Modular Open Architecture Controller for a Reconfigurable Machine Tool

QA Amra[a], J Padayachee[b], G Bright[c]

Received 14 January 2014, in revised form 13 June 2014 and accepted 18 June 2014

***Abstract** Modern manufacturing systems require machines and control systems that are flexible and reconfigurable to be able to match changes in product mix and demand. This paper presents research into the development of a modular Open Architecture Control System (OAC) for a Modular Reconfigurable Machine Tool (MRMT). The research involved the development of a novel modular control solution that links closely to the modular mechanical system to maximize reconfigurability. High levels of reconfigurability were achieved by the implementation of distributed microcontroller based drive modules in an open control environment. A class based OAC was designed in C#, which facilitates interoperability of control hardware from multiple vendors. The OAC also allows users to configure the software based on the MRMT structure. The OAC was tested on a MRMT to verify the reconfigurability of the controller and to test its performance.*

Additional keywords: Open Architecture, Reconfigurable Manufacturing Systems, Modular Reconfigurable Machine Tool, Open Architecture Control System.

1 Introduction

Reconfigurable Manufacturing Systems (RMS) aims to combine the advantages of Flexible Manufacturing Systems (FMS) and Dedicated Manufacturing Systems (DMS) to produce a system that achieves a high throughput in addition to the necessary flexibility, which will allow the system to evolve and change its production functionality efficiently and quickly as required[1,2]. RMSs are systems that are aimed at being inherently flexible and reconfigurable in their software and hardware constituents[3]. This inherent reconfigurability ensures that the system is able to match varying production requirements, while still maintaining a high throughput[4]. In order to achieve high levels of reconfigurability and a long system life, the design a RMS and its machines must be[1]: modular, convertible, scalable, integrable, customizable and diagnosable.

Mehrabi[5] has suggested that software issues proved to be the area of greatest concern for the successful development and implementation of machines for RMS. The primary motivation for this research is the advent of Modular Reconfigurable Machine Tools (MRMTs) as an advanced technology for RMS[6,7]. These machines require a control system that is modular and scalable. Advanced requirements such as interoperability and control openness are not met by modern, commercially available solutions. This paper presents the development of an OAC that facilitates the interoperability of electronic hardware from multiple vendors, allowing the openness necessary for user customization of the system. The paper proceeds as follows: section two presents the concept of a MRMT and their control requirements, section three presents recent developments on OACs, section four presents the OAC implementation, sections five presents the user interface and six present the test results. Section seven presents challenges to the industrial implementation of the OAC.

a. Discipline of Mechanical Engineering, Mechatronics and Robotics Research Group (MR^{2}G), University of KwaZulu Natal, South Africa, amra.aq@gmail.com
b. Discipline of Mechanical Engineering, Mechatronics and Robotics Research Group (MR^{2}G), University of KwaZulu Natal, South Africa, padayacheej@ukzn.ac.za
c. Discipline of Mechanical Engineering, Mechatronics and Robotics Research Group (MR^{2}G), University of KwaZulu Natal, South Africa, brightg@ukzn.ac.za

2 Control Interoperability and System Openness

MRMTs are machines that are assembled from a library of modules as shown in figure 1; in this library are modules that may be used to build the base, the axes, the work holder and the spindle of a machine. These same modules are assembled in different numbers and configurations to yield machines with different Degrees of Freedom (DOF), varying kinematic configurations and different processing functions.

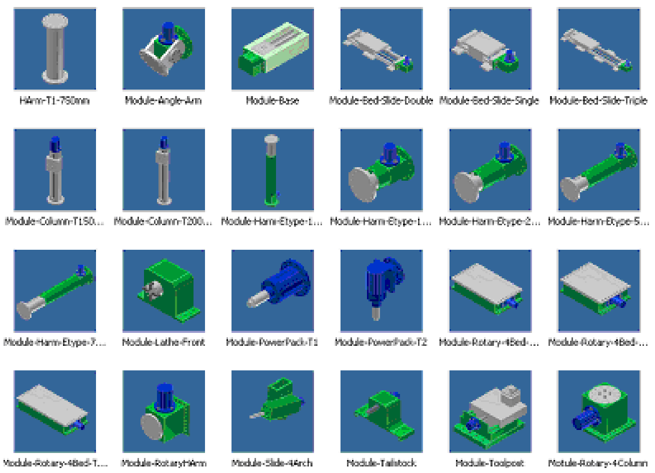


Figure 1 A library of mechanical modules for MRMTs[6]

The modular mechanical architectures of these machines allow the processing functions to be reconfigured, as shown in figure 2. In this illustration a three axis line boring machine is reconfigured into a three axis vertical milling machine through the interchange of a single module.

MRMTs are also scalable in their mechanical architectures[6,7]. Figure 3 shows how a three DOF vertical mill may be scaled up to a four DOF machine by the integration of an additional module.

The development of the OAC system, presented in this paper, aimed to match the mechanical reconfigurability of MRMTs. There are modular, scalable control hardware and software in the market today, available from companies such as SEW-EURODRIVE (Pty) Ltd, National Instruments Corporation (UK) Ltd and Siemens (Pty) Ltd. However, modularity and scalability alone are insufficient to meet the control demands of reconfigurable machinery. Researchers such as Mpofu *et al.*[8,9] have stated that the future of reconfigurable machinery lies in the ability to construct modular machines from Commercial Off The Shelf (COTS) mechanical modules. These modules may be sourced from multiple vendors and therefore interoperability of mechanical, control and software elements is an essential characteristic. The development of an open software control system facilitates the interoperability of hardware from multiple vendors while providing a consistent user environment and performance. The need for interoperability has been stressed by researchers such as Koren[10] and Pristchow *et al.*[11]. The necessity for control openness has also been stressed by these authors; this is discussed in the next section.

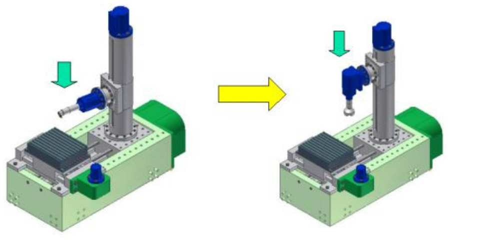


Figure 2 A Reconfiguration of Machining Functionality[6]

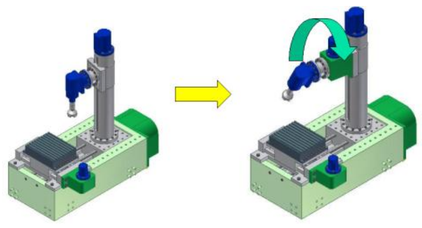


Figure 3 A Reconfiguration of Machining DOF[6]

3 Open Architecture Systems

The nature of RMS requires control systems that can quickly and reliably adjust its control functionality depending on the available hardware[5]. For the realization of RMS, the system is required to be open at all three levels, namely: system, machine, and control[10].

The openness of the system is key to the effectiveness of the overall system, where openness is characterized by portability, extendibility, interoperability and scalability[11]. Furthermore there needs to be a strong link between the software and hardware sub architectures of a RMS[13] therefore, modularity is also a key characteristic for an OAC system[1]. Figure 4 illustrates how the criteria mentioned affect the system both internally and externally.

An OAC should allow a user to be able to integrate user specific algorithms and programs. To facilitate this, the user will require access to the internal data structures and variables in order to implement such control algorithms. Koren *et al.*[10] highlight two basic types of controllers: a Vendor Specific (VS) controller, which is a closed system, and a Vendor Neutral (VN) controller which is an open system defined by a standard with the aim of allowing integration of modules and algorithms. Koren[10] and Pritschow *et al.*[11] state that for the success of OAC systems, the system must be: VN; based on well-established OAC standards; and provide well defined methods for data exchange and control reconfiguration. The architecture of the control system should therefore be designed to allow the user to add, swap or integrate new modules at any given time.

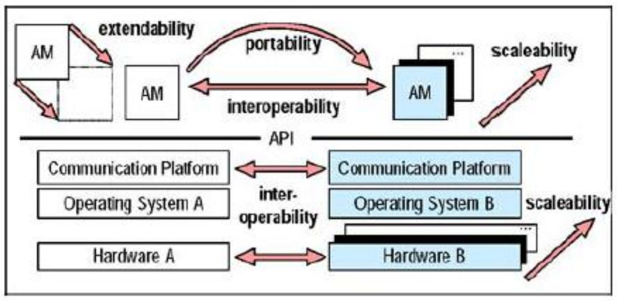


Figure 4 Criteria for Open Systems[11]

Another key aspect for OAC systems is the reusability of basic modules in the creation of more complex algorithms[10]. For example a basic limit switch software module may be used in a position control module which in turn will be used for interpolation. To accommodate for the effective re-use of software modules, a well-defined Application Programming Interface (API) is required for each module. Pritschow *et al.*[11] emphasize that the performance of the control system is influenced by the level of interoperability between the basic modules therefore, the development of well-defined API between the modules is crucial. The OSACA Reference model[13] presents an architectural blueprint for the implementation of an OAC system on PC hardware. The specifics of the computing system are encapsulated and the reference model is used to assist in control system portability and the interoperability between application modules.

Many attempts at developing OAC systems have been taken in recent years: the PC based software CNC system by Xiong-bo *et al.*[14], the PUMA robot system[15], the reconfigurable hardware-software multi-agent platform by Morales-Velazque *et al.*[12] and the implementation by Proctor *et al.*[16] are examples. Each of these has adopted different design approaches to achieve similar objectives with varying degrees of success. However, many of the attempts lacked a unified global standard or system. On the contrary the tried and proven systems such as Emerson's Delta V charm based solutions[17] and Siemens SIMATIC systems[18] are proprietary and not open. Proprietary control systems are either not plug and play or do not allow for reconfiguration[14].

4 Control System Development

4.1 Overview

To ensure maximum system reconfigurability, the following features were identified for development in the OAC: software openness, modularity, interoperability and scalability. These features would enable the control system to be easily reconfigured and expanded in minimal time.

To achieve these desired characteristics, a soft CNC architecture was implemented. A soft CNC architecture concentrates generic software routines such as user programming, text interpretation, program validation, trajectory planning and trajectory generation on a host computer. Hardware specific software routines are concentrated on distributed microcontroller based drive modules. A distributed drive module contains all the necessary software and electronic circuitry for the control of its corresponding mechanical module. Distributed drive modules connect to the host computer via a fieldbus; in this instance a CAN bus. An architectural overview of this system is presented in figure 5.

The significance of the distributed drive module is that it provides a physical separation of the hardware on which generic and module specific software routines are executed. This significantly simplifies the development of the OAC. At runtime the object orientated C# implementation on the host computer dynamically creates a specific software module for each corresponding hardware module using classes. Object orientated C# allows the modules to be defined from generic classes with a well-defined API. The inheritance feature of C# allowed the design to initially create one class for all modules and when distributed drive modules are detected, new classes are derived from the generic class set.

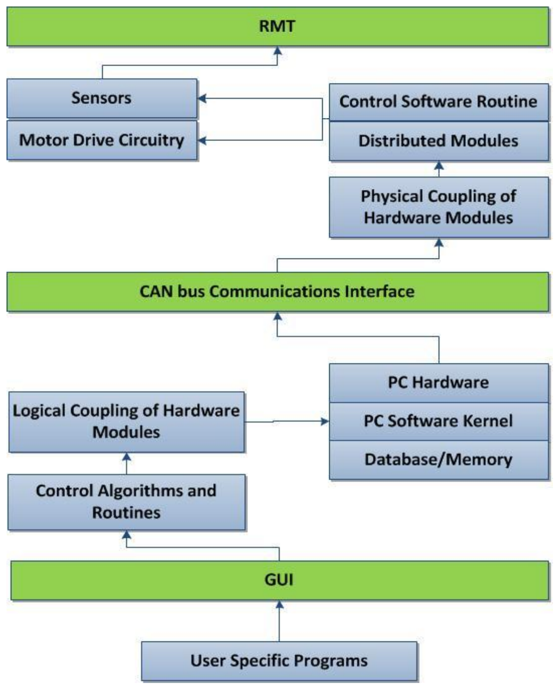


Figure 5 Architectural Overview of Control System

4.2 Distributed Drive Modules

Researchers have identified that the future of modular machines will involve the integration of mechanical and control modules from multiple vendors[8,9]. The challenge is to ensure seamless mechanical integration and consistent control performance. Researchers such as Abele *et al.*[19] have performed research on the development of standardized interfaces for the mechanical assembly of complete machines from modules. The concept of an OAC addresses the interoperability challenge presented when using digital control hardware from multiple vendors.

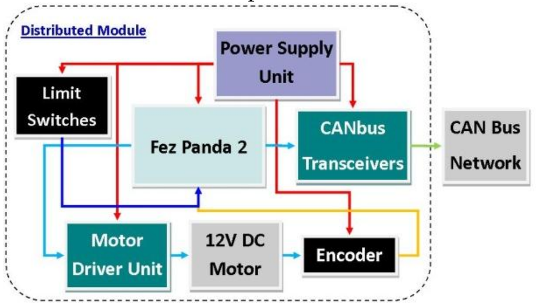


Figure 6 Components of a Distributed Drive Module

It is envisioned that in future, when a COTS mechanical module is purchased or rented, that module will be accompanied by its own dedicated drive module. Each distributed module contains the power electronics, communication and digital control hardware required for the operation of its corresponding mechanical module; figure 6 is a schematic representation of the components of the drive module.

The drive module contains the control algorithms for position and speed control of the mechanical module; however, the drive module serves more functions than a standard motor driver. It contains configuration information that enables "plug and play" functionality; allowing the OAC to instantiate new software modules to manage the reconfigured MRMT. Plug and play functionality is essential to enable the dynamic reconfigurability of the system.

Data such as module ID, module type, operating frequency, mechanical specifications and Homogenous Transformation Matrices (HTMs) are communicated to the OAC, allowing it to create a reference model of the hardware that is attached to the machine. The reference model is essential for instantiating the software functions that are necessary for the trajectory planning and trajectory generation operations on the host computer. An up-to-date reference model of the hardware on the machine is also necessary for validating user programs; an example of this would be to check that the range of motion programmed by the user does not exceed the physical range of motion of an axis.

4.3 Buffer Layers

The introduction of distributed modules creates two buffer layers in the system architecture, between the mechanical modules and the distributed drive modules, and between the distributed drive modules and the host PC, as depicted in figure 7. These interface layers allowed for the components at each tier to vary in types and architectures. Consequently all that was required to ensure system functionality is that the interface and data transfer between the tiers was consistent and standardized. The functioning of the OAC was therefore independent of the types of modules at each tier or what was located at each tier. The modules in a tier can therefore change and be upgraded without affecting system functionality.

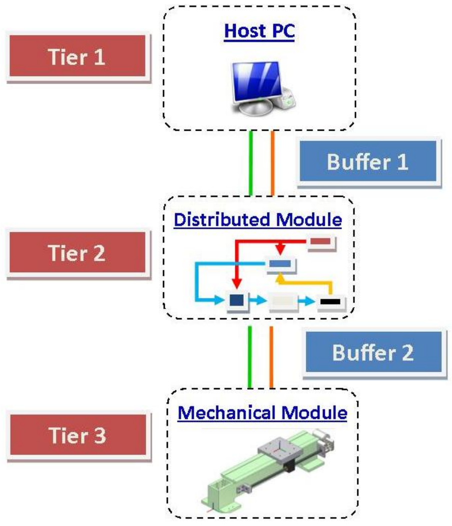


Figure 7 Mechatronic layers with buffers

4.4 OAC – Software Reference Architecture

The OAC software system is based on the multi-tier architecture illustrated in figure 8. The reference architecture outlines the necessary software routines and the interaction between these routines for the numerical control of a MRMT.

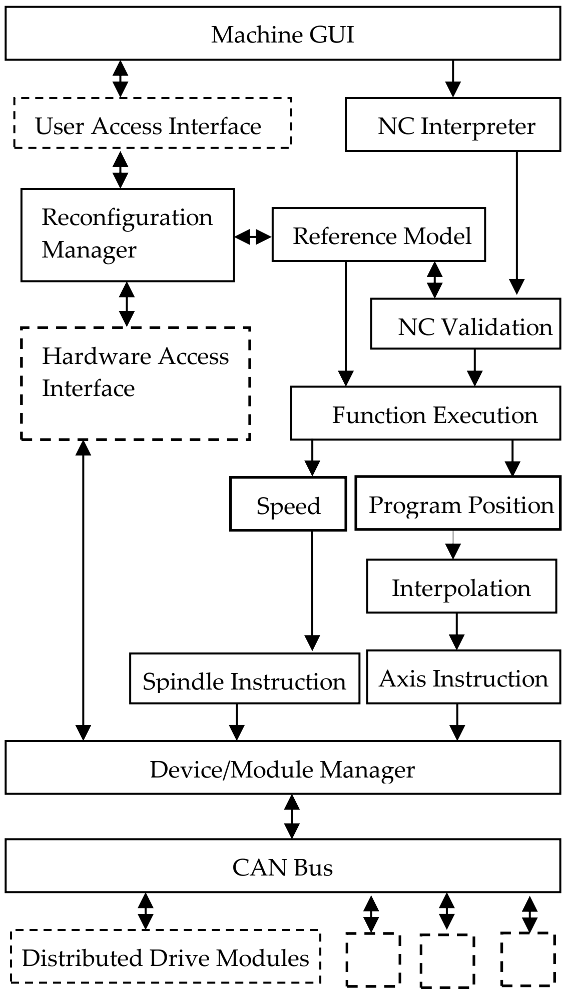


Figure 8 OAC Reference Architecture

The reference architecture contains many of the standard software functions of a typical CNC controller; these include text interpretation, program validation and interpolation. In addition to the standard NC functions, an OAC has two access interfaces that allow the functionality of the control system to be appropriately configured. These interfaces are what make this an open system.

4.4.1 User Access Interface and the Reconfiguration Manager

The high level interface is a user access interface that allows the user to configure algorithms. The user is able to configure algorithms as far down as the position and speed control algorithms operating on distributed drives by transmitting appropriate configuration messages via the module manager. The function of the module manager is to manage the communication between the OAC and distributed drives. The reconfiguration manager mediates between the distributed drive modules and the user to ensure that user selects configuration parameters that are available.

4.4.2 Hardware Access Interface and the Reconfiguration Manager

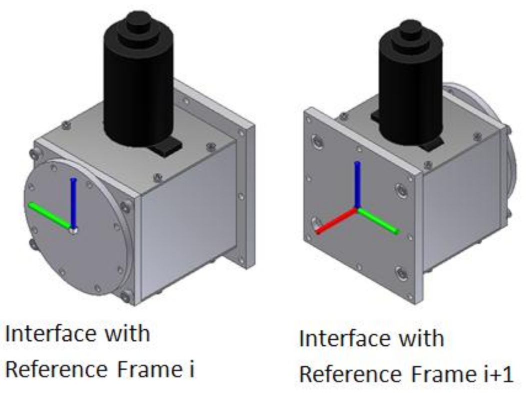
The OAC also has a low level interface which is a hardware access interface. This interface allows the distributed drive modules to transmit data to the reconfiguration manager. The reconfiguration manager actions the module manager to probe the CAN bus in order to determine the number of distributed control modules attached to the network. Distributed modules transmit hardware configuration data back to the reconfiguration manager such that it can appropriately configure the behavior of the OAC and build a reference model of the hardware that is attached to the system. This reference model is necessary such that the OAC can instantiate the appropriate software modules, containing the necessary routines to control the present MRMT configuration. As mentioned in section 4.1, these software modules are instantiated from predefined classes and the inheritance feature of C# allows the OAC to instantiate new derived classes according to the distributed drive modules connected to the CAN bus.

4.4.3 Hardware Reference Model

The hardware reference model enables the control and monitoring of individual axes. This model is an information model containing data on the spindle, the number and types of rotary axes and the number of linear axes. More importantly it builds up a new forward kinematic description of the MRMT each time the mechanical platform is reconfigured. The Homogenous Transformation Matrix (HTM) for any mechanical module is given by equation 1.

$$ {}^{i+1}_{\ \ \ i}M_n = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & x \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & y \\ -s\beta & c\beta s\gamma & c\beta c\gamma & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1) $$

The rotation component of the transformation matrix is represented by X-Y-Z Euler angles. The X-Y-Z Euler angle convention was selected as a standard method of representing the rotational component of a modules DOF, other similar conventions may have also been used.



$$ {}^{i+1}_{\ i}M_R = \begin{bmatrix} 1 & 0 & 0 & -138 \\ 0 & c\gamma & -s\gamma & 0 \\ 0 & s\gamma & c\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad -90^o \le \gamma \le 90^o $$

Figure 9 Mechanical Module with HTM

For example, the distributed drive corresponding to the rotary module in figure 9 would transmit it's HTM, together with its range of motion to the reconfiguration manager. Module transformation matrices are concatenated in the reconfiguration manager in order of module assembly from the cutting tool to the work holder, yielding a final matrix that describes the position of the tool holding module relative to the global reference on the work holding module. The forward kinematic model for any MRMT is given by equation 2.

$$ {}^{Work}_{Tool}T = M_n M_{n-1} M_{n-2} \dots M_1 \qquad (2) $$

This model is essential for establishing axis limits for the purpose of verifying and validating user programs. It is also essential for establishing the position of the spindle, relative to a reference point on the worktable. This is necessary for NC programming in absolute coordinates.

4.4.4 Interoperability via the Module Manager

The module manager operates on a set of standardized message protocols which creates interoperability. To demonstrate interoperability and openness, the OAC implementation was operated with three different distributed drive modules. The three different drive modules contained different microcontroller boards, which varied in system architectures, capabilities, development environments and bootloaders.

Table 1 summarizes the key differences between the chosen microcontroller boards. Variation exists in the electronic circuitry and digital control hardware of each distributed drive module. The set of standardized message protocols on which the modules and the module manger operates hides the variation on distributed drives from the OAC, allowing it to maintain interoperability and general functionality. This level of interoperability opens up the possibility for the integration of Commercial Off The Shelf (COTS) components from multiple vendors into any MRMT.

5 Graphical User Interface

Microsoft Visual C#, which is based on Microsoft's .Net Framework. The GUI in figure 10 shows the user several tabs that allow for the MRMT to be configured and programmed:

- Hardware Modules - After initialization of the CAN bus network, a bus scan is run to determine which modules are connected, after which the user is to enter the physical configuration of the MRMT to determine kinematic viability.
- Data Download - The host PC will download critical information such as control requirements, limitations and transformation matrices, from each connected module.
- Module Information - Displays information downloaded from each module.
- CAN bus - To assist with debugging and diagnosis. The tab displays the raw CAN bus data received
- Controllers - Controller selection tab.
- Motor Control - To assist with tuning and debugging. Individual control and movement of each connected module can be setup.
- Program Editor - To allow the user to enter a custom program.
- Algorithm Editor - To allow the user to edit parameters such as controller tuning. Interface to allow the user to customize and tune the control parameters on each distributed module.

Table 1 Comparison of Microcontroller Specifications

	FEZ Panda 2[20]	chipKIT MAX32[21]	Arduino UN[22]
Chipset	USBizi ARM7	PIC32	Atmel ATmega328
Operating Voltage	5 V	3.3 V	5 V
Frequency	72MHz	80MHz	16MHz
CAN Interface	1x CAN controllers	2x CAN controllers	SPI via CAN-BUS Shield
Programming Environment	Visual C#	MPIDE	Arduino IDE

The choice to have cross platform uniformity, standardization and the use of a common programming language meant that the OAC software was developed using Microsoft Visual C#, which is based on Microsoft's .Net Framework.

By creating a modular electronic hardware and software system with well-defined interfaces, the control system has the ability to be scalable. Distributed modules can be added on to the system with ease and by scanning the network, the new module will be detected and a class will be generated for it.

Multiple modules are capable of running on the system without conflict. The control system aims to address the customization requirements of open control systems by including three methods to customize programs and control algorithms. Users have the option to: program, compile and run custom applications for their MRMT. In addition, the performance of the controller can be evaluated and if need be the controllers can be tuned by the user. These methods provide flexibility, reconfiguration and customization to the end user, and if desired, additional methods can be implemented to allow further user customization.

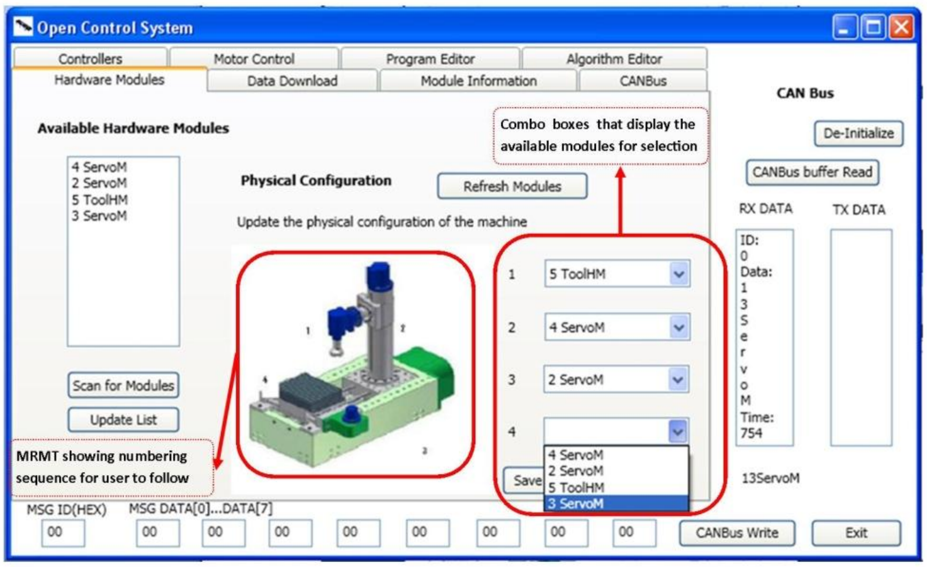


Figure 10 OAC GUI showing the configuration menu

6 Testing and Results

The MRMT was tested in for accuracy, repeatability, and module response time; these results are important discussion points for OAC systems. Results are presented here for a subset of MRMT modules, particularly the X, Z and A axes. Table 2 presents the repeatability in terms of worst case deviation from set points and the accuracy is computed and is shown in table 3.

Table 2 Worst Case Repeatability Results

	X Axis CW	Z Axis CW	A Axis CW
% Variation	0.42	1.97	0.88
	X axis CCW	**Z Axis CCW**	**A Axis CCW**
% Variation	0.21	0.99	0.94

Table 3 Worst Case Accuracy Results – Linear Axes

X Axis CW	Z Axis CW
0.42 mm	1.13 mm
X axis CCW	**Z Axis CCW**
0.27 mm	0.47 mm

It should be noted that the MRMT was a low cost experimental platform and backlash in the mechanical couplings and gears lead to lower accuracy and repeatability in the platform when compared to industrial systems. Table 4 summarizes the accuracy and repeatability for the rotary axis.

Table 4 Accuracy and Repeatability Results – Rotary Axis

	A Axis CCW	A Axis CW
% Variation	0.31	0.41
	A Axis CCW	**A Axis CW**
Deviation	1.3^{0}	1.45^{0}

Table 5 presents the response times and despite having the second fastest clock frequency, the Fez Panda 2 boards have the slowest response times. The Arduino UNO running at a clock frequency of 16 MHz, has a significantly faster response time although it operates at less than a quarter of the FEZ Panda 2 frequency. Despite the Arduino UNO running at a fifth of the clock frequency of the CHIPkit, it matches its response times. The Arduino UNO and the CHIPkit use similar Arduino bootloaders, unlike the FEZ Panda 2 board which runs on the Microsoft .NetMicro Framework. The response times are therefore not purely dependent on the microcontroller operating frequency, but are also based on the microcontroller architecture and microcontroller bootloader framework.

Table 5 Summary of Distributed Module Response Times

Message	Response Time (ms)			
Board	FEZ Panda 2	chipKit	FEZ Panda 2	Arduino Uno
General Call	5.2	1.1	6.7	1.5
Data Download	5.3	1.4	6.9	2.2

7 Challenges

The requirement for end user customization, VN controllers and interoperability of COTS components motivated the development of an OAC for the MRMT. However, if the system is open and this openness is not exploited within well-defined guidelines, it will ultimately have a negative impact on the system performance. The operating frequencies of distributed modules and the microcontroller bootloaders pose a challenge to the operation and performance of the system. The OAC showed that despite the differences in frequencies and bootloaders, the distributed modules managed to communicate with the host PC. The OAC uses the lowest common denominator in terms of module response times, to ensure uniform functioning. The performance of the system is therefore limited to the performance of its slowest drive module. Thus, performance trade-offs, as demonstrated by this implementation will not be acceptable until performance measures for distributed modules are standardized.

8 Discussion

MRMTs are an emerging paradigm in machine tool technology. These are machines that are assembled entirely out of modules that are reusable and can be rearranged to adjust the architecture of the machine. The flexibility, reconfigurability and modular structure allows the system to be gradually upgraded over time as production requirements change. This allows the system to be installed with the exact functions required at a given time, thereby minimizing the initial capital investment required. Researchers have stated that in future, modular machines should be able to integrate mechanical and control modules from multiple vendors while still providing consistent performance[8,9]. The OAC developed here demonstrated the integration and interoperability of control hardware from multiple vendors. Testing has revealed that with three different microcontrollers the system exhibited acceptable performance results but also revealed key challenges for the realization of OAC systems.

The OAC also demonstrated system openness by providing a user access interface and a hardware access interface. These interfaces allow for user customization through the integration of new mechanical and control modules. The C# language in which the OAC was developed, provided the facilities of modular software development and inheritance which made the OAC adaptable. Furthermore, the OAC has demonstrated openness and end user customization by allowing the access interface to change and modify performance parameters and the behavior of algorithms.

9 Conclusion

This research has shown that the need for openness, user customization, VN and flexibility can be addressed by an OAC. OAC systems are able to provide these desired characteristics; however they require well defined standards and guidelines for development. OAC systems in mobile phone industry had suffered as a result of a lack of unified standards for a vast number of development platforms. It was only after Android was developed for mobile platforms that there was cohesion and alignment in development that allowed for rapid innovation[24]. Similarly, the OAC systems for machine tools require standards and guidelines, if these are not developed through global collaboration; the performance of OAC systems will never match those of proprietary systems.

References

1. *Koren Y, Heisel U, Javone F., Moriwaki T, Pritschow G, Ulsoy G and Brussel HV*, Reconfigurable Manufacturing Systems, *CIRP Annals - Manufacturing Technology*, 1999, 48, 527-540.
2. *Koren Y*, Reconfigurable Manufacturing and Beyond, *Proceedings of the CIRP 3rd International conference on Reconfigurable Manufacturing*, Ann Arbor, Michigan, USA, 2005.
3. *Stecke KE*, Flexibility is the Future of Reconfigurability. Paradigms of Manufacturing—A Panel Discussion, *Proceedings of the 3rd Conference on Reconfigurable Manufacturing*, Ann Arbour, Michigan, USA, 2005.
4. *Malhotra V, Raj T, Arora A*, Reconfigurable Manufacturing System: An Overview, *International Journal of Machine Intelligence*, 2009, 1, 38-46.
5. *Mehrabi MG, Ulsoy AG, Koren Y and Heytler P*, Trends and Perspectives in Flexible and Reconfigurable Manufacturing Systems, *Journal of Intelligent Manufacturing*, 2002, 13, 135-146.
6. *Padayachee J*, Development of a Modular Reconfigurable Machine for Reconfigurable Manufacturing Systems, Master of Science in Engineering Thesis, School of Mechanical Engineering, University of KwaZulu-Natal, Durban South Africa, 2010.
7. *Padayachee J and Bright G*, Modular Machine Tools: Design and Barriers to Industrial Implementation. *Journal of Manufacturing Systems*, 2012, 31, 92–102.
8. *Mpofu K, Kumile C and Tlale NS*, Adaptation of Commercial Off The Shelf Modules for Reconfigurable Machine Tool Design. *Proceedings of the 15th International Conference on Mechatronics and Machine Vision in Practice*, Auckland, New Zealand, 2008.
9. *Mpofu K, Kumile C and Tlale NS*, Design of Reconfigurable Machine Systems: A Knowledge Based Approach, *Journal of KONBiN*, 2008, 8, 135-144.
10. *Koren Y, Jovane F and Pritschow G*, Open-Architecture Controllers for Manufacturing Systems - Summary of Global Activity, *ITA Series*, Milano Italy, 1998.
11. *Pritschow G, Altintas Y, Jovane F, Koren Y, Mitsuishi M, Takata S, Brussel HV, Weck M and Yamazaki K*, Open Controller Architecture – Past, Present and Future, *CIRP Annals - Manufacturing Technology*, 2001, 50, 463-470.
12. *Morales-Velazquez L, Romero-Troncoso RDJ, Osornio-Rios RA, Herrerra-Ruiz G and Cabal-Yepez E*, Open Architecture System Based on Reconfigurable Hardware-Software Multi Agent Platform for CNC Machines, *Journal of Systems Architecture*, 2010, 56, 407–418.
13. *Sperling W and Lutz P*, Designing Applications for an OSACA Control, *Proceedings of the International Mechanical Engineering Congress and Exposition*, Dallas, USA, 1997.
14. *Xiong-bo M, Zhn-yu H, Yong-zhang W and Hong-ya F*, Development of a PC-based Open Architecture Software-CNC System, *Chinese Journal of Aeronautics*, 2007, 20, 272-281.
15. *Farooq M and Wang DB*, A Reconfigurable and Modular Open Architecture Controller: The New Frontiers, *International Journal of Automation Technology*, 2008, 2, 205-214.
16. *Proctor FM, Shackleford W, Yang C, Barbera T, Fitzgerald M, Frampton N, Bradford K, Koogle D, and Bankard M*, Simulation and Implementation of an Open Architecture Controller, *Proceedings of the SPIE International Symposium on Intelligent Systems and Advanced Manufacturing*, Philadelphia, PA, USA, 1995, 196-204.
17. DeltaV Digital Automation System System Overview, Emerson-Process-Management Catalogue, Emerson Electric co, Austin, Texas, 2009.
18. SIMATIC Controllers: The Innovative Solution for All Automation Tasks, Siemens-AG Catalogue, Siemens-AG, Nürnberg, Germany, 2011.

19. *Abele E, Wörn A, Fleischer J, Wieser J, Martin P, Klöpper R,* Mechanical Module Interfaces for Reconfigurable Machine Tools, *Production Engineering Research and Development*, 2007, 1, 421-428.
20. FEZ Panda II Board - GHI Electronics, GHI Electronics Product Specification Sheet, GHI Electronics, LLC, Troy, Michigan, USA, 2011.
21. chipKIT™ Max32™ Board Reference Manual, Digilent, Inc. Product Reference Manual, Digilent, Inc., Pullman, Washington, USA, 2011.
22. ArduinoUno: http://www.arduino.cc/en/Main/arduinoBoardUno, 2012.
23. *Tilbury D and Kota S*, Integrated Machine and Control Design for Reconfigurable Machine Tools, *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Atlanta, GA, USA, 1999.
24. Andy Rubin, Android's founder, Leaves Project: http://www.zdnet.com/andy-rubin-androids-founder-leaves-project-7000012563/?s_cid=e589, 2013.