Jurnal Teknologi, bil. 24, Disember 1994 hlm. 9 – 21
©Universiti Teknologi Malaysia

A SOFTWARE ARCHITECTURE FOR MODULAR ROBOTIC SYSTEMS

S. ARIFFIN
R.H. WESTON
R. HARRISON
Loughborough University of Technology
UNITED KINGDOM

Abstract. Research is described which is leading to the specification and development of a motion simulation and design environment for modular robotic systems which enables the implementation of widely applicable software processes for machine control. Current investigation is focused on defining models of application tasks in modular robotic systems. This work is based on the Real-time Control System (RCS) reference architecture proposed by researchers at the National Institute of Standards and Technology (NIST) which was designed to support motion planning and implementation. However, this architecture is modified in such a way that it supports the concept of multitasking and inter-process communication. The emphasis of work is on the hierarchical structuring of solutions, this to enable the design and control of distributed motion elements. Also discussed in this paper is a strategy for achieving sensor-based modularization of modular robotic systems in a manner which facilitates fast and efficient response to changes in the functional or environmental requirements. The paper explains how an application software architecture is unified with the open systems design approach known as Universal Machine Control (UMC), which has been devised and developed at Loughborough University to enable reuse to software and control system components.

1 INTRODUCTION

Efficient design of manufacturing machine systems is a key requirement for the achievement of many industrial automation goals, as this leads to a reduction in the time-to-market of products which are competitively priced and closely meet customer needs. However, current approaches to the design of manufacturing machine systems require extensive human resources in terms of time, cost and expertise, thus preventing the full achievement of potential automation benefits. One of the key factors limiting the use of robotic systems as integral part of a manufacturing workcell stems from constraints imposed by their control systems. Indeed many forms of manufacturing machine (including robots) are supplied with low-capability simple controllers, which is restricted to a position control capability and simple input/output operations. Their restrictive computational architecture does not allow the implementation of flexible motion control strategies nor does it facilitate their flexible and effective integration into a host environment.

An automated manufacturing facility typically comprises a number of control computers separately located within a production facility to monitor sensors and issue appropriate actuator commands to associated production equipment. Software is required to control the machines and coordinate their operation so that production requirements can be satisfied. Suitable open approaches should be identified which formalise and structure the generation

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

of low-level software which realises motion control within manufacturing workcells, thereby enabling the design of higher functionality control systems (in terms of meeting individual application requirements) or achieving the same functionality levels with less effort and shorter development times. Invariably contemporary industrial approaches to system specification and construction lead to extensive effort required to produce the necessary real-time control software. Particular difficulties are experienced as the control software needs to concurrently control the operation a variety of mechanisms and processes in real-time.

This paper seeks to unify the use of architectural frameworks, which facilitate the structures of machine control task during system specification, with 'open' system implementation methods which facilitate system construction. This is to provide the creators of machine control systems with means of adopting distributed control techniques to realise solutions which are modular, reusable and extendable. The objective is to create an environment which enables the efficient implementation and experimental evaluation of various motion control functions in modular robotic systems. The investigation is focused on the definition of tasks for modular robotic systems based on the reference architecture proposed in Albus, McGain & Lumia [1, 2]. From the architectural viewpoint, it can be considered as being an implementation of version Albus, McCain & Lumia [1], hence it is designed to support motion planning and implementation in manufacturing processes. The emphasis here is on providing a functional task framework for motion synchronization in robotic control systems. This paper proposes an applications software architecture for machine control and its unifaction with the open system design approach referred as Universal Machine Control (UMC) which was developed by the Modular Systems Research Group (MSG) in Loughborough University.

2 PROBLEMS IN AUTOMATED MANUFACTURING

2.1 A Functional Machine Control Architecture

An architectural model of a control system should describe the functional and logical computational interrelations among its essential elements. Functional machine control (including robot) architectures are often organized within a control hierarchy although each architecture may embrace a variety of different control techniques. Most manufacturing machines on the markets are still restrictive in term of their functional capabilities and computational architecture which severely restricts the implementation of efficient and flexible control strategies, Duelen et. al. [3]. The basic operations performed by the elements of a machine must co-operate in performing a global goal (or high level task) which is defined or input at a highest hierarchical level. This global task should be successively decomposed through a hierarchy of control levels into simpler lower level commands and actions which are to be obeyed at subordinated levels. At the lowest hierarchical level, drive signals are provided for the actuators.

Current generations of machine controller are generally unable to meet the need for flexible manufacturing, offering limited scope opportunity for the reuse software, Harrison [11] and other control system components. There is a need for more efficient engineering methods which can formally and widely adopted by process designers and the control system engineers. The widespread acceptance of a problem oriented approach to the design and implementation of real-time control systems has however been much slower than in the case of business systems. The industrial application of these has been uncoordinated and unstandardised, probably largely because of the inherently high levels of heterogeneity involved.

The development of UMC is aimed at addressing the key implementation problems in machine control with particular emphasis on associated problems of using the current generation of proprietary control system elements. An overview of the UMC architecture is illustrated in Figure 1, which defines three hierarchical levels (machine, task and handler). Figure 2 illustrates the background problems underlying the formation of the UMC software systems. This illustration is based on a UMC Summary Document, Universal Machine Control [4] and a report on Integrated Machine Design and Control, Harrison, Moore & Weston [5] which outlines aspects of current research plans of the MSG.

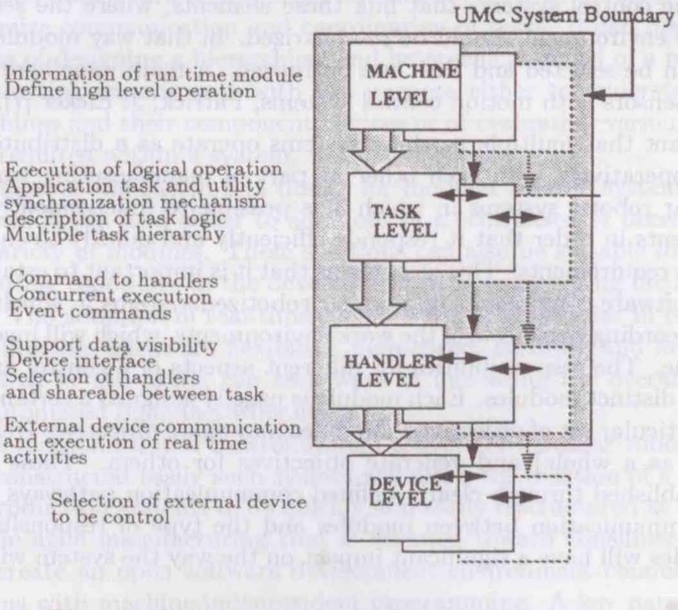
UMC System Boundary
Information of run time module
Define high level operation
MACHINE LEVEL
Ececution of logical operation
Application task and utility
synchronization mechanism
description of task logic
Multiple task hierarchy
TASK LEVEL
Command to handlers
Concurrent execution
Event commands
Support data visibility
Device interface
Selection of handlers
and shareable between task
HANDLER LEVEL
External device communication
and execution of real time
activities
Selection of external device
to be control
DEVICE LEVEL

Fig. 1 UMC overview and reference architecture

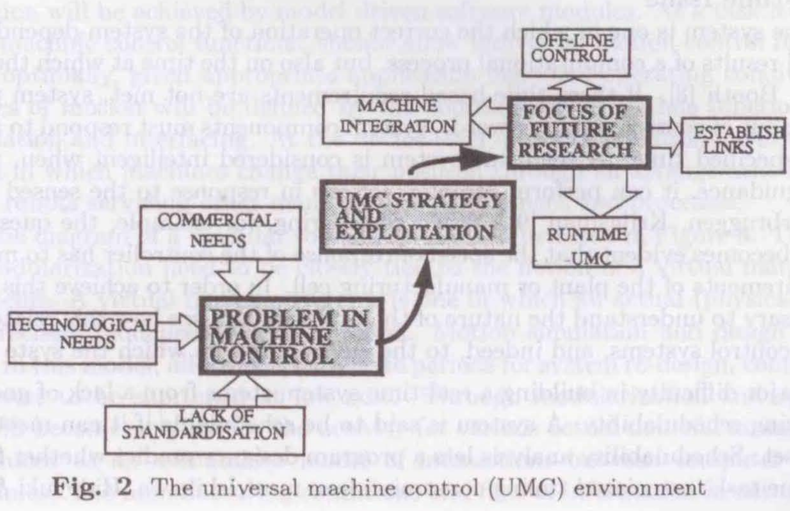

Fig. 2 The universal machine control (UMC) environment

2.2 Modularization in Robotic Control

In a robotized factory, each machine (including robots) can perform a subset of task commonly performed by humans (through use of their inherent motion and sensing-recognition capability) thereby automating manufacturing processes, such as assembly. On considering the robotized factory, a key issue raised concerns the way in which objects interrelate and fit with each other in a complex manner (such as in assembly) as this determines required motion profiles, such as the path traversed by a robot hand. Indeed difficulties of determining such profiles has slowed the introduction of robots, Tsukune, Tsukamoto et. al. [6]. A contributory problem is the current inability to integrate sensory systems in a generalised manner. To expand the usefulness of robotization, it is necessary to pursue sensor-based modular robotic control systems that link these elements, where the sensing function for recognizing the environment should be modularized. In that way modules which suit individual tasks can be selected and their use optimized. A main issue here is the integration of distributed sensors with motion control systems, Patrick, J. Eicker [7].

It is important that multiple machine systems operate as a distributed system so that they work co-operatively with each other at part of manufacturing workcells. This can lead to modular robotic systems in which it is possible to independently control multiple machines elements in order that it responds efficiently and quickly to changes in the work environment or requirements. This also means that it is important to establish architectural model of the software components of modular robotized systems in which control functions are designed according to aspects of the work environments, which will invetably change over a period of time. The responsibilities for different aspects of a control architecture can be divided among distinct modules. Each module is usually assigned a certain responsibility for achieving a particular set of objectives. Modules may share objectives (related to objectives of the system as a whole) and generate objectives for others. These dependencies can be usually established through clearly defined communication pathways between modules. The type of ommunication between modules and the type of responsibilities assigned to different modules will have a significant impact on the way the system will interact with its environment.

2.3 Real-time Issue

A real-time system is one in which the correct operation of the system depends not only on the logical results of a computational process, but also on the time at which these results are produced, Booth [8]. If these time-based requirements are not met, system processes will simply get out of control. In other words, system components must respond to a given events within a specified time. A real-time system is considered intelligent when, with minimal external guidance, it can perform complex actions in response to the sensed environment, Rodd, Verbruggen, Krijgsman [9]. When considering, for example, the question of direct control it becomes evident that the speed of response of the controller has to match the real-time requirements of the plant or manufacturing cell. In order to achieve this performance, it is necessary to understand the nature of the problem, and see how this relates back to the design of control systems, and indeed, to the environment in which the syste will execute.

One major difficulty in building a real-time system stems from a lack of good techniques for analyzing schedulability. A system is said to be schedulable if it can meet all dead line of a task set. Schedulability analysis lets a program designer predict whether for a given set of real-time tasks associated timing constraints can meet,Ishikawa, Hideyuki & Mercer [10].

The determination of such factors places a bound on the execution time of each task. To meet this requirement, the system must avoid priority inversion problems that occur when a higher priority task must wait while a lower priority task executes.

3 CONCEPT OF MODULAR ROBOTIC SYSTEM

Manufacturing machines normally incorporate of multitasking, high speed manipulation capability and should demonstrate flexibility and reconfigurability. Typically there may be a considerable communication problem among local machine control systems. Future machine control systems must embody adequate real-time machine control and integration capabilities to deal with the communication problem among machine control devices which often will be supplied from a variety of sources. Simulation is one of the tools which offers the possiblity to optimize communication and coordination of tasks. It can play an important part in the process of designing a hierarchical and heterachical model of a real-time control systems and conducting experiments with the purpose either to understand the motion behaviour of machines and their component devices or of evaluating various strategies and application logic required within a system.

Research into motion simulation and design for modular robotic systems is a real-time control problem which can lead either to solutions of a centralised or decentralised nature and comprise a variety of modules. These solutions can also be suitable for application in flexible manufacturing especially in the development of manufacturing facilities on a small or medium scale. The adoption of real-time software which is modular in construction can ensure easy modification, hardware flexibility, high-speed, accurate and low cost automation. Thus, research in this arena can seek ways of increasing the overall efficiency and productivity of automated manufacturing systems.

If both the the mechanical and control system aspects of modular robotic systems can be specified and constructed easily such system will allow the creation of a new generation of machine and production systems to be quickly and easily restructured at low automation cost and to enable agile manufacturing that is oriented toward consumer customization. It is possible to create an open software development environment conducive to creating distributed systems with machine-indenpendent programming. A key paradigm of the solution will be decomposition into modules, whereby reuse and sharing of goals, commands and information will be achieved by model driven software modules. At a task level, modularization of machine control functions, should allow individual motion control functions to be arranged optimally, given appropriate application logic and operating conditions. The characteristics of motion will be defined by the coordination of module functions such as sensing, actuation and interfacing. At the device level, a working motion environment will be developed in which machines change their position through an arrangement of multiple modularized robots servicing other manufacturing equipment and processes.

A schematic diagram of a modular robotic system is illustrated in Figure 3. The concept of robotic modularization need to be closely tied to the notion of a virtual manufacturing machine systems. A virtual machine systems is one in which an actual (physical) machine systems is precisely modelled using computers. Motion simulation and design should be manipulated in this model, allowing appropriate periods for system re-design, commissioning and changeovers to be shortened at low cost. Through modularization and openness of software it will become possible for the activity for various actual manufacturing processes to be reproduced as an executables model of interactions between computer controlled motions modules. The manufacturing conditions and flow of information in actual machine systems will always be reflected in the virtual machine systems. In other word, in a modular robotic system, the existence of a virtual machine system that accurately represents the real machine systems will be important. In the event of any differences arising between a virtual machine systems and actual machine systems, this will need to be resolved.

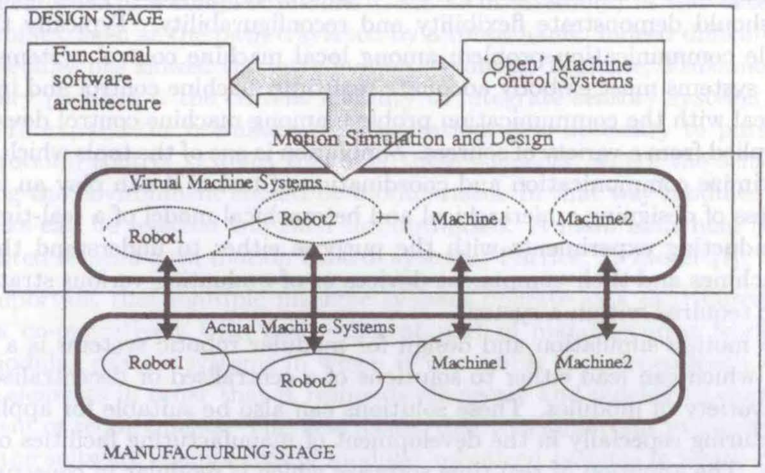

Fig. 3 Modelling structure for modular robotic control

4 ARCHITECTURE DESIGN AND MODULARITY

This research is focused on the task level of the UMC architecture, and seeks an appropriate application task architecture as illustrated in Figure 4. The research investigates motion coordination and modification of a hierarchical and distributed control approach for manufacturing cells, based on a unification of the NASREM, Albus, McGain, Lumia, Junberts, Hui-Ming Huang, Quintero, Zeigler [1, 2, 12, 13, 14, 16] and UMC, Universal Machine Control, Harrison, Moore, Weston [4, 5, 11] reference architectures. The UMC reference architecture is defined within three hierarchical levels (machine level, task level and handler level), Duelen et. al, Universal Machine Control [3, 4]. The current UMC run-time architecture is illustrated in Figure 5. A UMC machine consists of a number of concurrently executing processes together with mechanisms for communication and synchronization of the processes. Application tasks in the task level of the architecture are application dependent and user defined. The application tasks can have a heterarchical relationship although functionally one task can be a master task, Szabo, Scott et al [15]. The implementation of UMC uses the Microware OS-9 operating system as a platform. The OS-9 Events are used to synchronize the tasks, control the use of shared resources and for passing data between the tasks. Handlers provide a standardized interface between tasks and the external devices which they control. The advantages of using handlers are that they support device data visibility, provide a virtual device interface and are shareable between tasks. This opens up structured design and model driven approaches to distributed real-time control and maps it onto a set of proprietary (and modular) control system elements. The unification of the NASREM and UMC reference architecture as illustrated in Figure 4 can lead to a software design for machine control which should be flexible, reusable and easily modified as required. In order to design effective reusable application task software, one must understand how to decompose specific applications into potentially reusable modules and how to arrange these modules into an architecture to support reuse with minimal modification.

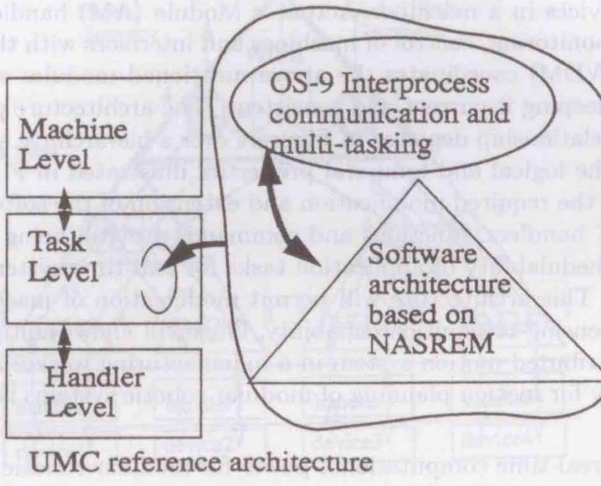
OS-9 Interprocess communication and multi-tasking
Machine Level
Task Level
Handler Level
Software architecture based on NASREM
UMC reference architecture

Fig. 4 The unification of NASREM and UMC architecture

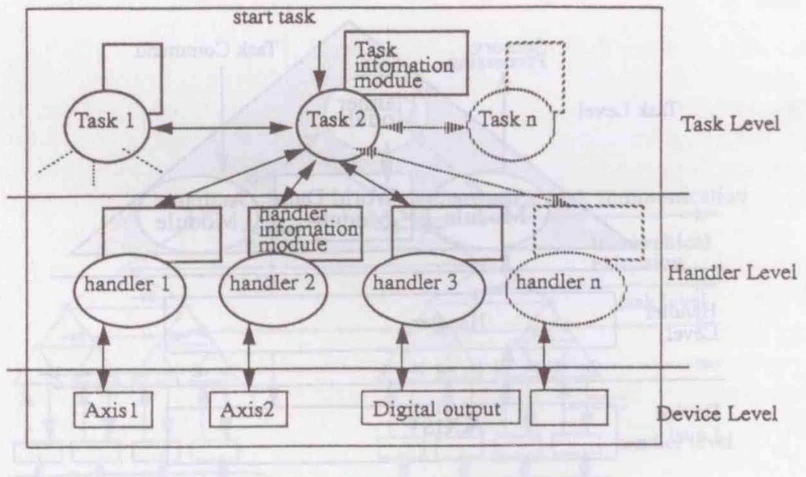

Fig. 5 The current UMC run-time architecture

The Task Level Architecture (TLA) proposed by the authors organizes the control modules so as to create the functional relationships and information flows in Figure 6, this essentially being based on the reference model proposed in, Albus, Mcgain, Lumia [1]. The Sensor Module (SM) processes sensory information to acquire and maintain an internal model of the devices in a machine. Actuator Module (AM) handles problems of motion-planning, task monitoring, control of machines and interfaces with the operator. The World Data Module (WDM) coordinates the above mentioned modules and maintains a knowledge database, keeping it current and consistent. The architecture proposed replicates and distributes the relationship depicted in Figure 6 over a hierarchical and modular computing structure with the logical and temporal properties illustrated in Figure 7. Figure 8 shows in greater detail the required modification and extension of the software architecture. Currently, the UMC handlers, functions and commands are still being utilised. This includes analysing the schedulability of application tasks for real-time systems requiring integrated motion control. This architecture will permit modification of machine control algorithms and a multiple sensing-recognition capability. This will allow multiple machine systems to operate as a distributed motion system in a manufacturing workcells.

The capability for motion planning of modular robotic systems is designed to enable the following:

- Flexible real-time computational power for modular robotic algorithms,
- Easy integration of multiple external sensors through UMC handlers,
- Communication with various input/out put devices and human interfaces,
- Independency from manufacturing machines (including robots) configuration.

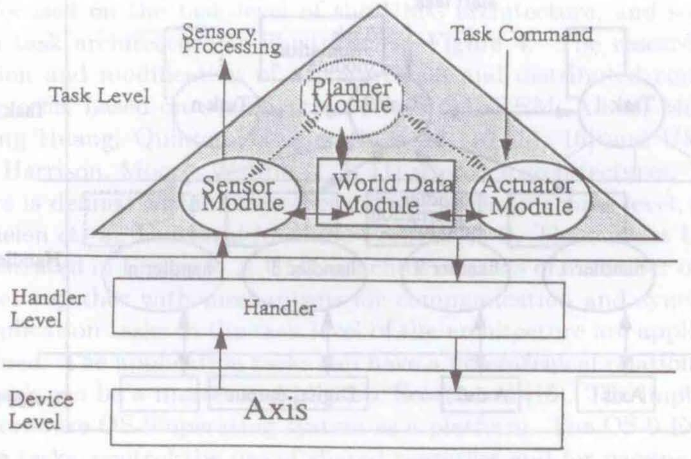

Fig. 6 The proposed elements of task control function (TCF)

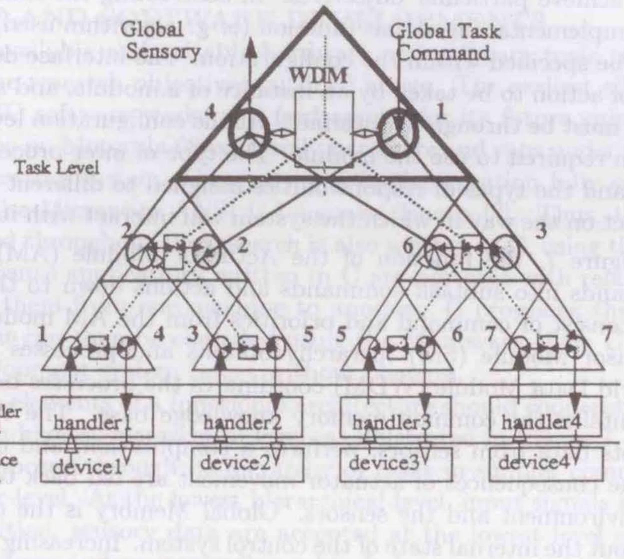

Fig. 7 The task level architecture

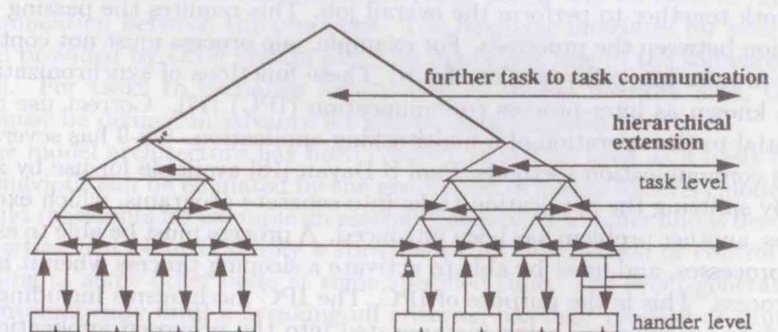

Fig. 8 The hierarchical and horizontal relationships in machine control systems

5 SOFTWARE MODULARITY AND INTERPROCESS COMMUNICATION

Responsibility for different aspects of the motion control problem is attributed within the above architecture in the manner illustrated by Figure 7. Each module will be assigned a given responsibility to achieve particular objectives. In abstracting the functional aspects of a module to enable implementation of that function (e. g. algorithm used), the interface of each module should be specified within the configuration. The interface defines both the type of data and kind of action to be taken by an instance of a module, and all communications with the instance must be through its interface. At the configuration level, this should be the only information required to use the module. The type of inter-process communication between modules and the types of responsibilities assigned to different modules could have a significant impact on the way in which the system will interact with its environment.

As illustrated in Figure 7, the function of the Actuator Module (AM) module is to decompose task commands into subtask commands and actions down to the lowest level. Input to AM module consist of command and priorities from the AM module at the next higher level. The Sensor Module (SM) hierarchy obtains and processes data from the system sensors. A World Data Module (WDM) coordinates the processes between SM and AM modules and maintains the common memory knowledge base. The WDM acts as a controller which accepts data from sensors, perform a computation, and uses the results to move actuators. The consequences of actuator movement are fed back to the controller throught the outise environment and the sensors. Global Memory is the database where knowledge is stored about the internal state of the control system. Increasing the modularity of the hierarchical and heterachical elements of the Task Level Architecture (TLA) plays a crucial role. Here the term 'modularity' is used to indicate a high degree of indepedence among individual elements, excellent reusability and ease of interfacing between elements.

Within the single triangular architecture as shown in Figure 6, one process is composed of a software process. It has the job of forking the other software process, which make up the application, and may themselves fork another process. In a multitasking application, processes must work together to perform the overall job. This requires the passing of data and synchronization between the processes. For example, one process must not continue its job until another process has collected data for it. These functions of synchronization and data transfer are known as inter-process communication (IPC) [17]. Correct use of these functions is essential to the operation of a multitasking application. OS-9 has several different inter-process communication methods, Paul S Dayan [18] available for use by application programs. By splitting the application tasks into separate programs, which execute as separate processes, another problem has been introuced. A process must be able to exchange data with other processes, and must be able to activate a sleeping process when it has data ready for that process. This is the purpose of IPC. The IPC mechanisms including signal, event, pipe and data modules are being incorporated into the proposed application tasks architecture. For example, the OS-9 events will play the role to synchronize the concurrent processes (between SM and AM), so that only one updates the WDM at a time.

The implementation of UMC software uses the Microware OS-9 operating system as a platform, Booth [8]. The proposed functional architecture utilises the concepts of IPC provided by the OS-9 operating system. It provides the mechanisms for task to taks and task to handler communications by utilising the OS-9 events, signals and user data modules. This provide synchronization and data transfer between modules. This research is investigating the functional capability of tasks in a hierarchical as well as in heterachical manner. The proposed architecture requires the use of multiple concurrently executing processes using the UMC handlers. Interprocess communication within the proposed system architecture is supported by a data-passing or message-passing system based on the UMC tasks commands.

6 HARDWARE AND SOFTWARE DEVELOPMENTS

Identification of available and suitable hardware and software tools is very important in order to realise the research objectives outlined above. The earliest effort was focused on realizing how UMC software works and feed results for its future enhancement. UMC is implemented to run on Motorola 68xxx family hardware and runs under the OS-9 multitasking, real-time operating system. Currently, the implementation fully supports application tasks written in the Microware ANSI C language, Booth [8]. Thus, the application task source code created throughout the research is also written in C, using the operating system UNIX. This is because applications written in C are portable with relatively little effort is required to move them from one machine to another. C programs that are written for a UNIX environment can then be compiled using the Microware Ultra C cross compiler and run on OS-9 development system target without changes.

The component elements of a functional architecture should cooperate to realize a global goal defined at the highest hierarchical level, as illustrated in Figure 7. This global task is successively decomposed through its hierarchy of tasks in simpler commands which are to be obeyed at lower level. At the lowest hierarchical level, input signals are provided for the actuators. In addition, sensory data are accepted at the lowest level and use to estimate the results and perform and corresponding subtask, by carrying out necessary planning and execution activities. The UMC handlers which provide standard interface between tasks and external devices, can still be utilised.

The software is being built in a modular and layered fashion, in conformance to the proposed functional task architecture. The main objects of attention are the interfaces rather than the contents of single modules. It is very important to define with precision interfaces or interaction points between the modules. These points will be used for synchronization and communication between the processes. The language interfaces for inter-process communication provided by OS-9 system plays an important role in the software development, Dayan [18]. For tasks to exchange information in shared memory area, the exact data structure must be defined in advance for shared data modules.

Once the model architecture has been created, it will be used as a basis for simulation. Process behaviour can be emulated by the generation of events for every node of the model. A node could represents for example an assembly station or a buffer and is described by a list of its properties. An event is simply a statement that information or control is transferred for processing at some other node at some specified time. The event-generating process is repeated automatically until a breaking-off criterion has been reached or simulation operation interferes. The approach will allow the simulation of real processes to be done either using physical model (e. g. scaled models in device setup) or computer models. Computer models will normally be selected since they have the advantage of making possible experimental examination and analysis which otherwise would be costly and time consuming. After experiments based on this model, the results has to be transferred to the real system and it ought to be guaranteed that these results are useful and collapse system build lead times. The results from the simulation should produce some form of virtual machine system. A virtual machine systems is one in which an actual machine systems is modelled in computational form.

Software programs are being created by utilising UMC functions an then compiled to executable task modules in C programming language. A number of UMC functions are available for use in application task programs. Application task source code is produced using the standard UNIX text editors and compiled using the Microware Ultra C Cross Compiler. Then the application tas objec code is transferred to the OS-9 target. The object code file of a task programme is forked by the UMC machine configuration utility, namely configuration editor (ce). This ce is the interactive terminal screen editor for creating machine, handler, map and profile edit files, Booth [8]. Execution of the programming output is displayed on a UMC Task Window. This task window is set to display real-time responses from various UMC handlers. This also means that the OS-9 configuration, UNIX text editors and UMC task window exist in the same computing environment.

7 CONCLUSION

The utilisation of a functional software architecture based on NASREM architecture gives a promising result in terms of extensibility, protability and software reuse. However, this architecture is modified in such a way that it is in compliance to the concept of multi-tasking and inter–process communication provided by the OS–9 operating system. This is important in order to build a flexible high-performance modular machine control systems which enables efficient implementation and experimental evaluation of different machine control devices using OS-9 operating system as a platform. The approach to 'open' motion and input/output synchronization will be contineud within the UMC reference architecture, which enables the implementation of reusable software processes for machine control. The UMC run–time environment remains on OS–9 development computers. The emphasis of work is on the hierarchical structuring of solutions and will enable the design and control of distributes motion elements. Hopefully distributed motion control in the run–time environment will be realised by exploiting the task level of the UMC reference architecture.

REFERENCES

[1] J. S. Albus, H. G. McGain & R. Lumia, *NASA/NBS standard reference model for telerobot control system architecture (NASREM)*, Nat. Inst. Standard and Tech., Tech. Rep 1235 (1989), Gaithersburg, MD.

[2] J. S. Albus, *Outline for a Theory of Intelligence*, IEEE Transaction of Sytems, Man and Cybernetics **21** (1991), 387–390.

[3] Duelen G. et. al., *An Advanced Robot Control System for Manufacturing Processes*, Annals of the CIRP **40** (1991), 387–390.

[4] –, *Universal Machine Control (UMC), Modular System Group*, Loughborough University of Technology, UMC Summary Document Issue 3, November 18th, 1992.

[5] R. Harrison, P. R. Moore, & R.H. Weston, *Integrated Machine Design and Control*, ACME/SERC Application: Case for Support (1992).

[6] H. Tsukune & M. Tsukamoto et. al., *Modular Manufacturing*, Journal of Intelligent **4** (1993), 163–181.

[7] Patrick, J. Eicker et. al., *Intelligent Systems and Technologies for Manufacturing*, AT&T Technical Journal (1991), 10–22 November/December.

[8] A. H. Booth, & A. J. Carrott, *UMC Reference Manual Version 3.0 June*, 1993.

[9] M. G. Rodd, H. B. Verbruggen & A. J. Krijgsman, *Artificial Intelligence in Real-time Control*, Engineering Application Artificial Intelligence **5 no 5** (1992), 385–399.

[10] Yutaka Ishikawa, Hideyuki Tokuda & C. W. Mercer, *An Object-Oriented Real-time Programming Language*, IEEE Computer (1992), 66–73 October.

[11] R. Harrison, *A Generalised approach to Machine Control*, A Ph.D Thesis, Loughborough University of Technology, 1991.

[12] J. S. Albus & M. Juberts, *RCS: A reference Model Architecture for Intelligent Vehicle and Highway Systems*, ISATA 1992, Florence, 1992, pp. 447–453 June.

[13] Hui–Min Huang & Quintero, R., *Task Decomposition Methodology for the Design of a Coal Mining Automation Hierarchical Real–Time Control System*, 5th IEEE Proceeding Symposium on Intelligent Control **2** (1990), 884–893.

[14] J. S. Albus & M. Juberts, *RCS: A Reference Model Architecture for Intelligent Vehicle and Highway Systems*, ISATA 1992 (1992), Florence, 447–453.

[15] S. Szabo & H. A. Scott et. al., *Control System Architecture for a Remotely Operated Unmanned Land Vehicle*, 5th IEEE Proceedings Symposium on Intelligent Control **2** (1990), 876–883.

[16] B. P. Zeigler, *Object-oriented Modelling and Discrete-Event Simulation*, Advances in Computers **33** (1991), Academic Press, Inc., 67–115.

[17] J. Albus, *Concepts of Reference Model Architecture for Real-time Intelligent Control Systems (ARTICS)*, NIST/NBS Tech., No. 1277 (1991).

[18] P. S. Dayan, *The OS-9 Guru 1-The Facts*, Galactic Industrial Limited.