A GENERIC MAN MACHINE INTERFACE

K. Fearn
Marconi Simulation
A Business Unit of Marconi Instruments Ltd.
Nr. Dunfermline, Scotland

ABSTRACT

The technology of Instructor/Operator stations has migrated from mechanical assemblies of meters, lamps, buttons and switches to computerized displays and controls based on graphics systems. These systems still seem to be produced using 'one-off' techniques. This paper looks at the characteristics that could be expected of a generic man machine interface that would be suitable for a range of training tasks. The difference between requirements for uses such as an Instructor's workstation, an Operator's workstation and a CBT student's workstation are examined.

In particular, this paper considers characteristics such as:-

Ease of use
Is this really a consideration?

User efficiency
How can a user get the best out of the system?

Context adaptability
Can the system fulfil the needs of many different users?

Re-configurability and applicability of the solution
How can a system be designed which is applicable to many different tasks?

Integration with data sources
How can a generic man machine interface be connected to a training system and what services need to be provided?

Cost-effectiveness
How can a cost effective solution for a wide range of tasks be provided?

The paper develops an architecture for a generic man machine interface which can satisfy the criteria discussed in this paper. Some 'standard' software and hardware is examined to see how it fits into this architecture and how much of the solution it can provide.

INTRODUCTION

The reasons for building a Generic Man Machine Interface (GMMI) are the classic ones for producing all standard components:-

1. **Timescales and Cost**
   A significant part of every simulator and trainer is tied up with the MMI, so any efficiencies in its design and production will benefit many projects in a large way. The benefits are reaped in the form of reduced timescales, reduced effort and hence reduced costs.

2. **Bigger and Better**
   As simulators and trainers become more ambitious and grander in scale, so they require bigger and better instructor interfaces to control them. This can be achieved by throwing more instructors at the task (at great expense in through-life costs) or by making better interfaces. In general better products are produced by standardized approaches based on proven methodologies and powerful tools. One of the prime aims of the GMMI is to make it easier to produce efficient interfaces which require fewer rather than more instructors.

3. **Consistency**
   The consistency of approach that a GMMI can bring to bear on the design of an MMI makes it *an ideal vehicle for supporting the concepts and* intentions of the various standards which are emerging for MMIs.

In order to be able to define the characteristics of a GMMI it is necessary to examine a representative range of the systems to which it could be applied. The classes examined are Instructor Operator Stations (IOS), Student Training Positions (STP) and Computer Based Training (CBT) workstations. The following descriptions are based on a wide range of trainers and simulators drawn mainly from the military field but also from the industrial and energy fields.

INSTRUCTOR OPERATOR STATION

A typical training simulator has the architecture shown in Fig 1.

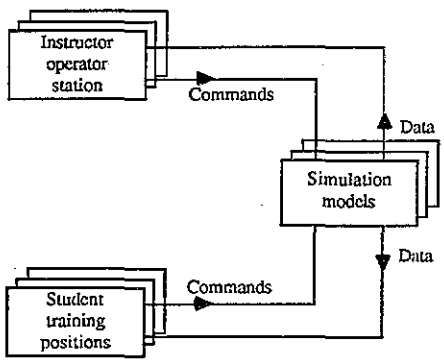


Fig 1 Architecture of a typical training simulator

In this architecture, the simulation models are the sole providers of data, with the IOS and STP providing different views of that data. In general, inputs at the STP and IOS either request data changes or presentation changes which can be made locally.

The personnel involved with the IOS are drawn from many disciplines. They range over subject matter experts, training experts, hands enlisted to man a console, and maintenance and support staff. There is a wide gulf of previous experience and expertise between the extremes of the various users. Also, a user may be deployed for anything from a few months to a few years thus widening the gulf between operators on a system.

The various IOS user's duties and the required displays include the following.

Maintenance of the System

The maintenance and support staff require an MMI which will allow them to locate faults rapidly and make effective repairs in a minimum time. This interface cannot be considered separately from that needed by the trainers since much of the vital diagnostic information is obtained during training exercises.

Configuration of Training System

Many training systems are reconfigurable to address different training objectives and to accommodate varying numbers of students. This is often carried out by the support staff and must be carried out quickly since any time spent in reconfiguration is considered as lost training time.

Preparation of Training Material

A great deal of the training staff's time and effort is spent in defining and preparing training material. This ranges from determining objectives to designing and building the exercises. In a well-designed system, the 'off-line' preparation system shares a lot in common with the actual training environment since all the pre-prepared material has to be validated by testing in the simulator.

Monitoring of Students

In order to control the exercise, the student's actions have to be monitored. A single instructor with a number of student operators in an advanced simulator can find himself with an enormous workload. Each student is required to manage the information coming in to his simulated workstation. In modern systems, this *information is often a very high workload for one man. If* little consideration is given to the instructor, he may find himself being expected to carry the workload of many men simply by trying to monitor many men.

Control of the Exercise

The exercise has to be able to be modified interactively to react to the student's actions so as to gain the best training advantage.

Debriefing the Students

It is usual to follow a training session on a simulator with a debriefing session in which the trainer can show the student how he performed and how he can improve his future performance.

Maintaining Records and Producing Assessment

Training without targets is aimless training, and the setting of targets almost always means maintaining records of achievement levels.

Displays and controls are required for all the various tasks. In particular the trainer may require the following types of display.

Strategic Map Display

Military simulators in particular require a birds-eye view of the training exercise. This is typified by being a

minimum of an XY or lat/long grid which is used as a visual aid to place symbols which represent the positions of important movable features of the simulation, like targets and buoys. There is often the added requirement for a coastline and other static features. The display scale can usually be changed. This is not the same as zooming a graphics picture since the static features rescale but the movable features are usually displayed at a fixed screen size irrespective of scale. Problems of cluttering of the screen usually require that classes of features can be selectively displayed. In many of the simulators this is a very commonly used display method.

Fault Injection Display

The recognition and handling of fault conditions is an essential factor in many training plans. The instructor can inject faults from the IOS using a variety of techniques, from command line input through to pointing to fault injection points on a plan of the equipment.

Tabular Display

When the instructor is looking at a lot of data it is often necessary to display it in a spreadsheet-like format so that similar data is neatly tabulated. This may not be the most readily assimilated layout but it is certainly true that large quantities of data can be accommodated with this approach. The ability to be able to highlight selected rows and columns can greatly enhance the legibility of the data.

Mimics Display

Plant mimics are often essential in helping the instructor to keep a clear idea of the current state of the process being simulated.

Student Monitoring and Exercise Control Display

The instructor's primary task is usually to train the student, but he often falls short in this task because of the excessive workload due to simply servicing the demands of the simulation to make course changes on targets, to change the wind speed and do the thousand other secondary tasks. Many of these tasks are amenable to some automation, even if only at the level of prompting. For example, intelligent targets which follow pre-planned courses or respond to the student actions are a good example of a technique to reduce the instructor's workload.

Student Repeat Display

When the student uses real equipment, there is still a common need to provide the IOS with a repeat so that the instructor can monitor the student. This can take the form of a simple tabular display with all the data presented as text. However, when the student uses a graphics screen for his display, the instructor can use the self-same picture for monitoring. His command interaction is however usually grossly different. For instance, the student may operate all the controls, whereas the instructor will not. Some functions cannot be addressed satisfactorily by the use of soft technologies and so the instructor often needs hard repeats of student displays. In addition, an intercom/rt is a common feature, often with the inclusion of facilities to modify the user's voice characteristics so that he can play the roles of many participants in conversations with the student.

STUDENT TRAINING POSITION

The users of the STP are as varied as the users of the IOS, and the simulator maintainers and support staff also need to operate it. The students may be experienced users who make regular use of the system for practice sessions, or they may be totally new to the training facility and the skills to be taught.

Clearly the opportunity for the use of generic technology is somewhat constrained by the training objective. At one extreme only 'real' equipment will do, whereas at the other extreme a personal computer with standard keyboard and mouse is perfectly adequate. In between these two extremes there are shades of solutions for which graphics screens mounted in special desks will provide a good compromise between realism and re-configurability. For the sake of this discussion, only those systems whose training objectives allow for the use of soft graphics representations are considered.

The STP is subject to many of the preceding comments on the IOS but introduces some special problems of its own which are then directly reflected back into the IOS. The following are the typical displays at an STP.

Tactical Map Display

This is essentially the same as the IOS strategic map except that the operator must be allowed to use a command technique similar to the real equipment.

Sensor Imagery Display

Specialized sensors can produce imagery which is entirely unique to the sensor and not amenable to generic methods. For example, radar displays come in all shapes and sizes with widely varying content. The display is either an ab-initio generic which is essentially similar to a tactical display, or highly system specific in which case it is outside the scope of a GMMI. Like radar, sonar displays are somewhat specialized and it is difficult to conceive of a generic facility. However, even these very specialized displays require control panels which can be provided generically.

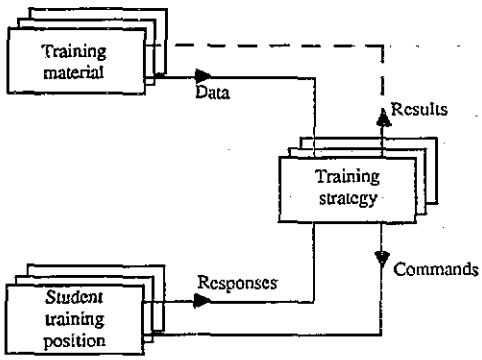
Equipment Control Panels

Most operator positions contain control panels which are made up of fairly standard components such as buttons, knobs, meters and keypads. These panels invariably require a repeat for the instructor to monitor the student. It is interesting to note the differences in the way that the instructor and student may use the same picture. To achieve the training objective it may be necessary for the student to interact using something close

to the real actions. To maintain this fiction a touch screen may be employed. This mandates that the 'buttons' are finger-sized. By contrast the instructor may only look at the picture or may interact using a more precise pointer such as a cursor driven by a rollball or mouse. The student may therefore require a bigger screen area for each button than does the instructor. To maintain the same perceived quality it may even be necessary to use a higher resolution for the same picture.

COMPUTER BASED TRAINING

An advanced CBT system has the architecture shown in Fig 2.



P5G-8/89

Fig 2 Architecture of an advanced CBT system

A CBT workstation is a special case of the STP where the IOS has been replaced by pre-prepared training material. This material is then delivered to the student under the control of an expert training strategy. The training strategy receives training material and displays it on the screen. In general, all screens are looking for responses which either answer a question or request a change in the presented material.

Marconi Simulation's Mandarin (TM) system is an advanced desk top training system which allows sophisticated simulations to be controlled from the context of a CBT lesson. This system already addresses some of the techniques which are envisaged as being necessary to produce a GMMI.

Since CBT systems are inherently generic in that they are general purpose training devices, the range of training delivered on a CBT system is potentially much wider than that delivered on a dedicated STP. This means that the potential student may be drawn from a much wider population. There may be no dedicated training staff associated with the CBT facility and system support may be an additional duty for someone otherwise unconnected with training. Many of the observations about IOS are true of CBT in that training material preparation is important and that the collection of student records is perhaps even more important.

Training material on a CBT system falls into a number of categories, including the following.

Tutorial

This is usually characterized by pictures and text which are presented in a form reminiscent of a viewgraph presentation where foils are laid over existing foils to build a picture. The student has control over the rate of presentation and is able to review material.

Drill and Practice

This tends to take the form of setting the student a mechanistic task which must be carried out in a suitable and timely sequence. This is not a simulation as often only the correct actions are explicitly recognized with all other actions being wrong and perhaps resulting in remedial training.

Simulation

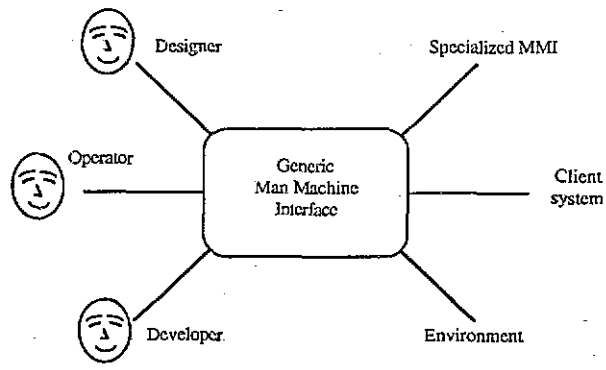
This is really no different from the sort of simulation that would be expected on a part task trainer, except it is usually of a shorter duration and set in the context of CBT by the use of preceding descriptons of the exercise and some sort of following analysis of the student's progress.

Testing

A feature of CBT is the ability to test the student by asking questions and assessing the results. The results of testing are gathered automatically for individual student assessment and for courseware improvement.

ARCHITECTURE

The GMMI must be designed for the widest applicability, and this means adopting standards wherever possible and defining them elsewhere. In order to assess the scope for standardization, look at a simple view of the system from the GMMI's point of view (see Fig 3).



P5G-8/89

Fig 3 External interfaces to the GMMI

The Operator

This is what the system is all about. The operator (instructor, trainee or student) is the end-user of the system, the MAN in the expression 'Man-Machine

Interface'. A key objective of the GMMI is to produce a highly effective operating environment for the end user. The system must therefore communicate with the operator in a manner that:-

1. Provides suitable information in an appropriate manner.
2. Provides suitable control over the system in an appropriate manner.
3. Is consistent, in that the operator only needs to learn a few simple rules to be able to operate the entire system without unpleasant surprises.
4. Is adaptive to the varying needs of different operators.
5. Makes the operator, regardless of experience, feel confident and in charge of the system.

Ease of Use. This is a much misunderstood and misused concept. The term 'user friendly' has come to be synonymous with 'usable with no training'. The down side of this definition is that the experienced user is often held at the level of beginner with no possibility of bettering his performance or improving his productivity. All too often 'beginner friendly' becomes 'expert hostile' and frustrations build. I suggest that 'efficient and effective' is a better objective in the design of an MMI.

Clearly a beginner requires more help than an expert and dialogue driven systems are excellent for this purpose. Unfortunately they are slow and an experienced user can rapidly tire of the slow pace of a dialogue driven system. By contrast, a system which allows an expert to operate at full speed may be totally inaccessible to a beginner. Looking back at the wide range of potential users of the considered systems, it should be apparent that the key to efficiency and effectiveness is adaptability. The interface must be adaptable to the needs of many different styles and standards of users.

Producing an MMI is difficult. Most computer-based MMIs are never actually designed, they are the consequence of a programmer taking the easy way out and providing an interface which, in essence, is produced for the machine's benefit so that it can talk to the man. If that is the case with existing interfaces then how much harder it would be to actually design an adaptive interface. This is of course where the GMMI really starts to show its power. If it provides sufficient in the way of high level picture building and control sequence definition, then it takes all the back breaking out of actually building the interface. The effort can be directed where it should be, ie in the design of the operation of the MMI looking in from the user's point of view and not out from the computer's point of view. Since the GMMI is the building block on which the specific MMI is built, it can make it easy to adopt conventions which keep the interfaces consistent, which is an important aspect of a good interface.

The benefits of a GMMI are that the end-user has an interface onto the application which is consistent, adaptive, efficient, effective, and may even be friendly to both a novice and an expert, in that they will both enjoy working with it since it helps them in their task and does not introduce additional burdens.

Slow response times make even the best designed interface unfriendly and very frustrating. There are three critical response times when looking at an MMI:-

Picture Change Time
This is the time to completely write a new picture which is replacing a previous one. In general, this is the time which comes in for the most operator criticism. It is noticable that operator perceptions of the times involved are often wildly at variance with the reality - two seconds can seem like ten and ten seconds like a minute.

Data Update Interval
This is the time to update the data in an existing picture. Most systems seem to maintain a one or two second data update rate for instructor displays which elicits little comment from operators. Faster update rates for many of the displays would only result in blurring of the presentation and actually result in a loss of information.

Feedback Delay
Operator input needs an 'instantaneous' response. A lack of timely feedback can leave the operator unsure if his input has been accepted. This leads to repeating the operation to be sure and ultimately to overshooting the required value because of all the accepted but unacknowledged inputs. The response can be anything from an audible signal to a cursor change or an instrument change. The important thing is that the input is acknowledged instantaneously (within 200msecs) in a consistent fashion.

The use of graphics displays for real equipment is only acceptable in certain circumstances. The conditions governing these circumstances are somewhat similar to those regarding simulators in general. All training requires the student's cooperation and simulators also require a 'suspension of disbelief'. The degree of this suspension depends on lots of factors but is governed mainly by quality of simulation and surroundings. For example, a graphical display of navigational instruments presented on a desktop in a classroom atmosphere may be quite acceptable to the student but that same quality of simulation may be unacceptable in an aircraft mockup. Important factors in the presentation are:-

Resolution
MMIs seem to concentrate on screen resolutions in the 512 by 512 to 1280 by 1280 range. Many of the interfaces would profit from using higher resolutions and it seems to be clear that they are designed around the best available technology.

Colours
The problems of the misuse of colour are well documented but it still goes on. The exhortations to use colour wisely and not gratuitously still seem to fall on deaf ears.

The Designer

The designer is responsible for defining the operator's environment and integrating it with the client system. In order to define the operator's interface, the designer needs to be able to define communications with the client system and the communications with the operator. The design process takes place in four steps:-

Definition of Data Requirements
It is necessary to analyse the user's requirements for data access and to identify its source within the client system. This phase does not define how the operator uses or views the data, only that it is required. The formal statement of the data requirement has to be agreed with the client system designers. This is a fundamental part of the design of the entire system and is not easily subject to later modification by an end-user and may have large consequences in the software if access to extra data is subsequently sought by a user.

Definition of Data Presentation
It is also necessary to analyse how all the potential operators are going to use that data and define an efficient and consistent interface. This process of defining the data presentation has to be carried out in conjunction with the end-users to ensure that appropriate information is being presented. The style of presentation will depend upon a number of factors, such as customer's requirements, user's requirements, ease of use, efficiency of use, consistency of use, adaptability of use, and of course practical graphical and processing considerations. It is highly unlikely that either the customer's or user's requirements will, on their own, produce a good user interface. It is the designer's task to interpret the requirements in the light of his experience of man-machine interface design and to ensure that a good interface is produced.

Integration
The MMI has to be integrated with the client system. In practice, this means producing initial conditions and start-up routines which configure the MMI to operate with the client system.

Maintenance
The adoption of a tool for the production of a GMMI makes it possible for end-users to modify an existing interface, either to simply reflect superficial changes in equipment appearance or to make real changes in the way in which the data is viewed by the user. This aspect is important since the way in which any system is used is likely to evolve with time and experience gained of its potential. It does seem to be true that most training establishments seem to be able to find *new and novel uses for their facilities.* Clearly, the capability for them to improve their MMIs and to develop them in keeping with the new uses is a great benefit.

In order to be able to carry out these design tasks the designer needs some support tools. Two fundamentally different approaches to attempting to provide a GMMI are possible:-

A Kit
This approach calls for the development of some tools, a library of subroutines, and a set of instructions on how to assemble these parts, in conjunction with project-specific items, into a working system to meet a specific objective. The major drawbacks of this approach are:-

(a) The supply of fragmented parts may lead to the supply of a fragmented solution.

(b) Instructions need not be followed and widely different interpretations will almost certainly cause wide divergence in individual usages.

(c) Although this approach minimizes the initial investment, it leaves a lot of work for subsequent users.

(d) Maintenance is likely to be difficult because of differing uses (and worse still, differing versions of the software).

(e) Each specific usage of the kit is a different system with recurring problems of integration.

A Black Box
This approach expands on the Kit solution and insists that the client system integrator has only a limited number of very well defined interfaces to the GMMI solution, that the GMMI solution is a closed system whose internal workings are invisible to the client, but that he has sufficient tools, documentation and support from the product to achieve his targets. The major advantages of this approach are:-

(a) A complete analysis of the problem is necessary to produce a complete solution which is totally compatible with other standards.

(b) Only a single implementation exists which covers all needs. The benefits of a single implementation are reduced maintenance costs, wider usage and therefore better testing and more cross-feed. An essential aspect of the implementation is the capability to extend the design as a direct consequence of user's experience and to make those extensions available rapidly to other users.

(c) The client's task is simpler and cheaper with the MMI sub-system available very early in the project.

(d) The solution can be thoroughly tested in its own right as a stand-alone item.

This black box approach is the preferred route forward. It is now possible to take a closer look at the process of designing a specific MMI.

Database. It is convenient for the interface designer to consider the system as a database containing all the required data. This database is created as a result of two separate activities:-

Software Requirements
One of the major activities of the software designer is in analysing the system requirement and reducing it to smaller problems. Usually these smaller problems are solved by separate programs which have to communicate with each other. The interface which each program presents to the rest of the system is its contribution to the database. In general, in keeping with good software design practices, the interface is kept to a minimum.

MMI Requirements
The database originating from the purely software design is sufficient to run the system. However, it is often the case that the instructor wishes to have access to data that would otherwise have been held as private within a program, so the MMI design activity extends the database requirement. It often not only extends the database but also the programs since much data is requested for display which otherwise would never be calculated. It is important that the MMI requirements are established early in a design since they can have a large influence on the design of individual programs.

For most purposes, it is perfectly acceptable to think of the MMI accessing a large database. However, a little insight into the workings of the database is desirable to understand some effects. When it is required to display a value from the database, that value is read from a regularly updated local copy of the database. When it is required to change a value in the database, it cannot be effected by changing the local copy. Instead, a request has to be sent to the owner of the data (the owner being the program which normally contributes that data to the database) asking for the data to be changed. If the request is approved, then the local copy of the database gets its regular update with the new value, otherwise the MMI may receive a request to display a error message stating why the data change request was refused.

The problem of communication between the GMMI and the individual sources of data is considered in detail later as part of the **Client System** description.

Presentation

The MMI has to be capable of being configured in such a way that powering up the computer causes the system to enter directly into the MMI without showing any intermediate environments. This turnkey technique is particularly suitable for an operator's workstation which is providing a soft simulation of genuine hardware. Any extraneous output can seriously impede the credibility of the system with an operator.

The MMI must also be available as part of a more general purpose environment, particularly for the maintenance and support staff who require ad-hoc access to operating system utilities. The windowed environment is very well suited to allowing the instructor to change his screen usage to precisely match his requirements at each stage of an exercise.

Controls. It is necessary to be able define how the user can control the system. This is done be defining the commands which should be executed in response to a specific operator action. These commands are single line instructions which perform particular specialized tasks such as changing the screen usage or altering the value of a data item. Commands can be grouped together in command lists which can be used in a number of places: in menus they describe what should be done when an item is selected; in picture definitions they describe what should be done when an operator interacts with an object in a particular way; at startup they are found in a configuration file to describe how the system should be configured; they can also be sent by remote tasks. It is interesting that a single unified set of commands can address all interaction with the GMMI. The various commands must cover the following topics:-

Access Rights
In a system with multiple users, it may be necessary to allow all instructors to look at the same pictures but only one to change the related data. A system of locks and keys is needed to provide this protection. If a command list contains a lock statement and the matching key is currently false, then the whole command list is not available, and so the operator is locked out of the control.

Configuration
At startup, any commands can be issued. There are, however, some which are specially required for use during startup. These set various defaults, such as the name of the current workstation, and wait for server tasks becoming available.

Data
A group of commands is needed to allow data and messages to be sent to remote tasks.

Development
The adoption of a GMMI makes it possible to provide a standard test harness for the development of the specific MMI so that simulated client system input to the MMI can be supplied manually. A similar standard test harness for recording actions makes the building of a test schedule far easier.

Operating System
An interface to standard operating system utilities, such as file copying, is required.

Errors
A systematic approach to error reporting is desirable. For example, the operator should be able to select how and when he views the error reports.

Menus
A full menu system which allows the usual banner and pulldown menus plus nested popups is required. Many simulation activities contain an element of categorization - this is a problem which is easily addressed by the use of nested menus.

Pictures
Pictures must be displayable in a window with an optional menu.

Windowing
Windows must be definable relative to an existing window or relative to the screen, and in a variety of windowing styles. They should be zoomable by a standard facility, and the operator has to be able to organize the usage of the screen to match the current requirements.

Picture Design. The following conceptual model of the development process for a picture separates the task into logical stages. There is nothing in the process which insists that all of one stage must be completed before the next can be started. Indeed, the GMMI picture editor must be designed to permit almost any way of going about a task. It is highly likely that, in practice, some objects would be entered into the picture and completely defined before going on to the next object:-

Style
A common problem encountered when many hands are involved in producing an MMI is that each implementor uses different colours and different interaction styles. The style editor allows the coordinator to decide on the colour usage strategy by identifying the key elements of the colour design and giving each a meaningful name. There may be a number of groups of styles, perhaps one for each group of simulated equipments. The individual styles may have names like 'radar panel' or 'radar small legends' so that the use of the style is identified rather than its appearance. A lack of uniformity of presentation style seriously mars an otherwise excellent interface and can make it seem that no care was taken.

Presentation
The picture is drawn with very little reference to data dependency. This is the mainly graphical stage which closely mirrors the usual graphical drawing packages. The drawing has to be produced in sympathy with the required data resolutions and control devices. There is no point in producing a picture which cannot address the required resolutions for the data display or makes the operator's control of the system difficult.

Data
The data necessary to drive a dynamic object is specified as an item in the database plus a scaling or conditioning treatment. Only a fairly small range of dynamic instruments is necessary, provided they are supplemented by a technique for the author to design his own dynamic instruments from the supplied primitives. Instruments have to be capable of recognizing potential misuse, such as insufficient screen area to supply the required resolution for the data presentation.

Commands
The operator can potentially interact with an object in many ways, each of which must be fully defined. The styles definitions not only contain the preferred use of colours but also the preferred control techniques. There may be a number of suggested techniques for each control to cover the various users. For example, it is not obvious how to operate a multi-position rotary switch using standard mouse techniques. The switch could be set up so that the experienced user knows to double-click on the left of the switch to step anti-clockwise, or the right to step clockwise. Perhaps as part of the standard interface for inexperienced users, the single-click anywhere on the symbol presents a popup menu with all possible options displayed. This menu could not only contain the options but also some descriptive text about each option. Supplement this technique with a HELP option as standard in the menu and the interface is now adaptable to at least three levels of expertise. Controls should be capable of recognizing potential misuse such as an inadequate screen area for the control. For example, a different minimum screen area is required for a pointer-driven interface than for a touch-driven interface.

Menus. It is not necessary to include menus in the design of a system, although it is hard to see why they should not prove to be powerful technique in an instructor's workstation. A menu structure consists of:-

Banner
A menu starts with a banner across the top of the window.

Pulldowns
A pulldown menu can exist only directly from a banner. A pulldown contains a list of entries which may be popups or a set of commands to be executed.

Popups
A popup menu can exist at any level of a menu structure. A deeply nested popup menu contains a number of title bars, one for each nesting level. A click on a title bar causes the menu to be de-nested to the desired level. A click on an option causes a further menu to be shown or a set of commands to be executed.

The Developer

The developer can easily become the forgotten man. Testing of an MMI is very difficult and very time consuming. The test schedule route is not the most satisfactory method since it imposes a strict discipline on the tester which no real operator would accept. It is the sheer perversity of the operator which makes it so difficult to fully test an MMI.

Having said that, the test schedule does play a part in testing isolated sequences in an MMI. A facility to allow a tester to interact with the system and automatically record his actions in an editable text file which can be subsequently re-used to drive the system can have a profound effect on testing strategy. Thus each interaction sequence can be automatically re-tested by replaying an automatic test schedule. This schedule can be developed and enhanced in response to each newly discovered problem so that all predicted faults and all subsequently discovered faults can be automatically verified by the schedule.

The benefits of this facility carry forward when testing the client system. An automatic test schedule which is taught to the machine can cut out the dreadful problems of testers who make keying errors and deviate from the test plan. By making the log an ordinary editable text file, it is possible to incorporate prompts which allow the tester to operate real hardware in response to instructions and to make it difficult to deviate from the plan.

Client System

The interconnection between the GMMI and client system is a major issue. Without a standard interface to the client it is virtually impossible to produce a GMMI. It is very easy to limit this discussion on interconnetion to networking and think that a solution has been found - not so. The question of connectability is more fundamental than networking and revolves around how any client task should talk to the MMI. Indeed, the problem can be taken even wider. How should any task talk to any other? This is of course at the very core of the perennial question of re-usable software. *Consequently this Inter Task Protocol* (ITP) has been the centre of much study.

It is useful to consider the problem in easy stages.

Data Update Characteristics. The bulk of the data update is of a routine nature and highly predictable. A real-time simulation has to be tied very closely to the real-time clock so that most tasks run at regular intervals. For example, a target manoeuvring task may run every 100ms to extrapolate the targets' tracks to their next positions and then make these new positions available to its clients. Since the task is regular, so is the updating to clients. This traffic is **routine** and accounts for the largest slice of all data.

Simulations and training systems also give rise to another class of message which is more closely geared to external events such as switch changes and operator/instructor input. This introduces the idea of **event** messages. There are also **command** messages which do not affect data anywhere but cause changes to the system state, for example 'freeze simulation'.

Language Independence. Large client systems may not only mean many computers but many different kinds of computers depending on different compilers and programming languages. Most assembler programmers are familiar with the problems of locating data within a high level language data structure. It can be necessary to expend considerable effort in understanding the compiler writer's data storage algorithms. To avoid these sorts of problems in moving data from one task to another in a large system, the adoption of a protocol which is independent of language and compiler implementation is necessary.

Task Insulation. It is usually considered that a good design aim is to minimize the connectivity between the different parts of the system. One objective of tasking is to partition a solution into separate activities which are insulated from other activities by the operating system. This insulation ideally incorporates the use of memory management to contain tasks within their own memory limits and to keep other tasks out. Communication between tasks has to be handled by operating system *functions. These typically take the form of message* passing and receiving mechanisms.

A fundamental objective of writing re-usable software is to insulate the individual module from its users. In the same way that a well-written subroutine will communicate with a caller by receiving parameters and returning a result, but never know who called nor know anything about the caller's own data, so an insulated task will not know who requires data from it. It will, however, know from whom it needs data.

A necessary corollary of this organization is that data belongs to a single task. For example, the position of a target belongs to the target manoeuvring task. Only that task can make changes to that data. Any other task, such as the MMI, can only request the owner to make a change.

System Integration. The integration of a large system can be made easier if it is well-designed, using a very well-defined interconnect mechanism between the various tasks. Even the best-designed systems hiccough during integration, perhaps a task has to be configured from one processor to another to balance real-time loads. If the interconnect mechanism handles this re-configuration rather than the tasks themselves then the integration is easier and more reliable.

With some understanding of the problem, it is possible to specify the cardinal points of the ITP:-

1. Communication paths between tasks are dynamically determined during system startup.

2. The startup has to be tolerant of missing tasks and varying sequencing of the startup.

3. Each task has a unique name by which it can be addressed by other tasks requiring access to the data it can supply.

4. Data updates are supplied by the owner of the data to all its clients, with a minimum of unnecessary run-time load.

5. Client tasks may request changes to data.

6. Commands may be sent to other tasks.

7. The data interchange must be language and compiler independent.

8. It is a recognized or de facto standard to permit connection to a wide range of applications.

During the study, no standards were found which completely addressed the problem. One of the best matches was found in Microsoft's Dynamic Data Exchange (DDE). Some modest enhancements to this protocol enable it to meet the requirement.

DDE sees the system database as a hierarchy of application, topic and item which maps well on to the simulation system as task, task instance and data item. So in a system where each target is managed by a separate instance of task 'target', the speed of target 3 might be 'target.3.speed'. This notation is used in spreadsheet applications to specify the contents of a single cell. Transfer of data from one task to another in this piecemeal fashion is far too time-consuming in a simulator. Almost inevitably, the fastest approach is 'give me what you've got and I'll pick out what I need'. This is accommodated by extending the protocol to accept a wildcard item (predictably '*') which requests all data exported by the task instance. This means that all parties to the transaction have to know the format of the message and this is accommodated by the data provider supplying a translation template when initiating the conversation on the data. This template described the ordering of the data in the message, the format of each element of the data, and the name by which all parties know that data. Thus each party to the transaction can build a simple lookup table to extract their required data from the message. This technique allows supplier tasks to redefine the layout of the message and to insert new information without any effect on consumers. It is also the basis of providing language, compiler and computer independence.

In its own right the ITP provides no networking capability, but this is as it should be. In keeping with the ISO OSI model, the communications are peer-to-peer with no knowledge of the underlying transport mechanism. This means that tasks talk (apparently) directly to other tasks. The advantages of this style of layered communication is that at each level, the interface is independent of lower layers and therefore inherently portable on to other implementations. To extend ITP to networking is conceptually simple, and is indeed simple to implement on most real-time systems. The extension is achieved by the provision of an ITP network interface task at each network node. This receives all local attempts at initiating a conversation and broadcasts them on to the network, where similar tasks in the other nodes relay the messages onto tasks in their systems. The local ITP Network task then acts as a surrogate for the remote task and the ITP operates over the network. It is possible for an ITP network task to optimize the network traffic by recognizing commonly required data and broadcasting that data to other ITP network tasks for them to relay individual tasks.

Specialist MMI

From the previous discussion on the interconnection to the client, it should be obvious that specialist MMI requirements, such as radar and sonar displays, can be handled by free-standing tasks which communicate through the ITP. In a similar way, it is possible to enhance the GMMI by the addition of additional free-standing tasks commonly associated with the MMI, such as record/replace, student logging, intelligent monitoring and automatic reaction to student actions.

The Environment

The environment is the Operating System and the host hardware. This has to be cost-effective and ideally should be widely available in a range of capabilities and costs to match the requirements of the client system. It is clear that the ideal solution is based on a WIMP (Windows, Icons, Mouse, Pointers). There are two obvious contenders in X-Windows in the UNIX world and MS-Windows in the DOS world. There can be little doubt that both approaches provide an excellent platform for GMMI since both provide complete insulation from the underlying hardware, a wide cost/performance range, and a commitment from a broad base of suppliers. The use of object-oriented languages like C++ in the implementation of the GMMI can provide further insulation from the windowing environment and generate a solution which is highly portable between the two environments.

The PC-based solution offers a highly cost-effective solution. Processors are currently offering up to 5 MIPS (33MHz 386) and, with the advent of the 486, may soon be delivering up to 20 MIPS. Coupled with graphics systems ranging from the VGA at 640 by 480 pixels, through high speed systems in the 1024 to 1280 bracket, to very high resolutions at 2880 by 2880, it is certain that an appropriate system can be chosen for any particular task.

In the case of a large simulator, the free-standing GMMI has to be connected to the rest of the system. The ITP is insulated from the actual networking system by its network interface task, but there are some key points in the choice of a network for connecting a GMMI to a simulator - the network must be fast enough to handle the predicted traffic and must also be predictable in its nature. At first sight this latter criterion may appear to rule out Ethernet. In practice, at low traffic rates, the network is predictable within acceptable limits. It may be possible to reduce network traffic substantially by broadcasting a

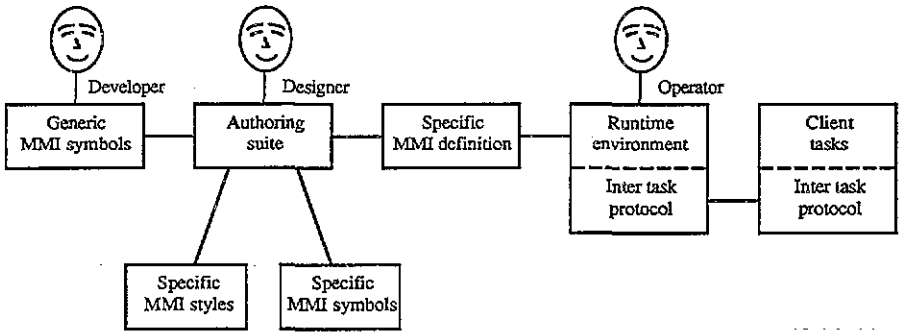


Fig 4 Block diagram of a GMMI

message once to all intended receivers. (Consider something like target manoeuvring which has to deliver its results to almost every task in a simulator.) The network may appear to have a massive bandwidth (Ethernet = 10Mbps), but this can be severely constricted by the efficiency and quantity of network software. Network protocols are enormous and can consume vast amounts of memory for the protocol and for messages. Similarly, they can consume vast amounts of local processing power. For this reason, a direct connection from the ITP network interface task to the network device drivers is probably the only sure means of guaranteeing performance.

CONCLUSIONS

The GMMI is conceptually simple, as shown in Fig 4.

The developer provides an 'authoring suite' which is based on a library of 'generic MMI symbols'. The designer enhances this library by defining his own 'specific MMI symbols' which are created using any existing symbols. He defines his required interaction styles in the 'specific MMI styles'. Using all of these, he creates a 'specific MMI definition', which can be run by the 'run-time environment' to produce the required MMI. This MMI uses the 'inter task protocol' to communicate with all 'client tasks' which may include simulation tasks, specialist MMI tasks, and utilities such as record/replay.

The hardware, operating systems and networks necessary for a GMMI are all available off-the-shelf, but the GMMI does not yet exist.

In the early Eighties, CBT began to migrate from general purpose programming languages to specialized CBT languages in order to simplify the 'authoring' task. These languages were used in conjunction with general purpose drawing packages which produced pictures which could be displayed by commands issued in the CBT language. The problems of interpreting operating input were entirely handled by the CBT program in a somewhat ad-hoc fashion. A few years ago Marconi Simulation produced Mandarin (TM) which took this 'authoring' a stage further by incorporating a specialized drawing package in its development system. This editor not only produces object-oriented pictures but allows the pictures to be built from active as well as passive components. The active graphics objects include simple instruments, user-defined symbols and input controls. This step allows the training material to be separated from the training-strategy, which is analogous to the required architecture for the GMMI: the MMI has to be separated from the simulation models Another step forward has to be taken in order to provide the simulation MMI designer with similar benefits to those already enjoyed by some CBT authors.

ABOUT THE AUTHOR

Ken Fearn is a Consultant Engineer with Marconi Simulation, having more than 20 years experience in the development of software, mainly in real-time training simulators. He is currently engaged in the design and development of a GMMI (on which work this paper is based). This activity is a natural successor to his previous project in which he led the engineering team which designed and developed the Mandarin (TM) computer based training system. This system varied from many other CBT systems in the manner in which it employed 'active graphics'. Mandarin (TM) is now in widespread use throughout the Royal Navy for many of their more demanding desktop training tasks.

Earlier activities have included the design and development of an artillery sound ranging simulator and sonar simulators for both military and civil use, and participation in the design and development of a parallel processor which was used to simulate an Advanced Gas Reactor in real time. Outwith the simulation sphere his activities have included the design and development of an operating system and communications software for advanced banking terminals.