1 client connected over the network, 4 data reports being serviced,
4) 1 client connected within the local CPU (interprocess comm),
5) 4 clients connected within the local CPU (interprocess comm),
6) 1 client connected within the local CPU (interprocess comm), one data report being serviced

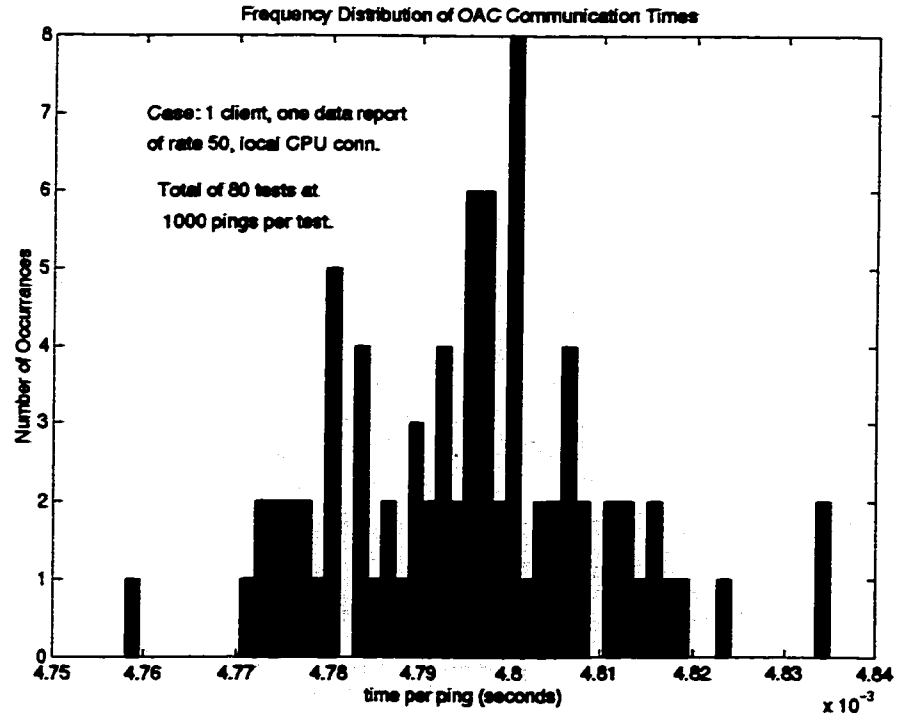

Frequency Distribution of OAC Communication Times
Case: 1 client, subnet connection.
Total of 160 tests at
1000 pings per test.
Number of Occurances
time per ping (seconds)
x 10^-3
0
2
4
6
8
10
12
14
16
2.4
2.6
2.8
3
3.2
3.4
3.6
3.8
4


Frequency Distribution of OAC Communication Times
Case: 4 simultaneous clients,
subnet connection.
Total of 160 tests at
1000 pings per test.
Number of Occurances
time per ping (seconds)
x 10^-3
0
2
4
6
8
10
12
14
16
18
7.5
8
8.5
9
9.5
10
10.5


Frequency Distribution of OAC Communication Times
Case: 1 client, 4 data reports
of rate 50, subnet connection.
Total of 160 tests at
1000 pings per test.
Number of Occurances
time per ping (seconds)
x 10^-3
0
2
4
6
8
10
12
2.4
2.5
2.6
2.7
2.8
2.9
3


Frequency Distribution of OAC Communication Times
Case: 1 client, local CPU connection.
Total of 160 tests at
1000 pings per test.
Number of Occurances
time per ping (seconds)
x 10^-3
0
10
20
30
40
50
60
4
4.2
4.4
4.6
4.8
5
5.2
5.4
5.6
5.8
6


Frequency Distribution of OAC Communication Times
Case: 4 simultaneous clients,
local CPU connection.
Total of 160 tests at
1000 pings per test.
Number of Occurrances
time per ping (seconds)
0
2
4
6
8
10
12
14
16
0.015
0.016
0.017
0.018
0.019
0.02
0.021


Frequency Distribution of OAC Communication Times
Case: 1 client, one data report
of rate 50, local CPU conn.
Total of 80 tests at
1000 pings per test.
Number of Occurrances
time per ping (seconds)
x 10^-3
0
1
2
3
4
5
6
7
8
4.75
4.76
4.77
4.78
4.79
4.8
4.81
4.82
4.83
4.84

APPENDIX 3
SURFACE ROUGHNESS TESTS

Surface Roughness Test Data

The following plots summarize the results of machining tests under the following conditions:

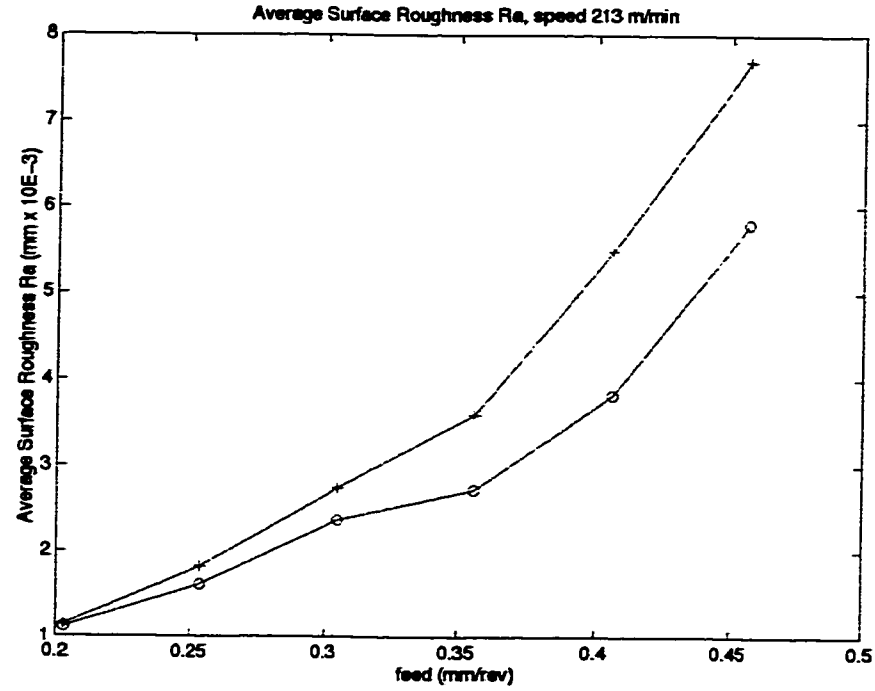
1) speed = 213 m/min, feed = 0.20, 0.25, 0.30, 0.35, 0.40, 0.45 mm/rev

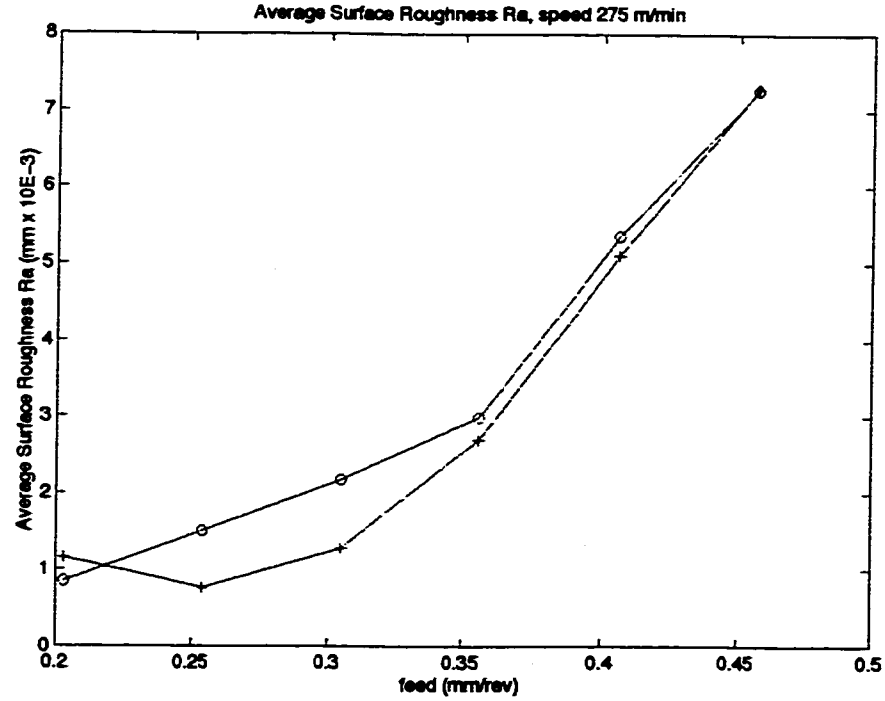
2) speed = 275 m/min, feed = 0.20, 0.25, 0.30, 0.35, 0.40, 0.45 mm/rev

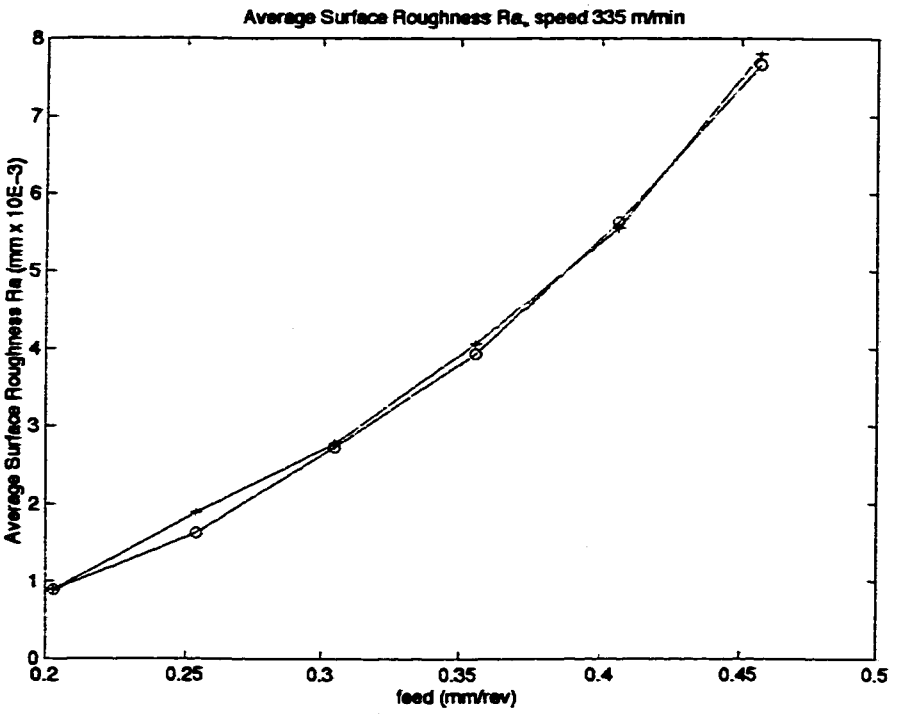
3) speed = 335 m/min, feed = 0.20, 0.25, 0.30, 0.35, 0.40, 0.45 mm/rev

The tests were performed on mild steel using P40 carbide inserts.

Each plot shown has two traces, each representing samples taken 180 degrees apart on the work piece circumference. Each individual scan involved a traversed distance of approximately 5 mm, with 6000 samples acquired within that distance.


Average Surface Roughness Ra, speed 213 m/min
Average Surface Roughness Ra (mm x 10E-3)
feed (mm/rev)
1
2
3
4
5
6
7
8
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Average Surface Roughness Ra, speed 275 m/min
Average Surface Roughness Ra (mm x 10E-3)
feed (mm/rev)
0
1
2
3
4
5
6
7
8
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Average Surface Roughness Ra, speed 335 m/min
Average Surface Roughness Ra (mm x 10E-3)
feed (mm/rev)
0
1
2
3
4
5
6
7
8
0.2
0.25
0.3
0.35
0.4
0.45
0.5

APPENDIX 4
TOOL WEAR TESTING

work material: AISI 1045 hot-rolled steel, as rolled
150 mm, 60 cm long

cutting tool: material - P40 tungsten carbide
geometry - triangular insert
0.8 mm nose radius
+6° rake angle
0° lead angle

cutting conditions: depth of cut - 2 mm

Series 1
tool 5 - 7

feed rate - 0.4 mm/rev
cutting speed - 183 m/min
time of cut - 10 s

Series 2
tool 8, 10, 11

feed rate - 0.6 mm/rev
cutting speed - 123 m/min
time of cut - 10 s

Series 3
tool 12, 13, 17

feed rate - 0.4 mm/rev
cutting speed - 183 m/min
time of cut - 5 s

Series 4
tool 14 - 16

feed rate - 0.6 mm/rev
cutting speed - 123 m/min
time of cut - 5 s

Cutting Series 1: Flank Wear

Tool 5			Tool 6			Tool 7			Average of 5, 6, 7		
total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)
0	0.00	0.00	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00
20	0.12	0.09	20	0.09	0.08	20	0.09	0.08	20	0.10	0.08
40	0.15	0.12	40	0.09	0.08	40	0.12	0.10	40	0.12	0.10
60	0.18	0.15	60	0.16	0.14	60	0.14	0.13	60	0.16	0.14
80	0.20	0.17	80	0.16	0.15	80	0.18	0.15	80	0.18	0.16
100	0.22	0.19	100	0.21	0.17	100	0.20	0.17	100	0.21	0.17
120	0.26	0.21	120	0.23	0.20	120	0.22	0.18	120	0.23	0.20
140	0.30	0.23	140	0.26	0.22	140	0.26	0.23	140	0.27	0.23
160	0.33	0.25	160	0.29	0.24	160	0.27	0.24	160	0.30	0.24
180	0.40	0.29	180	0.31	0.25	180	0.31	0.27	180	0.34	0.27
200	0.40	0.30	200	0.36	0.28	200	0.34	0.29	200	0.37	0.29
220	0.42	0.33	220	0.36	0.32	220	0.36	0.31	220	0.38	0.32
240	0.43	0.34	240	0.39	0.32	240	0.39	0.32	240	0.40	0.33
260	0.48	0.38	260	0.44	0.36	260	0.43	0.38	260	0.45	0.37
280	0.49	0.39	280	0.45	0.37	280	0.46	0.40	280	0.47	0.39
300	0.49	0.40	300	0.49	0.41	300	0.49	0.40	300	0.49	0.40

Cutting Series 2: Flank Wear

Tool 8			Tool 10			Tool 11			Average of 8, 10, 11		
total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)
0	0	0	0	0.00	0.00	0	0.00	0.00	0	0	0
20	0.06	0.04	20	0.06	0.05	20	0.07	0.05	20	0.06	0.05
40	0.07	0.07	40	0.08	0.06	40	0.12	0.09	40	0.09	0.07
60	0.09	0.07	60	0.10	0.09	60	0.13	0.11	60	0.11	0.09
80	0.11	0.10	80	0.12	0.11	80	0.17	0.14	80	0.13	0.12
100	0.12	0.10	100	0.14	0.12	100	0.17	0.14	100	0.14	0.12
120	0.12	0.11	120	0.14	0.12	120	0.19	0.14	120	0.15	0.12
140	0.14	0.12	140	0.16	0.14	140	0.20	0.16	140	0.17	0.14
160	0.15	0.13	160	0.21	0.16	160	0.22	0.17	160	0.19	0.15
180	0.15	0.14	180	0.22	0.18	180	0.25	0.18	180	0.21	0.17
200	0.17	0.15	200	0.25	0.19	200	0.32	0.23	200	0.24	0.19
220	0.18	0.15	220	0.26	0.21	220	0.37	0.26	220	0.27	0.20
240	0.19	0.17	240	0.43	0.29	240	0.44	0.29	240	0.36	0.25
260	0.22	0.18	260	0.51	0.31	260	0.60	0.36	260	0.44	0.28
280	0.24	0.21	280	0.64	0.40	280	0.60	0.39	280	0.49	0.33
300	0.26	0.21	300	0.69	0.40	300	0.64	0.41	300	0.53	0.34
320	0.28	0.22	320	0.73	0.45	320	0.65	0.45	320	0.55	0.37
340	0.29	0.23	340	0.77	0.51	340	0.67	0.47	340	0.58	0.40
360	0.30	0.24	360	0.81	0.52	360	0.71	0.49	360	0.61	0.42
380	0.34	0.26	380	0.85	0.57	380	0.71	0.50	380	0.63	0.44
400	0.36	0.28	400	0.87	0.58	400	0.71	0.51	400	0.64	0.45
420	0.36	0.28	420	0.87	0.58	420	0.72	0.52	420	0.65	0.46

Cutting Series 3: Flank Wear

Tool 12			Tool 13			Tool 17			Average of 12, 13, 17		
total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)
0	0.00	0.00	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00
10	0.11	0.09	10	0.08	0.06	10	0.09	0.07	10	0.09	0.07
20	0.12	0.10	20	0.11	0.10	20	0.09	0.08	20	0.11	0.09
30	0.15	0.12	30	0.13	0.11	30	0.12	0.10	30	0.13	0.11
40	0.17	0.14	40	0.15	0.12	40	0.13	0.11	40	0.15	0.12
50	0.20	0.15	50	0.19	0.16	50	0.15	0.12	50	0.18	0.14
60	0.26	0.20	60	0.22	0.17	60	0.18	0.14	60	0.22	0.17
70	0.33	0.24	70	0.26	0.20	70	0.19	0.16	70	0.26	0.20
80	0.36	0.25	80	0.29	0.21	80	0.26	0.18	80	0.30	0.22
90	0.38	0.27	90	0.32	0.23	90	0.27	0.20	90	0.32	0.23
100	0.39	0.29	100	0.36	0.26	100	0.29	0.21	100	0.35	0.25
110	0.41	0.30	110	0.39	0.30	110	0.30	0.22	110	0.37	0.28
120	0.43	0.32	120	0.43	0.32	120	0.34	0.25	120	0.40	0.29
130	0.45	0.34	130	0.43	0.34	130	0.36	0.28	130	0.41	0.32
140	0.46	0.36	140	0.49	0.36	140	0.40	0.30	140	0.45	0.34
150	0.47	0.37	150	0.49	0.40	150	0.43	0.31	150	0.46	0.36
160	0.49	0.39	160	0.53	0.43	160	0.44	0.33	160	0.49	0.38
170	0.52	0.41	170	0.55	0.45	170	0.46	0.35	170	0.51	0.40
180	0.53	0.43	180	0.58	0.47	180	0.49	0.39	180	0.53	0.43
190	0.56	0.44	190	0.60	0.50	190	0.51	0.41	190	0.56	0.45
200	0.60	0.47	200	0.61	0.52	200	0.54	0.44	200	0.58	0.48
210	0.63	0.50	210	0.63	0.55	210	0.55	0.46	210	0.60	0.50

Cutting Series 4: Flank Wear

Tool 14			Tool 15			Tool 16			Average of 14, 15, 16		
total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)	total cut time (seconds)	Maximum Flank Wear (mm)	Average Flank Wear (mm)
0	0.00	0.00	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00
10	0.05	0.05	10	0.07	0.06	10	0.07	0.05	10	0.06	0.05
20	0.08	0.06	20	0.08	0.06	20	0.08	0.06	20	0.08	0.06
30	0.09	0.08	30	0.08	0.07	30	0.10	0.09	30	0.09	0.08
40	0.10	0.09	40	0.09	0.09	40	0.09	0.07	40	0.10	0.08
50	0.12	0.10	50	0.11	0.10	50	0.09	0.08	50	0.11	0.09
60	0.12	0.10	60	0.11	0.09	60	0.11	0.10	60	0.11	0.10
70	0.14	0.12	70	0.12	0.11	70	0.12	0.11	70	0.13	0.11
80	0.15	0.12	80	0.12	0.11	80	0.15	0.12	80	0.14	0.12
90	0.14	0.12	90	0.15	0.12	90	0.15	0.12	90	0.14	0.12
100	0.16	0.14	100	0.17	0.13	100	0.16	0.14	100	0.16	0.14
110	0.18	0.15	110	0.21	0.15	110	0.18	0.15	110	0.19	0.15
120	0.19	0.16	120	0.22	0.16	120	0.19	0.16	120	0.20	0.16
130	0.21	0.17	130	0.23	0.17	130	0.20	0.16	130	0.21	0.17
140	0.23	0.19	140	0.24	0.17	140	0.22	0.17	140	0.23	0.18
150	0.24	0.19	150	0.27	0.18	150	0.23	0.18	150	0.25	0.19
160	0.25	0.20	160	0.28	0.19	160	0.25	0.19	160	0.26	0.20
170	0.26	0.21	170	0.30	0.20	170	0.26	0.20	170	0.27	0.20
180	0.28	0.21	180	0.32	0.21	180	0.28	0.22	180	0.29	0.22
190	0.29	0.23	190	0.33	0.22	190	0.31	0.24	190	0.31	0.23
200	0.33	0.25	200	0.35	0.24	200	0.32	0.24	200	0.33	0.24
210	0.36	0.27	210	0.39	0.26	210	0.33	0.25	210	0.36	0.26
220	0.39	0.28	220	0.41	0.28	220	0.36	0.26	220	0.38	0.27
230	0.40	0.30	230	0.45	0.29	230	0.36	0.28	230	0.40	0.29
240	0.42	0.31	240	0.49	0.31	240	0.39	0.30	240	0.43	0.31
250	0.46	0.33	250	0.50	0.32	250	0.43	0.31	250	0.46	0.32
260	0.50	0.36	260	0.54	0.35	260	0.45	0.32	260	0.50	0.34
270	0.54	0.38	270	0.57	0.37	270	0.47	0.33	270	0.53	0.36
280	0.56	0.39	280	0.60	0.39	280	0.51	0.36	280	0.56	0.38
290	0.58	0.40	290	0.62	0.41	290	0.54	0.39	290	0.58	0.40
300	0.60	0.41	300	0.63	0.42	300	0.54	0.40	300	0.59	0.41
310	0.61	0.42	310	0.65	0.45	310	0.55	0.41	310	0.60	0.43
320	0.62	0.44	320	0.66	0.46	320	0.56	0.43	320	0.61	0.44
330	0.63	0.46	330	0.66	0.47	330	0.58	0.44	330	0.62	0.45
340	0.63	0.46	340	0.65	0.47	340	0.59	0.45	340	0.62	0.46