Proceedings of the Institution of Mechanical Engineers,
Part C Journal of Mechanical Engineering Science, C415/019, pp189-197, 1991
Proceedings of the European Technology Congress and Exhibition, Birmingham

The logical and geometric modelling of a universal machine control reference architecture

R DOYLE, BEng, MSc and K CASE, BSc, PhD
Department of Manufacturing Engineering, Loughborough University of Technology

SYNOPSIS This paper reports on new research into the computer modelling and simulation of the UMC (Universal Machine Control) Reference Architecture. A key factor in the UMC Reference Architecture is the provision of aset of software configuration tools to facilitate the building of modular machines and machine control structures. In this respect the roles of geometric solid modelling and logical control modelling are emphasised in the paper.

1 INTRODUCTION

The UMC Reference Architecture presents a flexible and expandable strategy for modular machine design and machine control design. The software configuration tools to facilitate the building of modular machines and machine control structures are critical if this approach is to be successfully implemented. In particular the geometric solid modelling and logical control modelling tools are necessary in optimising the selection and aggregation of machine elements for new machines. For an established machine where product changes occur the configuration tools will reduce down-time through the application of an 'off-line' programming approach.

2 MODULAR MACHINES

Current trends in manufacturing lean towards smaller batches and reduced product life cycle times (1), which in turn demand a flexible approach to machine design practices. Highly complex machines which solve the problem of flexibility in a generic fashion inevitably have built-in redundancy and hence are relatively expens1ve. An alternative approach is to build machines exactly to requirement working from an array of low level and well defined machine and control modules (2). This approach has many clear advantages such as; rapid design and construction, simplification of the tools for construction, and easy replacement of worn elements. A further and perhaps more significant advantage is the ability of this style of machine to be easily re-configured in line with product changes (3). Such modifications can take several forms. The change could be a software change requiring the editing of a few parameters or perhaps the addition of new software modules. Alternatively the change may be made in the hardware where either the machine modules are relocated or new modules are added. The key is that modularity is maintained throughout the whole machine from the higher levels of machine control to the individual machine elements.

The potential applications of modular machines are broad. They perform well where a machine consists of a number of distributed actuators, and so typical areas of application include assembly, packaging, machine loading/unloading and quality control. Thus, there is a clear demand in manufacturing industry for flexible machinery to which a modular approach is ideally suited (4). However, many companies currently use special purpose machinery commissioned from external sources which inevitably leads to long changeover times between product variations.

3 THE UNIVERSAL MACHINE CONTROL REFERENCE ARCHITECTURE

Current modular machines are typically controlled by plcs (programable logic controllers) with either direct communication between plcs or indirect communication through a micro-computer. However, this type of control is very local and tends to lead to 'hardwired' solutions which lack flexibility (5). Clearly, there is a demand for a high level control which sits as an umbrella over all of the machine control elements. The main function of this high level of control is to provide co-ordination for the various machine control elements, work pieces, tools and sensors.

With this goal in mind the UMC (Universal Machine Control) Reference Architecture has been devised in the Department of Manufacturing Engineering at Loughborough University of Technology over a number of years through SERC (Science and Engineering Research Council) funded research (6).

The solution to this level of control is not trivial since there are many possible machine elements which must be consistently blended together. Even for a given machine element function there are often several manufacturers each with a unique custom control. The term 'Universal' in UMC illustrates one of the main goals of the UMC Reference Architecture to resolve the diverse range of controls into a consistent format in unison with the modular approach to machine building and control. Ultimately the UMC Reference Architecture is intended to become an open industrial standard which will benefit the builders of machines and the manufacturers of machine elements. 'Open' standards are currently believed to be vital across a whole range of computing activities. For example Open Software Foundation for UNIX (7) and IGES for CAD information exchange (8). Similar benefits should accrue from an open control structure which will lead to a larger market with consistent products which in turn must lead to advantages for both the suppliers and consumers of machine modules.

The UMC Reference Architecture describes the control and coordinating elements and the method of arrangement of those elements to form a hierarchical run-time control structure. The principal role of the run-time control structure is to coordinate and drive the modular machine with a number of custom control elements. The software which is generated as a consequence of applying the concepts embodied in the UMC Reference Architecture exists in a real time and parallel operating system, currently implemented under OS-9 (9)(10).

Figure 1 shows a simplified form of the run-time control structure, where the key components are tasks, data structures and handlers. Tasks are processes which run in parallel and describe the operation of the machine. They may also provide other functions such as machine monitoring. There can be many tasks active at any one time and coordination is achieved between them by a system of signalling events. Reference data and data transfer is achieved via a data structure which rests in the active memory of the computer. Access to the custom control devices of the machine elements and input/output (i/o) elements is made via the handlers, which are device dependent interfaces between the run-time UMC system and the device custom control. From the run-time control point of view the handlers are virtual, presenting a consistent interface to all elements in the machine.

Whilst the run-time control has a major role to play it must not be seen in isolation. The run-time control is the goal but the machine designer must be able to reach that goal in an efficient and sensible manner. Figure 2 shows an overall view of the UMC Strategy where it is clear that issues of machine definition, configuration and evaluation are of equal importance to the run time control.

Although it is possible to write and work with the run-time software directly this requires a deep understanding of the detailed workings of the run-time control. Normally machine designers will work at higher levels using configuration tools to aggregate the modules which form the run-time control (11). The overall logical machine structure can be defined using the rules in the UMC Reference Architecture. The individual elements of the control structure are defined by a combination of using design tools and extracting the information from libraries where they exist.

4 A DETAILED VIEW OF THE UMC STRATEGY

For the physical machine the natural focal point is the run-time control. However, the key decisions which determine the characteristics and performance of the machine are made at a much higher level. The UMC strategy allows the focal point to be moved to a higher level through its software based approach to control as opposed to the traditional 'hard-wired' approach.

Figure 3 shows the UMC design strategy. For even a simple machine there are many software and data sources which are required to properly describe the control of the machine. The fulcrum for design is where the aggregation of these sources occurs. The principal function of the aggregation phase is one of organization and reference. The user must describe the mach1ne in a manner which is consistent with the UMC Reference Architecture. The hierarchical nature of the UMC Reference Architecture naturally leads to the organisation of the machine description and high level task programming.

On completion of aggregation there are two possible routes which may be followed. For well established machines the code and data is pre-processed directly to generate the run-time control software. Following this step the software will still need to be proved on the machine but adherence to the UMC Reference Architecture should ensure this work is minimal.

For machine designs which are new or a radical modification of existing designs the logical and geometric modelling tools are employed to compare and evaluate designs. Continuity of design 1s ma1nta1ned by using the same aggregated information for both the modelling tools and the run-time control. As in the case for run-time control the aggregated data is pre-processed to a format suitable for the logical modelling software. The logical modelling software processes the aggregated data and emulates the real time operating system to generate data suitable for driving a geometric solid model of the hardware. The actions of the logical machine are described in two formats. The first is on an event by event basis. The second shows the action of the machine elements on a single time base.

The geometric solid modeller, based on the robotics modelling software Grasp (12), allows the user to describe each machine element from simple shapes such as cuboids and cylinders or alternatively machine elements may be extracted from existing libraries. Figure 4 shows a typical Grasp solid model of a UMC modular machine. The logical structure of the model is derived via the pre-processor from the description given at aggregation. Once complete the designer has the opportunity to manipulate and evaluate alternative configurations of the model. Where machine elements have the capability of motion the designer may move the elements through teach pendant like commands. (This is the technique used in the programming of 1ndustr1al robots where the robot is driven to particular positions using a controller accessed via a hand held 'pendant'. Robot postures may be saved in a sequence which when replayed represents a program.)

When both the geometric and logical models are complete the output from the logical model may be used to drive the geometric model so the effect of the design may be observed and analysed. The analysis may be numerical as well as visual giving the designer the opportunity to make comparative judgments on alternative designs. Since des1gn often requires iteration, the strategy allows the designer to return to the aggregation phase after analysis so that design optimization is achieved. The modelling tools also allow the machine to be balanced in terms of time and operating characteristics so that the parameters of the machine elements are properly specified.

5 AN EXAMPLE

The purpose of this example is to illustrate the main principles embodied in the UMC approach to machine control. The UMC strategy is capable of controlling machines of far greater complexity than is shown here. The example is set out in two parts. The first illustrates the organisation of a UMC machine. The second highlights the flexibility of the UMC strategy when compared with other approaches.

Consider the simple assembly cell shown in figure 5. The pick and place unit must select components from the two automatic feed hoppers and insert them in the assemblies which are delivered by the conveyor. The conveyor stops at an appropriate position as a result of signals from the sensor which detects the position of the assemblies. There are in total seven elements which the UMC system must coordinate; the four axes of the pick and place device, the single axis of the conveyor, the sensor i/o and the i/o to open and close the gripper.

Each moving axis is controlled by a motion controller and these are different for the pick and place unit and the conveyor. The i/o elements have their own unique interfaces.

The UMC system which coordinates all of the above elements exists in a real time environment running under the OS-9 operating system on a single Syntel micro-computer.

The interface between the UMC environment and the machine control elements is achieved via the UMC handlers, which are software modules provided as a part of the UMC system.

The character of the machine is given by a series of task programs which are generally written *in* the C programming language and run in parallel. For this part of the example there are two task programs. The first controls the conveyor in response to the sensor, whilst the second performs the insertion of the components into the assembly. Co-ordination between the tasks is achieved via the signalling of events. In this case the main events are; start, assembly in position, assembly complete and stop.

The cell and its function described above could be achieved using other methods, perhaps by using a plc based solution with an equivalent or slightly lower cost. However, the advantage of adopting the UMC approach is realised when an element of change is introduced.

Currently, it is unrealistic to expect a product to have a long run with no change if it is to remain competitive. Therefore, it follows that machines must be able to respond to this demand for product change. This is the area where the UMC approach shows a significant advantage over other approaches.

Consider the same situation described above 18 months on (see figure 6). The product has been changed so that there are a number of different assemblies requiring the same component insertions but in different locations. In order for the system to recognise the different assemblies as they arrive on the conveyor a vision system has been introduced. Also, the speed of components moving through the cell has been increased by replacing the pick and place unit with a higher specification unit from a different manufacturer.

In a plc type approach this situation is essentially a brand new machine, whereas with the UMC approach the changes can almost be dealt with as a change in parameters. Because the task software is device independent the change in pick and place unit can be accommodated by changing the handlers used. The variation in component location on the assemblies merely requires that additional location data modules be included in the data structure. The introduction of the vision system represents the bulk of the new work but is a relatively straightforward task since it has only to set an event indicating the detection of the assembly type.The Assembly task can then respond to this event by selecting the appropriate location data module.

6 CONFIGURATION TOOLS

When a designer devises a modular machine there are many aspects to consider such as specification of functionality, variability of function, performance in speed and cost terms, selection of modular elements etc. Each aspect is very different from the others in the way that it is viewed and applied, and yet at the same time there is considerable dependency between these factors. To build a machine directly without the aid of design tools requires the designer to have a great deal of experience. The interacting nature of the machine elements makes it very difficult to predict the operation and performance of the final machine on paper and inevitably paper designed machines require modification after they have been built. This approach also tends to result in a conservative approach to design with consequent losses in performance both in terms of cost and machine efficiency.

Initially, the UMC Reference Architecture aids the designer by offering a consistent platform on which to solve control and co-ordination problems. This frees the designer to concentrate on the real problem without having to be concerned whether control unit X will operate with control unit Y in response to sensor z. Following on from the initial specification phase the designer requires effective tools which help measure and compare designs. Under the UMC strategy these tools are provided in the form of the geometric and logical modellers. The tools allow the designer to describe design in as much or as little detail as is required. Designs can then be analysed, balanced and optimised in a dynamic fashion. Configuration tools are used to greatly reduce the amount of uncertainty of performance of the final machine.

7 GEOMETRIC MODELLING TOOLS

The ever increasing availability of computer processing power makes the computer based kinematic solid modelling of machines an attractive proposition. The value, in terms of visualisation, of constructing a solid model and displaying the actions of a machine is that it greatly improves the certainty that designs are correct.

The geometric modelling work in the project is being conducted in collaboration with BYG Systems of Nottingham who produce a successful robotic modelling and simulation system called Grasp (12). This software exhibits many of the features which are required for the modelling of modular machines. As a part of the collaboration a special version of Grasp has been provided so that features which are unique to the UMC approach can be included.

The first phase in making use of Grasp is the description of the physical machine. Machine elements are described by the assembly of a number of three dimensional shapes such as cuboids, cylinders, polyprisms etc. Models can be either defined in symbolic manner or in great detail. Once a model of a machine element has been defined it may be stored in a library for quick retrieval at a later date. Usually symbolic models are used in the early conceptual stages where there might be a number of designs to compare. The detailed models are produced as the design becomes more fixed and deeper explorations of the design are required.

Each element of the machine is defined in its own right and then associated with the machine as a whole. Each element must have a unique name and is placed relative to a reference object by position and orientation. The relationship between machine elements is described in terms of ownership. This relationship is important where one machine element is located on another element which moves. If the second element moves then the first must automatically maintain its position relative to the second.

The model as a whole is described in a data structure so that it is also possible to define data other than the geometric description and position. Often this is data which is required for reference by the kinematic modelling of the design, such as maximum and minimum positions, maximum velocity and acceleration.

Once the model is complete it can viewed from any point to ensure that it is correct. There are efficient editing tools available which allow alterations to the model. Where dynamic elements exist within the model these can be manipulated by making use of a set of robot pendant like commands. In general a machine element can be moved to any position in a range between its maximum and minimum positions. An element may also be sent to a pre-defined position such as a home position. Also, since the model can comprehend solids existing within three-dimensional space it is possible to detect if and where clashes occur between machine elements.

Although the checking of the physical layout is important, ultimately the real value of the geometric model is in the display of the consequences of the logical model. The logical model consists of a set of parallel programs which are written as a set of sequential programs which interact in a parallel fashion. Even with a small number of parallel processes it can be difficult to ensure that the interaction is right. The next section shows how the logical model is defined and how it is interfaced with the geometric model.

8 LOGICAL MODELLING TOOLS

The use of high level languages for the programming of robotics and associated equipment is well established (13) and at first sight could make a contribution to the UMC approach. Typical examples can be found in simulation packages such as Grasp (12) or in more direct robot programming packages such as SRL (15). However, these packages were initially designed to produce a single sequential program and then evolved to include parallel features which coordinate either other robots or elements such as conveyors. Also, the task programs which are produced are often post-processed into code to drive a specific robot.

Both the quasi-parallel programming and the non-generic software target prevent this type of approach from succeeding within the UMC approach. For simulation this role is undertaken by the logical control model and has the goal of emulating an actual UMC system as if it were running in real time. The principal requirements are as follows; concurrent programming, inter-task communication, virtual real time, a common data structure and an interface into the data structure of the geometric model.

The common thread which connects all of the elements of the UMC design strategy is the description of the tasks (or logical sequence of operation) which a given machine must perform. The logical model of a UMC must be directly related to the actual UMC code; firstly to ensure that the model is as realistic as is possible and secondly to maintain a direct relationship between the logical model code and the actual code. However, the same code cannot be used as there are clearly two separate environments, the actual and the virtual, each with their own unique requirements. For example, the time for an actual machine element to perform a move is determined by the physical device and its working context, which may vary. For the logical model a software routine must be provided which emulates the action of the machine and returns a value for the time of the required move.

The UMC design strategy accommodates both the similarities and, differences between the actual and virtual worlds by allowing the user to define tasks at a high level where there is overlap and then preprocessing to incorporate the differences. Users can write freely in C and make use of a relatively small, but powerful set of functions to achieve the UMC style of programming. A brief extract of typical UMC high level code is shown below.

```
{
.
.
while ( notfinished )
  {
  UMC OUTPUT ( rio, lamp, red );
  UMC SIGNAL_EVENT ( riouse );
  UMC WAIT_EVENT ( noboard );
  UMC MOVESLP ( load, fast );
  UMC WAIT_EVENT ( neffuse );
  UMC MOVESLP ( under, fast );
  UMC MOVESLP ( over, slow );
  .
  .
  .
  }
  .
  .
  }
```

Potentially the UMC code can be embedded in any language, but so far C is the only implemented language. The UMC function calls allow the user to provide proper control and co-ordination of the designed machine. The user may communicate and send instructions to specific axes or axis groups. For example, 'UMC MOVESLP (location, speed)' will send an instruction to the axis group associated with the current task to move to a position called 'location' with a 'speed' ratio of the maximum velocity and acceleration of the elements in the axis group. The user may also achieve inter-task communication and co-ordination through a system of events. For example, 'UMC SIGNAL EVENT (task_done)' sets the global event 'task done' which would allow any other-task with 'UMC WAIT EVENT (task_done)' to continue with its execution.

On completion the task programs are processed and compiled according to whether the programs are for use in a simulation or in an actual UMC machine. For an actual machine the resulting C code may be transferred onto the OS-9 system and then compiled or the code may be cross compiled on the UNIX workstation and then loaded onto the OS-9 machine.

The processed C code for simulation is compiled on the UNIX workstation. The task programs may then be run in two modes. The first is an event by event mode for a single task with the position of the machine elements being displayed by Grasp via an interface at the start and end of each event. This may be done either step by step or continuously. This mode is primarily intended for the development and debugging of task programs. A useful function is the ability to declare an unknown position in the task data. This unknown position can then be defined in an interactive manner through Grasp as the task program is run and then recorded in the task data. The second mode is for the display of a number of tasks running concurrently. In this case the simulation runs in virtual real time where the user declares a start time and time interval for an animated type display update to occur in Grasp.

9 CONCLUSION

The research work into logical and geometric modelling as reported here is at the midway stage and good progress has been made. The requirements for such an approach have been specified and many of the lower level constructions have been implemented within the special version of Grasp. Future work will concentrate on higher level aspects such as task programming, and off-line programming of modular machines will inevitably begin to assume equal importance with machine design and evaluation.

The scope of the UMC architecture itself is being widened constantly so as to for example encompass modern concepts such as software cams and gearboxes. These advances, although essential to the evolution of modular machines, are seen as providing a major challenge to the ingenuity of machine designers and programmers alike. Configuration tools such as those described in this paper then become essential to overcome the problems arising from the complexity and sophistication of machine design.

The importance of doing all this in the context of an open systems approach cannot be over-emphasised. The objectives and methodologies for modular machine design and implementation absolutely require a genuinely open approach. We believe that the concepts of the reference architecture implemented via graphical user tools meets this objective.

ACKNOWLEDGEMENTS

This work is being carried out under the Science and Engineering Research Council grant number GR/F 13492. The authors would also like to acknowledge the support of various industrial collaborators, especially BYG Systems and Quin Systems.

REFERENCES

(1) KRUSE, G. Excellence in Manufacturing, Production Engineer, Vol 67 No 4, April, 1988, pp 54-56.

(2) WESTON, R.H. HARRISON, R. BOOTH, A.H. and MOORE, P.R. Universal Machine Control System Primitives for Modular Distributed Manipulator Systems. Special Issue of International Journal of Production Research on Robotics, January, 1989.

(3) CHALLIS, H. Where is Robotics Going ? Automation, UK, May, 1989, pp 23-29.

(4) HARRISON, R. WESTON, R.H. MOORE, P.R. and THATCHER, T.W. A Study of Application Areas for Modular Robots. Robotica, Vol 5, 1987, pp 217-221.

(5) JASANY, L.C. PLCs into the 1990s. Automation, USA, April, 1989, pp 20-24.

(6) WESTON, R.H. HARRISON, R. BOOTH, A.H. and MOORE, P.R. A New Approach to Machine Control. Computer Aided Engineering Journal, February, 1989, pp 27-32.

(7) Anon, Standards One For All. CADCAM International, November, 1988, pp 39-42.

(8) OWEN, W. The Data Format Puzzle. CADCAM International, September, 1986, pp 29-32.

(9) DAVIS, D.F. (Ed.) The OS-9 Catalog January, 1989 (Microware Systems Corporation Des Moines, Iowa, USA).

(10) DIBBLE, P. OS-9 Insights: An Advanced Programmers Guide to OS-9/68000, 1989 (Microware Systems Corporation, Des Moines, Iowa, USA).

(11) BLUME, C. and JAKOB, W. Programming Languages for Industrial Robots, 1986 (Springer-Verlag).

(12) YONG, Y.F. GLEAVE, J.A. GREEN, J.L. BONNEY, M.C. Off-Line Programming of Robots. Handbook of Industrial Robotics, 1985, pp 366-386 (John Wiley and Sons, N.Y. USA)

(13) LAUGIER, C. Les Apparts Respectifs des Languages Symboliques et de la CAO en Programmation des Robots. Robotica, Vol 6, 1988, pp 243-253.

(14) BUSTARD, D. ELDER, J. and WELSH, J. Concurrent Program Structures, 1988 (Prentice Hall).

(15) BLUME, C. and JAKOB, W. Design of the Structured Robot Language (SRL). Proceedings of the Conference on Advanced Software for Robotics, Liege, 1983.

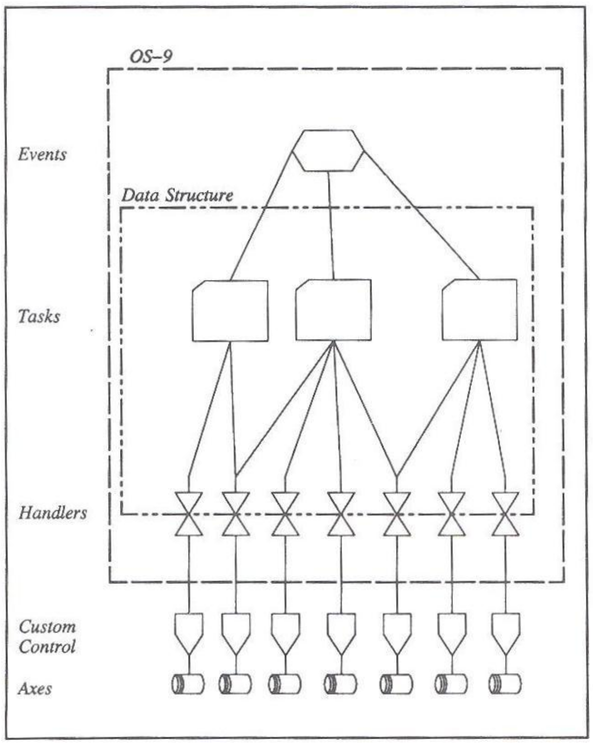


Fig 1 UMC run-time control structure

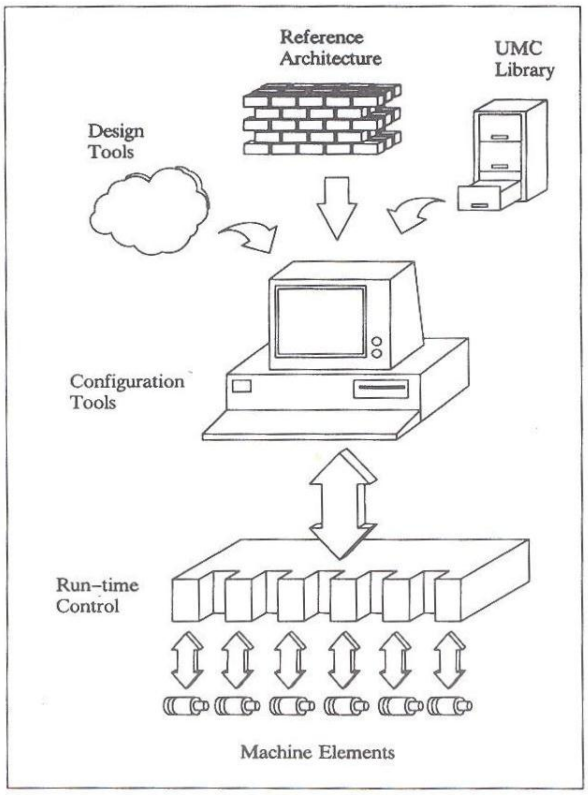


Fig 2 The overall UMC strategy

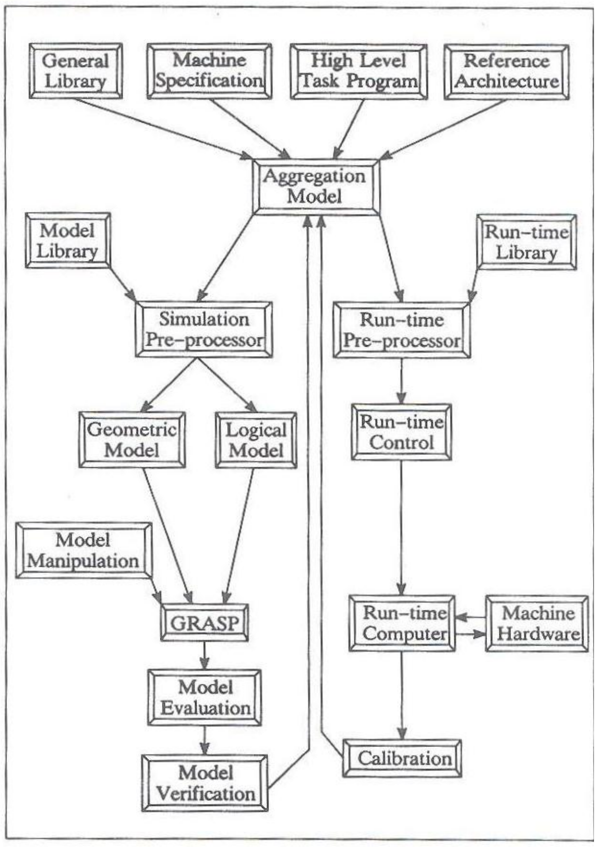


Fig 3 The UMC design strategy

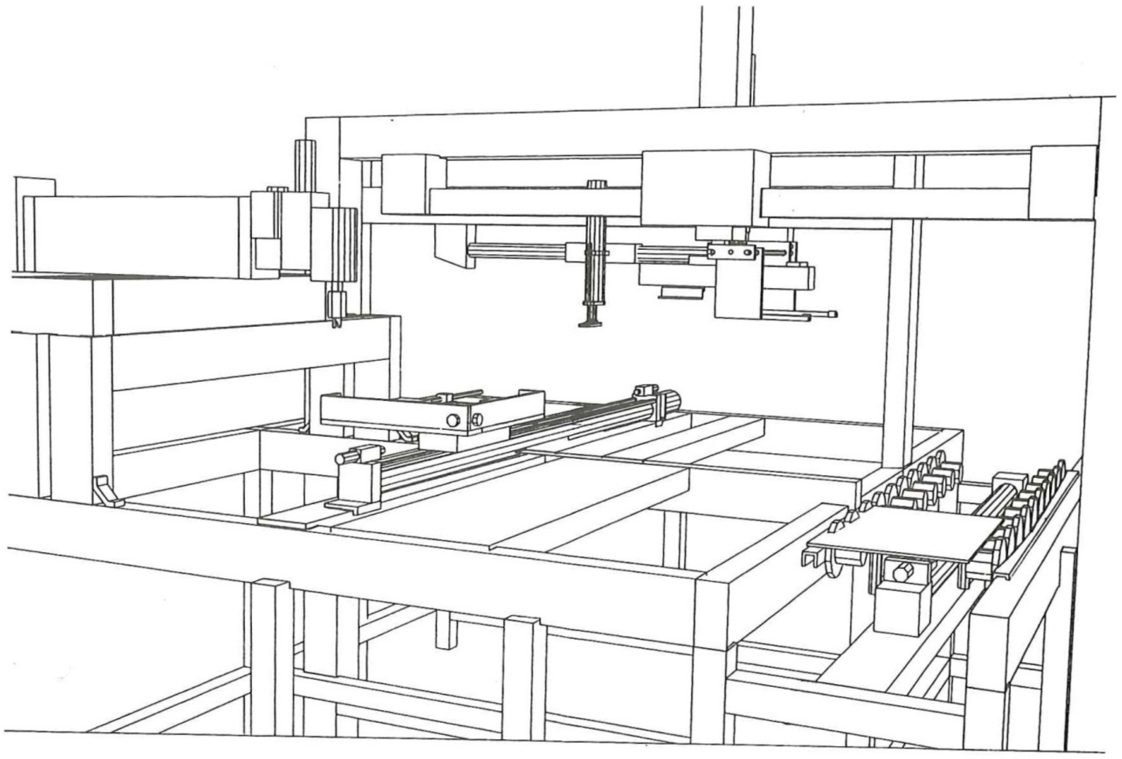
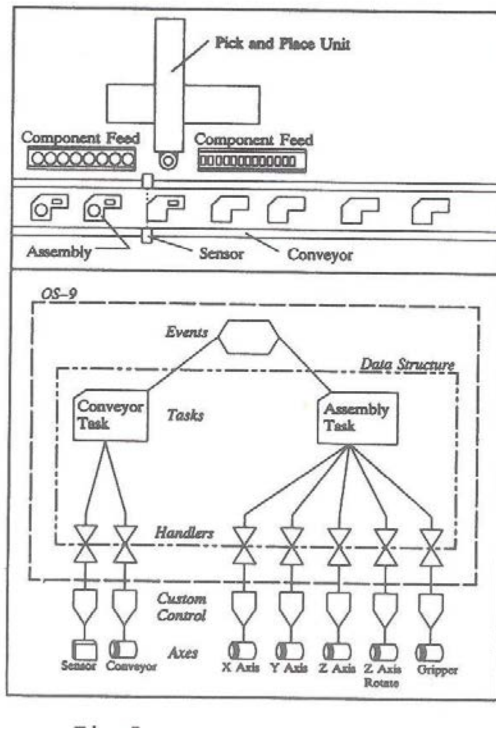


Fig 4 A typical Grasp solid model



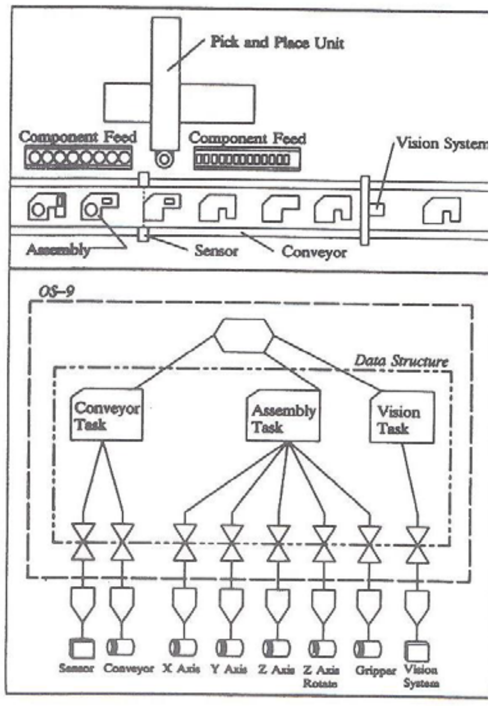


Fig 5 Example part A

Fig 6 Example part B