Design of a real-time open architecture controller for a Reconfigurable Machine Tool

I Masekamela, N S Tlale
Mechatronics and Micro Manufacturing
Council of Scientific and Industrial Research
Pretoria, South Africa
Email: imasekamela@csir.co.za, ntlale@csir.co.za

C M Kumile
Faculty of Built Environment and Engineering
Tshwane University of Technology
Pretoria, South Africa
Email: KumileCM@tut.ac.za

***Abstract*— The paper presents the design and the development of a real-time, open architecture controller that is used for control of reconfigurable manufacturing tools (RMTs) in reconfigurable manufacturing systems (RMS). The controller that is presented can be implemented on any controller hardware i.e. open controller. The controller is designed in such a way that it satisfies the time critical tasks required by reconfigurable manufacturing tools (RMT) i.e. real time controller. The kinematics analysis of a general 5-axis RMT is presented. Thereafter, software concepts that are used in the controller implementation are discussed. Software controller modules for RMTs are also discussed. The controller architecture for satisfying real time control requirements for RMTs is presented. The simulation results are presented using the different configurations of 5-axes RMT.**

I. INTRODUCTION

Globalization of the economy, saturated markets, rapid advances in technology and the increase in consumer awareness have precipitated the need for mass customization in production of higher quality products. Even though this has resulted in an increase in consumer satisfaction, it has brought about uncertainties in markets demands for consumer goods, with the current trend highly influenced by shorter life cycles. This has led to a lot of unpredictability in market demands and ultimately fragmentation of the market (size and time). Therefore manufactures of consumer goods need to respond to these volatile markets by timely producing high quality products at lower cost and to the exact specifications of the customers. In turn they need appropriate business strategies and manufacturing technologies to accomplish this task.

In recent years the new concept of *Reconfigurable Manufacturing Systems (RMS)* was conceived to address these problems. In terms of design, RMS has a modular structure (both in software and hardware) that allows rearranging of the manufacturing system to adapt it to the new requirements. The hardware is comprised of modular machines made of mechanical components (modules) that can be assembled and disassembled to meet the production requirements. On the other hand open architecture controllers (OAC) are aimed at eliminating the problem of implementation by creating a flexible control system which can be attached to a wide variety of machine tools. OAC's ensures integration of existing and new control devices and software modules, therefore adapting the manufacturing system to the different operating conditions. Thus modular machines and open architecture controllers are key enabling technologies in providing manufacturing systems that can integrate or remove hardware or software modules without affecting the integrity of the whole system, hence ensuring reusability.

The advent of faster processors for personal computers (PC) and a general reduction in their prices have increased the use of PC-based controllers [1]. PC-based controllers are generally flexible, open and can be easily integrated into other manufacturing functions. They also offer faster design cycles, lower down times due to the presence of diagnostic and simulation tools. These attributes help in enhancing productivity and reduces maintenance costs.

The basic constituents of a PC-based controller are hardware platform which comprises of a computer hardware, communications network, peripheral devices and sensors. They are capable of generating, receiving, transmitting, processing and storing data or signals. Software layer which includes operating system, device drivers, network communication and application software enables inter-task communication, multitasking, interrupts handling, network, memory management, system level errors and controls device input and output handling.

The paper is arranged as follows; Sections II describes the characteristics of open controllers including related work while section III introduces the aspects of real time controllers Section IV introduces the concept of controller architecture. Section V catalogues the kinematics and dynamics of a general 5-axes RMT with section VI discussing the controller

implementation. Finally section VIII concludes by stating the expected outcome of the project and future work.

II. CHARACTERISTIC OF OPEN CONTROLLERS AND RELATED WORK

An OAC offer services according to standard resources and standard rules that describe the syntax and services. They are capable of making changes with respect to functionality, performance and dependability as well as adapting to target platforms easily and cost effectively. The presence of support for the integration of new activities in the control environment makes open controllers very desirable. Moreover, they have modular structure with well defined communication interfaces enabling easy communication between modules. Furthermore their ability to detect failures and makes the implementation of recovery process easy.

The research into OAC for use in machine tools intensified in the 90's with organizations like Open Systems Architecture for Controllers within Automation (OSACA) [2] , Open Modular Controller (OMAC) and academic institutions like University of Michigan and University of British Columbia providing some useful platforms to date. Most of the controllers developed need a human machine interface (HMI) in order to alert the users of the processes involved and show the state of a machine at a particular instance. Some developers have suggested use of CAD/CAM systems for true simulation of the manufacturing processes [3]. For the proper execution of tasks, application programs are written for each task using C++, an object-oriented programming language known for its robustness. The application software requires a reliable Application Program Interface (API) to interact with the underlying hardware. A suitable API will abstract the underlying hardware specific architecture and encapsulate the assumptions of the software to allow different application software to run in different hardware platforms. Most developers designed their own API's but unfortunately some proved too hard to understand and they are only compatible to the developer's specially designed hardware.

The other common feature in the controllers was the communication architectures and networks. Some introduced novel communication architectures that have not been thoroughly tested and did not meet the requirements. However widely used communication networks include Ethernet, VentureCom, Process Field Bus (PROFIBUS) and CAN networks. Even though PROFIBUS is known for its high speed, most experiments involved preferred CAN which has proved to perform better in real-time boundaries. Furthermore another critical constituent of the controllers was found to be an operating system. A manufacturing environment uses machines that require motion controls which are hard real-time applications. Tasks like writing an application software are non real-time which can also be handled by a soft real-time system. Most front end operating systems for the controllers developed use Microsoft Windows NT/XP operating systems while both QNX and VxWorks real-time operating systems have enjoyed more success at a lower level implementation where deadline constraints cannot be violated. In some instances a real-time extension to the existing non-real-time operating system was used in order to conform to the requirements.

III. CHARACTERISTICS OF REAL-TIME CONTROLLERS

According to the Institute of Electronics and Electrical Engineering (IEEE) Portable Operating System Interface for Computer Environments (POSIX) Standard 1003.1b , "A real-time system is one in which the correctness of the result not only depends on the logical correctness of the calculation but also upon the time at which the result is made available" [4]. Therefore time is critical for successful implementation of real-time systems. Real-time capabilities can be classified into two main categories namely hard or soft. Hard real-time applies to situations where deadline constraints can not be violated and if this is not met the system can exhibit undesirable behavior. Motion control is one of the tasks which are classified under this category. On the other hand soft real-time tasks do not require strict time constraints.

Real-time operating systems (RTOS) are normally used to implement real-time systems [5]. RTOS have the ability to schedule tasks, meet deadlines, quickly recover from errors, fast switching between tasks and most importantly they are extensible. Further advantages include reduction in sizes and overheads. The presence of characteristics like multi-threadedness, preemptability, threads priority, predictable thread synchronization mechanisms, priority inheritance and predefined latencies (predictable).

IV. CONTROLLER ARCHITECTURE

A. Software architecture

Careful analysis of software implementation for the controller is required to ensure that the implementation adheres to the open and flexible environment required. So the problem solving involves utilising some of the tried and tested approaches used in the design of software for an open architecture. Proper structuring of the software through use of structures and architectural styles where a system can be decomposed into subsystems is the key to achieving a successful implementation of software.

A good architectural style ensures a well coordinated, synchronised and properly functioning system. A combination of styles normally creates a more robust system. Key

identifying characteristics of include flow of data within a system, mode of transmission and good synchronicity. Each architectural style has its own advantages and disadvantages but for this particular project data abstraction and object oriented style is the focus in software implementation. This style is characterised by encapsulation of data and its primitive operations into abstract data called objects [3]. The ability of an object to hide its representation from the clients makes it possible to change the implementation without affecting the client. The bundling of a set of accessing routines with the data they manipulate enables programmers to decompose problems into a collection of interacting agents.

In spite of favourable properties the object-oriented style posses some problems especially with interaction. The object must first establish the identity of another object it wants to interact with [6]. In the event that identity of an object changes it is required that the updates are passed on to the objects that invoke it. However mixing this style with other will ensure more robustness at the end of the implementation. Other styles include pipes-and-filters, layered, time-triggered, reactive, process networks, publish and subscribe, client server, process control and finite state machine.

B. Hardware Architecture

The implementation of hardware is one of the delicate matters to handle in ensuring a real time environment. Despite the use of the hardware that is real time compliant it is imperative to arrange the hardware in such a way that there are minimal communication errors and delays. This involves use of a hardware architecture that will be used in positioning the hardware modules of the system including sensors, microcontrollers and processors.

The proposed controller will use a hierarchical architecture where each axis will have its own servo controller and a supervisory controller which is likely to be a Wafer-Luke will be responsible for monitoring the entire system. The distributed nature and the multi-structure of this architecture will enhance reconfigurability. Even though the decentralized architecture seems to be more reconfigurable than the hierarchical architecture, its performance is highly likely to be affected by communication nodes and the decision was made to stick to a fail safe hierarchical architecture.

V. KINEMATICS AND DYNAMICS ANALYSIS OF A GENERAL 5-AXES RMT

Nowadays most machine tools are made of at least 5 axes commonly three translational degrees of freedom (DOF) along the X, Y and Z axes and two rotational DOF. Any of the axes or a combination of axes can be chosen to accommodate either a tool holder or a workpiece and the different configurations of the machine tool can be drawn from that aspect.

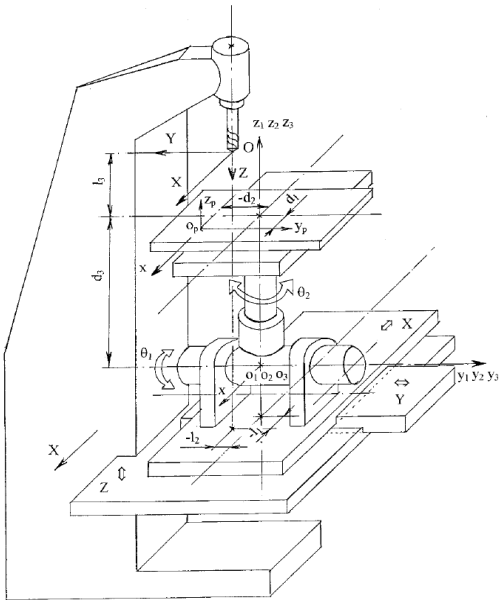

Figure 1: An example of a machine tool

The Denavit-Hartenberg representation, a 4x4 homogenous transformation matrix is used to represent the kinematics relationship between links [7]. A coordinate system is attached to each link as illustrated in fig. The representations for a rotary joint is

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, (1) $$

while the first two rows of the 4th column are replaced by 0's for the linear joint. θ_i and d_i are joint parameters for the i^{th} rotary and linear joints respectively.

In order to obtain the orientation of one link in relation to the other which is normally referred to as direct kinematics, a direct multiplication of transformation matrices will be done as follows;

$$ {}^0T_i = {}^0A_1\,{}^1A_2 \ldots\ldots\ldots {}^{i-1}A_i $$

$$ = \begin{bmatrix} x_i & y_i & z_i & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^0R_i & {}^0p_i \\ 0 & 1 \end{bmatrix} \quad (2) $$

where ${}^0R_i$ and ${}^0p_i$ is the orientation matrix and the position vector of ith link from the base/original link respectively. Even though the kinematics can be used to describe the motion of the links, they do not take into consideration the forces that cause the motion. Therefore further analysis of the dynamics of

the system was undertaken. The control torque is computed from

$$\tau(t) = D(q)\ddot{q}(t) + h(q,\dot{q}) + c(q), \qquad (3)$$

where q represents either one of the joint parameters θ_i and d_i while h(q,q) and c(q) are the Coriolis and gravity loading vectors respectively. D(q) is an nxn (where n represent the number of D.O.F) and is given by

$$D(q) = \left[\sum_{i=1}^{n} \{m_i J_{v_i}(q)^T J_{v_i}(q) + J_{w_i}(q)^T {}^0R_i(q) I_i {}^0R_i(q)^T J_{w_i}(q)\}\right]. \qquad (4)$$

with m_i being the mass of link i. J is the Jacobian matrix that determines the velocity relationships. It is calculated by

$$J_i(q) = \begin{cases} \begin{bmatrix} z_{i-1} \times {}^{i-1}p_i \\ z_{i-1} \end{bmatrix} \\ \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \end{cases}. \qquad (5)$$

The upper part of equation 5 is used if the joint is rotational while the lower part is meant for a linear joint. In order to achieve steady end effector motion along any coordinate axis, it is imperative to combine and run motions of various joint motors simultaneously at different time varying rates (Fu et al). This type of control is called Resolved Motion Rate Control (RMRC) with the equation of motion

$$\dot{q}(t) = A^{-1}J^T(q)[J(q)A^{-1}J^T(q)]^{-1}\dot{x}(t) \qquad (6)$$

derived from a non-linear

$$x(t) = f[q(t)]. \qquad (7)$$

VI. CONTROLLER IMPLEMENATION

The control functions are going to be implemented in software, therefore a good RTOS is essential. Real-time (RT) Linux will be used as the RTOS. In addition to being an open source platform, RT-Linux has short scheduling and interrupt latencies [1]. It was shown to have a worst case scheduling latency of 25 µsec on a 300Hz and 128 MB Pentium II machine while the PC's earmarked for the project have the processor speed of more than 1 GHz and are equipped with at least a 1GB of RAM.

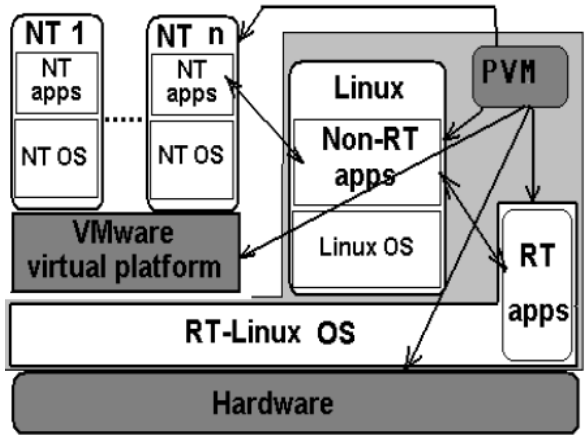

Figure 2: An example of the implementation [1]

As initially indicated the hard real-tasks will be directly implemented on RT-Linux. Other tasks will be implemented on the normal Linux operating system. It has been found out that a lot of manufactures use Microsoft windows based operating systems. One of the goals of the project was to create a controller that could be ported to existing systems without having to make major changes, hence it is necessary to create an interface for allowing using of windows based front end OS.

The hardware implementation would adopt the hierarchical architecture which promotes reconfigurability. Also proper positioning of sensors would lower the communications delays while at the same time enhancing real-time properties.

A communication network which will be responsible for providing reliable and temporally predictable message passing [6] between nodes will be adopted. The broadcast and bus topologies are currently the most common as they provide simultaneous arrival of signals at different tasks and are cost effective. At lower level Serial Real-time Communication system (SERCOS), a digital interface for communication between industrial controls and input/output (I/O) devices would be further investigated for possible implementation. Moreover the ability of SERCOS to coexist with other protocols like Ethernet makes it more suitable for implementation. Process Field Bus (PROFIBUS) is another communication interface which is normally used to operate sensors and actuators will also be investigated. Moreover Controller Area Network (CAN) bus, a computer network protocol would be explored.

A separate project is being carried on to create a library of the components of machine tool so as to provide the necessary equipment for generating different machine tool. The resultant control system should be able analyse a CAD model of the generated machine tool and perform the dynamics analyses. A

suitable control law will be selected for any particular configuration of the machine tool. Fig shows the desired procedures for the motion control implementation.

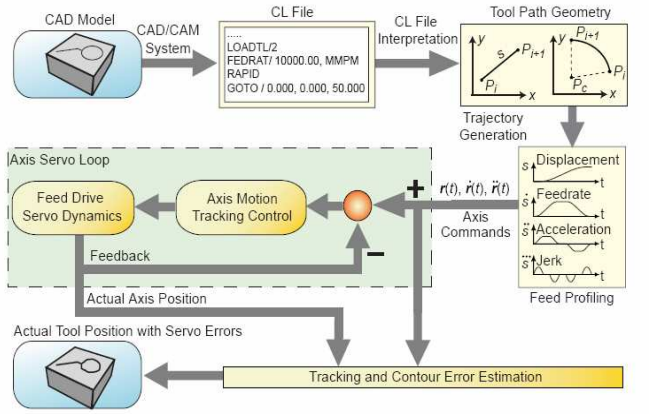

Figure 3: Motion control [8]

The end user should just have a detailed graphical user interface (GUI) illustrating the actions of the machine tool. Fig is an example of the GUI.

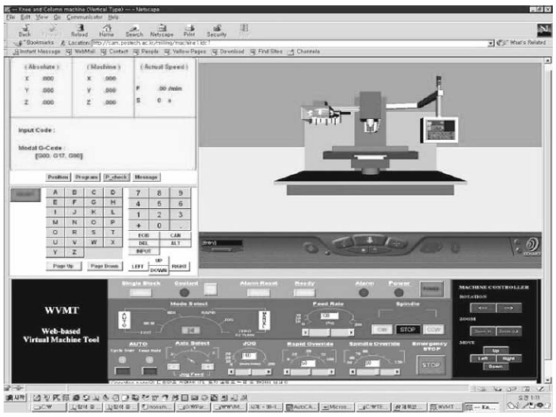

Figure 4: A graphical user interface [9]

VII. CONCLUSION

The need for a manufacturing system that can be quickly changed both at hardware and software level to adapt to rapidly changing markets intensified the need for modular machine tools and open architecture controllers.

The paper presented the steps that are going to be followed in developing an open architecture controller that will be used for controlling different machine tools or different configurations of machines. Although a number of similar project have been undertaken they have fallen short in terms of openness of the controller and in meeting the real-time requirements. It is expected that at the end, a controller that is capable of handling kinematics and dynamics configurations will be produced. It is expected that the architecture of the controller will be open to new activities within the manufacturing environment and it will be flexible enough to quickly adjust to varying production requirements.

The proposed controllers will be first modelled using the above mentioned tools and tested on virtual platform. Upon successful implementation on a virtual platform it would be tested on a real testbed which is still to be developed.

ACKNOWLEDGMENT

I would like to thank both my supervisors Dr Nkgatho Tlale and Chris Kumile for their guidance. I would like to acknowledge the both Council of Scientific and Industrial Research and Advanced Manufacturing Technology Strategy (AMTS) in providing financial support for undertaking the project.

REFERENCES

[1] S.Kommareddy, Y. Kazuo, K.Yoshihito , “PC-based open architecture servo controller for CNC machining”, Proceeding of the Second Real-Time Linux Workshop, Orlando, USA, 20-27 November 2000.

[2] J. Nasca, “Comparison of three different open architecture controllers”- Proceedings of International Federation of Automatic Control(IFAC)-MIM, Prague, Czech Republic, pp. 134-138, 2-4 August 2001.

[3] R. Atta-Konadu, “Design and implementation of a modular controller for robotic machines”, Saskatoon-Doctorate Thesis, University of Saskatchewan,, 2006.

[4] M. Barabanov, “A Linux-based Real-Time Operating System”, Masters dissertation, New Mexico Institute of Mining and Technology, 1 June 1997.

[5] A. Gambier, 2004, Real-time Control Systems: A Tutorial. Control Conference, 5th Asian, Volume 2, , pp 1024-1031, 20-23 July 2004.

[6] D Chen, “Architecture for Systematic Development of Mechatronic Software Systems”- Royal Institute of Technology (KTH), Stockholm, Sweden, Masters Thesis 2001.

[7] K.S.Fu, R. C Gonzalez, C.S.G Lee, “ROBOTICS Control, Sensing, Vision, and Intelligence, McGraw-Hill Book Co, Singapore, 1987.

[8] K. Erkorkmaz, Y. Altintas, C-H Yeung, “Virtual Computer Numerical Control System”, CIRP annals, vol 55, no 1, pp 399-402, Elsevier, 2006.

[9] S-H. Suh, Y. Seo, S-M. Lee, T-H. Choi, G-S. Jeong, D-Y. Kim, “Modelling and Implementation of Internet-Based Virtual Machine Tool”, International Journal of Advanced Manufacturing Technology, vol 21, pp 516-522, Springer-Verlag, 2003.