MODELLING WITH THE INTEGRATED PERFORMANCE MODELLING ENVIRONMENT (IPME)

Anna M. Fowles-Winkler
Micro Analysis and Design, Inc.
4949 Pearl East Circle, Suite 300
Boulder, CO, USA 80301
E-mail: awinkler@maad.com

KEYWORDS
Discrete-event simulation, High-Level Architecture, human performance, simulation tool, workload.

ABSTRACT

The Integrated Performance Modelling Environment (IPME) is a commercially-available, Linux-based discrete-event simulation software application for building models that simulate real-life processes. With IPME models, users can gain useful information about processes that might be too expensive or time-consuming to test in the real world. Some example application areas for simulation modelling using IPME include the following:

- Evaluating procedures, workload, and staffing issues for aircraft and ships, including workstations.
- Modelling adaptation and alertness to changing time zones.
- Analysing information flow, staff sizes, and workload for a staff of 15-20 people in a military control centre.

IPME provides a plug-and-play model environment to allow for flexibility in evaluating different environments and crews. The built-in workload functionality accounts for operator resource demands in systems. Finally, IPME's communications interoperability provides an easy way to reuse existing models or simulations.

INTRODUCTION

Micro Analysis and Design, Inc. began IPME development in 1995 with support from QinetiQ Centre for Human Sciences. In 1998, Defence Research and Development Canada—Toronto (DRDC-Toronto) joined the development program. Built from the software base of Micro Saint, a Microsoft Windows-based commercial discrete-event simulation product, IPME is an integrated environment of plug-and-play component models designed to analyze human system performance. From the beginning, development has focused on two goals: 1) providing a flexible modelling environment, allowing users to choose which components they needed for their analyses, and 2) allowing IPME to communicate with other simulation applications.

This paper will provide a basic understanding of IPME's plug-and-play models, human performance workload features, and interoperability capabilities.

PLUG-AND-PLAY MODELS

An IPME model, or *system*, is a collection of models and data that represent what the user is currently analysing. A system is comprised of the following component models:

- An environment model
- A crew model
- A performance shaping model
- A task network model
- An experimental suite
- An external model

The only required component model is the task network model; all other models are completely optional. This section will describe all models except for the external model. The external model is discussed later in the section titled "Interoperability."

An environment model describes external factors such as physical, crew, mission, and threat factors. Physical factors include humidity and temperature. Crew factors can define how well the group performs as a team, including factors such as leadership. Mission factors include elements like intelligence and weapons reliability. Finally, threat factors describe the degree of threat from an enemy, and the position of the target. Each factor may have an associated expression that is evaluated at each simulation event.

By default an environment model includes a basic set of variables. Customizing an environment model by adding user-defined external factors allows the model to fit particular environments under consideration. Environment models of different areas can be developed and plugged into existing systems, providing a simple way to compare operator performance under different environmental factors. For example, a model of a military's ground forces could have one environment model to represent a jungle, and a second environment model to represent a desert.

A crew model defines individual operator roles, and includes operator characteristics such as non-physical traits (fitness, training), states that change during simulation (boredom, hunger), and physical properties including anthropometry (weight, height). Because

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

states can be updated during a simulation run, each operator defined in a crew model can have unique characteristics. Traits are generally held constant during a simulation run, but can be varied for a block of runs using the experimental suite, described later in this section. Currently, operator anthropometry cannot be varied as traits can, but a future version of IPME will include this functionality.

A performance shaping model is a collection of user-defined functions called performance shaping functions (PSFs) that modify the time it takes to complete a task, or the probability of task failure. The PSFs are linked to individual tasks through a task taxonomy, allowing one PSF function to be dynamically applied to any similar task in a model. Since PSFs can use operator states as expression variables, models can discriminate performance results as a function of operator characteristics. Therefore, simulations can have two operators performing the same task type with different, and therefore more realistic, task time and probability of failure outcomes.

The task network model is a graphical display of the system processes or tasks. Because IPME's task network model is based on the same technology as Micro Saint's task network model, the terminology used to describe the task network model components are clearly based in human performance modelling. A task network model consists of networks and tasks. Figure 1 shows a sample task network model.

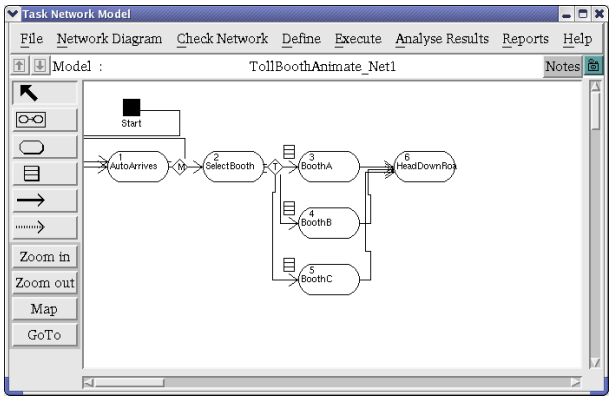

Figure 1: IPME Task Network Model Dialogue

Networks may be a sequence of tasks performed by an operator or a series of processes that define an organization. Tasks generally denote human activities, but they might represent other, non-operator processes, or logic to support the simulation. Tasks contain timing information, conditions for execution, and operator assignments. Operators from the crew model may be statically assigned to particular tasks, or they may be dynamically assigned depending on aspects such as what operators are available to perform the task.

A task has a set of expressions associated with it to control when the task executes, to control the state of the system when the task begins or ends, and to specify what, if anything, should happen if a task fails to execute. These expressions may contain user-defined variables and functions. Variables and functions are defined globally, and may be used in any expression in any model.

While it is optional to use the environment, crew, and PSF models, using all three in combination with the task network model takes advantage of the plug-and-play nature of IPME, and the interconnectedness of the models. External factors from the environment model and operator states from the crew model may affect operator performance, with that performance being defined as a PSF. If an analysis requires different environments or crews to be analysed, those different models can be easily used with a single task network model. Figure 2 shows the relationship between these component models.

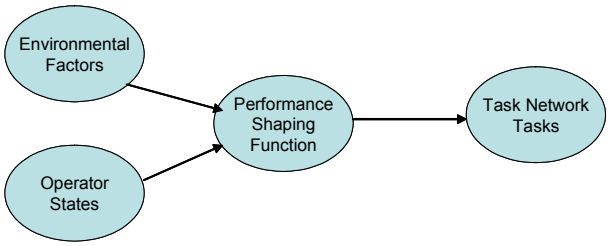

Figure 2: Component Model Relationships

After a model has been developed in IPME, the experimental suite can be used to vary system variables to show equipment, design, operator, or procedure variability. The user specifies independent and dependent variables and their values for each simulation run or for a block (a series) of runs. For example, the environmental physical factor *temperature* can be configured to have the value –10°C for the first block of runs, 0°C for the second block of runs, and +10°C for the final block of runs. Human trait values such as *fitness* can also vary across simulation runs and potentially represent differing human populations to be used in the model. The experimental suite provides a user-friendly method of running multiple experiment blocks with varying variable values.

Once a model has been developed, an analyst may decide to share portions of that model with other analysts. The built-in master library allows users to store and distribute operators, performance shaping functions, environment models, and networks. These models are under revision control in the master library, indicating new models and when models have changed. Users link to models in the master database in a read-only mode, preventing modification without permission. To modify a master library model, a user must either have permission to modify the library, or the user may unlink the model from the library. By unlinking a model, a user is then able to modify a local copy of the model without modifying the library version of the data.

WORKLOAD

IPME includes the following built-in workload methodologies: Prediction of Operator Performance (POP), Visual, Auditory, Cognitive, and Psychomotor (VACP) and Workload Index (W/Index), and Information Processing/Perceptual Control Theory (IP/PCT).

POP, developed by the Defence Evaluation Research Agency (DERA) 1992-1995, predicts performance degradation from interference between concurrent tasks (MAAD 2003). Input (visual or auditory), central (mental operations), and output demands (manual or vocal) are considered for each task. Figure 3 shows the POP Workload Percentages portion of the task assignment and workload dialogue for a sample task.

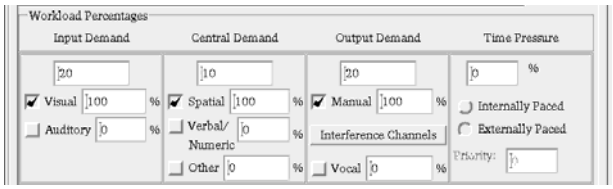

Figure 3: POP Workload Percentages

The task assignment and workload dialogue includes a *Workload Percentages* section for POP workload data. The workload demand values are entered for the input, central, and/or output channels, with 100 representing maximum workload before overload. A task demand multiplier (TDM) is calculated based on the fraction of time a single resource for a single task is available to the time that it is actually used. The TDM then lengthens task time when it is multiplied with the task mean time. The implementation in IPME is symmetric; therefore, resources are divided symmetrically among tasks during simulation execution.

Tasks can be externally or internally paced in the POP algorithm. An externally paced task is scheduled by external source, for example, a supervisor requesting a report. An internally paced task is self-scheduled by the operator. Externally paced tasks are higher priority than internally paced tasks; therefore if an operator is overloaded, internally paced tasks will be rescheduled to accommodate the operator completing externally paced tasks in the time available.

VACP and W/Index measure the resource demand imposed upon an operator. The VACP algorithm measures the task loading for an operator within visual, auditory, cognitive, and psychomotor channels (McCracken 1984, Bierbaum 1987). The VACP portion of the task assignment and workload dialogue is shown in the upper portion of Figure 4.

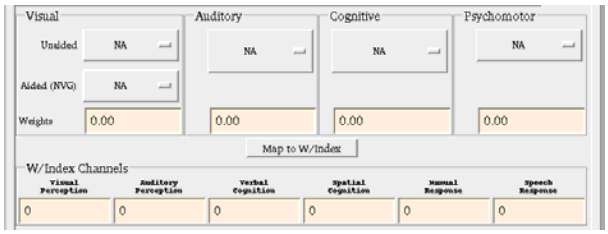

Figure 4: VACP and W/Index Values

VACP values are selected for each of the channel categories, and then the corresponding weights are automatically generated. Each task is rated based on the weighted task demand within the channels. During simulation execution, attentional demand is calculated for each operator

The W/Index algorithm (Sarno 1992) measures the resource demand imposed upon an operator within six resource channels, and supports interference between channels. The VACP resource values can be automatically mapped to W/Index values, as seen in Figure 4 by the "Map to W/Index" button. Operator workload is then calculated during simulation execution. Figure 5 shows the instantaneous workload and demand per operator in a system. In this particular example, there is a single operator (Operator 1) in the system, with three tasks being executed concurrently by the operator. This dialogue also graphically shows the attentional demand ratings and workload values as they change.

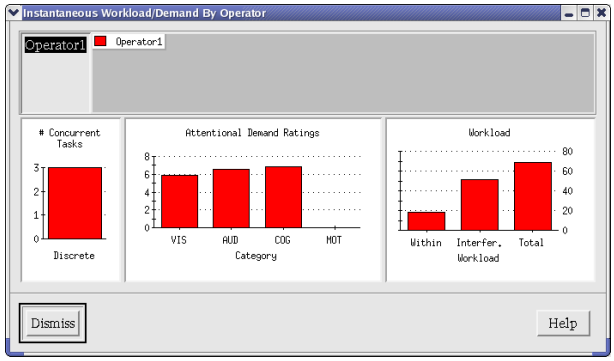

Figure 5: Instantaneous Workload and Demand

IP/PCT was developed by Keith Hendy of DRDC-Toronto. This workload methodology posits that all factors that impact human cognitive workload can be reduced to their effects on the amount of information to be processed (an operator's cognitive limits), and the amount of time available before the task must be completed (an operator's time pressure). According to IP/PCT, human operators change their processing strategy to reduce the amount of information to be processed, or increase the time available.

In IPME, the IP/PCT scheduler places tasks on the event queue based on how a human would order tasks. For example, the most important tasks are addressed first,

and tasks that are almost finished are completed before starting new tasks. Each task assigned to an operator is categorized according to time priority.

Additionally, cognitive and structural interference are considered. A memory limit is set for all operators, which affects the number of tasks an operator has in prospective memory, the memory before working memory. Tasks may be shed from prospective memory, meaning that the operator has forgotten about those tasks.

Structural interference between tasks appears when an operator is required to operate separate controls with a single limb, when visual focus is required for images too far apart, or when an operator is required to speak two (or more) distinct messages at the same time. The IP/PCT scheduler determines which task to execute based on time priority and available channels. Tasks that cannot execute due to interference may be delayed until they can execute, or they may be shed.

Visual, auditory, cognitive, or psychomotor interference in IP/PCT is determined by values set by the user for each task. The IP components tab, shown in Figure 6, allows the user to select the domain(s) used for the particular task, and the domain category. For example, a cognition task could be a passive monitoring task, or a skill-based task.

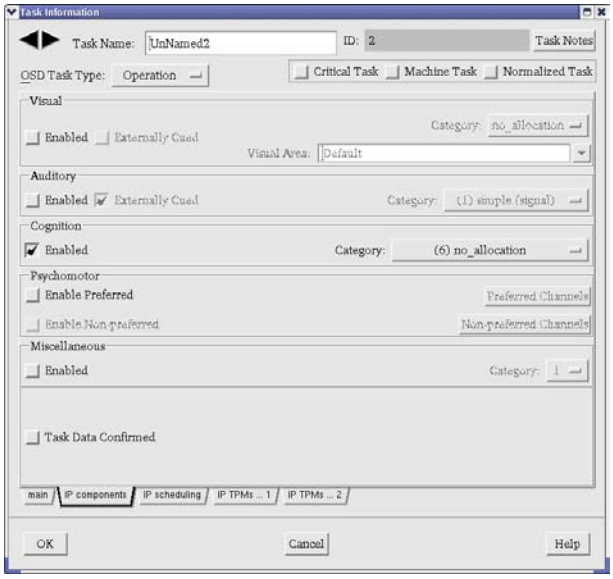

Figure 6: IP/PCT Component Domain Selection

Both IP/PCT and POP affect task execution, and may cause tasks to require more processing time, or to be delayed, rescheduled, or shed, depending on an operator's workload and other concurrent processes. These modifications to the simulation event queue occur automatically as a result of operator overload or interference, depending on the workload algorithm enabled.

INTEROPERABILITY

The external model allows IPME to communicate with other existing models or applications. There are two supported communications protocols: a TCP/IP sockets interface and High Level Architecture (HLA).

The external TCP/IP sockets interface is a simple protocol that allows IPME to communicate with other simulations (IPME to IPME or IPME to custom simulations). This functionality extends the capabilities of the overall IPME system by allowing an IPME model to share data with custom simulations such as model optimisers, cognitive models, and applications that animate IPME models. IPME may be run as either the server or the client. When IPME is run as a client, the custom application controls IPME model execution by dictating when events can happen. More typically, IPME is run as a server to control event execution in the custom application.

There are three phases of communication: 1) Registration, 2) Event Processing, and 3) Termination. Because IPME is the interface controller, messages are received from a custom application during Registration and at scheduled event times during Event Processing. During the Registration phase, the custom application identifies itself to IPME and configuration information is exchanged. The custom application also identifies which IPME simulation variables it wants to be informed of each time events are executed. Optionally, during the Registration phase, the custom application may request that IPME's simulation time synchronise to the host clock. This functionality allows for real-time simulation execution.

Event Processing occurs after Registration. This phase involves IPME notifying the custom application when it may execute its next event (when IPME is running as a server), plus the exchange of variables. After Event Processing, the Termination phase terminates communication between IPME and the custom application.

HLA is the other communication method supported by IPME. The HLA implementation in IPME uses the Defense Modeling and Simulation Office (DMSO) Run-Time Infrastructure (RTI) version 1.3NG version 3.2. IPME does not yet define a Federation Object Model (FOM), as is typically required by federations. It is anticipated that with further interest and participation in using IPME as a federate in a federation, at that time a FOM will be developed.

IPME has a defined Simulation Object Model (SOM), required by federates. The SOM details how IPME interacts with other federates. The object class *IPMEVariable* is defined in the SOM to represent model execution variables. An *IPMEVariable* may be one of the following data types: integer, float, or string. When IPME is configured as a federate, the user selects which

system variables to exchange with other federates, as shown in Figure 7. Variables may be read or updated by IPME.

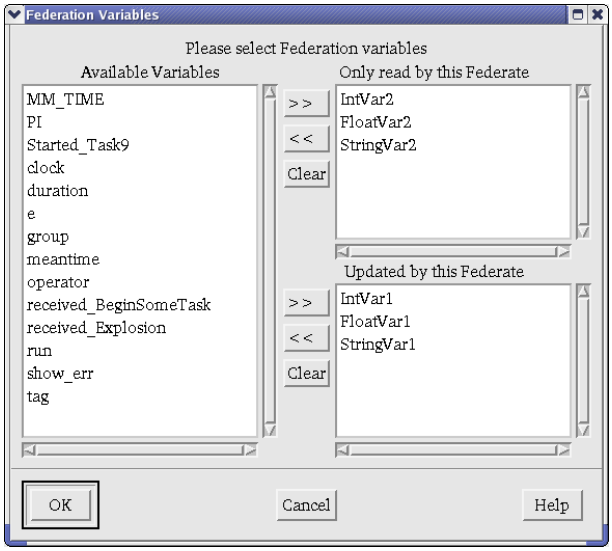

Figure 7: IPME Federation Variables Setup Dialogue

IPME sends variable values when the values are updated. When another federate sends a variable update to IPME, those values are then updated at their timestamp time during simulation execution.

IPME supports user-defined, model dependent interactions that subclass from the interaction *IPMEInteraction*, defined in the SOM. User-defined functions created in IPME may be used as interactions, with the function parameters acting as interaction parameters. Interactions provide a time-based method for inserting expressions into the IPME simulation event queue. Each interaction has a timestamp, therefore, when an interaction is received by IPME, the corresponding IPME function expressions are evaluated at that timestamp time.

Figure 8 shows the HLA Interaction Setup dialogue in IPME. All available user-defined functions in a task network model are listed in this dialogue. The user may then select which functions to send as interactions, and which to receive as interactions from other federates. Function parameters do not need to exactly match interaction parameters—extra parameters are either ignored (if the extra parameter is in a task network function) or set to a value of 0 (if the extra parameter is sent from another federate).

Although a sample Federation Execution Details (FED) file is provided, this file should be modified to specify user-defined objects and interactions. Both the *IPMEVariable* and *IPMEInteraction* classes are instantiated in the FED file to support data and interaction exchange between a particular federate and other federates.

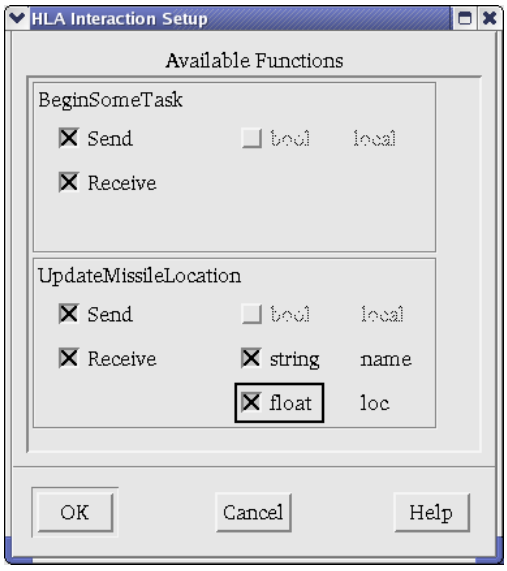

Figure 8: IPME HLA Interaction Setup Dialogue

IPME implements two time management strategies by being both a time-regulating and time-constrained federate. A time-regulating federate associates events to the federation time, determining when other federates may execute their next events (DOD 2000). IPME is also time-constrained, meaning that it expects to receive events with a timestamp (DOD 2000). The time lookahead value used by IPME can be specified by the user, and does not change during a simulation run.

SUMMARY

This paper has focused on the three important aspects of IPME: plug-and-play models, built-in workload methodologies, and interoperability. The plug-and-play model framework unique to IPME allows the user to easily try different environments and crews with a single task network and collection of performance shaping factors. Built-in workload methodologies simplify task network model development by reducing workload calculations that might be otherwise required. IPME also supports two communications protocols, TCP/IP sockets and HLA, allowing an IPME system to take advantage of pre-existing models and simulations. IPME provides a powerful and flexible environment for model development in order to analyze human performance and stressors. Future development for IPME includes varying operator anthropometry and bundling useful, supportive client applications with IPME.

REFERENCES

Bierbaum, C.R.; Szabo S.M., and Aldrich T.B. 1987. "A comprehensive task analysis of the UH-60 mission with crew workload estimates and preliminary decision rules for developing a UH-60 workload prediction model." Technical Report ASI690-302-87[B], Anacapa Sciences Inc., Fort Rucker, Alabama.

DOD (Department of Defense), Defense Modeling and Simulation Office. 2000. "High Level Architecture Run-

Time Infrastructure: RTI 1.3 – Next Generation Programmer's Guide Version 3.2."
Hendy, K.C.; and P. S. E. Farrell. 1997. "Implementing a Model of Human Information Processing in a Task Network Simulation Environment." Defence and Civil Institute of Environmental Medicine, DCIEM No 97-R-71 (December).
McCracken, J.H.; and T.B. Aldrich. 1984. "Analysis of selected LHX mission functions: Implications for operator workload and system automation goals." Technical Note ASI479-024-84. Fort Rucker, AL: Army Research Institute Aviation Research and Development Activity.
Micro Analysis and Design (MAAD). 2003. Integrated Performance Modelling Environment: User Guide, ver. 2.5.4, (May).
Micro Analysis and Design (MAAD). 2003. IPME Task Network: User Guide, ver. 2.5.4, (May).
Sarno, K.; and C.D. Wickens. 1992. "The Role of Multiple Resources in Predicting Time-Sharing Efficiency: An Evaluation of Three Workload Models in a Multiple Task Setting." Technical Report ARL-91-3/NASA A31-91-1, NASA AMES Research Center, Moffett Field, CA.

AUTHOR BIOGRAPHY

ANNA FOWLES-WINKLER is a Principal Software Developer at Micro Analysis and Design where she manages the IPME project. She has a Bachelor of Science in Computer Science from the University of Maryland. Ms. Fowles-Winkler is currently pursuing a Master of Arts in Linguistics from the University of Colorado. She started working on the IPME project in 1999 as a software developer, and moved to managing the project in 2001. Ms. Fowles-Winkler is interested in furthering IPME's interoperability, including exploring cognitive modelling frameworks. Her e-mail address is : awinkler@maad.com. For more information about IPME, see `http://www.maad.com/ipme`.