Reduction of machine tool times through a software/hardware integrated solution

D. R. Bevan* and S. A. Mouton*
*Centre for Rapid Design and Manufacture, Buckinghamshire Chilterns University College, High Wycombe, Buckinghamshire, HP11 2JZ, U.K.

Abstract

Toolmaking is an industry that creates metal moulds, generally of durable tool steels, for producing vacuum-formed and injection moulded plastic parts, which are used in thousands of everyday items such as mobile phones. At present toolmaking is labour intensive with each machining operation requiring manual supervision. The FASTOOL project was a European Union funded collaboration that was aimed at reducing the manpower content of mould toolmaking, and extending the working day. This paper focuses on one element which utilised specially created scheduling and control software that operated an automated overhead gantry system and could remotely start the workshop machines. The software was completely object oriented which allows future proofing by creating new objects for new machines. The results demonstrate that this manufacturing process can be automated, leading to better working conditions for employees and an increase in efficiency and profitability.

Introduction

The toolmaking industry creates metal moulds which are negatives for many of the complicated shapes seen in common use today. In its simplest form a mould consists of a core and cavity, each with half the design cut out of it. Molten plastic is injected into the space between the core and cavity and allowed to cool and solidify. Special ejector pins are used to remove the part from the mould once it has opened. Complex shapes like mobile phones, computer mice and children's toys are made in this fashion. There are several different methods of creating the mould, including standard milling with hard cutting tools, erosion by electron bombardment from a die sinking tool or from a wire (die sinking or wire Electro-Discharge Machining – EDM). Combinations of machining methods are often needed to create the desired moulds.

At present each machining operation is carried out separately under manual supervision. This results in a large amount of idle time and high manual labour costs. A common issue with modern toolrooms is how to schedule the flow of work. If the person scheduling the machines gets it wrong, two moulds could be competing for the same machine tool while others sit idle.

In the Far East where labour costs are significantly lower than in Europe and the USA (by an order of magnitude), toolmaking is a rising industry. To help remain competitive it is necessary for Western countries to shorten production times or reduce the amount of manual labour required for each mould.

FASTOOL was a European Union funded collective project (IPS-2001-80010) in Framework V that aimed to increase the competitiveness of European toolmakers. Issues addressed were the reduction of manual labour content, increasing machine tool utilisation and lengthening the effective working day. An automated facility was constructed consisting of an overhead gantry with a grabber mechanism, and a robot for manipulating moulds destined for the EDM machine.

The hardware was built using existing off-the-shelf components, with the intention to keep the entire cost of the system below that of a standard entry level robot manipulator

currently on the market. By using standard interfaces the system also allows substitutions of some elements to enable custom fitting to a particular user's workshop. In order to control this hardware a specially designed piece of software needed to be created. Ideally this software would be able to extend the working day of the workshop without incurring the penalty of manual labour costs. The software should be able to schedule the work for the toolroom, in an effort to maximise the use of all machine tools. In essence, the software would contain a specially constructed scheduling layer, and a hardware controlling layer. The scheduler would be able to direct the actions of the hardware layer, thus enabling remote activation of the overhead gantry system and the milling machines at specific times.

MATERIALS AND METHODS

Programming language

The core of the software was written using Java from Sunsoft. Java was chosen because of its platform independence. The operating systems that were targeted were 32bit Microsoft Windows e.g. Windows 95, 98, XP and 2000. Some minor modifications to the software will allow it run on any other operating system that supports JAVA.

Due to the complexity of scheduling algorithms a third party application was purchased. This provided a skeleton framework which was built upon to cater specifically for the needs of modern toolmakers.

Identifying a work item

In order to design software that will automate a toolroom, it is essential to understand how toolmaking activities are organised. One tooling order for example can consist of many machining operations. It is often the case that a core or cavity is partly milled, then moved to an EDM machine to cut fine detail, and then returned to the milling machine to produce other features.

The work is split into three distinct software objects. The first is referred to as a tool job, and it simply refers to a particular order. It is given a name by the designer and automatically assigned an ID number from a database. The second software object is the part. Each part must belong to a job, but there can be any number of them. Again the designer will enter a name to identify the part. A typical example would be one part referencing a core, and another part referencing a cavity. Finally there is the definition of the process. Each process must belong to a part, but there can be any number of them. The process refers to a single machining or finishing operation. An example of a toolmaking assignment broken down into constituent software objects is shown in **Figure 1** below.

In order to gain quick and reliable access to the data it was decided that it should be stored in a database. Initially the program was created with a Microsoft Access database holding the information. This was seen as a piece of software that many toolrooms would have installed. For heavy load users, the program was also configured to run from a Microsoft SQL Server 2000 database. The SQL Server 2000 program provides vastly increased performance time, but is costly to purchase. A simple change will allow the program to run from SQL Server 2000 if available.

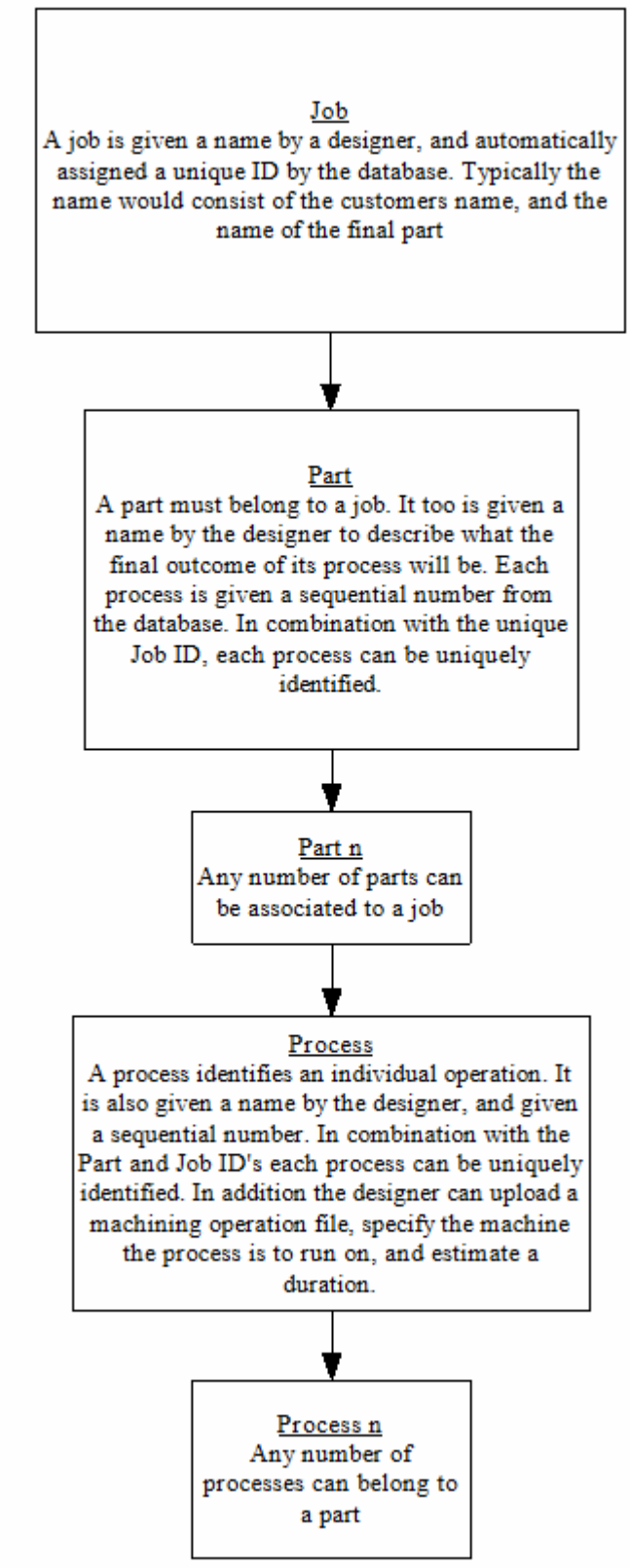

Figure 1: Job hierarchy as used by the FASTOOL scheduling software. The Job unit collates all the individual parts of the tooling job, and in turn each part maintains links to all the machining processes required to create it. The number of process to each part is unlimited, as is the number of parts to a job.

Program structure

There are five main parts to the scheduling software. These are: an Administrator interface, a Designer interface, a Technician interface, the visual job scheduler and the process manager.

As with most modern programs that will have multiple users, there should exist an easy method of setting access rights on an individual user basis. Three key roles were defined and differentiated: Administrator, Designer and Technician. The Designer is the person who inputs data about the machining job, specifies which machine tool the work is to be performed on, and an approximation of the time the machining operation will take. The Technician function is split into two levels. Both have an interface which shows them a Gantt view of the current work schedule. From here they can view information about any scheduled job, such as the location of its machining operation file, its starting and finishing times etc. The higher level technician has extra facilities. They can re-arrange the schedule simply by dragging any job to a new starting time, or they can force an entire job to the front of the current work schedule. This last feature is seen as particularly important in toolrooms because customers often desire their work be completed earlier than initially requested. **Figure 2** shows the Technician interface with some parts in the schedule. The final user type is the Administrator. The Administrator role can encompass designers or senior technicians, with rights to create, modify and delete user accounts.

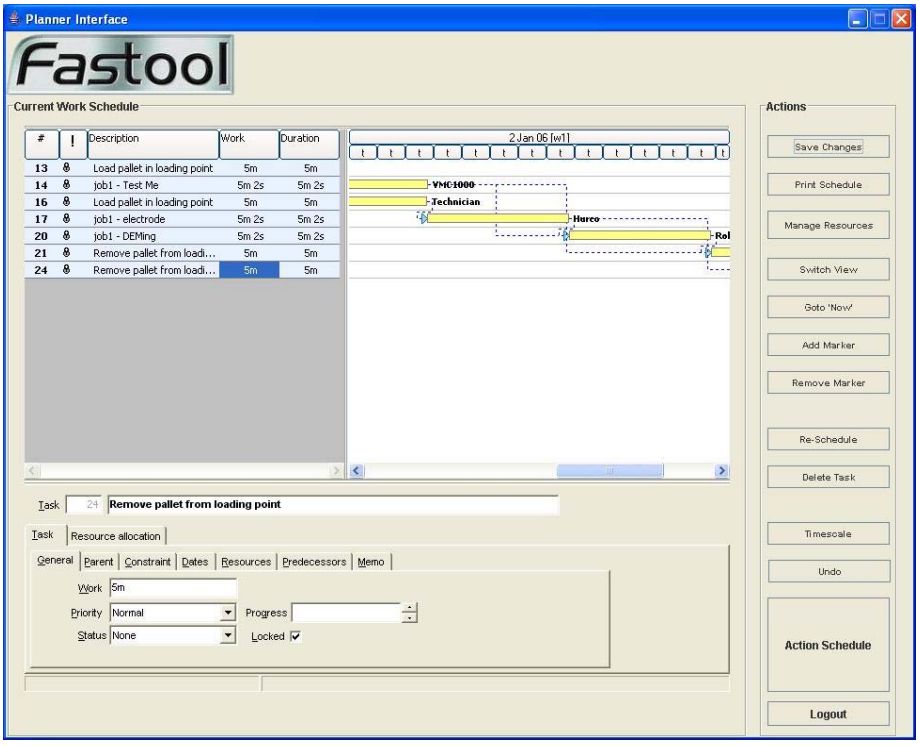

Figure 2: A screen shot of the FASTOOL program running in Technician mode. The name of the parts can be seen in the table on the left hand side, and a visual Gantt representation on the right. Further details can be accessed by the controls on the bottom and right hand side of the window.

One of our design goals was to reduce the amount of on-screen clutter typically associated with modern complex programs. To make the system easy to use the interface eschewed the use of file menus and replaced as much as possible with buttons on a graphical

user interface (GUI). The Technician users do not even need a keyboard to operate their interface.

The Scheduler

The third party scheduler component of the software contained a very useful grounding for our extended work. One of its most important features was the ability to set working periods for every resource in the toolroom. For the machine tools this might be set to 24 hours a day, 7 days a week. However for technicians working in the toolroom this would need to be reduced to 8 hours a day, 5 days a week. In addition to this it is possible to set extra working periods, for example if it is observed from the Gantt chart screen that work will not be completed on schedule, a technician could be requested to work over the weekend to expedite matters. Ordinarily, the technician would be asked to report for work for an undetermined length of time, which could introduce a large surcharge on the labour cost. The schedule however will be able to specify exactly how long the technician would be required for and when. So powerful is the software system, that one could even set individual lunch breaks for the technicians in the toolroom. As well as the large level of control over resources, the scheduler allows users to set constraints on particular processes. For example a process could be tagged with a 'must finish by' constraint. Once this is applied the scheduler will try to ensure that the process is indeed finished by that deadline. If it proves to be impossible given the current work load, the scheduler will notify the user. This can prove invaluable in helping to decide whether to bring in technicians on overtime, or to contact the customer and inform them of the impending delay. There were a large number of modifications that needed to be done to the third party software. The module had been designed as a closed program, so the ability to communicate with Java had to be created from scratch. In addition to this several issues and memory leaks that had to be fixed.

One of the main issues with scheduling software is the amount of time it takes to process a large volume of work. Although the computerised system is faster than a human by a large margin, it could still be inconvenient if the procedure were to take a long time. On a 3GHz Pentium 4 computer it took just under 5 minutes to reschedule 2,000 processes. In order to best suit a dynamic toolroom environment, the scheduler was programmed to only schedule a week's work at a time. Any processes that have a start date longer than a week away are ignored. However, it has the flexibility to be able to schedule over a longer period, if required.

It should be pointed out at this stage that whenever a new set of processes are entered into the Gantt chart and rescheduled they will be optimally scheduled, based on the current schedule. So the initial sequencing of the processes will make for an efficient workflow, and this will only degrade if the tasks in the current week's work do not conform to the schedule.

The process manager

The process manager is the real heart of the FASTOOL automation software. This controls which messages go to which machine tool and manages the state of the entire system to prevent bottlenecks and conflicting calls. The process manager consists of many separate modules. One of the most important of these is the one that controls the overhead gantry. It is absolutely vital that once the gantry is started on an operation, that it sees it through to completion. If a request to move the gantry is received whilst the grabber is inside a machine, it could cause immense damage to that machine and the support structure. The manager also

contains references to each of the resources available in the toolroom so that it can request machining operations from them.

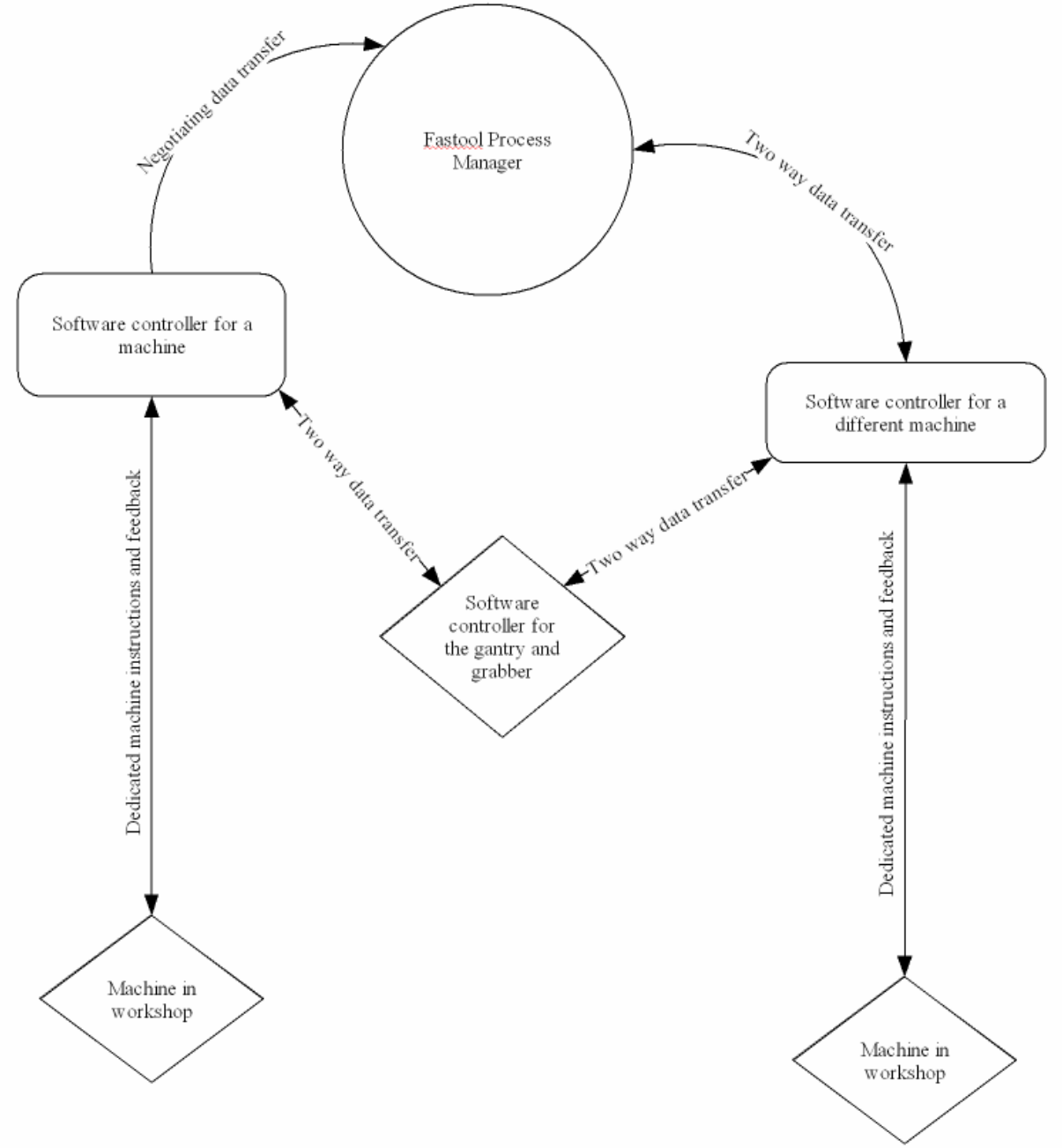

Figure 3: The process manager has two way communications to software controllers that in turn have two way communications with the machines they control. All software controllers have access to the gantry and grabber object which organises and manages it's own workload. New software controllers can easily be added to enable the software to keep up with current hardware.

The system was designed to be fully updatable as new machines become available on the market. In order to do this a certain amount of abstraction is called for when designing the representations for each machine. The manager will only request that a machine object commences a job and tells it the location of the machining code it will need to use. The machine object handles requests to the gantry, e.g. picking up a work piece and loading it into the machine. In order to keep consistency, a base class was created for machine objects. This

class is able to request a gantry operation and also notify the manager when it has finished. Every new machine object that is created must extend this object so that it inherits all the properties needed for integration with the process manager. The only change between each object therefore is how the machining information is sent to the machine. **Figure 3** shows a representation of the FASTOOL system.

Schedule processing

The process manager constantly checks the database of processes to see if any of them need to be started. When it finds one that is scheduled to start in the next minute, it negotiates transfer of the relevant data (CAD file, estimated build time, etc.) to the software object that controls the required resource. The software objects for the resources deal with every process in a strict first come first server order. This ensures that the schedule is executed as designed. By distributing the load to other software modules, the main process manager never becomes burdened with controlling many processes. This feature means that the FASTOOL software is a scalable solution. Once the software object for the resource receives the message it requests that the overhead gantry load the hardware in the toolroom with the specified pallet. If the gantry is busy, the request is added to the gantry's queue of jobs. Again this queuing ensures that the order of the schedule is not disrupted even if the execution time is. Once the gantry has successfully loaded the pallet into the hardware, the software object locks the pallet to the machine bed pneumatically. The milling operation code is then transferred to the hardware as dictated by the software object, and the operation commences. Upon completion the software object requests that the gantry remove the work piece and then processes the next job in its queue.

Hardware interfacing

The overhead gantry was built by using off-the-shelf components. This enabled us to procure robust units with good communication abilities at a relatively low cost. Essential parts that needed to be monitored by the process manager were: the location of the gantry on the overhead system, the height of the grabber, the state of the grabber (open or closed) and the speed of descent of the grabber. All of the devices that reported this functionality could communicate via RS-232 or RS-485 interfaces. Most modern computers only have one serial port, and it would be difficult to manage a wire that trailed all the way around the overhead system. To overcome this situation a serial port hub was employed that could translate RS-232 messages into TCP packets. This enabled communication to be made via TCP/IP using a standard network interface card. In addition, a device that can send TCP/IP packets through the electric cable powering the gantry was purchased, thereby negating any need to manage trailing cables around the overhead system.

Older milling machines tend to have RS-232 communication ports, and new models have Ethernet ports. Ethernet permits rapid and easy transport messaging from the base computer to the target machine tools. In the case of legacy RS-232 controllers a host computer was connected to them, and a small relay program was run on the computer which would accept messages from the main program manager, and pipe the instructions out of the serial socket to the machine tool. Although this may appear to introduce extra expense, almost all legacy machine tools have a computer attached to them to send machine code anyway, so the FASTOOL system does not require additional hardware purchases.

The majority of toolroom machines can be loaded from the top using the overhead gantry. For some machines however this is not possible. The Charmilles EDM machine is a good example of this. To enable the overhead gantry to work with machines like this a custom

robot was built especially for the project. The robot had a moving conveyor belt that could accept a pallet from the overhead gantry, and then pick up and manipulate the pallet so that a side loading would be possible. This robot was controlled via a USB connection. **Figure 4** shows a schematic representation of how the gantry ran around the toolroom, with all the machines available for top loading.

RESULTS

The aim of the project was to produce a working model capable of demonstrating that the concept was sound. To this end, the goal was achieved.

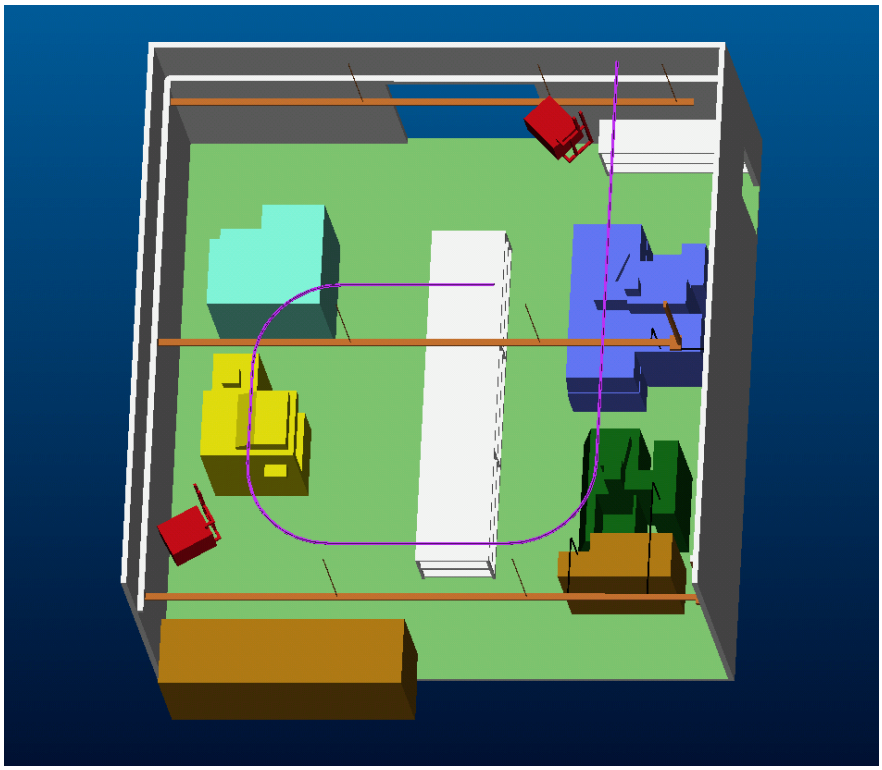

Figure 4: A schematic representation of the gantry rail (purple) running around the toolroom. The dark blue machine represents a Bridgeport VMC1000, the green machine a Hurco BMC 20 and the yellow machine a Charmilles Roboform 31

A demonstration machining job was constructed that consisted of two processing stages. The first stage would entail one process - machining on a Bridgeport VMC 1000 milling machine with a modern Heidenhain machine controller[1] (which communicates via Ethernet). The second stage would consist of two processes, the first being a milling operation on a legacy Huro BMC 20 milling machine (which communicates via RS-232). The second process comprised a simulated EDM operation using a Charmilles Robofom 31 (RS-232 communication). It was only possible in this instance to simulate the EDM operation as it was not possible to get the Roboform 31 EDM to accept remote messages within the timescale of the FASTOOL Project, even with significant on-site help from Charmilles engineers. Toward the final stages of the project it became apparent that communication with the Charmilles would be possible, but there was insufficient time left to implement and test it. However for loading the EDM machine it was still necessary to utilise a specially commissioned robotic handler (which communicates via USB). Although this exercise was simple in terms of the number of jobs involved, it still demonstrated the capacity for the Scheduler to perform and synchronise multiple tasks.

The Scheduler

The software proved to be robust and did not make any communication errors to the hardware, nor did it try to activate more than one process on one machine at any one time. The processes were started within 5 seconds of their intended start times. It was also possible to test the robustness of the algorithm by reducing the estimated milling time of the processes to ten seconds. This was not sufficient time for the work to be completed. However, the software perfectly handled the fact that the allotted time had overrun, and executed the remaining schedule, in the correct order.

Of particular importance was the way the software controlled the overhead gantry. The hoist could lift up to 500kg but was limited to a maximum of 250kg in the trials. Hardware and software controls were implemented to ensure safe operation and collision avoidance.

The scheduler provides great flexibility for future iterations. For example, the Gantt screen view could be updated every few minutes, with progress bars being reported from each of the machine tools. This would allow the Technicians to observe if the estimated time of the job was accurate, and if not, they could reschedule the remaining tasks efficiently. One of the major problems of having a distributed control system, i.e. computers that were controlling legacy machine tools being connected to other computers, then via the network to the controlling computer, is that on a system reset each machine tool needs to be restarted in the correct order.

Machine tool hardware control

The Bridgeport VMC-1000 uses an advanced Heidenhain controller which can report a large amount of information about the machine tool operations. It also allows simulated user key presses. Heidenhain supplied an ActiveX control which could be called from the program manager whenever a process for this machine tool arrived. The first step was to transfer the specified file from the local computer to the Bridgeport's hard drive. Once the file was transferred it was possible to simulate the key presses of the user to set that file as being the one to load, and then start the milling operation. As soon as the milling was completed, the ActiveX control would fire an event to let the operator know. This proved to be very easy to implement.

The Hurco BMC-20 was a lot more difficult to activate. Although this machine tool allowed file upload via RS-232 it did not allow the simulation of any button presses or the sending of any non-milling commands to it. To overcome this problem, a small program was created that constantly sends a wait command to the Hurco. When the BMC-20 software object needs to send milling instructions to the hardware, it inserts the code amongst the wait commands. To all intents and purposes the Hurco is only ever processing one job at a time. But the contents of that job are dynamically created by the small sending program. Since the Hurco does not inform the sender that it has finished the milling process, it was necessary for us to devise a monitoring method. When a job is sent to the Hurco, it stores the received information in a memory buffer. Each line of code that is sent takes up a certain amount of this memory. Because the length of each line of code sent to the Hurco is known and because the size of the onboard memory is also known, it is possible to calculate when the controller has processed an entire buffers worth of instructions. When the system has finished sending milling instructions and the number of bytes transmitted matches the size of the memory

buffer, the system knows that the Hurco has finished all the milling commands, and is now executing the wait commands. In other words, all milling operations have ceased on the hardware. With the Hurco BMC-20 this process takes under 3 minutes, so it has very little effect on increasing the milling time for the process.

Robot

The robot is controlled via a USB interface which was connected to a remote computer. This computer had a re-routing program on it, much like those controlling the RS-232 machines. The software that controlled the robot was embedded into the EDM controller object. This ensured that the scheduler could issue commands to the machine just like it would any other device. Whenever the EDM machine received instructions from the scheduler, it would co-ordinate the robot by itself. The robot performed as expected proving that the software objects allowed any amount of abstraction to be handled by the FASTOOL scheduler. In essence any number of additional manipulators could be added to any machine, as long as the software module is programmed correctly to deal with the additional mechanism.

Storage bays

Some advanced logic had to be incorporated into the scheduler to prevent lock-ups. It would be possible for the scheduler to have a work piece being machined on resource A, with another piece being machined on resource B. When A finishes it is to go onto machine B, and when B finishes it is to go onto machine A. The scheduler keeps track of which resources are loaded, and which are not. It would therefore not unload the first piece until resource B was free, and vice versa. To stop this from happening storage bays needed to be introduced. These are small areas that are located under the gantry that have sufficient space to hold a work piece. As soon as a resource has finished its machining operation it requests that the gantry move the work piece to a storage bay. **Figure 5** below is a picture taken whilst the FASTOOL machinery was operating under the demonstration schedule.

DISCUSSION

Software

The software scheduler was optimised to schedule jobs based on resource (machines, technicians) availability. Extra logic was needed to ensure that the scheduler also took into account the number of pallets loaded into the system, and how many storage bays were free. The programming of this was not trivial, as the number of pallets loaded into the system is constantly changing through the scheduled work. It was also required as a security feature that the system knew exactly where each pallet was at any given time. This information was written to persistent storage in the form of a database. In the event of a power outage, the system would be able to recover from where it had left off (although part machined work would probably need human intervention). The tracking also enabled the scheduler to be aware of whether a machine was already loaded with a pallet or not. Whilst a machine has a pallet on it, its computer model is considered locked by the system and will not accept any requests placed onto it. With this logic applied it would be impossible for the system to try and stack one pallet on top of another.

The logic of the scheduler was significantly more complex than expected. The third party scheduler had been purchased to alleviate the development of this particular module; however major changes still needed to be made to ensure that it could reschedule with additional constraints imposed. The distributed communication between the FASTOOL

software objects is incredibly flexible and allows detailed data transfer. It would be possible to have the FASTOOL program visually show the process percentage completed on the Gantt chart. By monitoring data such as this it could also be possible to have the scheduler automatically reschedule itself should a particular job take far longer than expected. With the right level of monitoring, it is theoretically possible to have a fully self-updating system.

Figure 5: A photograph of the finished FASTOOL grabber and trolley in action. The commands are being relayed via a Windows PC in another room of the building.

Storage bays

The storage bays were added to prevent machine lock ups (pallets competing for a locked resource), but they also introduced an additional benefit. Manual operators (technicians) are treated by the scheduler as being resources in exactly the same as the machines are. The main difference is that technicians have a limited work schedule applied (Monday – Friday, 9:00 – 5:00) whereas machines don't. In a workshop that doesn't have storage bays, a schedule that is to be run over the weekend could only have one pallet going through the system at any time, unless either a technician came in on the weekend just to load the pallet, or another entry point to the system was created. If has to be employed to come in for ten minutes just to put a pallet into the system, the object of automation starts to break down.

By forcing the scheduler to move a pallet from any resource to a loading bay, the automation can be helped whilst ensuring smooth running of the system. Imagine the same scenario as before but this time the workshop has storage bays. Friday afternoon can be spent by the technicians loading work pieces into the system which are then taken to storage bays. Even if the machined work won't start for several hours. Once in the storage bays, the system has complete control of when it can pick the pallets up, thereby negating the need for a technician to load the pieces over the weekend.

Improving the grabber

The grabber for the overhead gantry was engineered to compensate for pendulum motion when lowering, safety when moving a pallet around and safety when raising and lowering. To achieve these features the grabber was controlled via a small microcontroller which was built and programmed by us. This microcontroller was mounted into the top of the

hoist and its RS232 communications port connected to a TCP/IP server. Switches located on the grabber confirmed to the FASTOOL software whether the grabber was open, closed, loaded or unloaded. Unfortunately our inexperience in microcontrollers led to us underestimating the amount of noise that would be generated on a 5m stretch of cable from the switches to the controller. The scheduler often reported instances of switches being pressed simply because a small voltage had been detected on the line. Despite efforts to overcome this programmatically, it was still present. However, this problem is easy to overcome with the right electronic equipment.

Improving the software

The FASTOOL software in its current state is advanced, robust and reliable. The core of the program is largely future proof due to its usage of separate objects to control the various machines in the workshop. Updating the system whenever new machines are added to a workshop would be a simple and quick process.

The interactions between the machine software objects and the main logic centre currently exist only to allow simple two way communications about when a job should start, and when it has finished. Extending this would allow many possibilities. The most useful would be rapid updates on the state of a job. A progress bar could be used to get a dynamic view of the workshop containing up-to-the-minute information. This architecture also enables the possibility of a massive network that is easily scaled and maintained.

CONCLUSION

Although there are many scheduling software solutions available, very few offer any level of hardware connectivity or control. Those that do require expensive additional machinery[2]. With the majority of robotic manipulators it is necessary to re-arrange hardware to suit the robot, rather than a toolroom. In contrast the FASTOOL overhead gantry can be fitted to suit virtually any toolroom. The FASTOOL software has proven that it is possible to automate a workshop that is difficult to manage, and contains legacy machines. As long as a device has a computer communications port it is theoretically possible for an object to be written to control it. For machines that adopt the same communications standard e.g. the Heidenhain controller, additional software objects are not needed. The flexibility and scalability in the software means that although it was written specifically for the tool making industry, it can be applied to any workshop that has a combination of manual labour and machines.

References

1. http://www.heidenhain.com/?/c:2238;177,Products+and+Applications:479/c:479;177,Controlling+Machine+Tools:583.html
2. WorkShopManager (http://www.system3r.com/)