NASA/TM—2016–219478

The Man-machine Integration Design and Analysis System (MIDAS) Software Training Documentation

Brian F. Gore
NASA Ames Research Center

October 2016

NASA STI Program…in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, and organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question via the Internet to help@sti.nasa.gov

- Phone the NASA STI Help Desk at (757) 864-9658

- Write to:
  NASA STI Program
  STI Support Services
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199
  Fax: (757) 864-6500

NASA/TM—2016–219478

The Man-Machine Integration Design and Analysis System (MIDAS) Software Training Documentation

Brian F. Gore
NASA Ames Research Center

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California

October 2016

Acknowledgements

NASA would like to acknowledge the significant work that was done to create this software and its documentation. The team included NASA Ames Research Center (Becky Hooey), San Jose State University (Eric Mahlstedt), and Alion Science and Technology (Shelly Scott-Nash, Christopher Wickens, Connie Socash, Mark Brehon, Marc Gacey, and Mala Gosakan).

Available from:

NASA STI Program
STI Support Services
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

This report is also available in electronic form at http://www.sti.nasa.gov
or http://ntrs.nasa.gov/

Table of Contents

List of Figures

The Man-machine Integration Design and Analysis System (MIDAS) v5: Software Training Documentation

Brian F. Gore, Ph.D.

Abstract: This document is a training manual for MIDAS v5 that takes an user through the hardware and software requirements for the MIDAS V5 software, the steps required to download the software, and the steps that a user needs to take to create a MIDAS simulation. The training guide also illustrates how the models interact to generate MIDAS predictions of operator performance along task, workload, and situation awareness timelines and provides the test routines that were conducted to verify the operation of the integrated MIDAS models. The training guide shows the user how the MIDAS task model interacts with and controls an anthropometric model through its use of behavioral primitives. The training documentation also illustrates one approach that has been used to filter and analyze MIDAS output.

Chapter 1: Training Overview Steps

The training overview walks a first time MIDAS user through the basic steps of installing the MIDAS software, loading a file, and getting a simple MIDAS model executable to run. The Jack™ environment is pre-built for this exercise. This section is followed by sections that more fully detail how to build the Jack™ environment from scratch.

Installing MIDAS

MIDAS v5 requires the MIDAS software (executable / .msi file), MicroSaint Sharp **(**.msi file), and Siemens' Jack.

MIDAS Initial Install

- You need to have the free .NET environment (.NET Framework 3.5 Service pack 1) to run MIDAS.
- If you do not have the .NET software:
  - go to Microsoft
  - select "Download"
  - download and install
- You also need to have the most recent version of the free Java runtime environment. Locate the Windows Installer Package, Midas-5.0 Beta with an .MSI extension.
- Copy the .MSI file to the machine you want to install MIDAS on
- Double Click on the .MSI file. If you inadvertently did not check whether you have the .NET software available on the machine you are installing MIDAS on, you may be prompted to download the latest version of .Net Framework.

Updated install (when new releases of the MIDAS software are released)

- Open Control Panel and select Add/Remove Programs.
- Remove MIDAS
- Follow the instructions above to install the updated MIDAS version
- ".midas" analysis files you've created with the previous version will not be automatically removed.

Setting up MIDAS

Start/Open Midas

1. Locate Midas desktop icon and double click (see Figure 1).
   If you happen to double click the MIDAS icon **twice**, it will open two versions of MIDAS. This is not as problem; simply close one of the open MIDAS' if you do not want two versions open.

™ Jack is a UGS trademark name

Figure 1. MIDAS Icon on Desktop to Launch the Software.

2. Default screen should look like Figure 2.

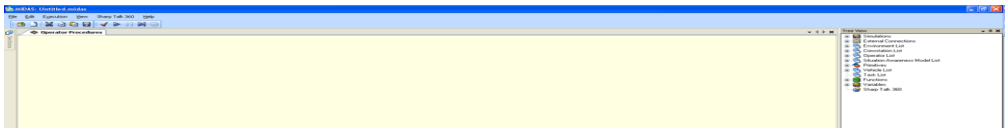


Figure 2. Default MIDAS screen.

Start/Load MicroSaint Sharp

MicroSaint Sharp and MIDAS interact and both need to be opened to run a simulation as Sharp sends MIDAS the X,Y,Z, coordinate data about the environment.

1. Locate MicroSaint Sharp desktop icon and double click (Figure 3).

Figure 3. MicroSaint Sharp Icon to Launch MicroSaint Sharp.

2. Default screen should look like Figure 4.

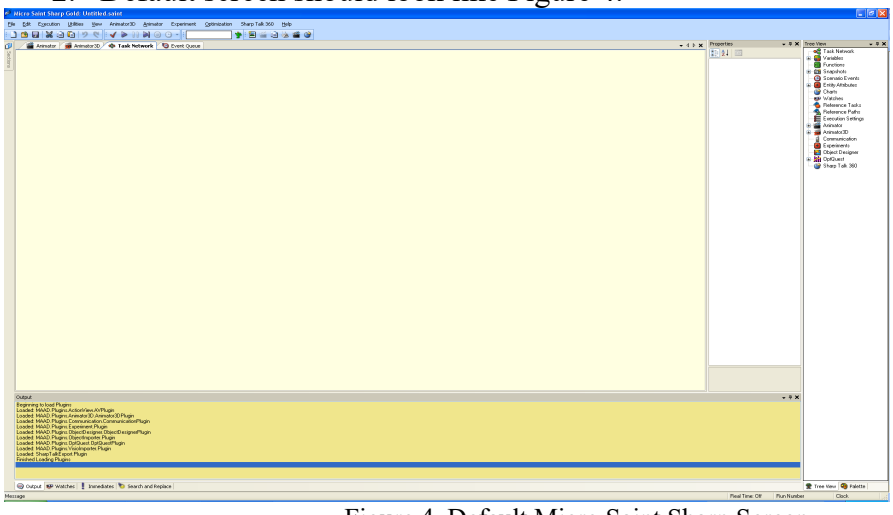


Figure 4. Default Micro Saint Sharp Screen.

Using the Anthropometry Software (Jack™)

Jack is an independent piece of software. It needs to be purchased from Siemens, Inc. Once you have this anthropometric software, you will be able to view the CAD environment. This CAD environment "talks" to MIDAS through a socket called the

CPORT module in Jack™ and through the external connections option in the MIDAS "properties" window (see next page).

Start Jack™

- MIDAS operates with Jack™
- Double click the Jack™ icon
- You should see an empty Jack™ environment

Start the Jack™ License server (if you use one version of Jack on a server with multiple versions of MIDAS on different lab machines)

- Double click the lmtools icon
- Click on the Start/Stop/Reread tab
- Click the Start Server button

Side Bar: If you find that you are unable to load the Jack software it could be that the server needs to be restarted.

Start the CPORT Module

- Click on the Modules menu
- Select Plug-ins
- In the Registered Modules menu click on CPort
- Click the Load button
- Verify that the CPort menu option now has an asterisk in front of it
- Click the Ok button

Open the CPort Command Window

- Click on the Modules menu
- Select CPort -> Open Command Port
- Verify that the mode is Normal
- Verify that the state is Active
- Verify the status is Waiting in yellow text (this will change to green when the port is sending information)
- Note the port number for later, then press dismiss

Get the Jack™ IP Address (you will need this for later)

- Open the main Windows Start menu
- Navigate to All Programs -> Accessories -> Command Prompt
- In the window that comes up type ipconfig
- Find the line that says IP Address . . . . . .
- Note the number in that field for later (this should be a number like 169.254.19.61 or 127.0.0.1 for local machine)

Start up Jack and Load Jack files

- Use the File/Open menu in Jack™ to load the Training Environment for this exercise. Look for a file named TrainingOct112006.env

- Note: you will need to have all of the Jack™ figure files on your local system or network so that the Jack™ software will be able to generate the desired images.

Hints for Navigating in Jack™

- Moving the viewpoint of a window will let you zoom in/out or rotate the scene.
    - In the toolbar at the top, click on the eye icon, the text below it will read View Control
    - In the window that appears, click on the Move button
    - Use the left mouse button to rotate the view right and left, or up and down
    - Use the right mouse button to zoom in and out
    - Use the middle mouse button to move the view up and down
    - The buttons can be used in combination
- Moving an object
    - In the toolbar, be sure Fig is selected in the drop-down menu
    - Press the icon of the hand with the pointing finger
    - Scroll the mouse over the object until it turns yellow
    - Click on the object to select it
    - Verify that the object you want is the one listing in the text box left of the hand icon
    - Hold the left mouse button and move the mouse back and forth to move the object in the X-direction
    - Hold the middle mouse button and move the mouse back and forth to move the object in the Y-direction
    - Hold the right mouse button and move the mouse back and forth to move the object in the Z-direction
    - Move a few of the images you loaded around to get a feel for how it works, and to separate them
- Rotating an object
    - Select object you want to rotate, click on the object with right mouse button (arrows will appear under the origin of the object)
    - Press and hold shift key on keyboard
    - Press left, right or center mouse button to rotate along the X, the Y, or the Z plane
- Getting an object's location
    - In the toolbar, be sure Fig is selected in the drop-down menu
    - Press the icon of the hand with the pointing finger
    - Scroll the mouse over the chair until it turns yellow
    - Click on the chair to select it
    - Verify that the object you want is the one listing in the text box left of the hand icon
    - Read the X, Y, and Z locations out of the cm: textboxes on the right side of the toolbar
- Getting the location for a piece of an object
    - In the toolbar, be sure Site is selected in the drop-down menu
    - Press the icon of the hand with the pointing finger

    - Scroll the mouse over the object until the text reads the name of the piece you are looking for
    - Click on the object to select it
    - Verify that the object you want is the one listing in the text box left of the hand icon
    - Read the X, Y, and Z locations out of the cm: textboxes on the right side of the toolbar
- Saving the environment
    - Go to the file menu and select save scene
    - Name the filename you would like to call the scene (e.g. INL-Training). Note, the .env extension will automatically get added to the filename.

Chapter 2: Using MIDAS

This training manual is organized in a manner consistent with how one would design a simulation and attempts to track the organization of the tree view that appears in the MIDAS software.

Overview
The MIDAS simulation environment can be thought of as possessing three primary organizational structures; the inputs, the processing structure, and the outputs. These are similar to the information-processing model of the human. MIDAS inputs include the operational environment (e.g., flight profiles, scenario objects and events, etc.), the operator tasks and operator process models (e.g., algorithms that represent operator characteristics such as expertise). The MIDAS processing model is comprised of a task manager model that schedules tasks to be completed, definitions of the state of models within the physical simulation, a library of "basic" human primitive models that represent behaviors required for all activities such as reach, and cognitive models such as operator perception. The MIDAS output model generates a runtime display of the task network, the anthropometry as well as mission performance.

The task network environment as well as the interaction among the various models when MIDAS is opened can be found in Figure 5. Figure 5 illustrates the operator procedures (light yellow), the properties (blank white column), the tree view of the components in MIDAS (simulations, external connections, environment list, crewstation list, operator list, situation awareness list, primitives list, vehicle list, task list, functions, variables and sharp talk 360), and the code output (dark yellow) pane. If you inadvertently close a window or want to get back to this default layout, select toolbar View > Layout > Reset Screen Layout

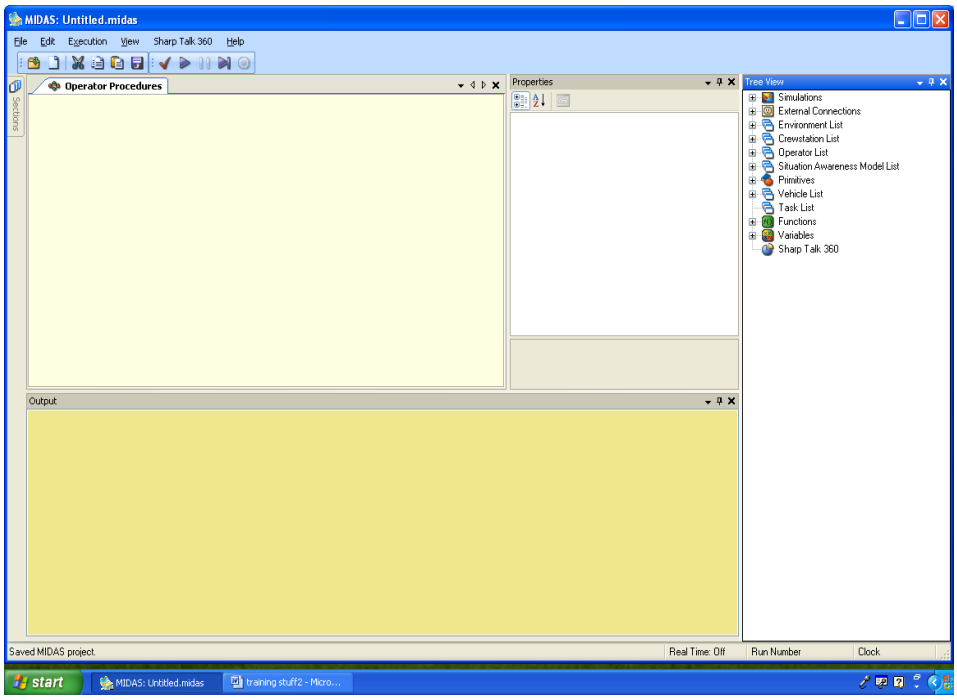


Figure 5. Default MIDAS Four Panel Screen.

Creating a Project

A project is a collection of simulation definitions. Simulation definitions are a collection of models where models are operators, operator procedures, vehicles, crew stations, environments, events and situational awareness weightings. This allows the user to reuse operators, procedures, vehicles, crewed stations, or environments.

- Select the Save As… menu option under the File menu
- Navigate to a folder where you would like your projects to be saved
- Enter a name for your project in the File name text box and select the Save button

Sidebar: All the simulation definitions and models are stored in files with the .MIDAS extension. If you would like to give your project to a colleague, you will need to give them this file plus any files created on the Jack machine. It's useful to keep all your Jack files for a given analysis in a subfolder for easy portability.

Tip: Select the floppy disk icon on the tool bar from time to time while working to protect your work should you loose power or experience an irrecoverable software error.

Creating a Simulation

- Select the simulation created by default when the project is created. Look for User Simulation 1 in the Tree View under Simulations
    - Click on the little '+' symbol beside simulation.
    - This will expand to a user simulation where the user will define the various elements that they would like to link in their simulation (the environment, Crewstation, operator, vehicle, functions, etc.
    - The default screen view should also present the properties that are associated with the user simulation

- the property grid should include the categories of name, miscellaneous (this is where you will define the speed of the simulation), the models that will be included as properties of the simulation(crew station environment operators and vehicle), (a field that allow use the user to insert notes about the properties of the user simulation), runtime settings that includes animate simulation (this is the flag that tells Midas to communicate with Jack),display messages(this is used for debugging and increases the amount of time it takes to run a simulation; so turn this off for demos), flags to enable the coordinated situation awareness, the SEEV settings, the number of runs, whether runtime charts are turned on or off, the increment of simulation time (every one hundred milliseconds), and whether workload management is turned on or off (See Figure 6).

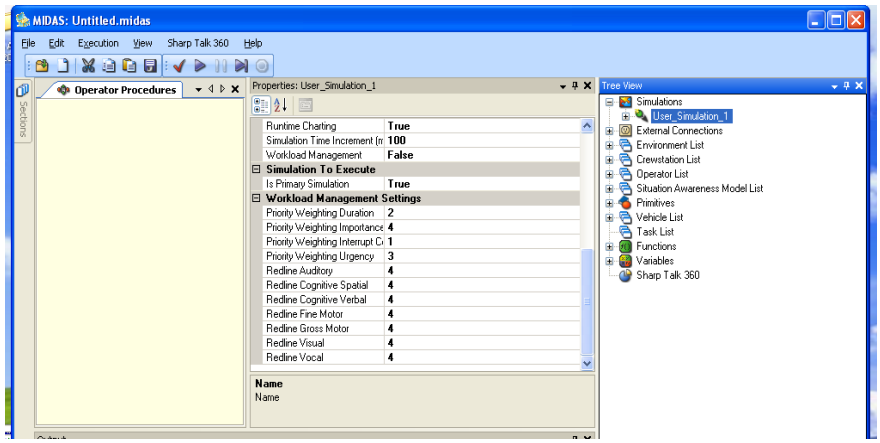


Figure 6. Properties of the Simulation (Definitions).

- In the property grid, type the name you would like for your user defined Simulation in the Name field under Identification (the white rectangle can be typed in).
  - Note, this portion of the tree view, will get populated with the definitions that you will be creating in the environment, the crew station, the operator, and the other functions that will be part of your simulation.
  - Set Animate Simulation to True (this allows you to see the visualization from the jack software)
  - You'll also have a number of settings for the workload management model in the event that you would like to manipulate and delay the onset of multiple, concurrent tasks.
- **After** you have created Models below (Environment, Operator, Vehicle, Event Set), you will be reminded to assign them to this Simulation

Side bar: If you'd like to create an additional Simulation, right click on Simulations in the Tree View and select Add Simulation from the context menu.

Side bar: The red X's indicate that there has been no model definition applied to the component model (e.g. no vehicle, environment, event set, or operator have been defined yet). In the beta version of MIDAS, there needs to be one model defined in each component classification

Simulations Node

The simulation node is a node that contains all of the elements that will be in your simulation (model). It is the part of the MIDAS architecture that links the external connections, the environments, the crewstations, the operators, the SA, the vehicles, and the functions together to create the full simulation. It also lists the Sharp Talk 360 software that is used to communicate to the External Environment coming into MIDAS (see Figure 7).

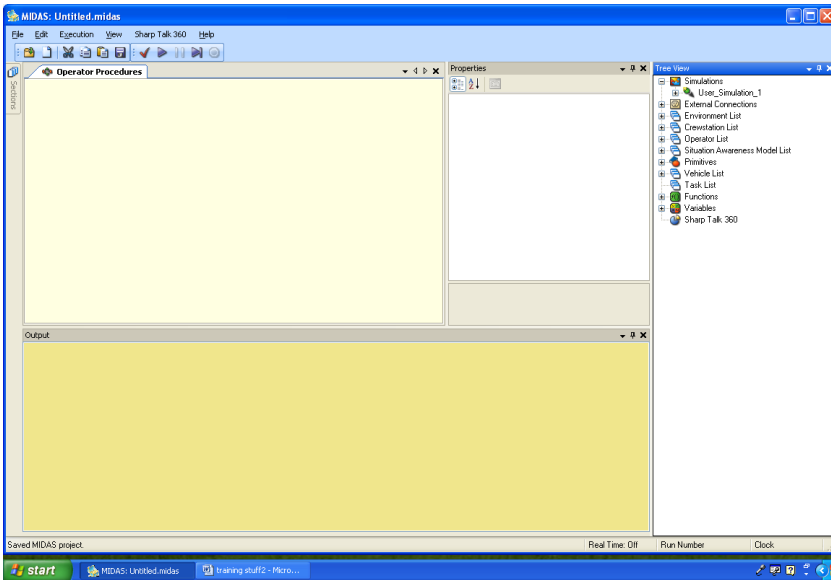


Figure 7. User Simulation Tree View Expanded.

Open the Simulations Node – move cursor to the little plus icon and expand the contents of the node. As a default (when a new simulation is being created) contained within the simulations node, you will see an environment model, 'environment' and an Operators 'operator1' node. You will see a red X on the operator node. This is okay as you have yet to define an operator for the simulation. The red X will go away when you define the operator (see Figure 8).

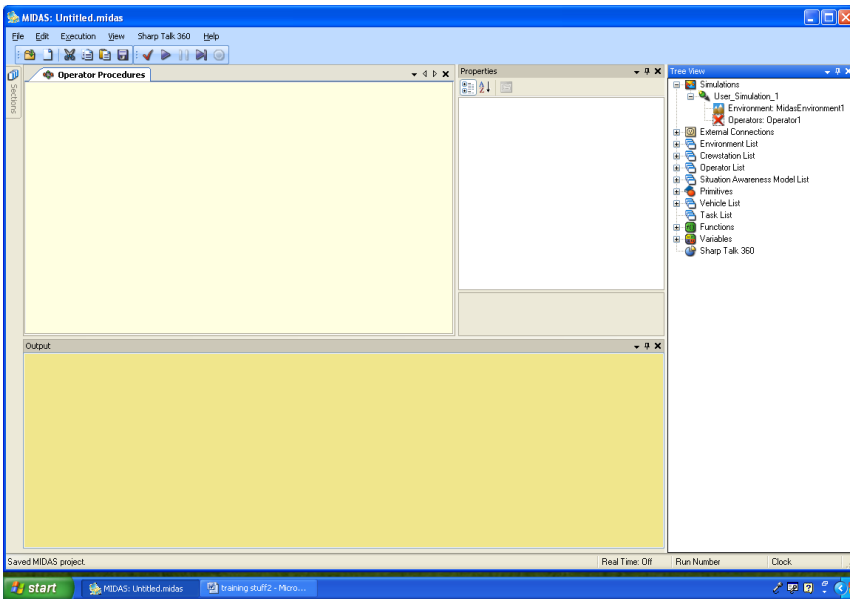


Figure 8. Expanding the Simulations Node.

External Connections:

This is a node where you define whether the MIDAS architecture will be linked to an external piece of simulation software to represent the physical human shape (anthropometry) and CAD renditions of the environment (see Figure 9). To open, click on the + sign, then select the anthropometric software (in this case, Jack). This will bring up a properties window that defines the ip information for the anthro software.

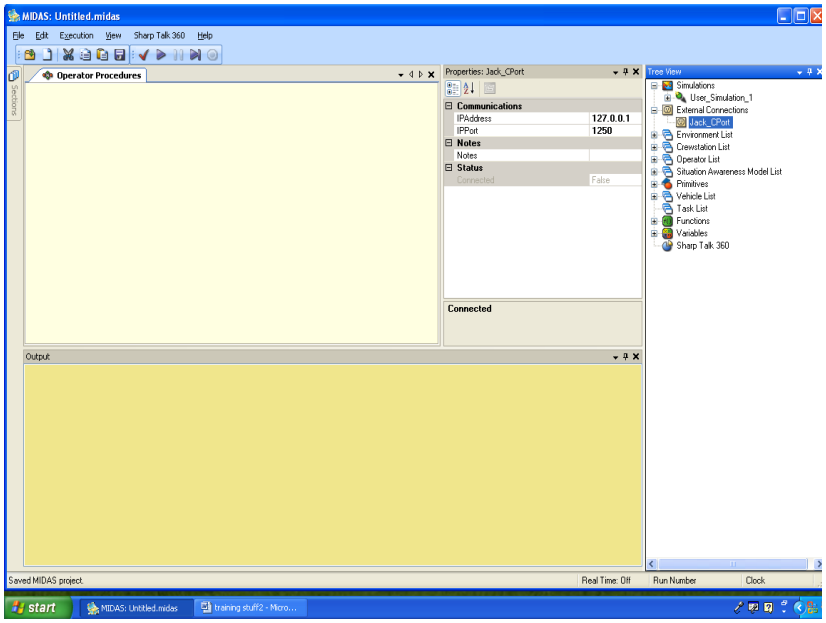


Figure 9. External Connections Node in the Tree View.

- Expand External Connections in the Tree View by clicking on the + to the left of it (see Figure 10)
- Set up the Jack™ CPort

- In the Properties window, type the Jack™ IP Addresses in the IPAddress field.
- **Hint**: to get the Jack™ IP address. Go to the Jack™ machine
  - Do Start from Windows and select "Run…"
  - In the cmd prompt, type ipconfig
- Type the Jack™ Port into the IPPort field.
  - Hint: This is usually 1250 for Jack™.
  - You can verify by loading the CPort plugin in Jack™.

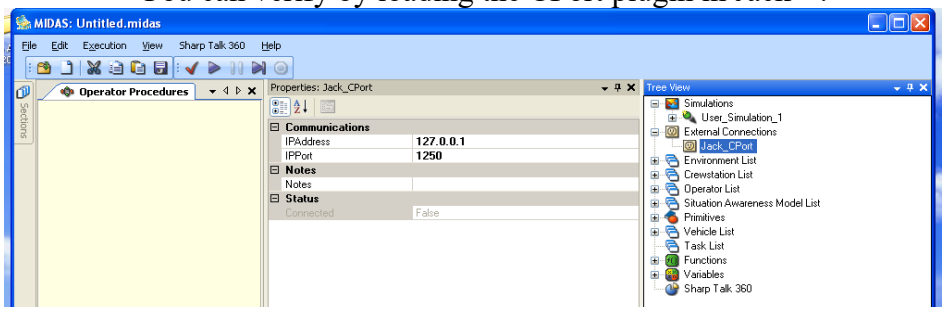


Figure 10. Setting up External Connections for Anthropometric Character.

Environment List

The MIDAS software automatically generates a default environment when a new simulation is created. This is a node where the environment that the simulation will use is defined. Multiple environments, each with different characteristics (lighting, terrain, visibility, features as shown in Figure 11) can be used, and selected in the simulation, thereby allowing whether the environmental variables impact the model's performance (operator's time to respond, detect information, workload patterns, and awareness). Each of these environmental characteristics impacts the basic MIDAS behavioral models according to empirical data. To set up the Environment List, 'Right Click' on the Environment List, Add Environment Model.

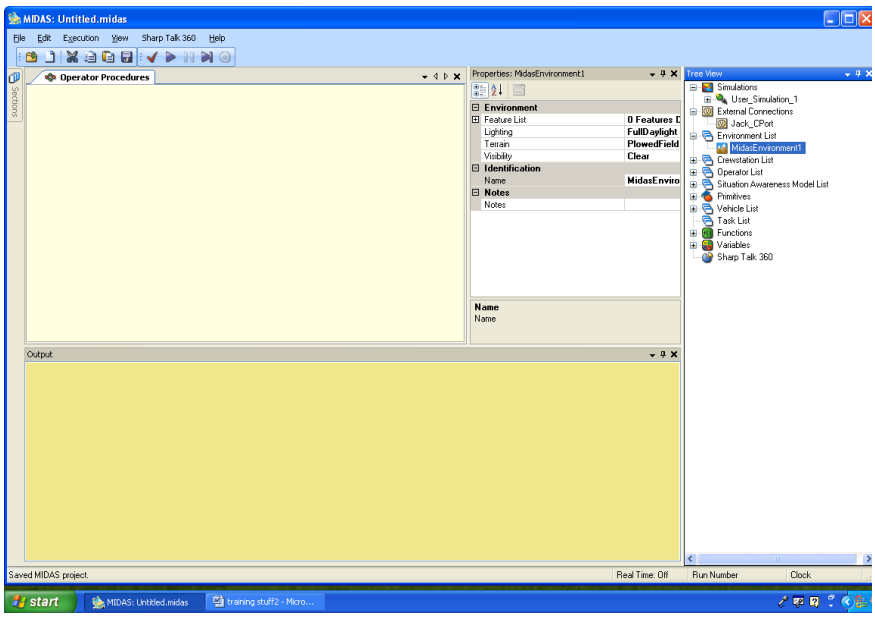


Figure 11. Setting up the Environment to be used in the MIDAS Scenario.

Crewstation List

The next section describes the steps needed to create the crewstation, the workstation that the operator will reside in and interact with. Before all of the component models that will be in the scenario can be populated, how the modeled operator detects the information from the crewstation and the environment will need to be decided upon, as this decision will determine how the operator's scan pattern is created and how the scan pattern interacts with the crewstation and the environment. Two choices exist for this decision. The operator scan pattern can be driven either probabilistically or by the SEEV (salience, expectancy, effort and value; Wickens & McCarley, 2008) model. For a full description of the SEEV model's implementation in MIDAS, see Gore, Hooey, Scott-Nash and Foyle, (2008).

If the scan pattern is a probabilistic scan, the model will need flags set in the initialization settings of the simulation branch to outline that the probabilistic scan is being used for visual attention. All of the fixation points according to the context in the scenario being modeled will need to be defined. Select the rules for the decision on the path that the model will select to generate the visual scan (probabilistic, tactical, or multiple). To generate the probabilistic logic for the model, determine the probability values from empirical literature, and then enter the values in the routing task by double clicking on the task node. Figure 12shows the options that become available when double clicking the routing task. The values in the path define the percentages that the model will use when the task fires.

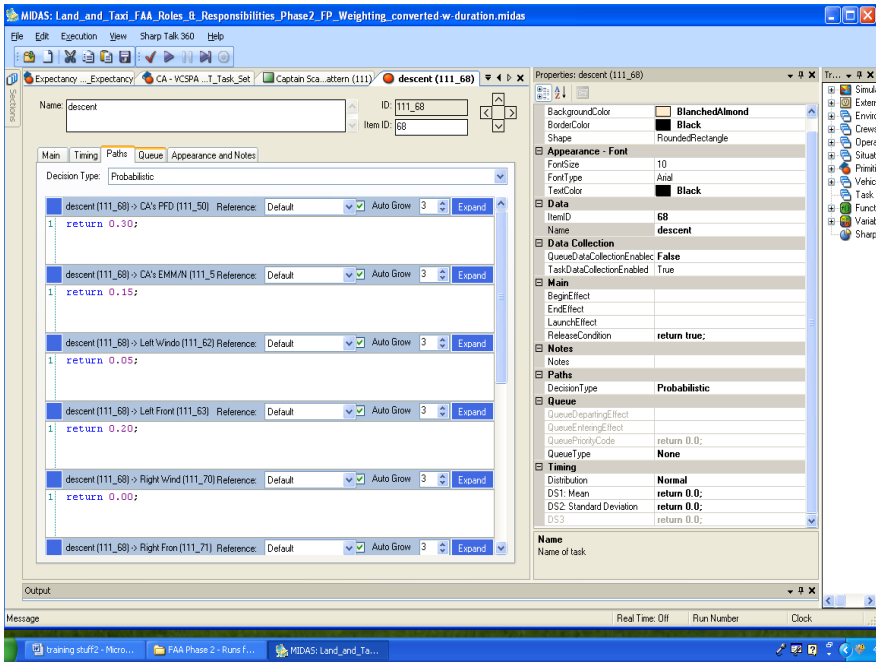


Figure 12. Example of a Probabilistic Scan in a Model.

If the SEEV attention model is desired, then the tasks in the operator procedures will need to be bracketed with a start SEEV task at the beginning of all of the operator tasks and an end SEEV task at the end of the operator tasks. In addition, a SEEV start flag is needed in the visual attention / scan pattern network (illustrated in Figure 13). To ensure

a continual state of workload due to a continuous task as the one on a cockpit, the model assumes the operators are each performing both a visual SEEV scan and an auditory scan continually throughout the entire model without interruption in addition to the procedural tasks that need to be performed throughout the different phases of the flight. The operator when not performing a procedural task typically engages in a cockpit-scanning pattern and also continuously monitors for any audio communications. In order to capture this workload accurately, an operator procedure is needed in the model for every operator at the beginning of the network diagram in the top level of the network. These are typically called:

- Captain Scanning Pattern Primitive (task ID)
- First Officer Scan Primitive (task ID)

Within each primitive, the following two operator primitives appear:

- SEEV Fixate
- Auditory Monitor

A path drawn from each primitive back onto itself accomplishes the continual looping of these scans.

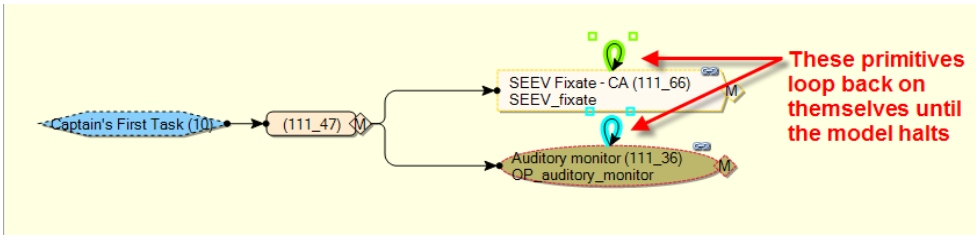


Figure 13. SEEV Fixate Start Task in Two Looping SEEV Model Applications (Visual, Auditory).

The SEEV weightings on each of the fixation points in each of the scenario contexts along the dimensions of expectancy (Figure 14), relevance of the AOI (task importance), and situational elements (Figure 15).

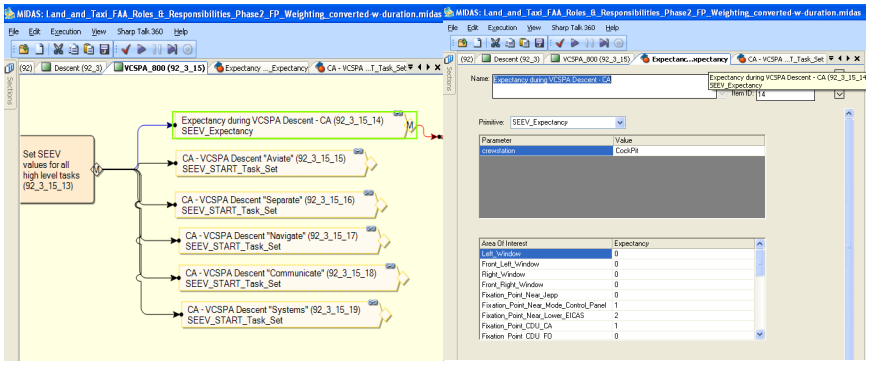


Figure 14. Example of Expectancy and AOI Weightings.

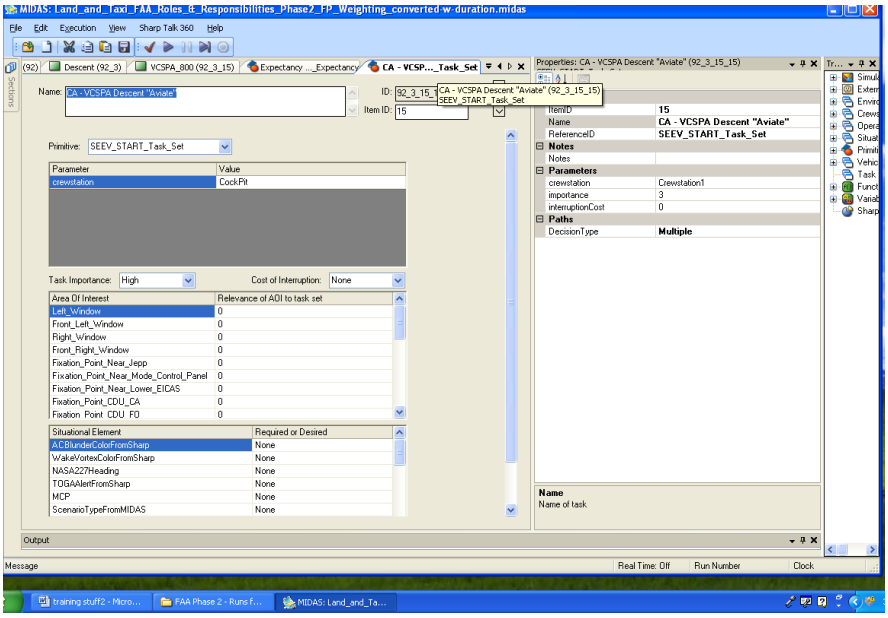


Figure 15. Task Importance, Relevance of AOI, and Situational Element.

SEEV Fixation Point Priority

For any point along the model where it has been determined that a crewmember should purposely focus on a specific fixation point, the modeler may add an immediate weighting for that fixation point. This weighting factors into the SEEV calculations. If weighted heavily enough, the crewmembers eyes will be drawn to that specific location for an amount specified.

The SEEV Fixation Point Weighting parameters have been added to several Operator primitives. For those points in the model where a fixation is required but the operator is not performing any operator-based primitives, a user-defined primitive can be defined that has the same parameters (fixation point, weighting, duration; Figure 16).

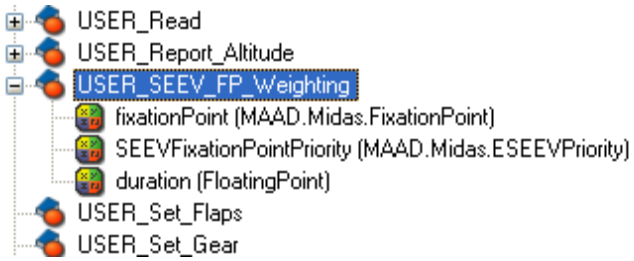


Figure 16. SEEV Fixation Point Parameter Weightings.

For each instance of the primitive, a modeler may set the following (see Figure 17):

- **fixationPoint**: any fixation point in the crewstation
- **SEEVFixationPointPriority**: high, medium, low or none
- **Duration**: the number of seconds the weighting should last

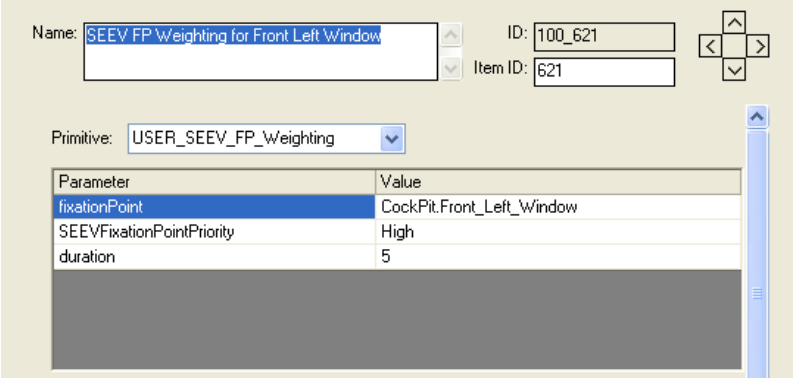


Figure 17. Setting SEEV Parameters Values in the MIDAS Task Network.

In case of user defined primitives, the modeler needs to set in the time field the following line of code: return duration;

Creating the Crewstation List

This is a node where the crewstation and its associated details that the simulation scenario will use are defined. Multiple crewstations, each with different layouts/details can be created. The one to be used in the simulation given the scenario requirements can be selected by clicking on the desired crewstation. This allows different crewstation layouts, configurations, component attributes (colors, noise, etc.) in order to evaluate the layout's impact on the model's performance (operator's visual attention, time to respond, detect information, workload patterns, and awareness) to be saved off and selected for comparative analysis. Each of these environmental characteristics impacts the basic MIDAS behavioral models. The MIDAS software automatically generates a default crewstation.

When crewstation is selected with the cursor (shown in Figure 18), the properties window will populate with attributes, component states (discrete state components), coordinated SA, geometry, identification, and memory (WM/LTM decay-able). The geometry bounding box information will come from either: 1) the visualization software that you are using (Jack), or, (2) from manually entering the information.

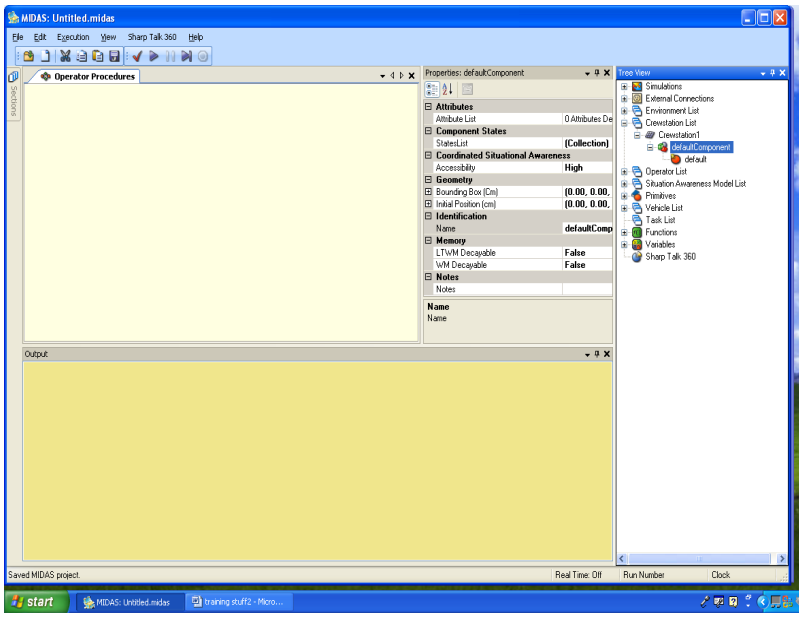


Figure 18. Crewstation Definition in MIDAS v5.

The crewstation attributes can be populated/defined. To do this, click on the component, name it according to a desired name, and then open the attributes list (top line) in the properties window. This will bring up the Attributes collector editor (see Figure 19).

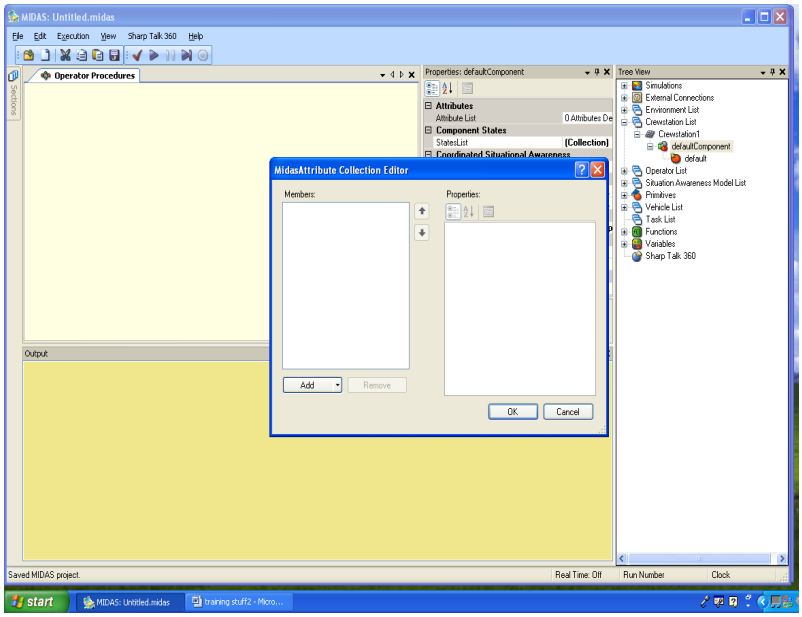


Figure 19. Example of Assigning Attributes through the Attributes Editor in the Crewstation.

In the Attributes collector editor window, the visual or auditory attributes associated with the crewstation element can be created and defined by selecting the “add” attribute button (Figure 20).

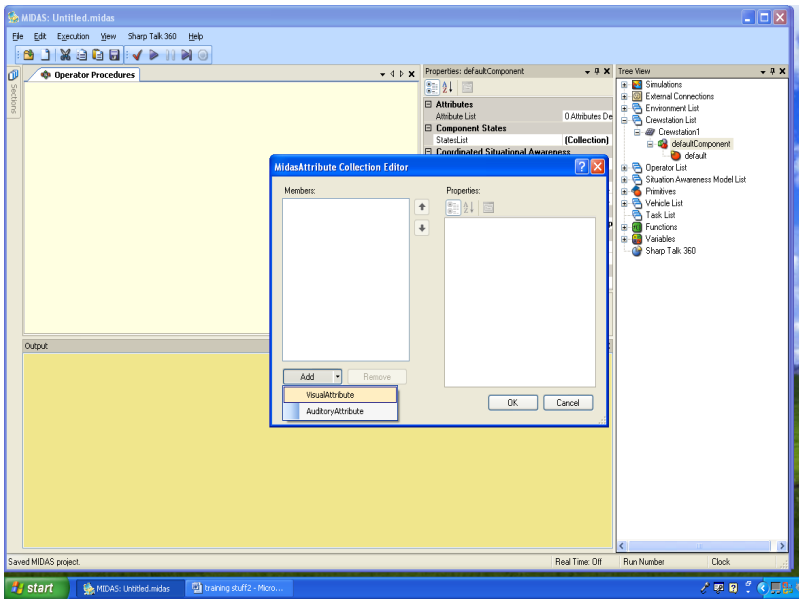


Figure 20. Example of Add Attribute Function to a Crewstation Component.

To define the properties, select the desired attribute, then define it via the pull down menu from the 'collection …' option in the software as shown in Figure 21. Note, Collection can be defined according to the following: undefined, size, shape, texture, color, contrast, transparency, text, digital value, spatial value, spatial symbolic value, light emitting, heat emitting, class, reflectance, visibility, and lighting condition.

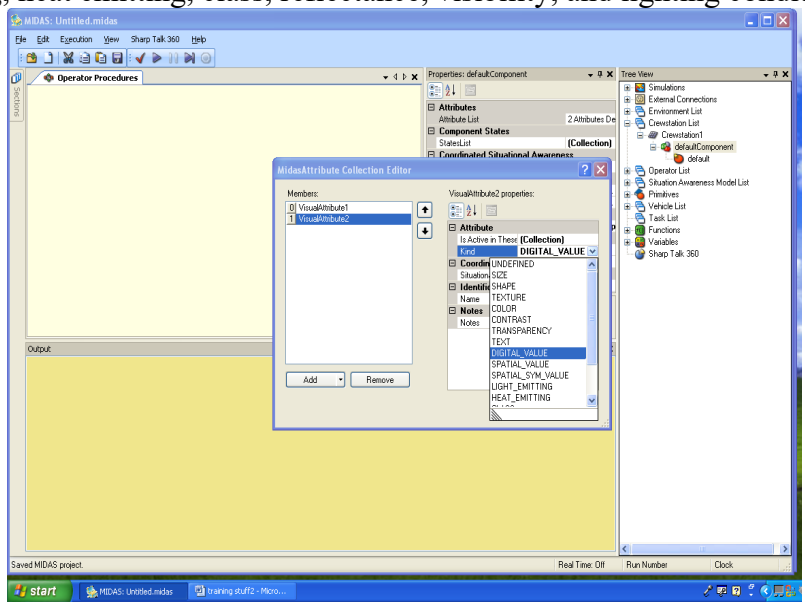


Figure 21. Attribute of a component definition in MIDAS v5.

When the attributes window is closed, the attributes list will now have the number of attributes defined in Figure 21 (see Figure 22).

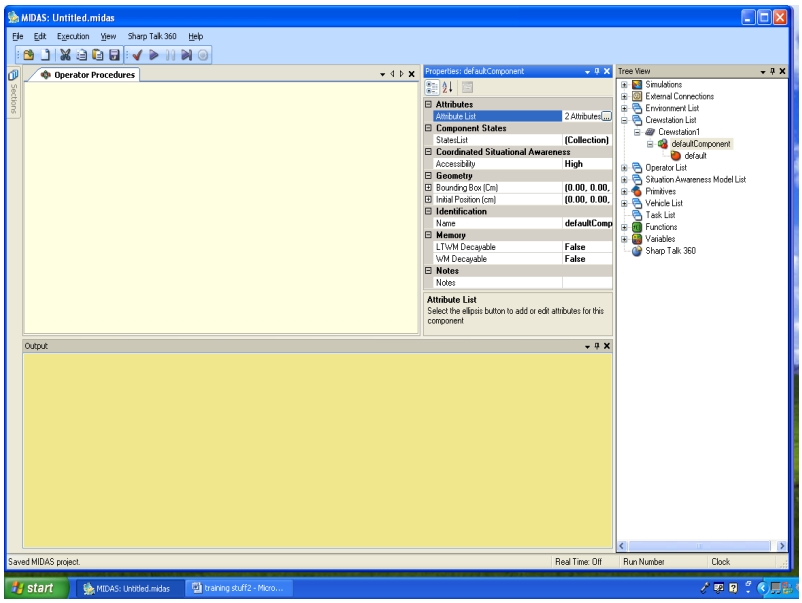


Figure 22. Populated attributes list.

Define Component States

The MIDAS simulation that you are creating automatically generates a default component with a default state for you. Component states are the conditions of the objects that are contained within your scenario. When a component state is created, click on the default component in the tree view list, and right click the mouse button to define the component definitions. These can be fixation point based, equipment component based, or even imported from the Jack Software (shown in Figure 23).

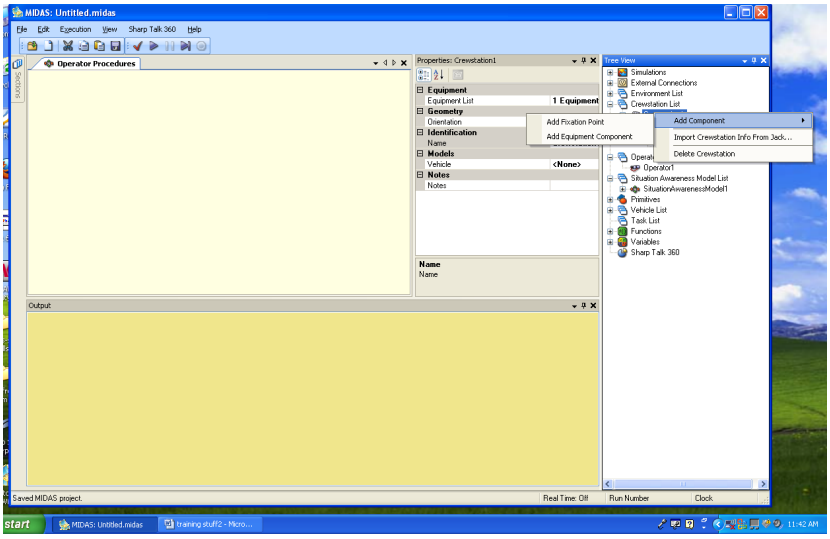


Figure 23. Defining component states in MIDAS v5.

Once a component state (object) for the scenario has been created, it will need to be defined as illustrated in Figure 24. The states of component models can be defined through the properties window of the component. The default state of the component is set to true. Add a second state if you want to define a state associated with false.

Sidebar: the geometry information will be brought in from the CAD software package if you are using one. If you are not using a CAD package, then you can insert the XYZ coordinate information by hand.

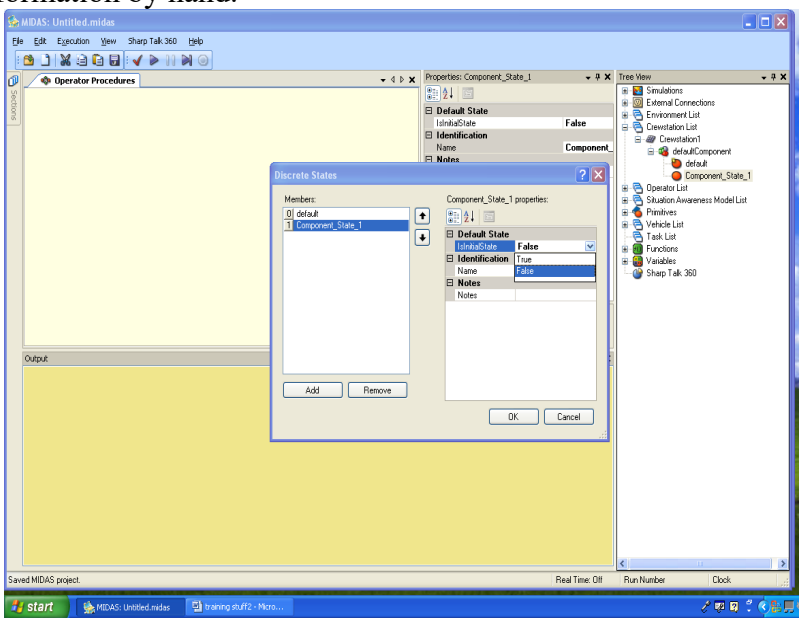


Figure 24. Component states definition.

Operator List

The MIDAS simulation automatically generates a default operator. This will be labeled Operator 1. This name can be changed by replacing Operator 1 with a name of your choosing. This operator will be behaving inside of a crewstation and as such, will need to be assigned to a crewstation. As shown in the figure below, the operator is assigned to the desired crewstation by selecting the name Operator 1 in the 'models' portion of the properties page, then hit return. This will add the operator to the crewstation and will bring in the XYZ coordinate information from the crewstation. The crewstation will now have a little '+' sign beside it. If this is expanded, the crewstation will be listed here (See Figure 25).

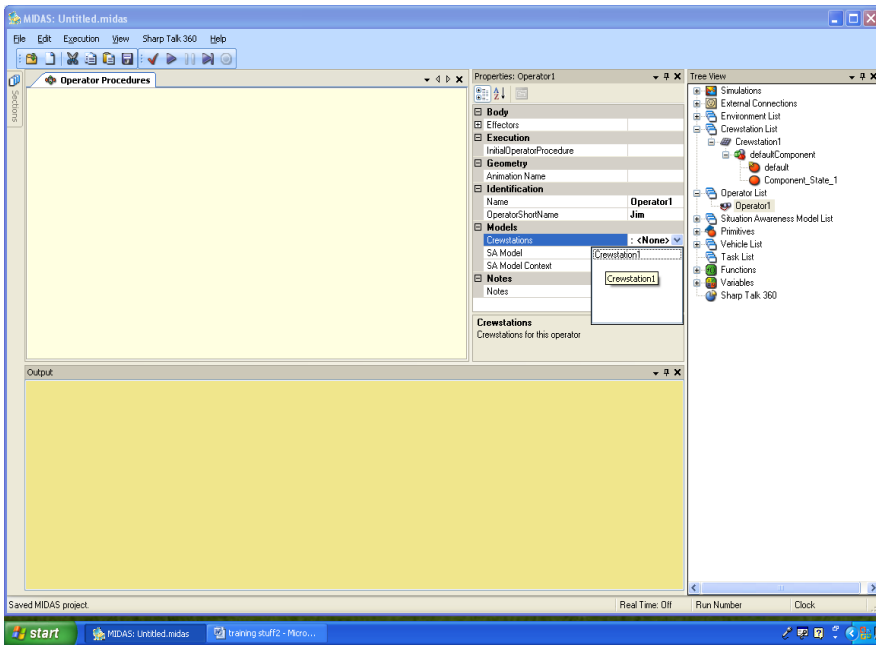


Figure 25. Operator model definition.

Situation Awareness Model List

The Situation Awareness (SA) model is made up of three primary components; the contexts, categories, situational elements (SEs) as illustrated in Figure 26. The situational context refers to the operator's goals and the task importance. The environment is broken into SEs, the elements necessary to complete the higher-level task as defined by required/desired to complete the goal and information accessibility. For example, for the task "Aviate", the SE 'altitude' is required, but the SE 'angle of attack' (display of the wing angle relative to the wind to warn of stall condition; Hooey et al., 2009) is desired.

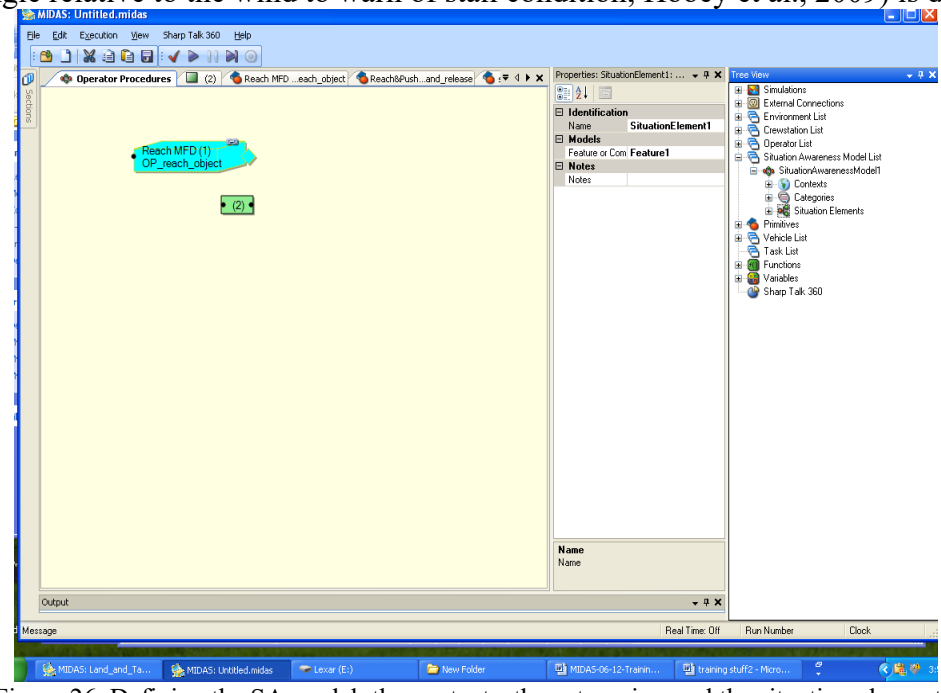


Figure 26. Defining the SA model; the contexts, the categories, and the situation elements.

MIDAS Situation Awareness Model: Information Accessibility and Data Limits
The Accessibility, Detection to Comprehension calculation, and Data Limits are designed to expand the resolution of crewstation component design, to allow for finer-grained resolution of the model and more accurate component testing. Currently there are two Display Elements, which impact times to comprehension: *Accessibility* and *Text Length of Messages*.

Accessibility

Currently accessibility has four levels [None, Low, Moderate, and High], which represent the difficulty/time required to access a given component (see Figure 27).

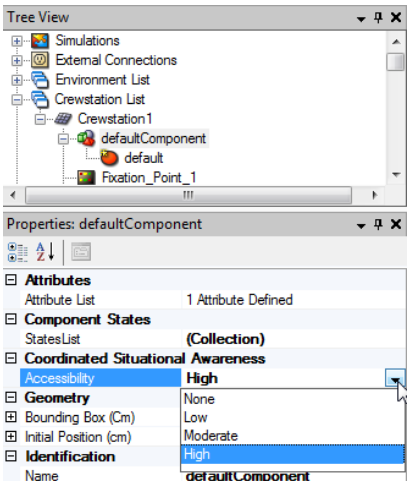


Figure 27. Current component accessibility.

Text Length of Messages

Text length is currently the only Detection to Comprehension (DTC) parameter in MIDAS. It is accessed when a display has an attribute set to TEXT and the *EQUIP_set_attribute_value* sets the attribute's value in the *value* field. The number of characters in the text is the parameter for calculting the DTC (Figure 28).

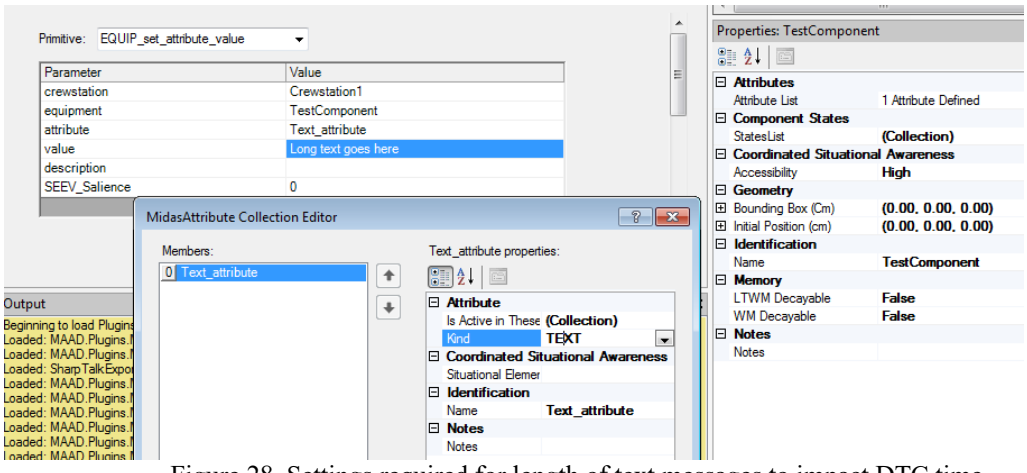


Figure 28. Settings required for length of text messages to impact DTC time.

The three components of the library of display elements and how they are planned to be implemented in MIDAS will each be discussed separately below, followed by a brief discussion of issues with combining multiple display elements.

Accessibility

The Accessibility is a measure of the information availability that assesses a time penalty for information acquisition based on whether information is embedded in a deep level of a display. Since this time penalty occurs before even detection can occur, this penalty effectively lengthens the number of fixation to get from Undetected to Detected.

Data Limit Library

Currently, all of the data limits have three values: None, Mild and Severe. The effect of a parameter is to "cap" the SA, making the Perception/SA of the attribute decay from Comprehended to Detected faster (i.e. it is in LTWM for a shorter period of time, although the slope of SA decay remains the same.) It will have no effect on the decay from Detected to Undetected, using either WM or LTWM as shown in Figure 29.

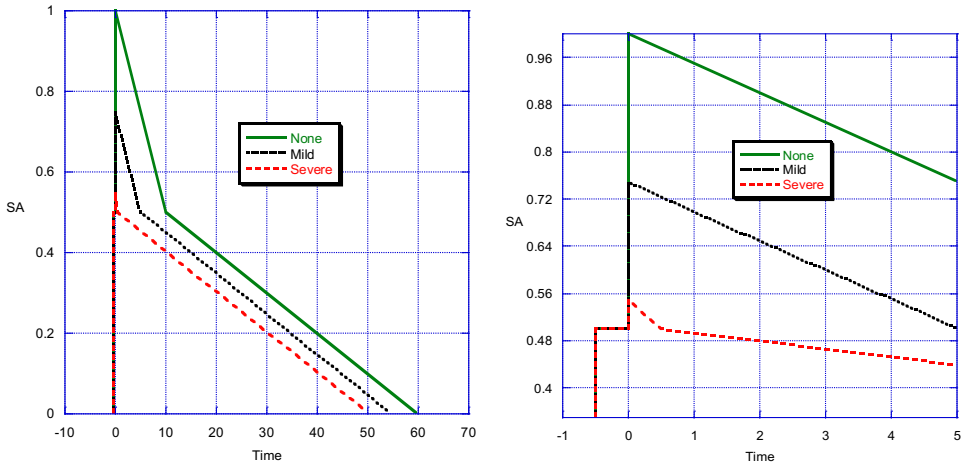


Figure 29. Effect of data limit on perception/SA (close-up on right).

Visual and Auditory DLs

Two of the DLs, Spatial Incompatibility (Auditory) and Working Memory Load are auditory in nature. These two will appear under the DL heading when an auditory attribute is selected. The other seven: 3D ambiguity, Legibility, 2D Display Tracking, Spatial Incompatibility (Visual), Symbol Unfamiliarity, Flight Path Projection, and Time Projection will appear under the DL heading when a visual attribute is selected.

Combining DLs

For Combining DLs, take the most severe of the DLs as the effect on SA.

Changing DTCs

Data Limits depend on the actual physical properties of the display and do not need a primitive to change Data Limits during a model run.

Defining Operator Procedures

Operator procedures are all of the tasks that are contained within the simulation. These include the operator primitives, operator procedures / tasks that the modeled operator will be set to undertake, the SEEV primitives, the equipment primitives, features, animation, user-defined, routing tasks, comments, as well as other capabilities like copy, cut and paste (see Figure 30).

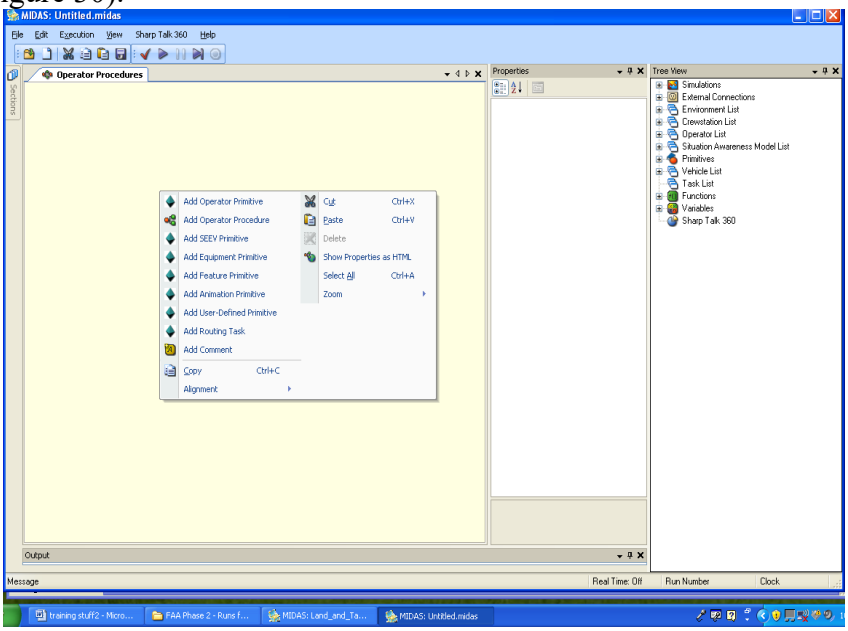


Figure 30. Right Mouse Click in the Operator Procedures Window Brings up Procedure Eleven Definition Choices.

Operator primitive

Right click the mouse and move the pointer to Add Operator Primitive – a little operator primitive task node will appear (Figure 31). It is numbered as a sequential number as a default.

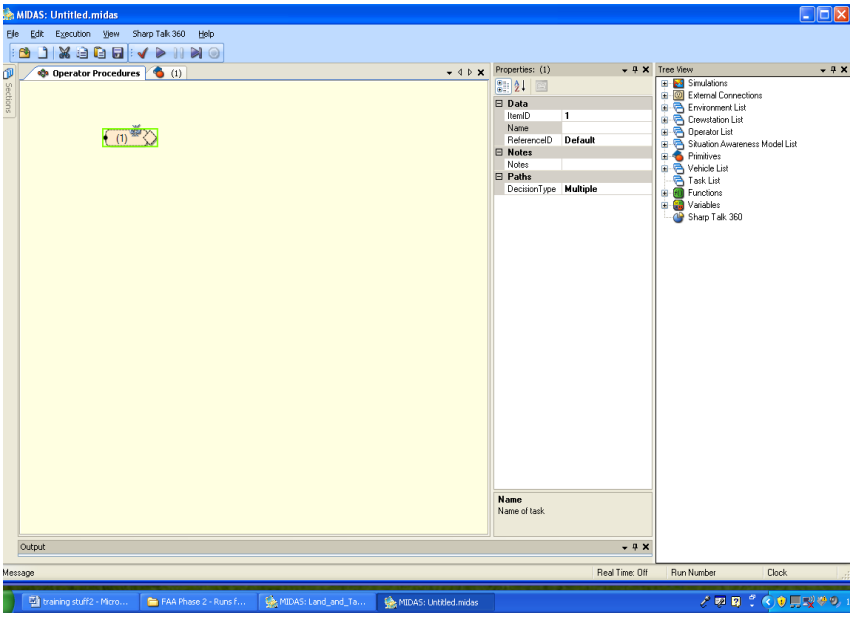


Figure 31. Adding an operator primitive into the operator procedure window.

Double click the task primitive node and a definition page will appear (see Figure 32). Here a name can be entered (e.g. Reach MFD). This task will then need to be tied to a MIDAS Operator Primitive (basis of behavior). For example, Reach MFD would be tied to OP_Reach_Object primitive.

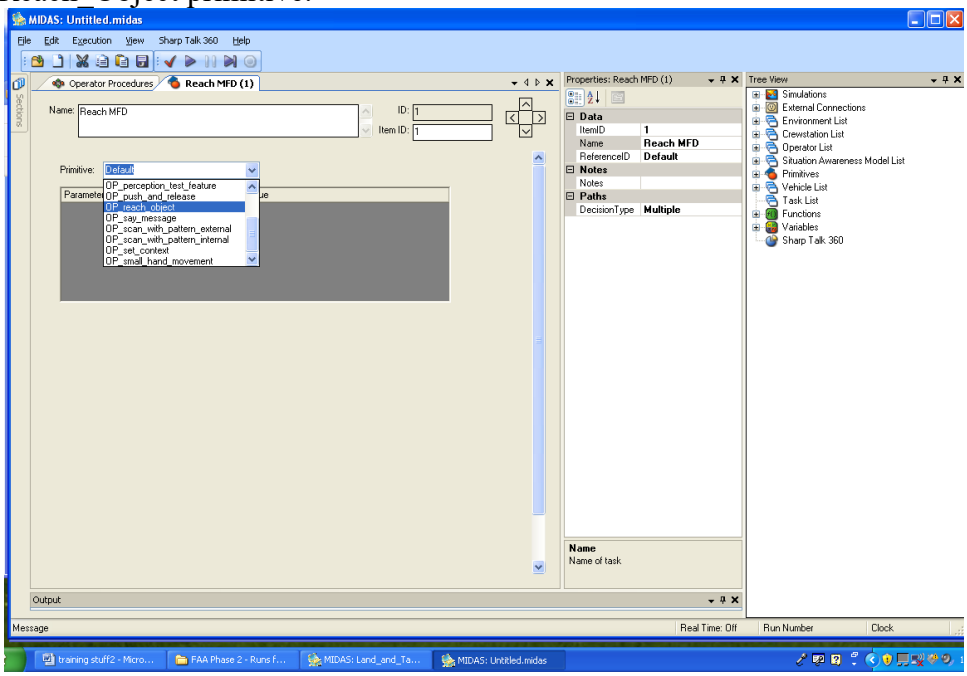


Figure 32. Example of a task primitive definition.

Once the primitive has been selected, a number of parameters appear (see Figure 33) that will need to be defined. The parameters are defined according to the models that have been created in the simulation (as a default, these are all blank fields that contain pull down options where models have been defined). These parameters include the crewstation, the equipment, the end effector of the anthropometry (hand or foot), the MIDAS task, the fixation point, and the SEEV fixation point priority (Figure 33). The data that populate the parameters come from the definitions created during the model development process. An example of the kinds of data that can be entered from these pull down menus can be found in Figure 34. The SEEV priority definitions can be found in Figure 35.

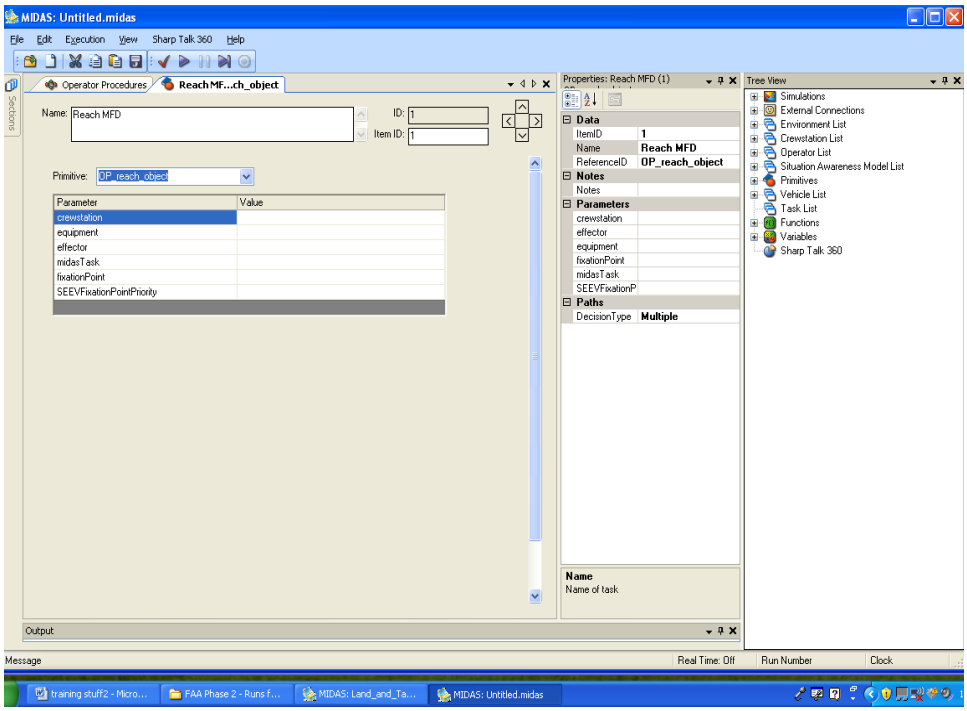


Figure 33. Parameters that appear when an operator primitive is created.

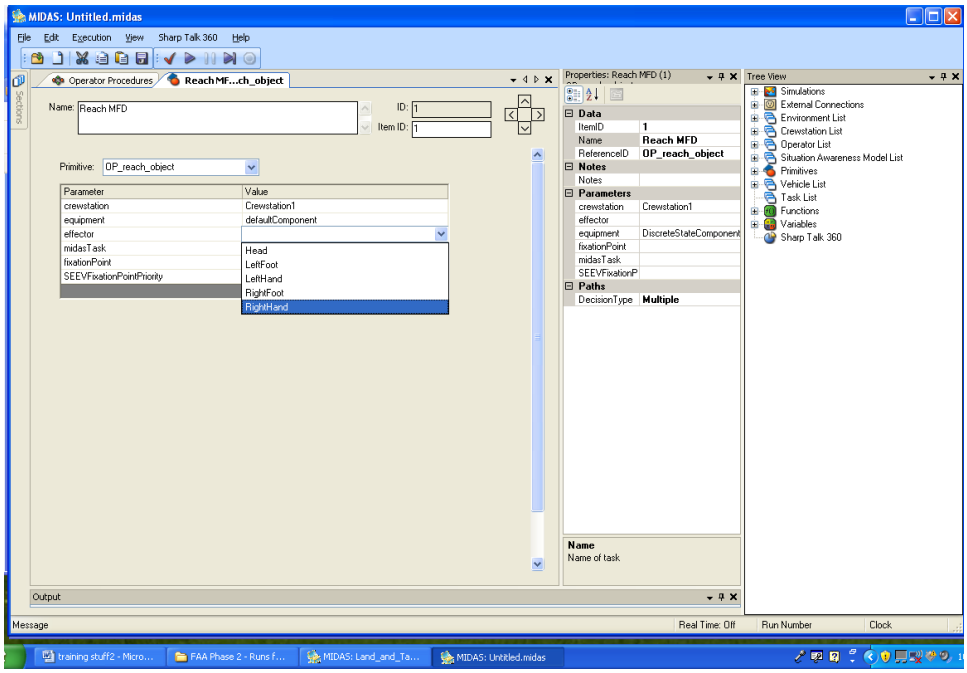


Figure 34. Example of the data that can be selected from the pull-down menu.

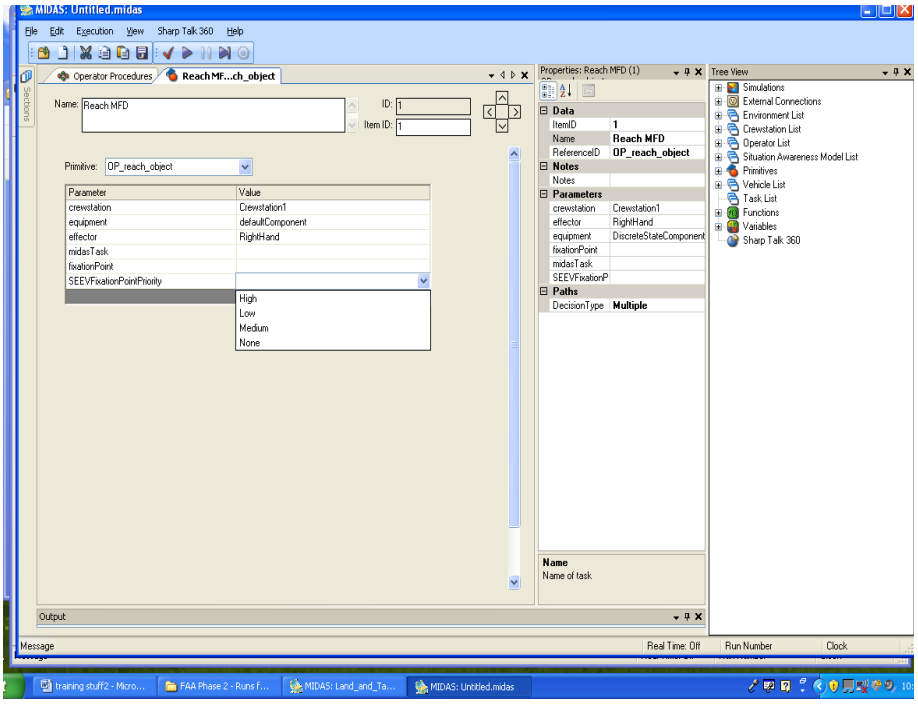


Figure 35. Example of the SEEV importance setting on the operator primitive assignment.

Operator Procedures

It is also possible to nest tasks together to create a "procedure". When the operator procedure is selected, a new window appears where you can link multiple operator primitives together to create a procedure. The top-level procedure will inherit the name given to the lowest level task. The procedure illustrated in Figure 36 is one that is a push and release procedure that is made up of a 'reach object' followed by a 'push object' (Figure 37). The reach object uses the Welford version of Fitt's Law while the small hand movement uses Shannon's version of Fitt's Law.

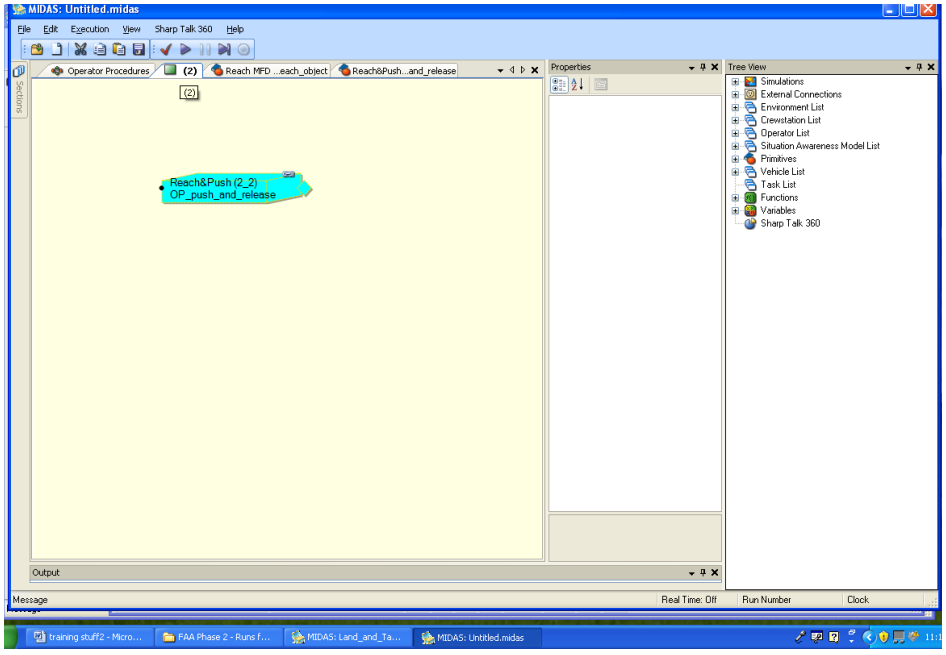


Figure 36. Creating an operator procedure (comprised of an embedded network).

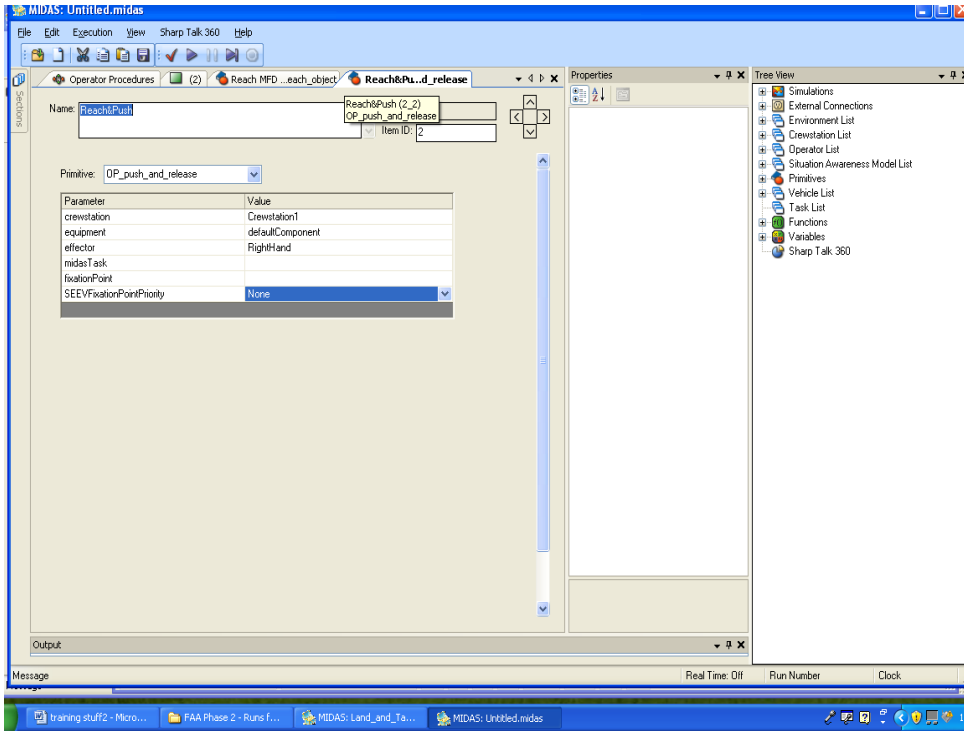


Figure 37. Creating an operator procedure (comprised of an embedded network).

SEEV Primitive

SEEV primitives bracket a task set (to drive the attention model around the environment) or to define the fixation points required of the attention model (Figure 38, Figure 39).

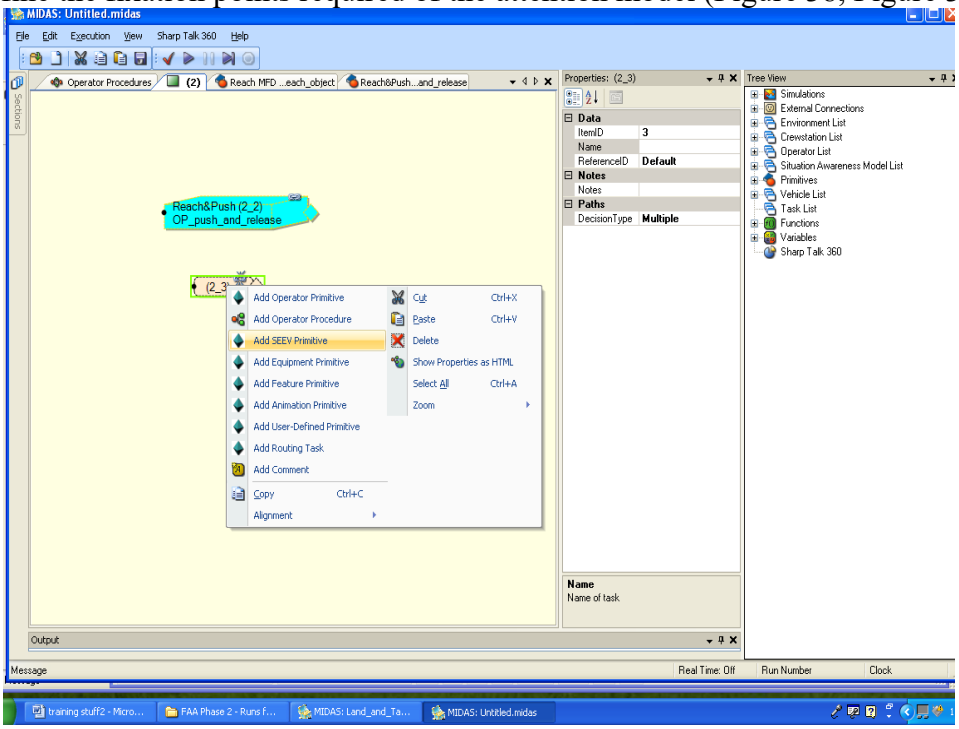


Figure 38. Parameters to define when using a SEEV primitive.

The screen snapshot in Figure 39 illustrates the FP selection and the required parameters to drive the eyeball around the environment. This particular snapshot ties the SEEV fixate to the crewstation1 because that is where FP1 is located.

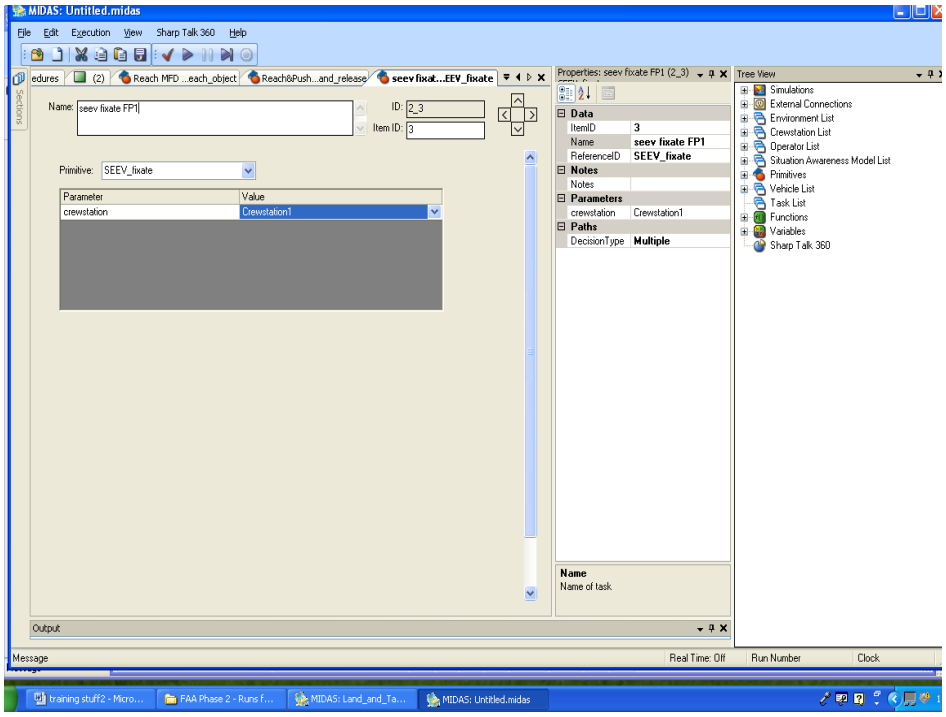


Figure 39. Parameters required for a SEEV fixation point.

Equipment Primitives

Equipment primitives are definitions that are provided to the CAD model in the crewstation. Equipment refers to all of the elements that change inside the cockpit as a function of the vehicle movement in the environment (e.g. displays, levers, dials, buttons, knobs, switches, keyboards, input devices, etc.). In order to define the equipment, add an equipment primitive (see Figure 40).

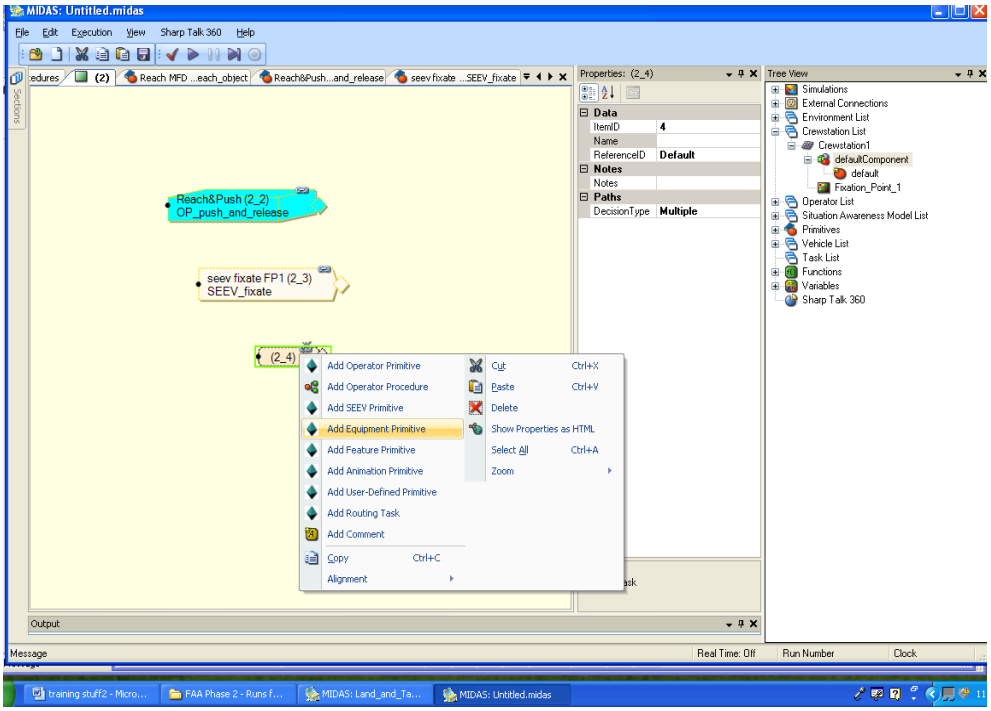


Figure 40. Parameters to define when setting / adding equipment primitives to the operator procedures network.

Then tie the appropriate equipment primitive to the equipment task created in the above step. The equipment primitive can be set to test attributes defined in the components of the crewstation (e.g. visual attributes). All definitions are available through a pull down menu (see Figure 41, Figure 42).

- Set Attribute value – this option allows a value to be set and upon which the visual attention model will compare against for driving the attention based on the salience of information. See the two figures below that illustrate the settings required to complete this set of embedded tasks.
- Set component accessibility – this option allows for the accessibility of information to be modified which in turn impacts the SEEV attention model.
- Set default state allows the MIDAS attention model to decide on whether information is changing rapidly or slowly (this influences the bandwidth, which in turn impacts the rate at which MIDAS attention decides to look at the information).

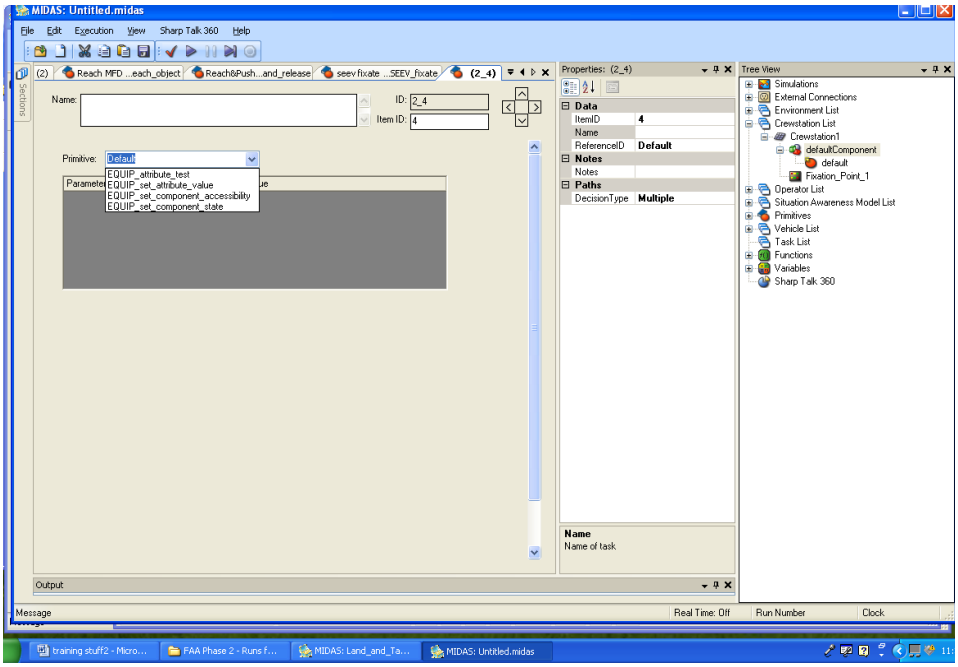


Figure 41. Parameters to define when setting attributes using the attribute primitive on a crewstation component.

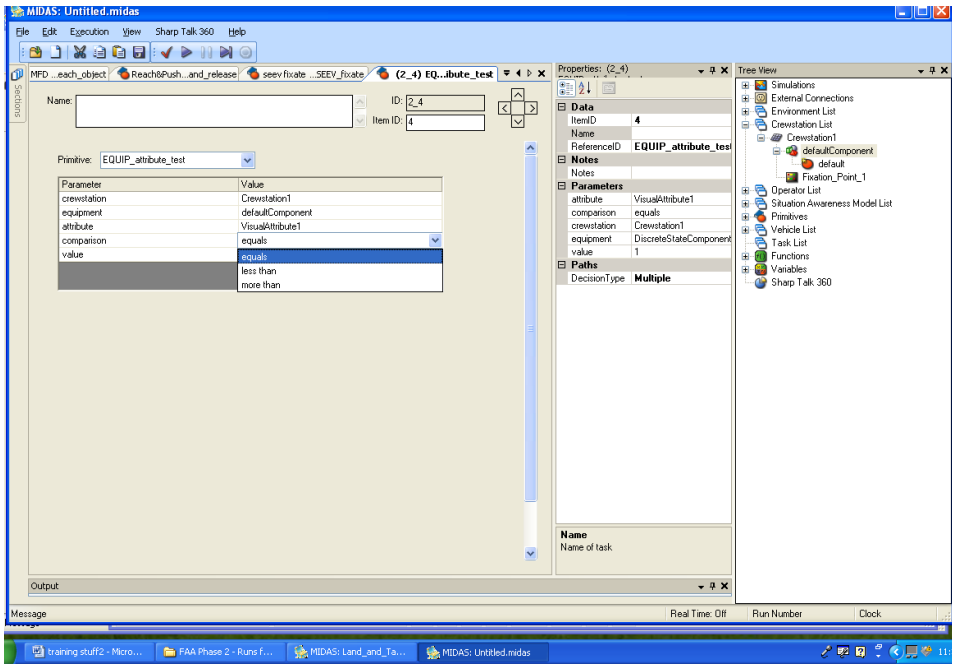


Figure 42. Parameters to define when setting value attribute on a crewstation component.

Feature Primitives

These are items that are defined in the environment, they are external to the MIDAS crewstation elements – items such as external signage, runways, other objects). To set these features of an environment, define the features in the environment by going to the environment node. Click on the + sign beside the feature in the properties window and populate the desired features (attributes can be placed on these as well) (see Figure 43).

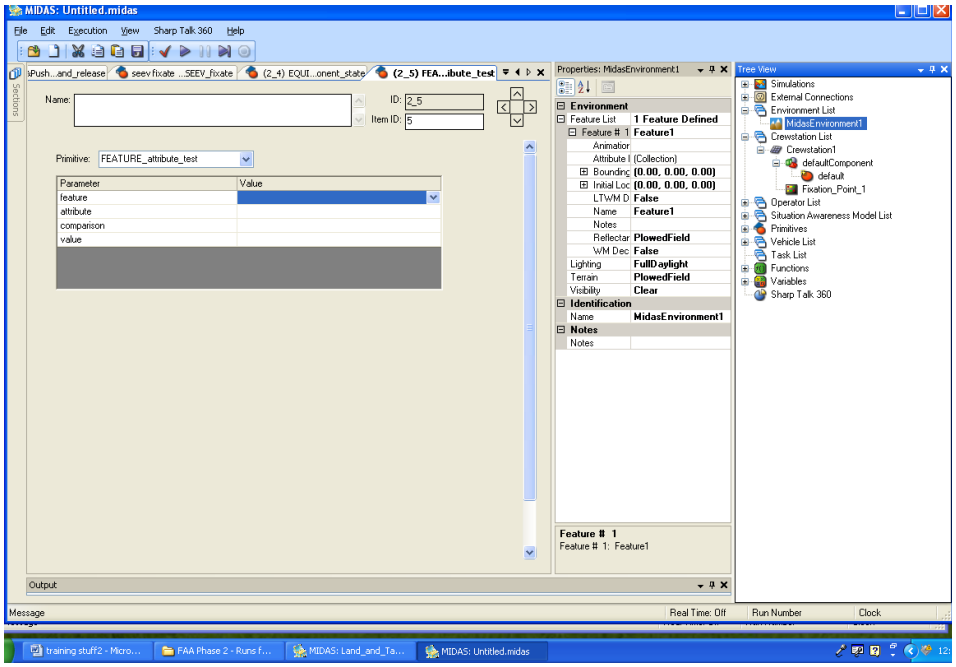


Figure 43. Parameters to define when adding features to an environment.

Once you have defined a set of environmental features, the model can use these in assessing whether features meet a criteria upon which to perform a desired set of actions. Select the features primitives from the available options using the right mouse click (see Figure 44).

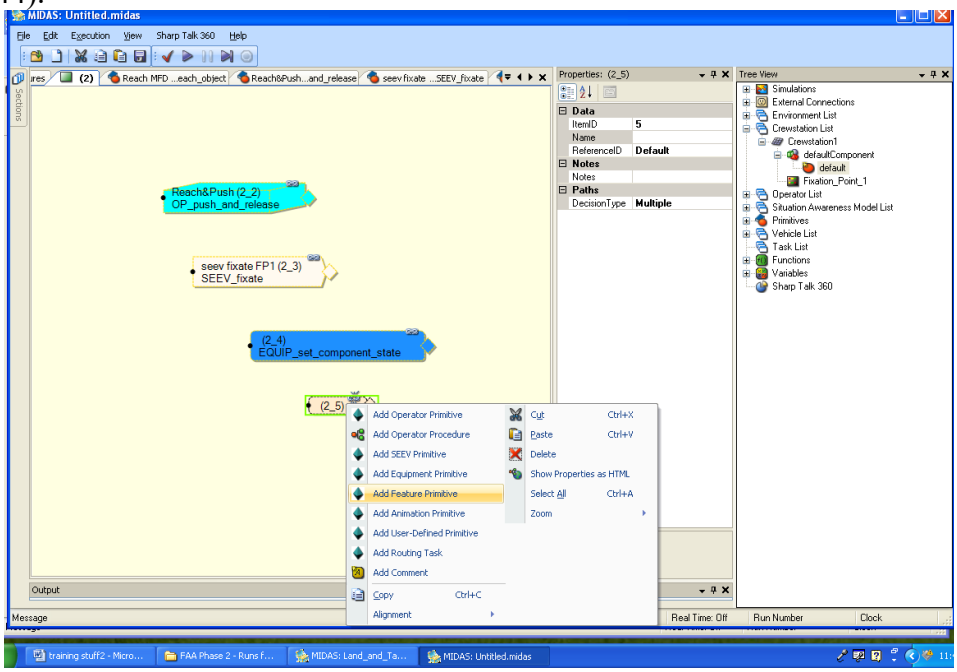


Figure 44. Accessing the features primitive from the operator properties window.

Once a features primitive has been placed in the task network model, criteria can be defined. In the current case, the feature is the feature that has just been placed into the environment and the attribute is the ‘previously-defined’ attribute (visual attribute 2). A

value is placed into the text slot, and a salience value is assigned (reminder, salience drives the operator attention) (see Figure 45).

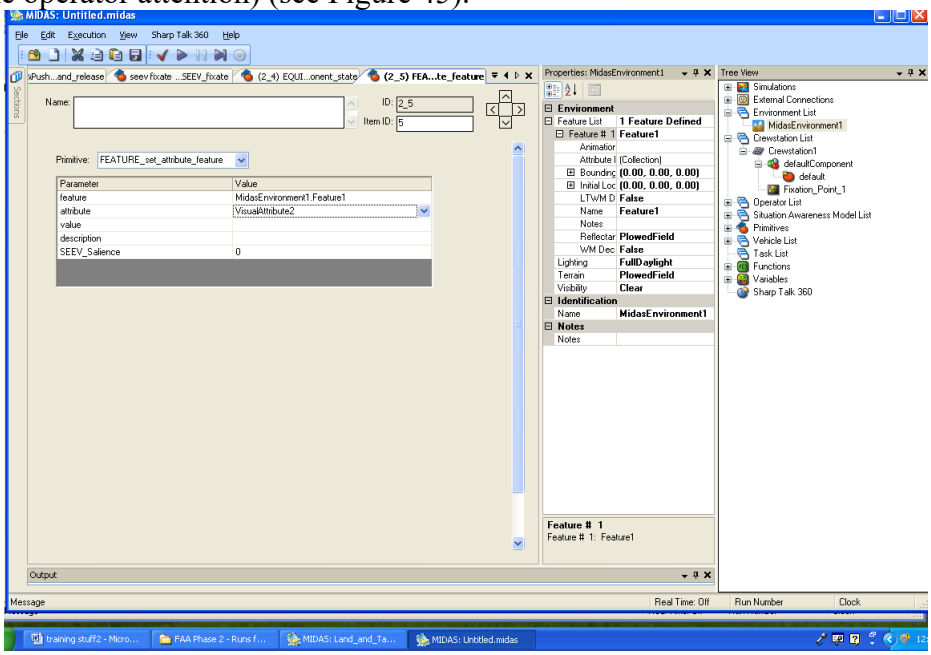


Figure 45. Parameters to define when using Features in the properties window.

Animation Primitive

The animation primitive is used in concert with the CAD model to define times when the model is available/viewable in the environment (e.g. a display that becomes active at a specific altitude). The parameters of Figure name and whether the figure is shown or hidden is defined in the Animation Primitive definition page (Figure 46).

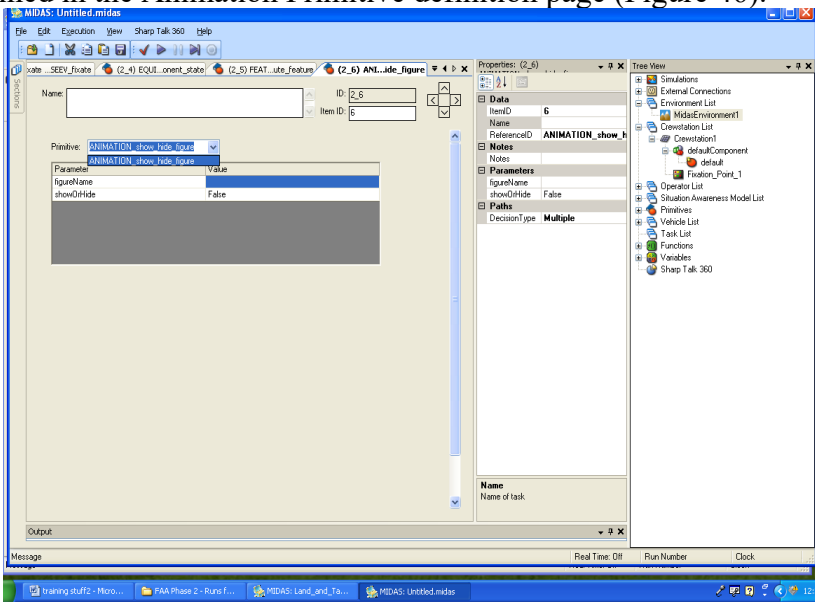


Figure 46. Parameters to define when using an animation primitive.

User Defined Primitive

An user defined primitive is used when no primitive model exists that can be used to represent the activity desired by the model analyst. For instance, “walk” is one such

primitive that is not contained within the base MIDAS models. A walk primitive would need to be populated with empirically determined data in terms of the amount of workload this requires, and amount of time per step (or at the desired level of fidelity), the SA (see Figure 47).

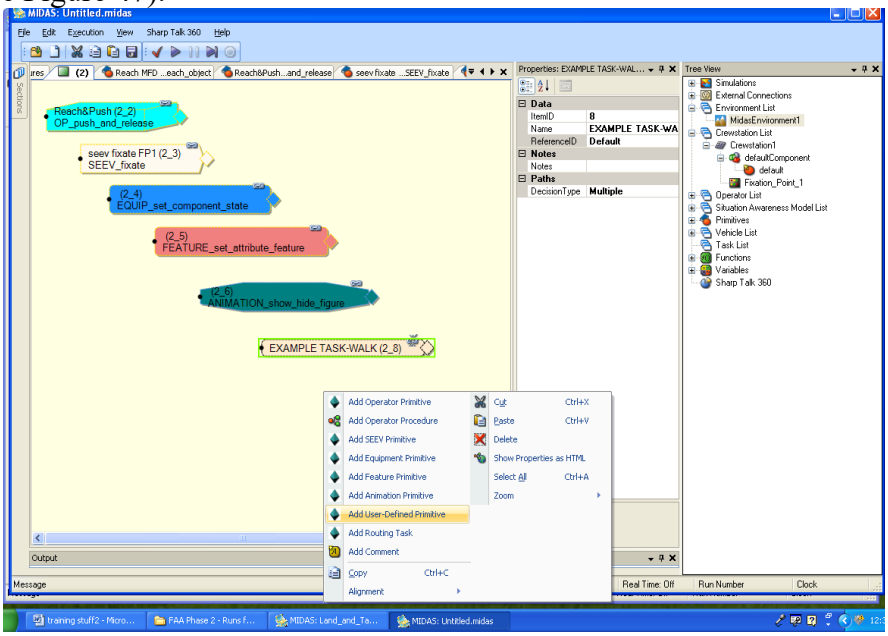


Figure 47. Setting an user-defined primitive.

Routing Task

A commonly used task in MIDAS is the routing task. This task is used to define the path that the model should logically follow when the model nodes are linked together. As a default, the routing task will show the user the information as illustrated in Figure 48. This figure illustrates that the user will be able to enter release conditions, beginning effects, ending effects, and launch effects.

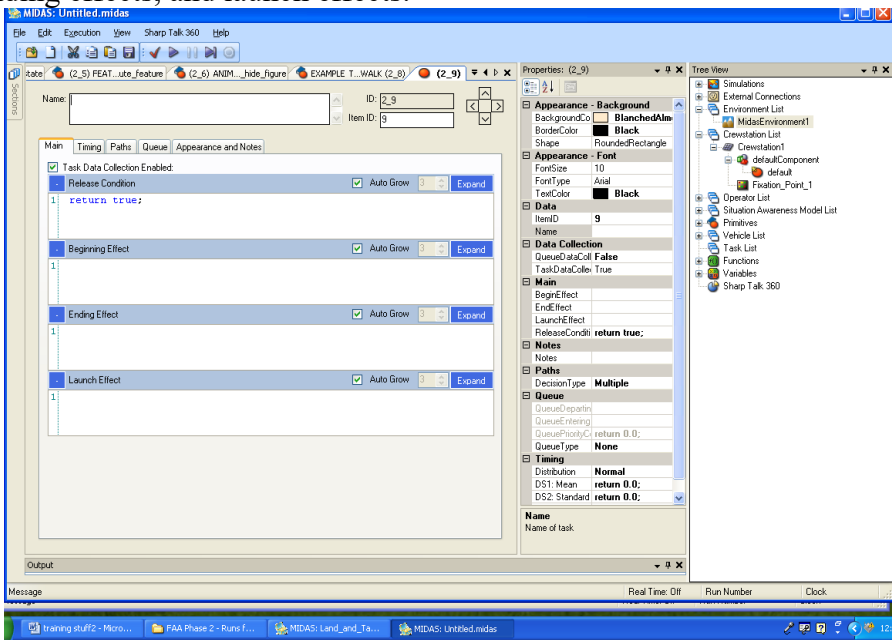


Figure 48. The routing task (main tab and properties).

The timing tab shows the place that the user enters task times (means and SD) (Figure 49).

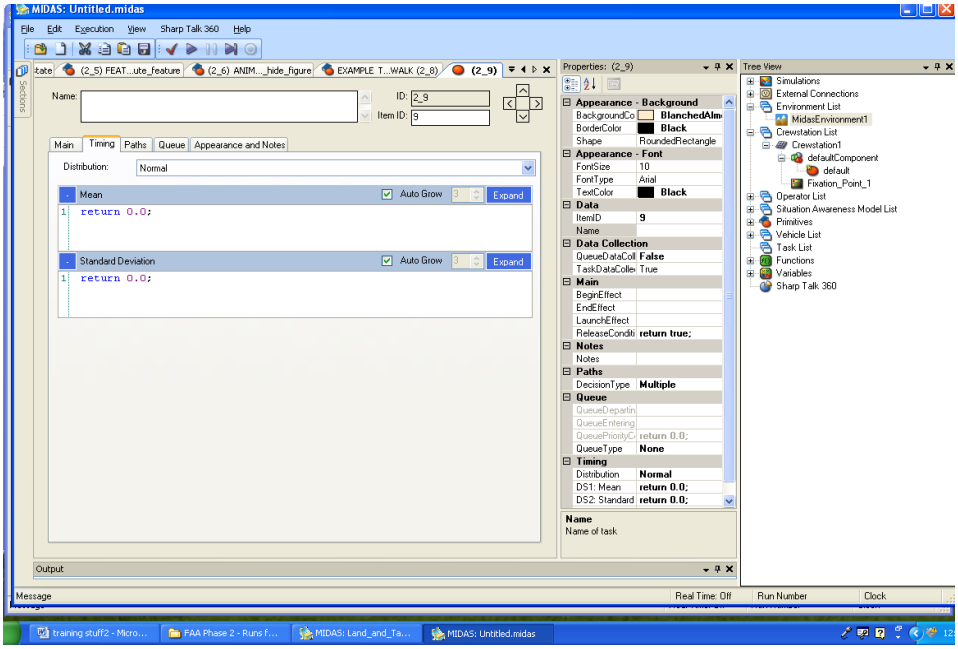


Figure 49. The routing task (timing tab and properties).

A routing task is dropped into the Operator Procedure window and a set of tasks is created from that routing task. The user creates a path between the routing task and the first operator task by pressing the left mouse button on the routing task and dragging the cursor towards the operator task. A path will be generated. Repeat this process for the second routing task. When the second path is created (and for every subsequent path), a decision task node (diamond shape) will appear. The default of this decision node is to multiple (meaning all routing tasks will fire), but tactical and probabilistic are options as well. The paths are defined according to the number of paths that go out of this routing primitive (see Figure 50). If four paths exit this routing primitive and each has an equal probability of success, then the probability will be 0.25 for each path (which represents 25% or ¼; see Figure 51). When the MIDAS simulation operates in Monte-Carlo simulation mode, this routing task logic and its percentages becomes critically important.

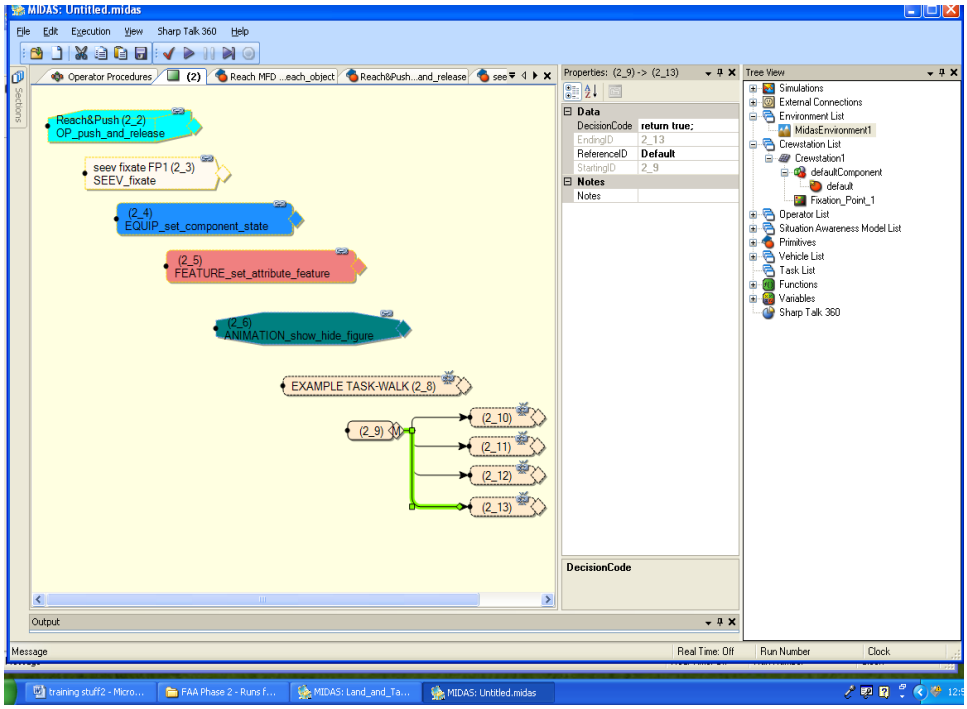


Figure 50. Steps to define the routing paths in a scenario.

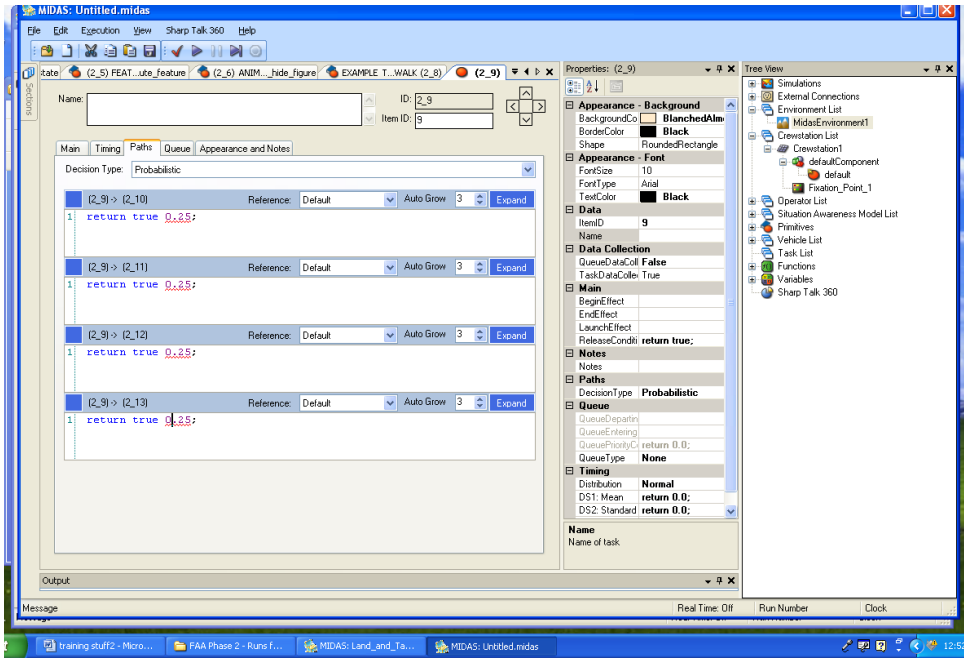


Figure 51. Path definitions in the path tab of the routing task.

Task dependencies can also be entered here in the queue tab. First a queue type is selected, whether it is FIFO, LIFO, none, or sorted. The queue is defined as a variable in the model. Logic of the queue defined how long the queue strong needs to be before the task fires. In addition, the queue priority, and queue duration are defined here (see Figure 52).

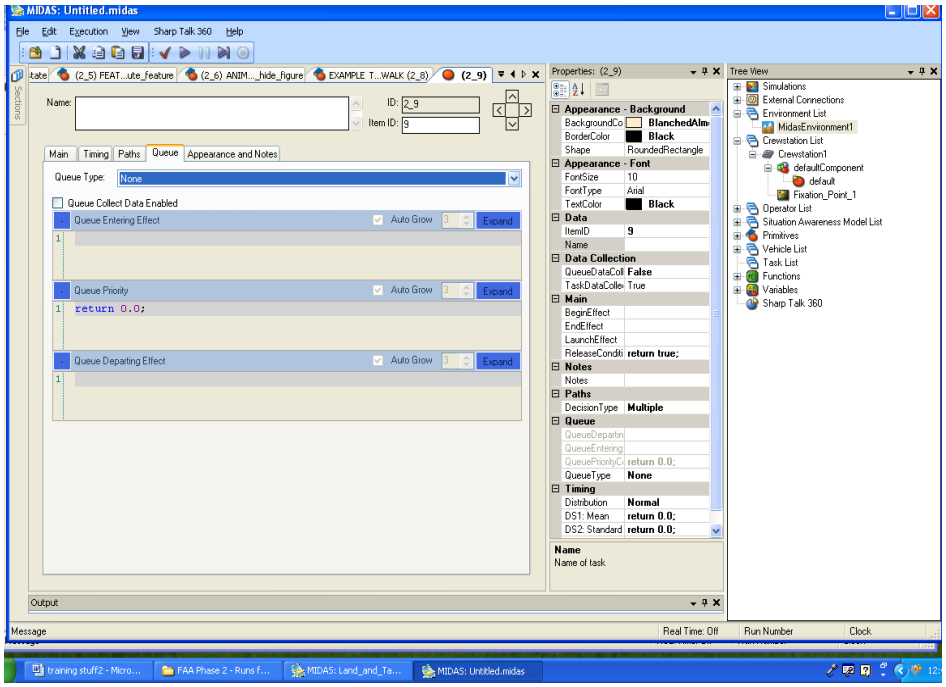


Figure 52. Queue definitions tab of the routing task.

Sidebar: To search for a model definition state, an object, or any piece of the model within a complex model (that possesses multiple layers), bring up the search functionality by CTRL F. Typing Control F will bring up a search window over the code output window as illustrated in Figure 53.

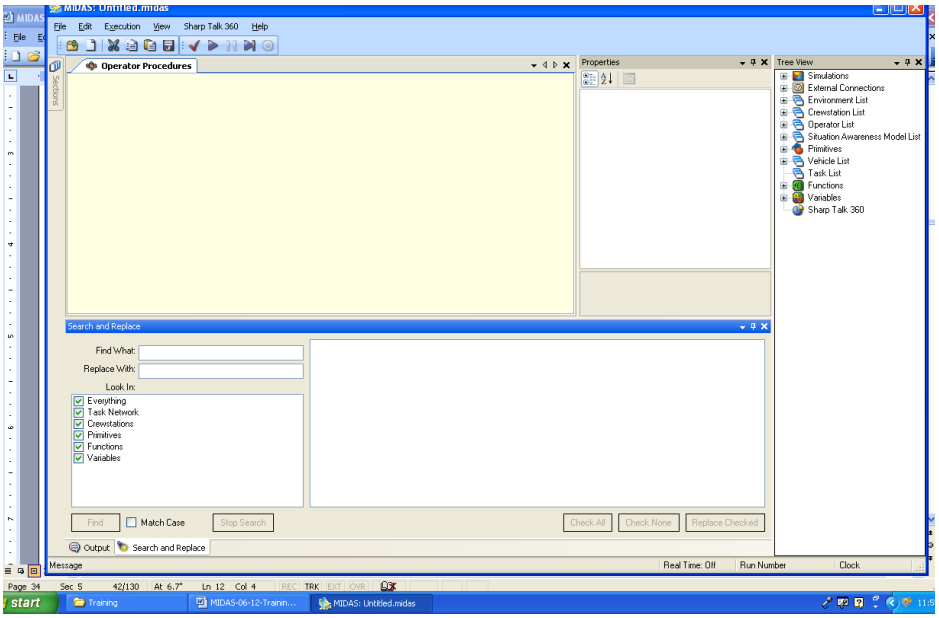


Figure 53. The MIDAS search feature.

Chapter 3: To Run a MIDAS Simulation

1. Open Midas and select your desired .midas file
2. Open MicroSaint Sharp and select your desired .saint file
3. Press the play button (Control+G) in Midas. This will open Sharp Talk 360 (see Figure 54).

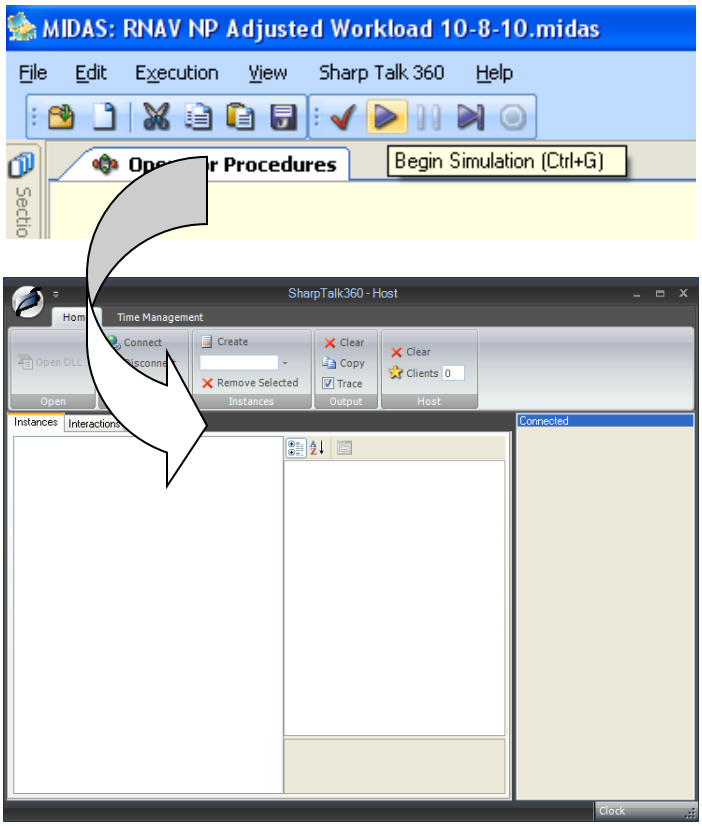


Figure 54. First step to run a MIDAS model: screen illustration of starting a MIDAS simulation and Sharp Talk being launched.

4. In the Midas toolbar, select Sharp Talk 360 drop down menu > Connect (see Figure 55).

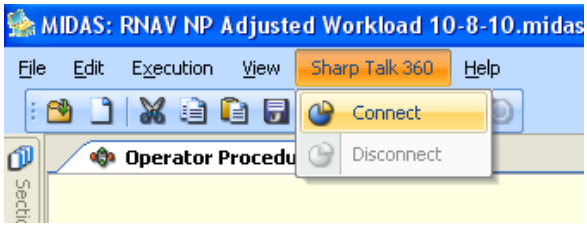


Figure 55. Second step to run a MIDAS model: Screen illustration of the steps required to activate Sharp Talk.

5. In the MicroSaint Sharp toolbar, select Sharp Talk 360 drop down menu > Connect

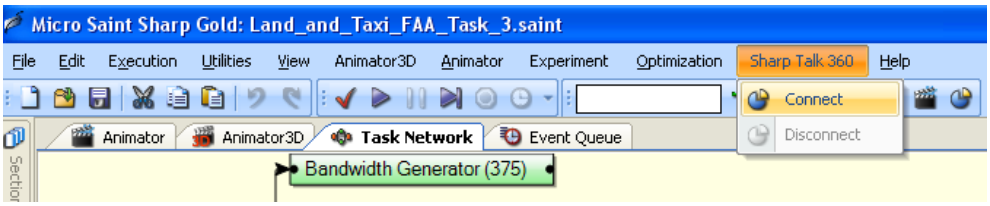


Figure 56. Third Step to run a MIDAS model: connect MIDAS to Sharp Talk.

6. You should now see that both Midas and MicroSaint Sharp are connected in the output box of Sharp Talk 360 – Host (see Figure 57).

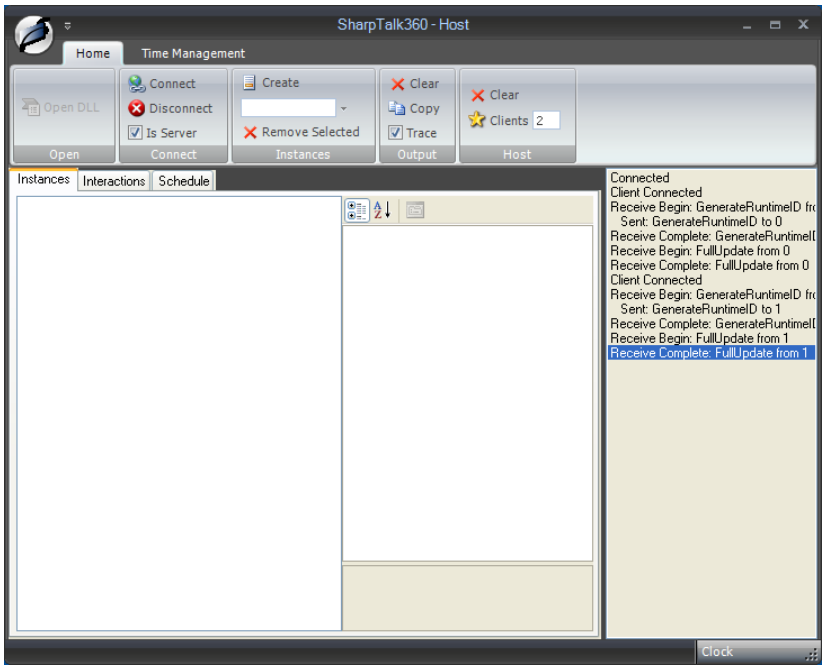


Figure 57. Fourth step to run a MIDAS model: Sharp Talk display of successful connection between MIDAS behaviors and Sharp's environment.

7. Select the time management upper tab in Sharp Talk 360 – Host and Press the blue play button in the control box on the top left (see Figure 58).

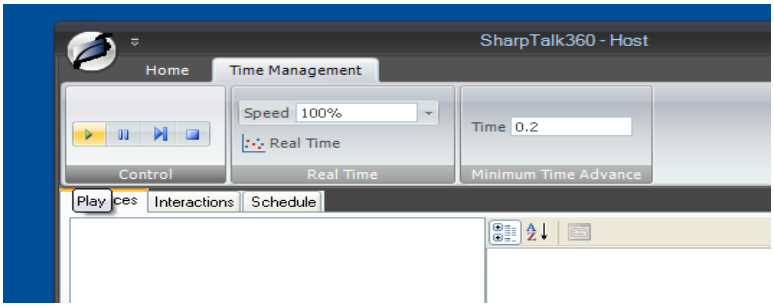


Figure 58. Fifth step to run a MIDAS model: starting the MIDAS model (from the Sharp Talk window).

8. The simulation has now been initiated. It usually takes about 30 seconds before you can see the runtime processes in the MicroSaint Sharp Window. Figure 59 illustrates the runtime processes that commence.

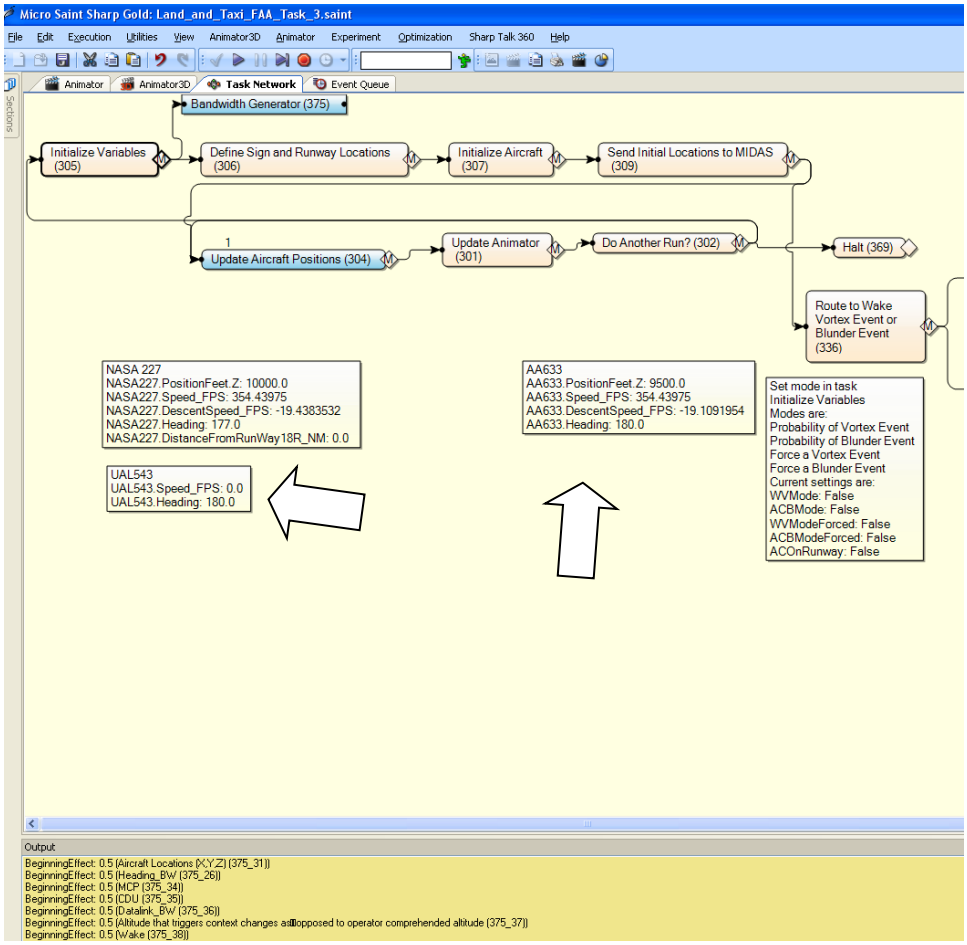


Figure 59. Sixth step to run a MIDAS model: Sharp Talk activated.

SharpTalk 360 – Host is used to control the simulation. You can pause it, stop it, step through it, speed it up, slow it down, etc. If the trace box is checked in the output box of SharpTalk 360, you will see a dynamic list of tasks on the right side (above the clock). Depending on your model, typical time from start to finish is about 15 minutes. Next we will see what the data look like.

Side bar: don't leave the folder open where your Midas file is during simulation because it can cause the simulation to crash.

Side bar: the simulation will run about 50% faster if you minimize the MIDAS and SAINT windows because it doesn't have to load the gui's.

To Run a Monte Carlo MIDAS Simulation

1. Open MIDAS and select Initial routing task (e.g. Set Number of Runs & Other Settings (75), see Figure 60)

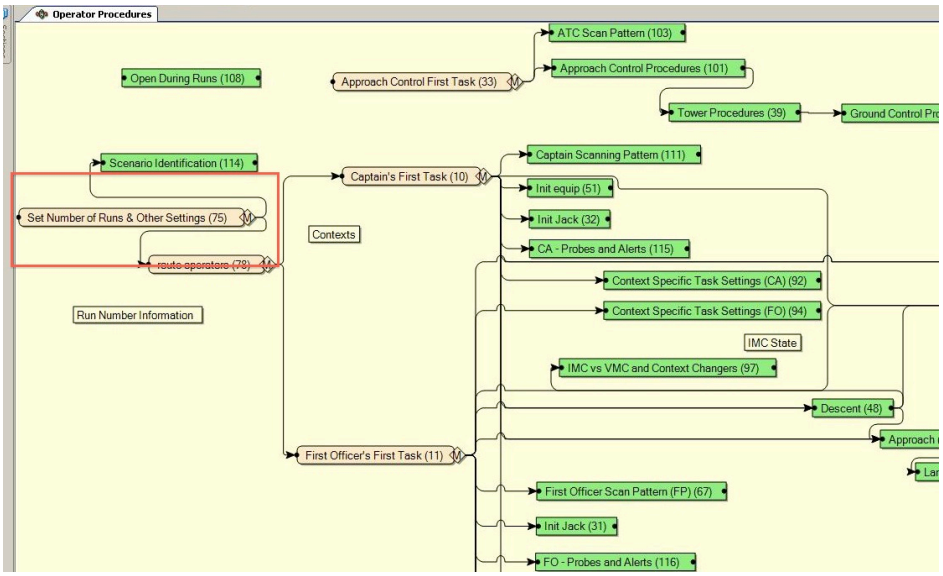


Figure 60. MIDAS initialization settings in the first routing task.

2. Adjust NumberOfRuns by retyping your desired number of runs and change comment to usually 1. A typical number for monte carlo runs is 10 (see Figure 61).
Note: do not adjust this anywhere else in MIDAS.

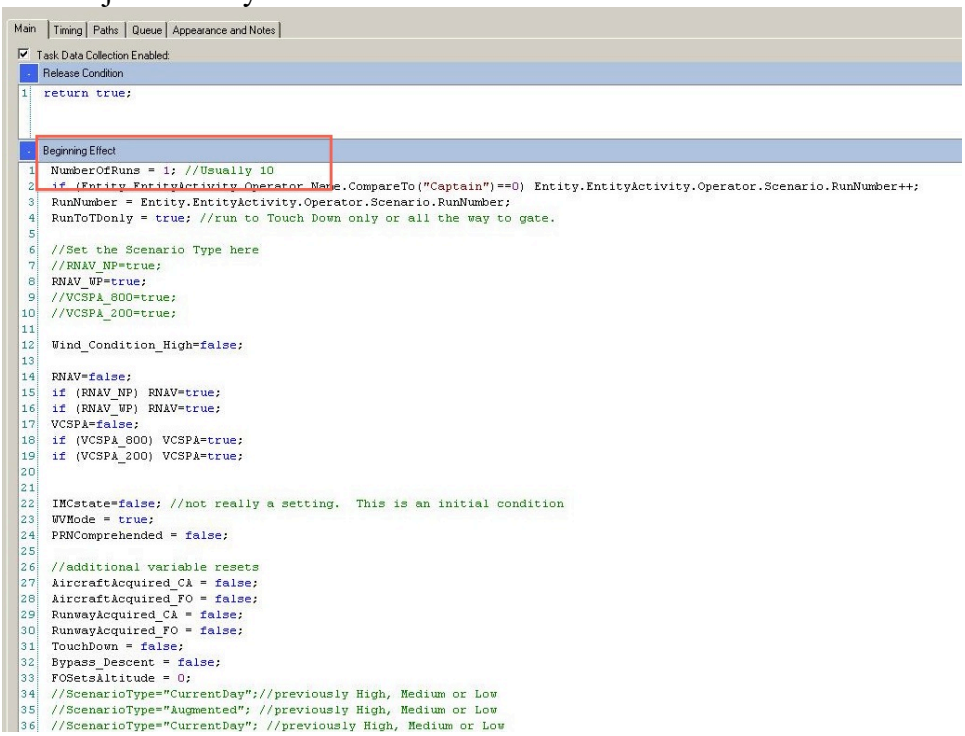


Figure 61. Settings for entering Monte Carlo simulations runs.

Manipulating MIDAS Scenarios

1. Open MIDAS and select Module: Set Number of Runs & Other Settings (75) (see Figure 62).

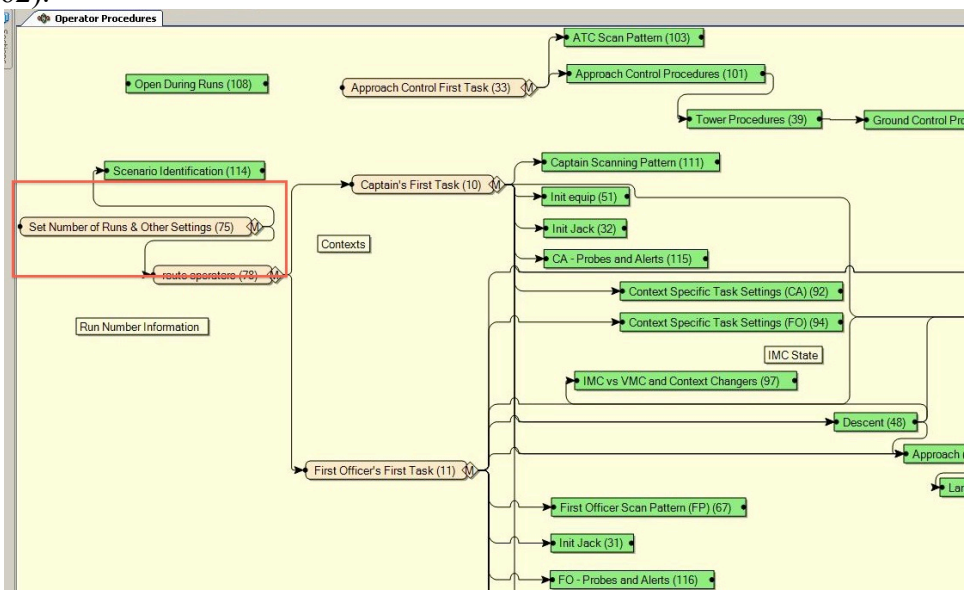


Figure 62. Manipulating an existing MIDAS scenario.

1. Adjust code in beginning effect box (see Figure 63). To turn something off, type two "/" marks consecutively to comment it out. To turn something on, delete the two "/" marks

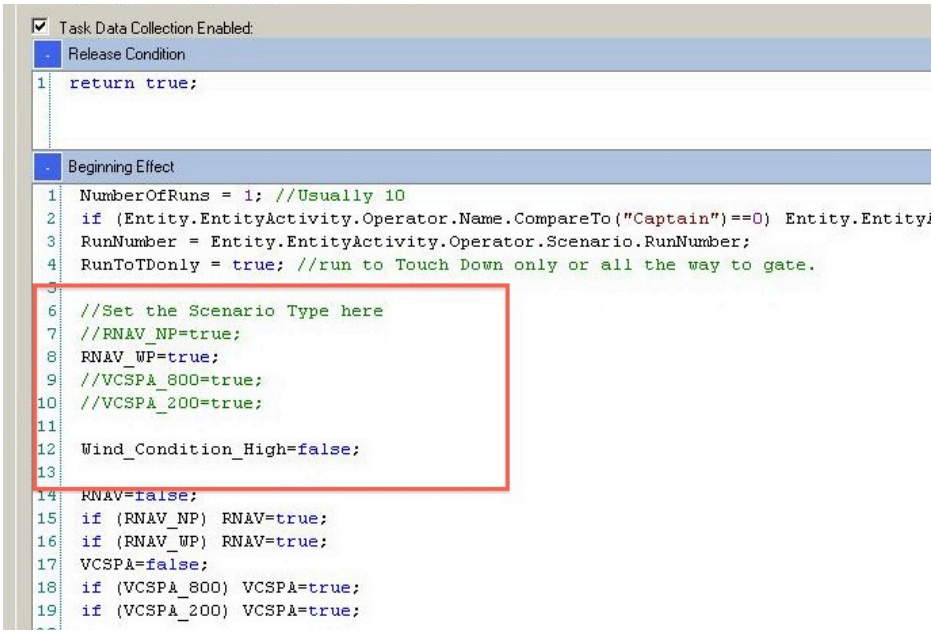


Figure 63. Example of turning off simulation settings in C Sharp in MIDAS.

Manipulating SAINT Scenarios

1. open microsaint sharp and double click on Initialized variables (subnetwork 305 in Figure 64) .

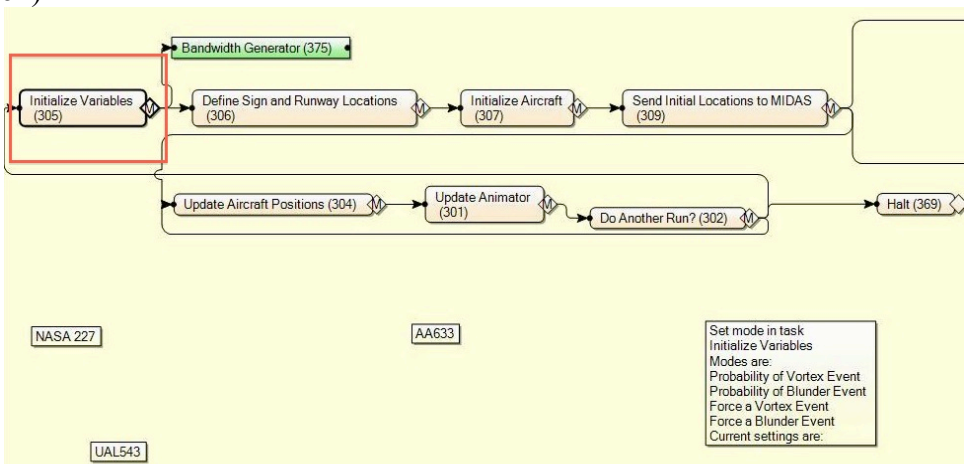


Figure 64. Example of Saint Scenario definition and modification.

2. Adjust code in the ending effect box. Comment it out to turn it off (remove the two "/"; Figure 65)

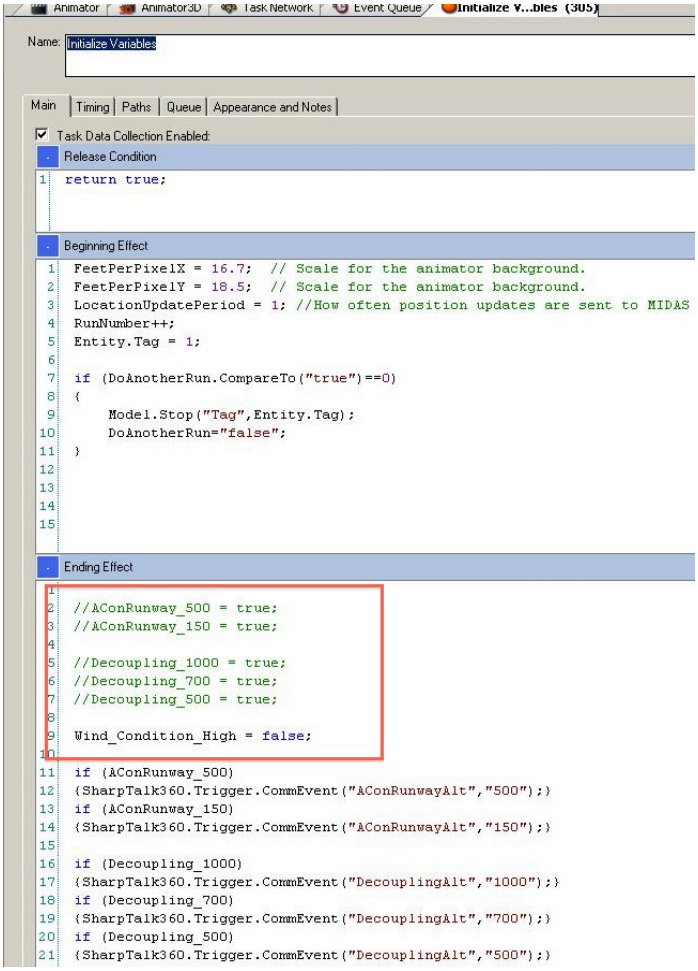


Figure 65. Illustration of commenting out code in Sharp.

Chapter 4: Models Contained Within MIDAS v5

MIDAS Attention
MIDAS represents attention as a series of basic human primitive behaviors that carry with them an associated workload level determined from empirical research [9,10,11]. Actions are triggered by information that flows from the environment, through a perception model, to a selection-architecture (that includes a representation of human attention loads), to a task network representation of the procedures that then feeds back into the environment. Actions carried out by the MIDAS operator impact the performance of the model in a closed-loop fashion. The Salience, Effort, Expectancy, and Value (SEEV) of information in an environment (Gore, Hooey, Wickens, & Scott-Nash, 2009) drive the allocation of attention that allows for dynamic scanning behaviors in MIDAS. SEEV estimates the probability of attending, P(A), to an area of interest in visual space, as a linear weighted combination of the four components - salience, effort, expectancy, and value. Attention in dynamic environments is driven by the bottom-up capture of Salient (S) events (e.g., a flashing warning on the instrument panel) and inhibited by the Effort (E) required to move attention (e.g., a pilot will be less likely to scan an instrument located at an overhead panel, head down, or to the side where head rotation is required, than to an instrument located directly ahead on a head-up display (HUD) (Wickens, Goh, Helleberg, Horrey, & Talleur, 2003).

MIDAS Perception
MIDAS represents perception as a series of stages that information must pass through in order to be processed. The perception model includes visual and auditory information. Visual perception in MIDAS depends on two factors – the amount of time the observer dwells on an object and the perceptibility of the observed object (Gore, Hooey, Wickens, & Scott-Nash, 2009).

The perception model computes the perceptibility of each object that falls into the operator's field of view based on properties of the observed object, the visual angle of the object and environmental factors. In the current implementation of MIDAS, perception is a three-stage, time-based perception model (undetected, detected, comprehended) for objects inside the workstation (e.g., an aircraft cockpit) and a four-stage, time-based perception model (undetected, detected, recognized, identified) for objects outside the workstation (e.g., taxiway signs on an airport surface). The model computes the upper level of detection (i.e., undetectable, detectable, recognizable, identifiable for external objects) that can be achieved by the average unaided eye if the observer dwells on it for a requisite amount of time. For example, in a low-visibility environment, the presence of an aircraft on the airport surface may be 'detectable' but the aircraft company logo on the tail might not be 'recognizable' or 'identifiable' even if he/she dwells on it for a long time.

MIDAS Memory
Tasks from the MIDAS input process also require knowledge held either in the operator's memory (working, long-term working, and long-term) or available from the environment to be consulted and used to determine subsequent tasks to be completed (Gore, Hooey,

Wickens, & Scott-Nash, 2009). Memory in MIDAS is represented as a three stage, time decay model (Gore, 2011). The stages are working memory (WM), long-term working memory (LT-WM), and long-term memory (LTM). The decay rates cause memory to be above or below a "retrievability" threshold based on the time since the information was last accessed. The retrievability thresholds incorporated into MIDAS are 5 seconds for WM and 5 minutes (300 seconds) for LT-WM. The WM decay rate is faster than the LT-WM decay rate. Information that falls below the retrievability threshold is forgotten. This causes the perception level to be set to Undetected for external visual and auditory information or Unread for internal visual information. Newly perceived and recently refreshed attributes will be retained in LT-WM only if a node with newly updated or referenced attributes leaves WM before its attributes have decayed below the retrievability threshold. An operator may retain newly perceived information after it leaves WM, at least for a while, until it decays below the LT-WM retrievability threshold. If the information necessary for activity performance is available, and its priority is sufficient to warrant performance, then the schedule within the model operates according to heuristics that can be selected by the analyst. In most cases the heuristic is to perform activities concurrently when that is possible, based on knowledge and resource constraints.

Implementation of SEEV in MIDAS - Overview

The integration of SEEV into MIDAS allows dynamic scanning behaviors by calculating the probability that the operator's eye will move to a particular area of interest (AOI) given the tasks the operator is engaged in given a multitask context.

Areas of Interest

In the parlance of MIDAS, AOIs are fixation points. In the land and taxi model, examples of AOIs are Internal Fixation Point Near PFD or External Fixation Point at Front Left Window.

An AOI could be an individual display such as an altimeter or as in the land and taxi scenario the fixation points could be at the screen display level, as in the Primary Flight Display or Navigation Display. It depends on the question the analyst is trying to answer. The analyst should set the fixation points appropriately to the question and level of detail required by the analysis.

SEEV Components

These are Salience, Effort, Expectancy and Value. These are explained in more detail below.

Salience

Think of the salience of events as properties that can be measured physically.

- The loudness of an utterance but not the content of the utterance
- The frequency of flashing
- The color of an indicator

An example of salience could be a proximity indicator on the navigation display that flashes when another aircraft comes too close.

An important guideline is to always think about salience in the context of a salient **event.** In MIDAS, delay between a salient event as defined by the analyst and the first fixate on the display that exhibited the attribute change related to that event is reported. One important implementation detail in MIDAS is that because MIDAS has a perception model unlike previous SEEV implementations, the delay is specifically timed to the point at which perception exceeds 'Undetected'. In other words, the end of the fixate primitive as opposed to the beginning is the trigger for considering a salient event mitigated. Simple heuristics aid the analyst in setting the salience level. e.g. 1 = change with no luminance increase, 2 = change with luminance increase, 2 = change in position, 3 = change in position and luminance increase, 4 = repeated onsets (flashing) (see Figure 66).

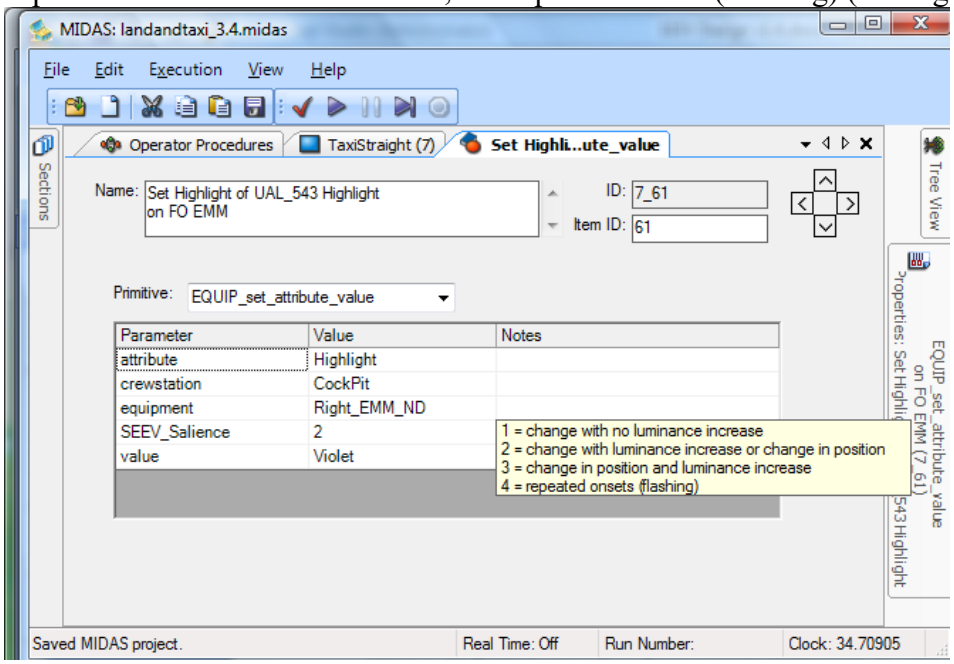


Figure 66. Salience heuristics to aid the analyst are displayed as a tool tip from the notes field.

Effort

Effort impacts the likelihood of traveling from one AOI to another. Effort is the only inhibitory factor in the SEEV equation.

In MIDAS an Effort rating between 0 and 1 is calculated for each AOI relative to the currently fixated AOI and is based on the angular difference. Any AOI that is 90 degrees or more from the current AOI is set to the maximum. Any AOI that is less than 90 degrees is divided by 90 degrees.

Before the SEEV calculation is run, effort is normalized between 0 and 1 by subtracting it from 1. So if the captain is currently fixating the PFD, then the angle is 0 and effort is 1. The captain will not be as likely to look away from the PFD. Likewise, the angle from the PFD to the left window is around 75 degrees and the effort is around .2. The captain will be less likely to turn from the PFD to the window.

Expectancy (Bandwidth)

Expectancy is described as the event frequency along a channel (location) or as bandwidth. An example from the Wickens' literature is the frequent oscillation of attitude of a light plane when encountering turbulence. The pilot expects the horizon line on the attitude indicator to be jumping about wildly. Because of this, he'll be watching it closely.

In comparison to the attitude indicator is the altimeter during a controlled descent. The pilot expects it to be dropping down at a constant rate and therefore has a low expectation of seeing changes in descent rate. At least until they get close to the ground at which point Value escalates. One guideline we can take away from this example is that when the rate of change is constant, then the bandwidth or frequency of oscillation is zero. In SEEV applications, bandwidth (or event rate) is always used as a proxy for expectancy.

In MIDAS, Expectancy is implemented as a SEEV primitive (see Figure 67). Expectancy can be set for each operator in the model at any time. However, each successive expectancy setting overrides the previous one for the given operator.

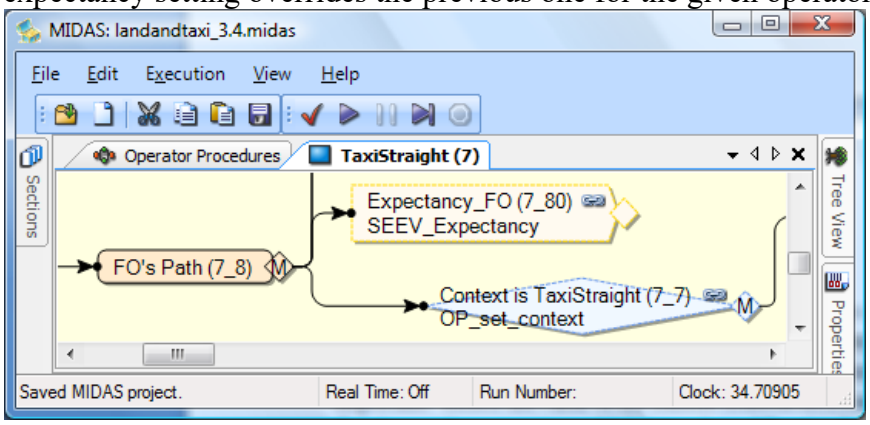


Figure 67 - Expectancy as a SEEV primitive.

The user sets expectancy for each AOI to a number between 0 and 6. Before the SEEV calculation is run, expectancy is normalized between 0 and 1 by dividing by 6. (See Figure 68 for an example on entering this information into the SEEV spreadsheet in MIDAS).

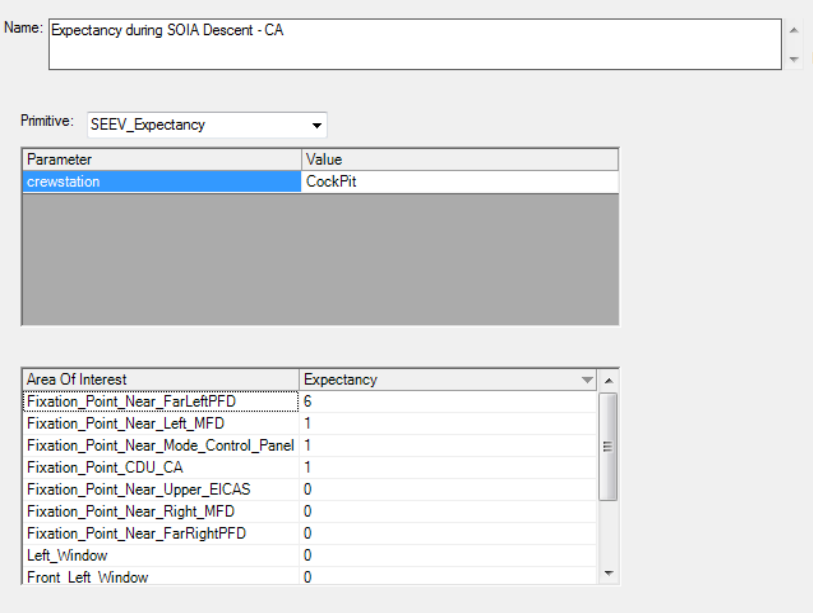


Figure 68 - An example of setting Expectancy for First Officer.

Value (Importance)

The level of Value denotes the importance of attending to an event or task or the cost of missing it. An example from the Wickens' literature is that in the context of driving, lane keeping is arguably more important than dealing with an in-vehicle navigation system. Selecting a radio station is the least important. Hence the value of the outside fixation necessary for lane keeping is > value of a head down fixation on the navigation system which is > value of fixating the radio (e.g., values 3,2,1). In land and taxi, avoiding an in air collision on approach is more valuable than maintaining heading and speed.
In Wickens' approach, the sum of the product of the task value and the relevance of each AOI to the task is used to compute the value (importance) of the AOI (see Table 1).

Table 1. Table of Importance Values for Populating the SEEV Parameters.

		Importance of AOI to task			
Task	**Task Value**	front window	left window	near PFD	near ND
Avoid collision	.8	.6	.4	0	0
maintain speed / heading	.2	.1	.1	.4	.4
Value of AOI		=(.8*.6)+(.2*.1) .5	=(.8*.4)+(.2*1) .34	=(.8*0)+(.2*.4) .08	=(.8*0)+(.2*.4) .08

In MIDAS, Value is implemented using SEEV primitives in order to bracket sets of primitives. In the example illustrated in Figure 69, two task sets are running in parallel and happen to begin and end at the same time although beginning and ending at the same time is not a requirement. The SEEV calculation considers all tasks sets that are active until they are explicitly ended by a SEEV end task set primitive.

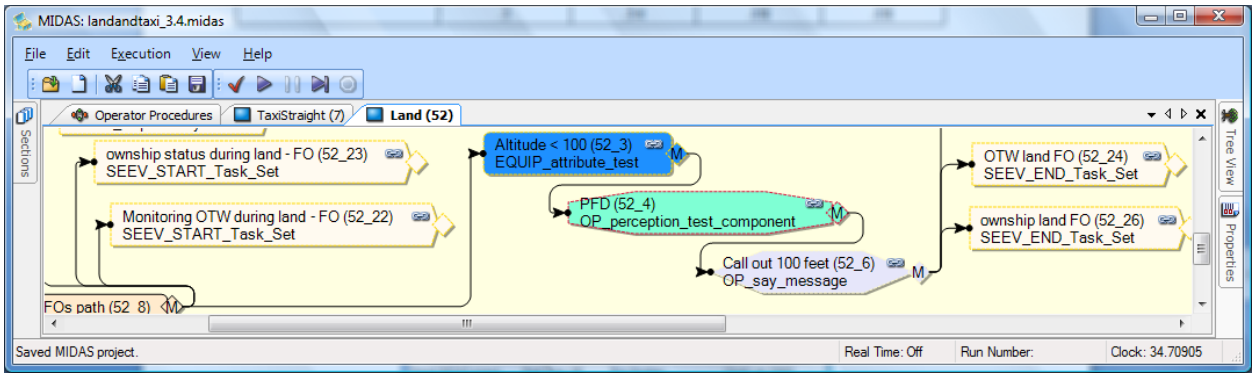


Figure 69 - SEEV Task set encapsulating a set of MIDAS primitives

For each set, the user sets an overall importance. In the example in Figure 70, monitoring out the window is of high importance to the primitives bracketed by the Monitoring OTW during land – FO task set.

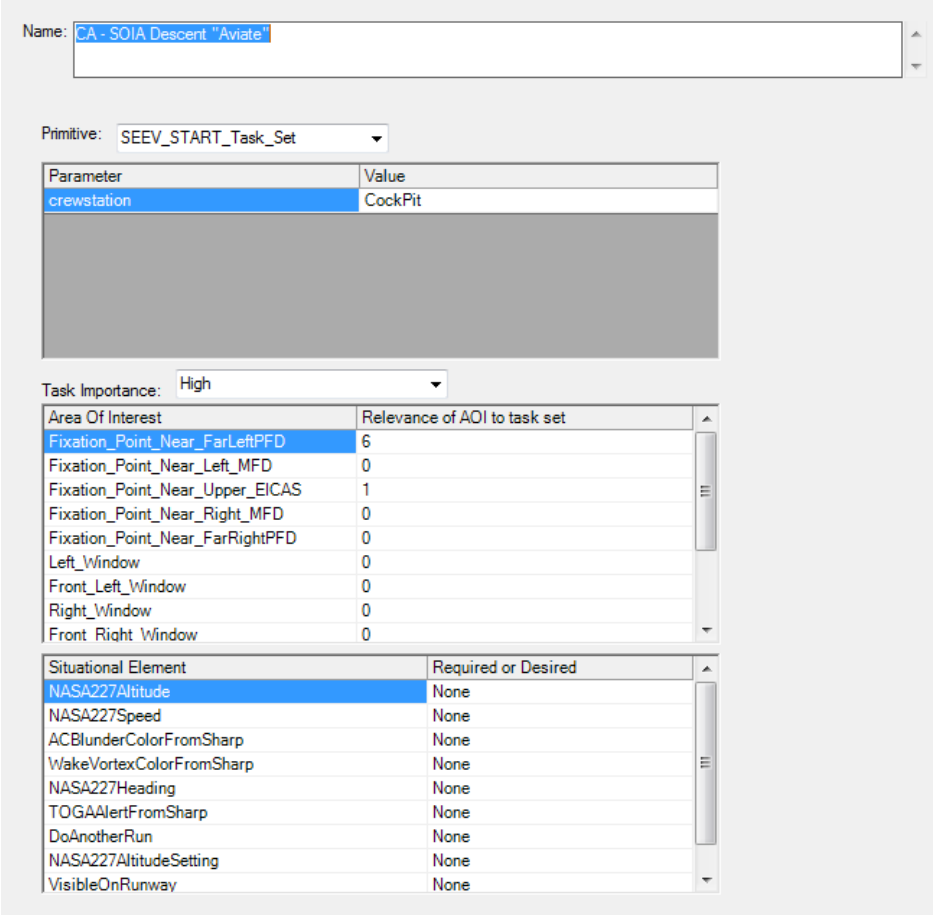


Figure 70 - Example of starting a SEEV task set

The user can indicate a relevance using a number between 0 and 6 for each AOI in the task set. In addition, the user can specify none (0), low(1), moderate(2) and high(3) importance rating for the entire task set. Before the SEEV calculation is run, the task set

importance is normalized between 0 and 1 by dividing by 3 and dividing by 6 normalizes the relevance.

Equation and Coefficients

The SEEV equation can be found in Equation #1:

Equation 1. SEEV equation.

$$P(AOI) = s*S - ef*EF + (ex*EX + v*V)$$

The EF, EX and V parameters drive the eyeball around the displays. However, if a salient event happens, then P(A) is may be offset by the AOI exhibiting the salient event until the time that that AOI has been fixated.

Wickens suggests that these weightings are TBD but that they should be part of the MIDAS immutable parameters that are reported but not changeable by the analyst. They should be the same for all MIDAS type models.

In the MIDAS implementation, Expectancy, Effort and Value are all values between 0 and 1 as explained above. Salience is unconstrained. The implementation of the SEEV equation in MIDAS can be found in Equation #2.

Equation 2. SEEV Implementation in MIDAS

$$P(AOI) = S + .25*EF + EX + V$$

Equation #2 is calculated for each AOI across all tasks the captain is performing. As an illustration consider the following likelihoods. The sum total of all likelihoods is calculated which is 2.1 (see Table 2).

Table 2. Likelihood of Sampling the AOI in a Scenario.

AOI	Likelihood	Sum
PFD	.8	.8
ND	.6	1.4
EICAS	.3	1.7
Left Window	.2	1.9
Front Left Window	.2	2.1

Next a random number between 0 and 1 is selected and multiplied by the total (2.1) to produce a threshold (see Table 3). Again as illustration for multiple sequential fixations this might turn out to be the following.

Table 3. Random Number Assignment of Visual Fixation Likelihood.

5 Sequential Fixates	2.1 * a number between 0 and 1 (threshold)
1	.3
2	1.2
3	1.8
4	.5
5	.9

Finally, the AOIs continue to cycle until the sum of the likelihoods surpasses the threshold (see Table 4).

Table 4. Example of the Random Number Comparison to the Sum of Likelihood to the Threshold.

5 Sequential Fixates	AOI fixated
1	PFD
2	ND
3	EICAS
4	PFD
5	ND

This sequence illustrates that AOIs with a higher likelihood are fixated more often because a random number is more likely to land in their larger portion of the sum of the SEEV values.

A note on Effort

In order for this algorithm to work, the SEEV equation must result in a number 0 or greater for each AOI. The only way that can happen is if effort is subtracted from the equation and if effort is greater than salience, expectancy and value. Therefore, effort must be a value between 0 and 1 as well. To do that, instead of subtracting effort, we subtract effort from 1 and add it to the equation. Hence AOIs that are near the current AOI have a greater SEEV value than AOIs farther from the current AOI if salience, expectancy and value are equal.

Integration with Current Models

Scanning

Currently in MIDAS, a scan procedure can be built from fixate primitives through implementing probability or decision logic to drives the fixation pattern/sequence. SEEV is selected at the simulation level (from the simulation node). If SEEV is enabled then user-entered fixate primitives will be skipped. Instead, the SEEV Fixate primitive for each operator will be used. The SEEV model is called at the end of each SEEV Fixate. The timing and task load of the SEEV fixate is exactly the same as for the non SEEV

Fixate. In this way, the user doesn't have to build two different models to if they want to flip back and forth between attention guiding mechanisms (see Figure 71).

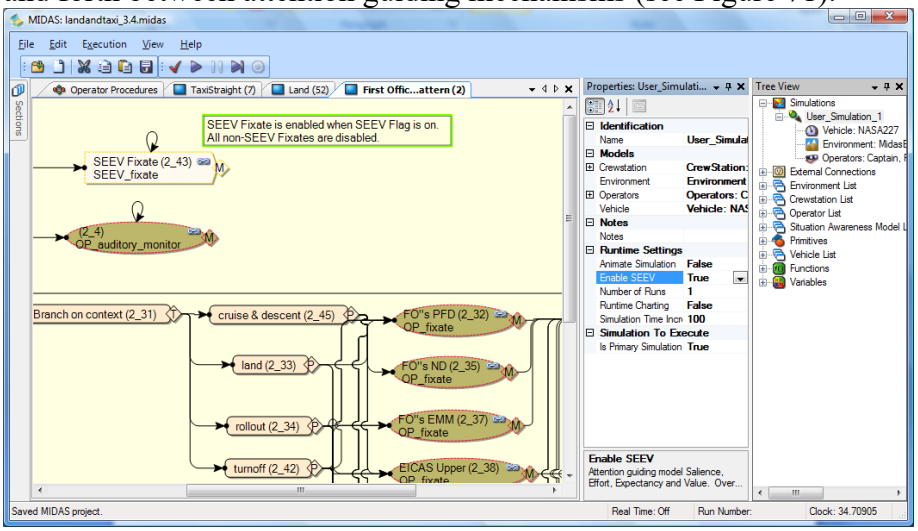


Figure 71. Illustration of the Probabilistic and the SEEV Model in the Same Operator Procedure Model.

Situation Awareness (SA) Model

From the point of view of the code base, the SA model is independent of SEEV. However, the SEEV driven scanning pattern will directly impact SA since SA is dependent on perception and perception is dependent on scanning. One important note is that in this version the memory model is operational and does affect the SA model.

The MIDAS Situational Awareness Model

Situation Awareness and its Interaction among Cognitive Models in MIDAS

Situation Awareness (SA) is tied to operator perception, attention, working memory (WM), long term working memory (LTWM), and long term memory (LTM). The same is true for the MIDAS modeled operator. MIDAS WM decays over 10 seconds from comprehension of perceived information and LTWM decays over a total of 60 seconds from comprehension. It is important to note that the LTWM decay actually starts from the time of Comprehension, so it takes 50 seconds to decay from Detected to Undetected. The decay function of SA is based on the following five assumptions:

(1) Current thinking is that the loss of information in SA can be approximated by the decay function of LTWM (Ericsson & Kintch, 1995; Wickens, 2008; Durso Rawson & Gerrato, 2007). This has a rate of forgetting somewhere between the rapid decay of WM, and of the slow decay of LTM.

(2) The exponential decay rate, characteristic of all human memory systems can be closely approximated by two linear functions, as shown in Figure 75, such that the knee of the curve approximates the time constant of a linear decay.

(3) The particular choice of where the knee (discontinuity) of the function lies is based on SAGAT SA data (Gugerty, 1998). In this study, it was found that the accuracy of SAGAT query responses in a driving simulation, declined by approximately 25% when, on average, such queries were responded to 7 seconds following the screen blanking characteristic of SAGAT administration. We note, on the green line on the right side of the figure, that the two parameter (slope) function approximates this value.

(4) The slope of the LTWM decay function at approximately 20% accuracy/20 sec delay is replicated in a second unpublished study (data provided by Gugerty). Using a similar paradigm to that examined in Gugerty (1998), three SAGAT delays (7 sec, 17 sec and 27 sec) were imposed, with these showing respective accuracies of 80%, 70% and 60%, yielding a near identical slope to Figure 75.

(5) While these particular values establish the reliability of the LTWM decay slope estimate, they can serve to position the knee of the curve at a slightly higher accuracy (and shorter time). A combination of the two sources might place the knee at 7, rather than 10 seconds.

Figure 72 provides an example of the output from the *MIDAS_FixationChanges* report of a simple model that exercises the different decay rates.

MIDAS_FixationChanges_run1of1.4DC97063.csv [Read-Only]

	A	B	C	D	E	F	G	H
1	RunNumb	Time	Duration	Context	Operator	Int/Ext	Fixation Point	
2	0	0.6	0.432334	Context1	Operator1	interior	Fixation_Point_1	
3	0	2.9	0.279294	Context1	Operator1	interior	Fixation_Point_2	
4	0	3.3	0.176795	Context1	Operator1	interior	FP_EmptySpace	

Figure 72. MIDAS_FixationChanges Report for simple model

The *MIDAS_Situation_Element_Accessibility_and_Perception* report presented in Figure 73 shows that the Perception of the Attribute Component 1 reaches Comprehension (a

value of 1 reached on line 6). It then decays in a linear fashion such that after 10 seconds the perception is at a value of 0.5, still above the Detection threshold but no longer above the comprehension threshold. The Attribute further decays linearly for another 50 seconds (60 seconds total from Comprehension) to a value of 0, corresponding with Undetected. In the second case (Component 2), the SA of the Attribute reaches Detection before the Operator looks away. The SA then decays linearly from 0.5 (Detected) to 0 (Undetected) over the course of 10 seconds.

MIDAS_Situation_Element_Accessibility_and_Perception_Operator1_run1of1.4DC97063.csv [Read-Only]

	A	B	C	D	E	F	G	H	I
1	Operator: Operator1								
2	RunNumb	Time	Context	Situation	Component	Accessibi	State	Perception level	
3	0	0.3	Context1	SE_1	Component_1	High	default	UNDETECTED	
4	0	0.3	Context1	SE_2	Component_2	High	default	UNDETECTED	
5	0	0.5	Context1	SE_1	Component_1	High	default	DETECTED	
6	0	1.4	Context1	SE_1	Component_1	High	default	COMPREHENDED	
7	0	2.8	Context1	SE_2	Component_2	High	default	DETECTED	
8	0	12.7	Context1	SE_1	Component_1	High	default	DETECTED	
9	0	13.3	Context1	SE_2	Component_2	High	default	UNDETECTED	
10	0	62.7	Context1	SE_1	Component_1	High	default	UNDETECTED	

Figure 73. MIDAS_Situation_Element_Accessibility_and_Perception Report for a Simple Model.

The results of the previous report are shown in Figure 74. This graph demonstrates that without additional fixations, the perception decays as expected with the decay rates outlined.

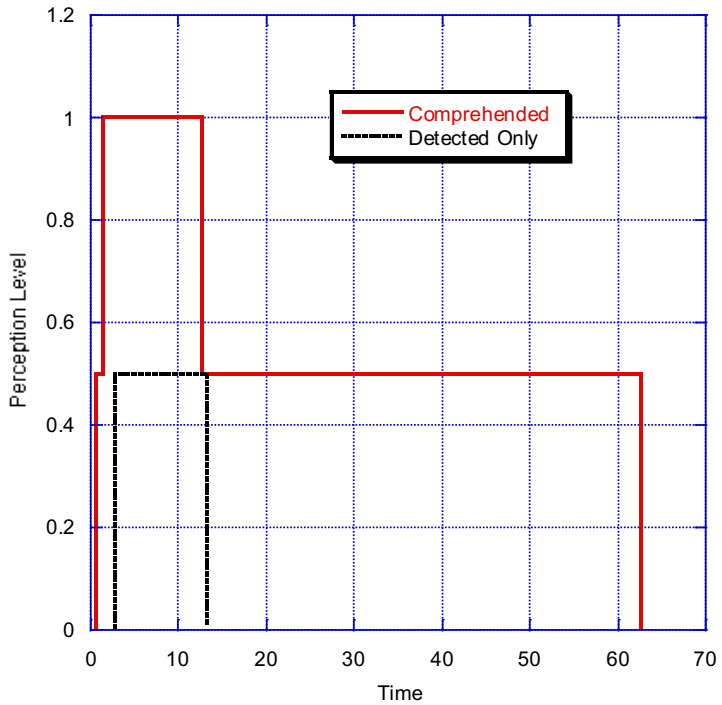


Figure 74. Decay rates for WM and LTWM.

The two-slope linear model of decay

SA decay in MIDAS v 5 is dynamic over the course of a simulation; **no longer** operating according to a step function (Figure 75), and is mapped to the decay in Perception.

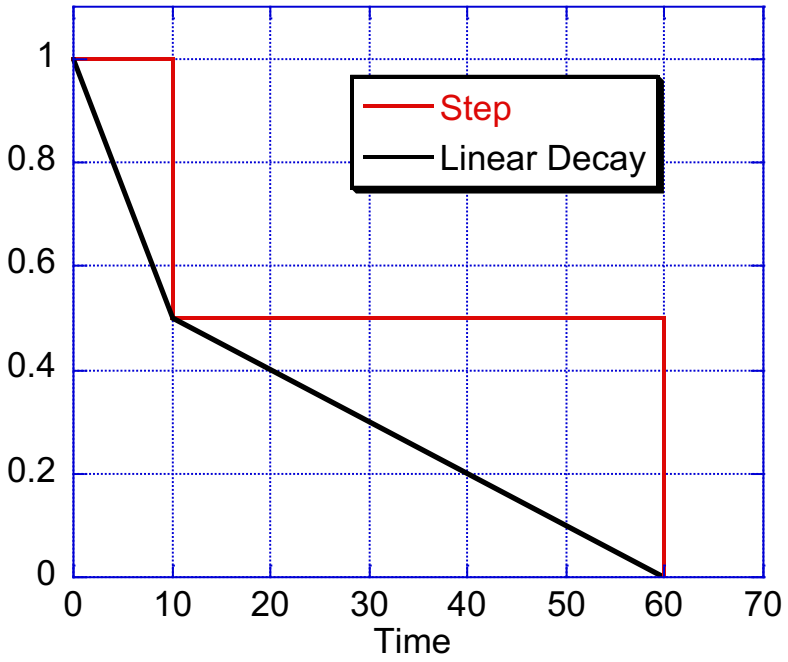


Figure 75. Linear Slope Decay mapped to Perception levels.

Figure 76 presents the data output from a model run with the linear SA decay curve.

MIDAS_Situational_Awareness_by_Task_Operator1_run1of1.4DC97063.csv [Read-Only]

	A	B	C	D	E	F	G	H
57	0	3	Context1	TS_12	2	1.96	0.98	2
58	0	3	Context1	TS_215	2	1	0.5	2
59	0	3.1	Context1	TS_12	2	1.95	0.975	2
60	0	3.1	Context1	TS_215	2	1	0.5	2
61	0	3.2	Context1	TS_12	2	1.94	0.97	2
62	0	3.2	Context1	TS_215	2	1	0.5	2
63	0	3.3	Context1	TS_12	2	1.93	0.965	2
64	0	3.3	Context1	TS_215	2	0.99	0.495	2

Figure 76. MIDAS_Situational_Awareness_by_Task_Operator

Figure 77 demonstrate both linear decays of component 1 and component 2 from the simple model presented in Figure 73 and Figure 74, with the blue line representing a Display that has reached Comprehension and then decays using WM and LTWM decay rates to Undetected, and the green line representing a Display that has reached Detection only and then decays using WM decay rate only to detected. Note that the SA actual is "2" at fully comprehended and "1" at Detected, due to the fact that the associated Situation Element is set to "required" which doubles its SA value.

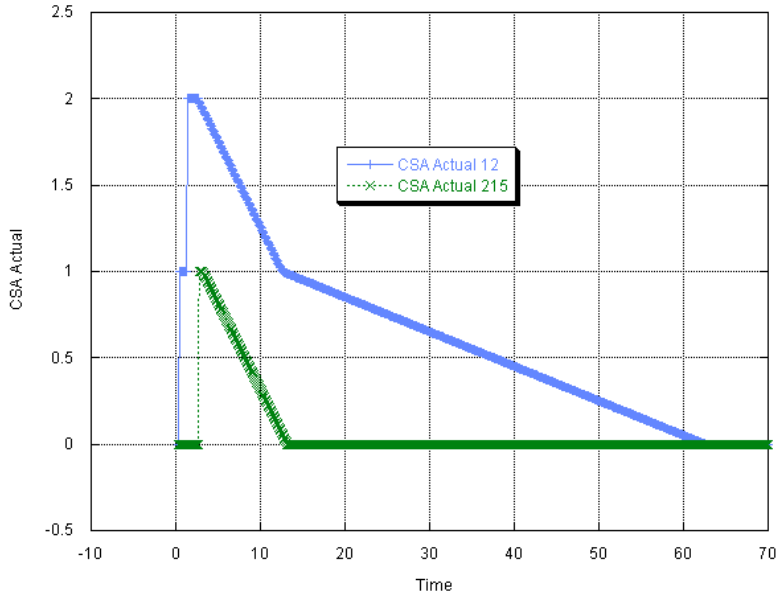


Figure 77. Graph of WM and LTWM SA Decay.

Normal Decay from Detected to Undetected

There has been only one fixation to push the perception level of the component to Detected. It then decays using WM to Undetected after ten seconds. An intermediate fixation during LTWM decay resets the start of the LTWM decay time, but maintains it in LTWM, such that the decay from Detected to Undetected using LTWM always takes 50 seconds from the last fixation (Figure 78).

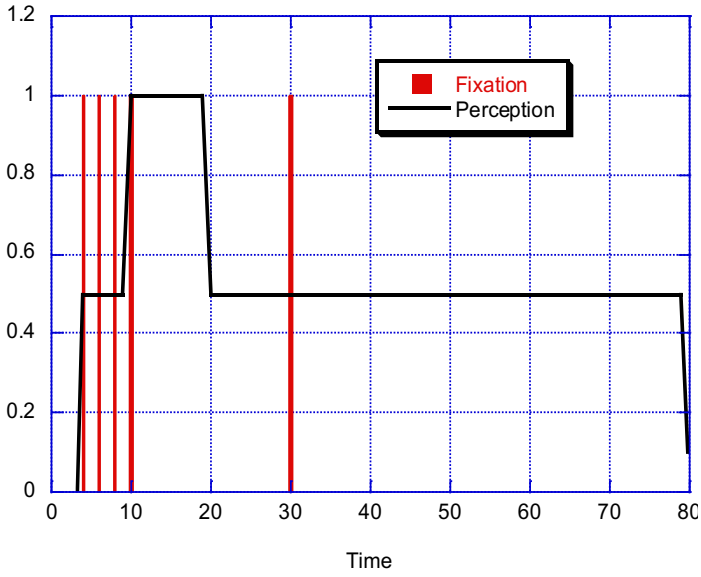


Figure 78. Correct Decay from Comprehended with Fixation to Detection.

MIDAS Situation Awareness Model: Information Accessibility and Data Limits

There are two forms of parameters affected by display or information properties that influence perception and comprehension, and hence attaining situation awareness in MIDAS.

(1) **accessibility costs** impose a delay before detection can even begin. For example keystroke access to a piece of information like a route planning tool requires the pilot to do some manual action before a displayed element can be viewed, and this action will take some time to accomplish.

(2) **detection to comprehension** time (DTC) describes certain display or information properties that simply require a longer time (more fixations) to gain full comprehension after the initial fixation that accomplishes detection. For example a longer word, or longer string of instructions will impose a greater DTC to read.

For both accessibility and DTC costs, time is the critical metric, and this time is either a fixed penalty imposed by some discrete property, or can be represented as an equation, if there is a quantitative (rather than categorical) variable that influences this time.

(3) **data limits** are properties of the information display that simply limit the maximum level of situation awareness that can be obtained from a fixated display attribute , no matter how long it is fixated. An example would be very small text that cannot be read, no matter how closely the eyes examined it. For data limits, we specify a physical property (or quantitative level of such a property) that either produces **mild data limits** or **severe data limits.** The distinction between mild and severe is the extent to which fixations achieve full SA. For mild, fixation achieves 75% of the full SA level. For severe, fixation achieves 50% of the full level. This difference has implications for how rapidly the information is lost by decaying SA memory in the absence of subsequent fixations. Also, if MIDAS were to directly compute **errors of comprehension**, rather than time, these two levels would provide an important basis for error prediction.

The concept of data limits is directly based on the dichotomy drawn by Norman and Bobrow (1975) between data limits and resource limits in multi-tasking. If a task is data limited, perfect performance (in this context, zero errors) cannot be obtained no matter how many resources are supplied. If a task is resource limited, then investment of more resources can ultimately achieve perfect performance. If the only resource involved in limiting task performance was time, then, in the current context, DTC would describe resource limits.

The Accessibility, Detection to Comprehension calculation, and Data Limits are designed to expand the resolution of crewstation component design, to allow for finer-grained resolution of the model and more accurate component testing. Currently there are two Display Elements, which impact times to comprehension: *Accessibility* and *Text Length of Messages*.

Information Accessibility

Currently accessibility has four levels [None, Low, Moderate, and High], which represent the difficulty/time required to access a given component (see Figure 79).

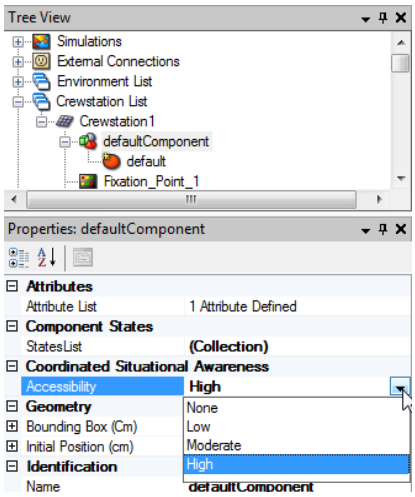


Figure 79. Current Component Accessibility.

Text Length of Messages

Text length is currently the only Detection to Comprehension (DTC) parameter in MIDAS (see Figure 80). It is accessed when a display has an attribute set to TEXT and the *EQUIP_set_attribute_value* sets the attribute's value in the *value* field. The number of characters in the text is the parameter for calculting the DTC.

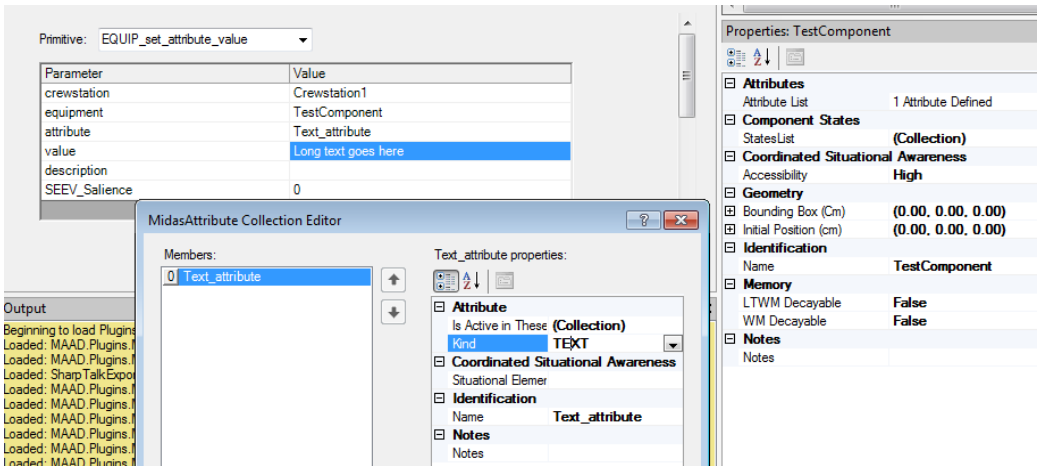


Figure 80. Settings required for Length of Text Messages to impact DTC time.

The three components of the library of display elements and how they are planned to be implemented in MIDAS will each be discussed separately below, followed by a brief discussion of issues with combining multiple display elements.

Accessibility

The Accessibility is a measure of the information availability that assesses a time penalty for information acquisition based on whether information is embedded in a deep level of

a display. Since this time penalty occurs before even detection can occur, this penalty effectively lengthens the number of fixation to get from Undetected to Detected.

Data Limits

Currently, all of the data limits have three values: None, Mild and Severe. The effect of a parameter is to "cap" the SA, making the Perception/SA of the attribute decay from Comprehended to Detected faster (i.e. it is in LTWM for a shorter period of time, although the slope of SA decay remains the same.) It will have no effect on the decay from Detected to Undetected, using either WM or LTWM as shown in Figure 81.

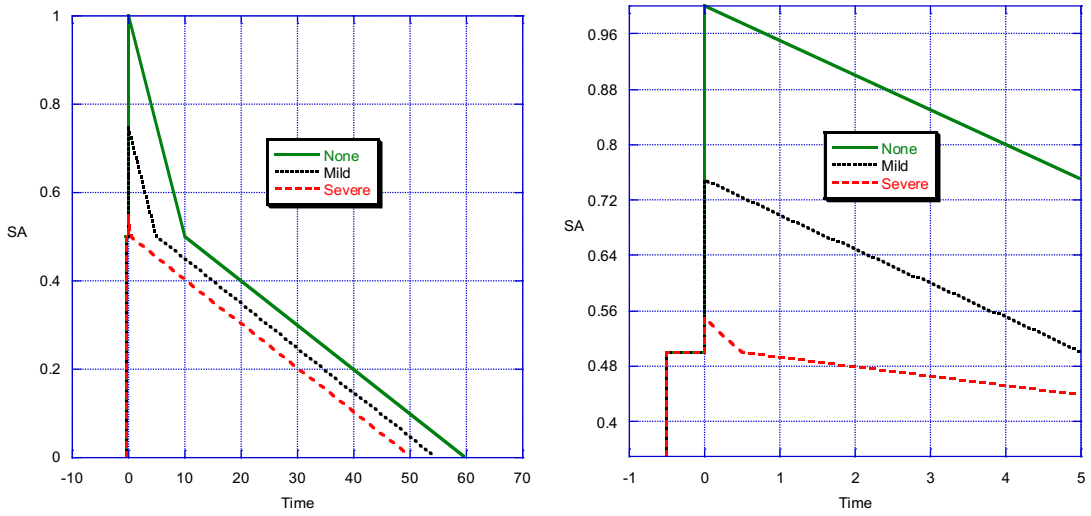


Figure 81. Effect of Data Limit on Perception/SA (close-up on right).

Visual and Auditory DLs

Two of the DLs, Spatial Incompatibility (Auditory) and Working Memory Load are auditory in nature. These two will appear under the DL heading when an auditory attribute is selected. The other seven: 3D ambiguity, Legibility, 2D Display Tracking, Spatial Incompatibility (Visual), Symbol Unfamiliarity, Flight Path Projection, and Time Projection will appear under the DL heading when a visual attribute is selected.

Combining DLs

For Combining DLs, take the most severe of the DLs as the effect on SA.

Changing DTCs

Data Limits depend on the actual physical properties of the display and don't need a primitive to change Data Limits during a model run.

The SA Output Reports

We extended the current MIDAS_Situational_Awareness_by_Task report to include CSA Optimal, CSA Actual, Ratio, and CSA Optimal degraded by Accessibility.
The new MSAT report the run number, the time, the context, the Task column (representing the SEEV tasks), CSA Optimal, CSA Actual, Ratio, and CSA Optimal degraded by Accessibility (Figure 82) .

MIDAS_Situational_Awareness_by_Task_Captain_run1of1_with_SEEV.4DCAA35B.csv

	A	B	C	D	E	F	G	H
1	Operator: Captain							
2	RunNumber	Time	Context	Task	CSA Optimal	CSA Actual	Ratio	CSA Optimal degraded by Accessibility
3	1	0.2	approach	CA - VCSPA Approach "Systems"65_43_13_11_57	6	0	0	0
4	1	0.2	approach	CA - VCSPA Approach "Communicate"65_43_13_11_56	3	0	0	-6
5	1	0.2	approach	CA - VCSPA Approach "Aviate"65_43_13_11_53	8	0	0	-9
6	1	0.2	approach	CA - VCSPA Approach "Separate"65_43_13_11_54	18	0	0	-17
7	1	0.2	approach	CA - VCSPA Approach "Navigate"65_43_13_11_55	16	0	0	-35
8	1	0.3	approach	CA - VCSPA Approach "Systems"65_43_13_11_57	6	0	0	0
9	1	0.3	approach	CA - VCSPA Approach "Communicate"65_43_13_11_56	3	0	0	-6
10	1	0.3	approach	CA - VCSPA Approach "Aviate"65_43_13_11_53	8	0	0	-9
11	1	0.3	approach	CA - VCSPA Approach "Separate"65_43_13_11_54	18	0	0	-17
12	1	0.3	approach	CA - VCSPA Approach "Navigate"65_43_13_11_55	16	0	0	-35
13	1	0.4	approach	CA - VCSPA Approach "Systems"65_43_13_11_57	6	0	0	0
14	1	0.4	approach	CA - VCSPA Approach "Communicate"65_43_13_11_56	3	0	0	-6
15	1	0.4	approach	CA - VCSPA Approach "Aviate"65_43_13_11_53	8	0	0	-9

Figure 82. MIDAS_Situational_Awareness_by_Task (MSAT) report.

The MIDAS Workload Model

Introduction

The challenge associated with the measurement and management of workload from an empirical perspective has led to many different conceptualizations on the degree to which workload should influence an operator's performance. There is little question that workload does impact nominal performance but there is less agreement on precisely how workload influences performance. Some individuals thrive under periods of high task load while others fail under periods of low task load and vice versa. Representing this divergent empirical performance computationally is needed so that model analysts generate accurate representations of human-system interactions. The Man-machine Integration Design and Analysis System (MIDAS) possesses two distinct approaches to represent workload and its impact on operator performance both of which rely on the Multiple Resource Theory (MRT) operating behind the scenes to impact performance. The first is an unconstrained representation where MIDAS completes the tasks and outputs workload according to a conflict matrix with task degradation functions based on the MIDAS behavioral primitives linked to the task. As there are no upper limits or redlines on operator performance, this mode of operation allows model analysts to see where the model predicts workload spikes. The second is a representation that includes a task management model that uses task priority to determine task schedule.

MIDAS Representation of Workload

The MIDAS workload model computes the workload of a multi-tasking operator. As described previously (Gore, 2010), MIDAS uses the Task Analysis / Workload (TAWL; McCracken & Aldrich, 1984; Hamilton, Bierbaum, & Fulford, 1991; Hamilton, Bierbaum, & McAnulty, 1994; Mitchell, 2000) task-load specification structure, which was developed and validated by the U.S. Army. Basic behavioral primitives are created and workload estimates are assigned to the behavioral primitives using the TAWL. These behavioral primitives are then tied to MIDAS tasks in the task network model that are then exercised in various operational contexts. The MIDAS workload model also contains the Multiple Resources Theory (MRT, (Wickens, 1984, 2002) of human attention for situations where multiple tasks share attentional resources.

Workload Channels

MIDAS defines task loads according to seven channels: two input channels (Visual, Auditory), two cognitive processing channels (Cognitive-Spatial and Cognitive-Verbal), and three output channels (Voice, Gross Motor, and Fine Motor). The MIDAS implementation extends McCracken & Aldrich's original four channel representation by breaking out the cognitive information (verbal and spatial) and the output (vocal, gross and fine motor) because the increased fidelity was deemed more consistent with the assumptions of the MRT (Gore, 2010; Hooey et al., 2010) and with many of the application domains (Gore, 2010; Gore, Hooey, Wickens, & Scott-Nash, 2009; Gore & Smith, 2006; Hooey, et al., 2010).

Task Primitives

MIDAS breaks tasks down to a set of basic behavioral *primitives* that contain workload defined along the seven channels outlined in the previous section. There are two kinds of

primitives: Operator Primitives (OP) and User-defined Primitives (UP). There are ten Operator Primitives that represent basic (non domain-specific) human behaviors (e.g., reach, push and release, say message, information seeking using SEEV). The MIDAS user can also define User Primitives that are tailored to the domain application (e.g., acquire lead aircraft). In both cases, the TAWL (Hamilton & Bierbaum, 1992; McCracken & Aldrich, 1984) is used to determine the task load values. The TAWL values are empirically determined and based on US military personnel in the Army Light Helicopter Experimental (LHX) Program (McCracken & Aldrich, 1984), further tested / validated using Army tank operators (Mitchell, 2000), and validated with commercial airline pilots (Gore et al., 2011).

Conflict Matrix and the Multiple Resource Theory (MRT):

Wickens (2002) outlines the requirements for a computational representation of the MRT. In his article, Wickens outlined that the MRT must possesses four features. The first is that each task can be represented as a vector of its resource demands at a qualitative (type of resource) and a quantitative level (number of resources). Second, channel load is task dependent. Third, loss of performance on one or both tasks from its single task level is determined by a performance penalty if: (a) the total demand on both tasks is high, and (b) both tasks compete for overlapping resources (common levels on one of the dichotomous dimensions) within the four dimensions of the multiple resource model (or within the dimensions of whatever other model is selected). Fourth, the amount of performance lost on the tasks can be established by an allocation policy. If both tasks have equal priority, each task will share equally in the performance decrement.

The MIDAS model's implementation of the conflict matrix and the MRT assumes that task interference is directly proportional to the predicted workload. The workload model requires a matrix of resource coefficients (Table 5), which was estimated using MRT (Wickens, 1991). The conflict matrix presents coefficients of the amount of conflict, between resource pairs across tasks. This matrix is heart and soul of the `multiple' aspect of MRT. MRT assumes that if two tasks cannot share a resource, the conflict value is 1.0 (e.g. two tasks simultaneously demanding voice resources). The value is 0 if two tasks can perfectly share the resources in question. These coefficients are used by MIDAS to estimate the workload of the tasks coded in the model. Historically, MIDAS's operator workload model was unconstrained and did not account for the fact that in actuality, operators use workload management strategies to delay or shed tasks when overloaded.

Table 5. MIDAS MRT Conflict Matrix (updated).

		Channel Visual	Auditory	Cog Spat	Cog Verb	Fine Motor	Gross Motor	Vocal
Channel	Visual	1.0	0.4	0.6	0.4	0.5	0.2	0.2
	Auditory		0.8	0.4	0.6	0.2	0.1	0.6
	Cog Spatial			0.8	0.5	0.5	0.2	0.2
	Cog Verbal				0.8	0.2	0.1	0.6
	Fine Motor					0.8	0.6	0.5
	Gross Motor						0.4	0.4
	Vocal							1.0

Workload Calculation

Workload is the summation of the task loads of the cumulative set of primitives being performed with conflict assessed should one exist. Given two concurrent tasks, the demand values for each possible pair of resource channels in which neither demand is zero, are summed and added to the sum of the coefficients from the conflict matrix for the corresponding channels. That is:

Workload = K_1(Sum of demand) + K_2(sum of the conflict components)

The weightings K_1 and K_2 can then be adjusted as needed to reflect differences in weighting of demand and conflict, or both kept at 1.0 as a default. Stated formally, the workload equation is shown in Equation 3

Equation 3. Workload Calculation in MIDAS v5.

$$W_j = K_1 \sum_{i=1}^{7} (a_{t1,1} + a_{t2,1} a_{t3,1}) + K_2 \sum_{i=1}^{7} (cij1 + cij2 + cij3)$$

where

$Wj = instantaneous\ workload\ of\ channel\ i\ at\ time\ t,$

$j = channel$

$K_1 = channel\ constant$

$t1, t2, t3 = operator\ tasks$

$a = load\ of\ channel\ i\ to\ perform\ task\ t$

$K_2 = conflict\ constant$

$c_{i,j} = channel\ conflict$

Workload Computation Example

	V	A	CS	CV	GM	FM	V
t1 (fixate-spatial-object)	5.9	0.0	3.7	0.0	0.0	0.0	0.0
t2 (push-and-release)	3.7	0.0	1.2	0.0	2.2	0.0	0.0

V Workload = (1)(5.9 + 3.7) + (1)(1.0 + 0.6 + 0.2) = 11.4
A Workload = (1)(0.00) = 0.00
CS Workload = (1)(3.7+1.2) + (1)(1 + .7 + .2) = 6.8
CV workload = (1)(0.00) = 0.00
FM Workload = (1)(0.00) = 0.00
GM workload = (1)(0.0 + 2.2) + (1)(0.2 + 0.2) = 2.6
V Workload = (1)(0.00) = 0.00

Output

Workload is output for each channel and can be visualized through the MIDAS run-time displays or can be collected for post-run analyses. The workload model output in MIDAS is presented with the workload scale along the Y-axis and the simulation timeline along the X-axis. Workload spikes are readily observable in the MIDAS output in the context of the model events across the model time. Workload spikes are areas that MIDAS predicts the operator to be in a vulnerable state given that the modeled operator's

workload exceeds a theoretical threshold and is therefore likely to miss critical signals, or have increased time to complete a task given competing tasks according to. For example, Figure 83 displays the Captain's workload across time during approach and landing over all seven channels.

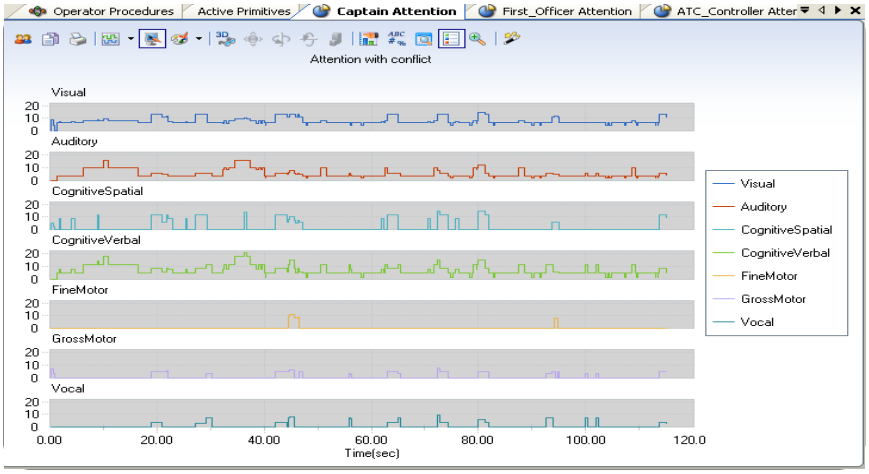


Figure 83. Example of Runtime Display of Workload Experienced by the Captain.

Comparing the workload inputs with the timeline and the task list output from the model tests the baseline workload model's performance.

Workload Management Model

The MIDAS workload model was augmented to reflect the *strategies* that actual operators use when faced with a workload-overload situation. These strategies use MRT as its theoretical underpinning and implement a conflict matrix to delay task performance based on multiple, simultaneous task performance.

Defining Workload-Overload

A workload threshold, or red line value, for each individual workload channel is set by the MIDAS analyst in a 'simulation settings' window of the MIDAS graphic user interface (Figure 84). The computed workload value is compared against the corresponding channel threshold to determine if the operator is overloaded. If the workload value exceeds the threshold value then the operator is considered to be overloaded, and completes the primitive with the highest priority (defined below).

Workload Management	True
Simulation To Execute	
Is Primary Simulation	True
Workload Management Settings	
Priority Weighting Duration	0
Priority Weighting Importance	4
Priority Weighting Interrupt Cost	0
Priority Weighting Urgency	0
Redline Auditory	4
Redline Cognitive Spatial	4
Redline Cognitive Verbal	7
Redline Fine Motor	4
Redline Gross Motor	4
Redline Visual	11
Redline Vocal	4

Figure 84. MIDAS Graphic User Interface for Workload Management Model Settings.

In other words:
If (operator workload[in any one channel] >= set threshold for that corresponding channel), Operator is OVERLOADED

Every time a primitive tries to start, all release conditions of all waiting primitives are recalculated and ranked by priority. Then working down the priority list the next primitive that has task load that won't force an operator channel over the threshold is started.

Determining Task Priority

The strategy adopted by the operator in MIDAS is to work on primitives with the highest priority. The MIDAS analyst assigns a value to represent Importance, Urgency, Duration, and Interrupt Cost for each operator task. The MIDAS analyst then determines the relative weighting of those four factors by assigning a Priority Factor Weighting Score.

The priorities for each primitive are computed based on an algorithm created by synthesizing Freed's (Freed, 2000) task scheduling research with that of Wickens & McCarley (Wickens & McCarley, 2008). The algorithm that was developed is as follows:

Priority = PriorityWeightingImportance * Importance'
+ PriorityWeightingUrgency * RelativeDelay
+ PriorityWeightingDuration *(1-RelativeDuration)
+ PriorityWeightingInterruptCost * InterruptCost

The following section discusses the four factors - Importance, Relative Delay, Relative Duration and Interruption Cost - and the weightings that are applied to them.

Importance
MIDAS NextGen models have five high-level tasks: Aviate, Separate, Navigate, Communicate, and Systems Management. Each task primitive inherits importance from one of these high level tasks, as assigned by the MIDAS analyst. There is always a one to one mapping between a primitive and a high level task. Should a primitive support two high level tasks, the primitive inherits the higher importance task. Each of these high level tasks is assigned an importance value of none (0), low (1), medium (2), or high (3). These values are subsequently converted as follows:

Importance' = Task Importance/3.

Therefore, a high importance task would have an importance' of 1 and a low importance task would have an importance' of .333.

Urgency (Relative Delay)
MIDAS uses the amount of time a task has been delayed as a proxy for urgency. The calculation used in MIDAS is as follows:

RelativeDelay = total time a primitive has been delayed/total time of delayed tasks

Where *total time a primitive has been delayed* is defined as the sum total a particular primitive spends waiting in queue before the operator starts working on that primitive; and *total time of delayed primitives* is defined as the total time of all the primitives at any given time that are waiting in queue for the operator's attention.

Duration
The factor that contributes to the priority of a primitive is Relative Duration. Relative duration is defined as follows:

RelativeDuration = duration/total time of duration of delayed primitives

Where *duration* is derived from the 'duration field' within each MIDAS primitive. The primitive durations vary depending on the kind of primitive. For example, a "user_read" primitive has duration where the total duration for that primitive is defined as (0.06 seconds) * (the number of characters in the string). So if the user_read primitive has 50 characters, then the total duration = (0.06) * 50 = 3 seconds. *Total time of duration of delayed primitives* is defined as the sum total of the duration of all the primitives in queue.

Interruption Cost
Interruption cost refers to the time cost incurred if the operator is distracted and then must reacquire the information needed to complete the task. Similar to the task importance factor, the interruption cost is inherited from the high-level tasks (i.e., Aviate, Separate, Navigate, Communicate, and Systems Management). While the importance values chosen by the user is descriptive in the sense that the MIDAS analyst selects high (3), moderate(2), low(1) or none(0), these values are subsequently converted as follows:

InterruptionCost' = InterruptionCost/3.

Therefore, a task with a high interruption cost would have an InterruptionCost' of 1 and task with a low interruption cost would have an InterruptionCost' of .333.

Priority Factor Weightings
Each of the factors (Importance, Relative Duration, Relative Delay and Interruption Cost) discussed above can be weighted individually through the simulation settings. Through the use of the weights the user can influence which factor is used as the deciding factor to calculate priority. If all factors are weighted equally, the prioritization algorithm (as presented above) accounts for all factors.

MIDAS Workload Management Model Verification

To test each of the MIDAS workload management strategies discussed above, specific models were created using the following steps:

- An existing model was chosen.
- New workload-bearing primitives were then added to the network to specifically cause the operator to experience points of overload during the model run.
- The **Importance**, **Duration** and **Interruption Cost** was set for each of the primitives by either specifying the value directly or by linking the primitive to a task with these values in place. **Delay** was addressed in the network design.
- The **prioritization** of the four management strategies was set for the entire simulation.
- The **redline** values for each workload channel (the point over which the operator is considered to be in overload) were set.

Workload/Task Management Strategy Tests

The remainder of this document describes all test models in detail and the results, which confirm the success of the workload strategy implementation. The tests used in this study are based on the FAA-NASA Land and Taxi 2010 MIDAS model (Gore, et al., 2011). To efficiently test the strategies, however, only context of flight, namely **Descent**, was used. Furthermore, a much simpler network of primitives to make testing of the strategies more manageable replaced the existing network of primitives in this Descent context. Each strategy tested was put through five tests:

- Strategy **disabled**: this test demonstrates that workload management strategies have no effect when the Workload Management flag in the simulation settings is set to **False**. All simultaneously encountered primitives execute at the same time as expected and none are delayed. Because an unlimited number of primitives are allowed to execute, workload for these tests is much higher along the simulation timeline compared to the strategy-enabled tests. Furthermore, because more primitives are allowed to execute simultaneously, these runs are in general much shorter than those from the strategy-enabled tests.
- Strategy tested in **isolation** (all other strategies zeroed out) with redlines threshold values **exceeding** workload total for two simultaneous primitives. This test demonstrates that a strategy will limit the operator to two simultaneous primitives at a time. This test also demonstrates that an isolated strategy works as expected if all other strategies are zeroed out.
- Strategy tested in **isolation** (all other strategies zeroed out) with redline threshold values **totaling** workload total for two simultaneous primitives. This test demonstrates that a strategy will force the operator into a 1-task-at-a-time situation. This test also demonstrates that an isolated strategy works as expected if all other strategies are zeroed out.
- Strategy tested as the **highest priority** over all other strategies with redline threshold values **exceeding** workload total for two simultaneous primitives. This test demonstrates that a strategy will limit the operator to two simultaneous primitives at a time. This test also demonstrates that weightings from other strategies (provided they are lower) do not affect the priority strategy.

- Strategy tested as the **highest priority** over all other strategies with redline threshold values **totaling** workload total for two simultaneous primitives. This test demonstrates that a strategy will force an operator into a 1-task-at-a-time situation. This test also demonstrates that weightings from other strategies (provided they are lower) do not affect the priority strategy.

Some of the strategies tested include additional tests which check to see how the strategy behaves if a task is already in progress prior to a workload conflict.

1. Task Importance Test

The Task Importance Test uses several instances of the same *User_Read* primitive. This primitive involves workload in the Visual and Cognitive Verbal Channels. By itself a single instance of this primitive contributes a visual workload value of 5.0 and a cognitive verbal workload value of 3.0. Each additional instance of the primitive adds the same amount of workload to the same channels.

In the network in Figure 85, all three primitives start at the exact same time and take the exact same amount of time: (0.06) x (number of characters in the string), which in this case is 0.06 x 100 = 6 seconds.

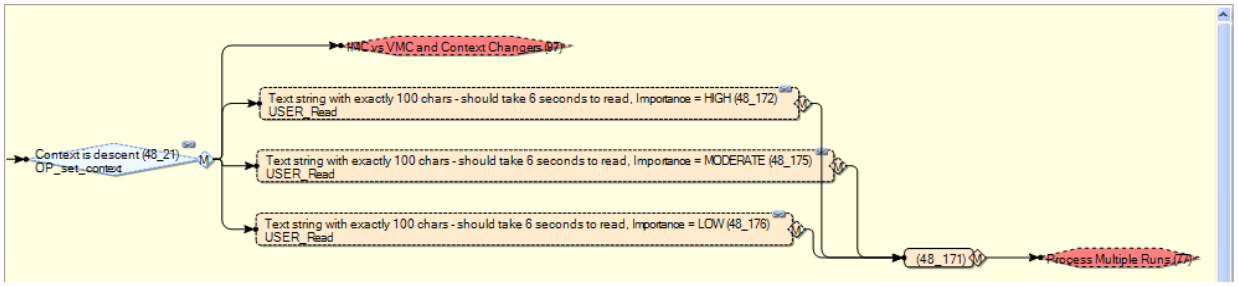


Figure 85. Network illustration of three SEEV start times.

Each primitive is linked to a specific SEEV Start task whose importance value is then inherited (Figure 86and Figure 87). The linked task is listed in the **midasTask** field for each primitive:

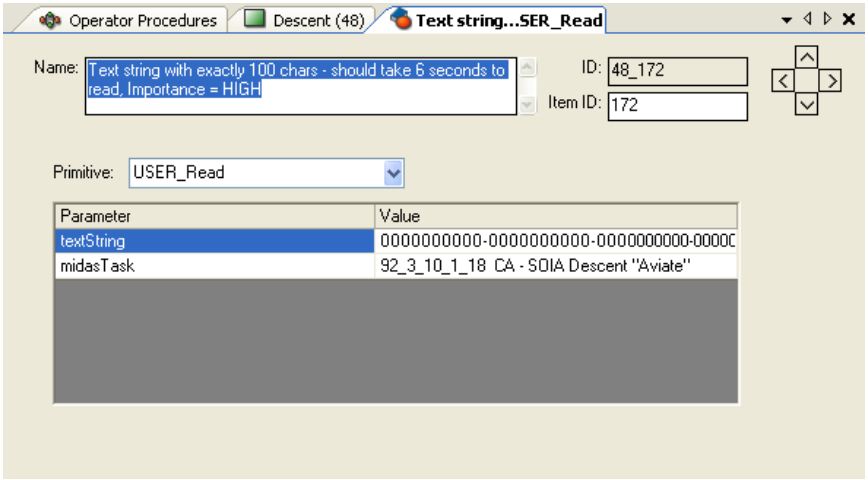


Figure 86. Example of the High importance text string, the MIDAS primitive and its parameters.

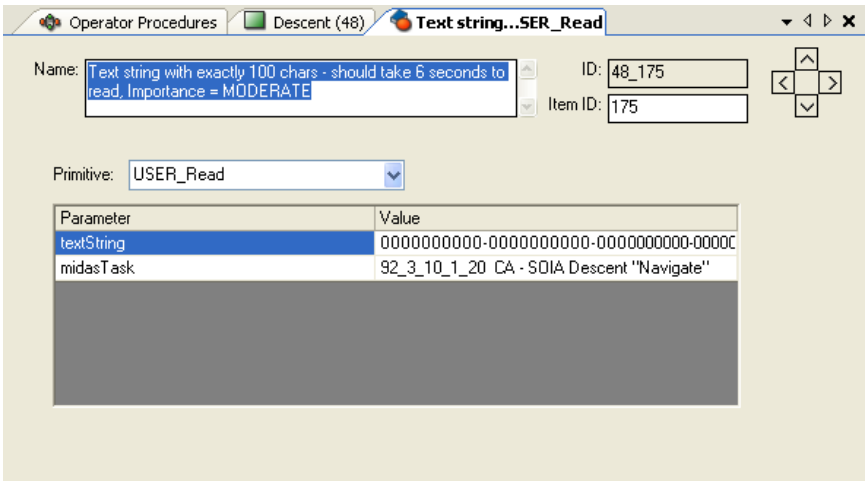


Figure 87. Example of Moderate importance text string, the MIDAS primitive and its parameters.

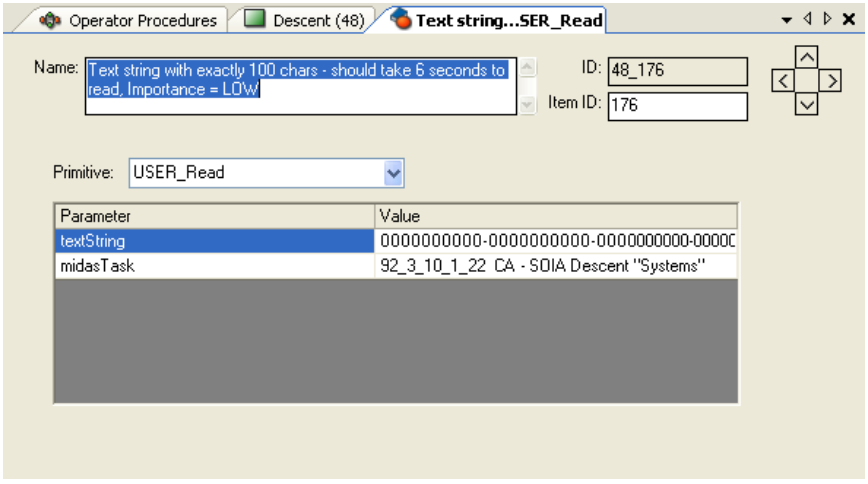


Figure 88. Example of the low importance text string, the MIDAS primitive and its parameters.

The SEEV Tasks referenced in the primitives above may be found in the Captain's SEEV Settings for the Descent context, SOIA Scenario under IMC Conditions (Figure 89):

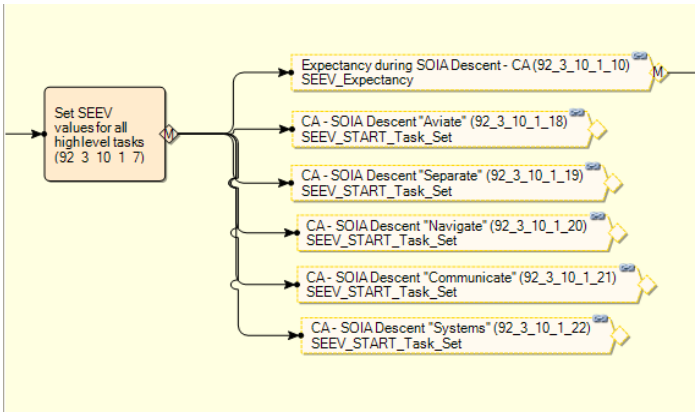


Figure 89. SEEV settings called by the user read primitive broken down per context.

In this test, the value of the Task Importance was modified while all other values (Interruption Cost and Duration) remained the same (Figure 90).

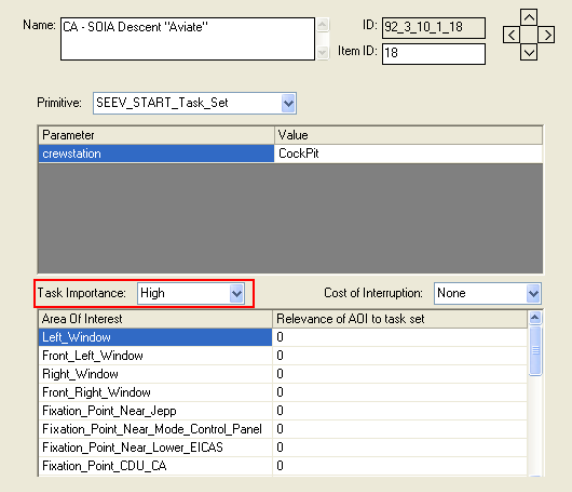


Figure 90. Manipulation of the task importance of the SEEV start task in the context of aviate.

Next, the prioritization of the Workload/Task Management Strategies were set by accessing the User Simulation Settings in the Figure 91 (a) MIDAS tree, clicking on it, to show (b) its properties:

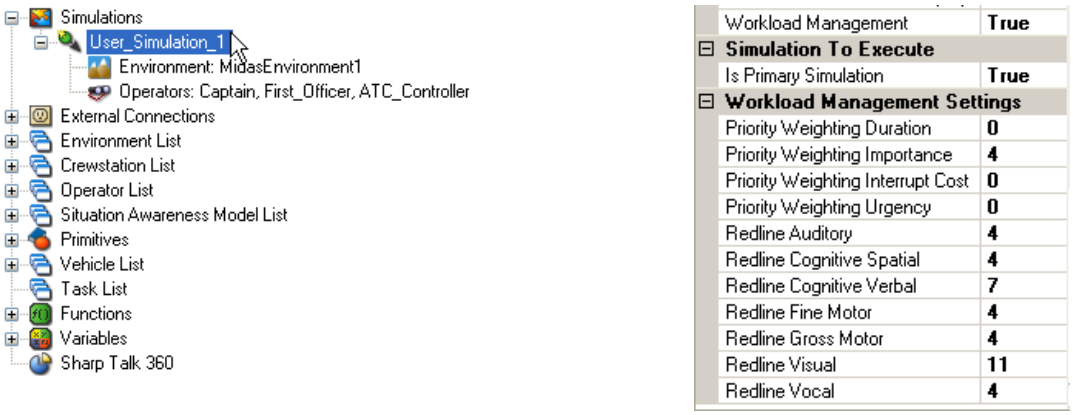


Figure 91. Assignment of workload Redline Values.

- **Priority Weighting Importance:** set to the highest value of 4 in order that the task importance determine the priority of the next primitive to occur.
- **Redline Visual:** depending on the test being run, this value is set to 10 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 11 to allow a maximum of two User_Read primitives to happen simultaneously.
- **Redline Cognitive Verbal:** depending on the test being run, this value is set to 6 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 7 to allow a maximum of two User_Read primitives to happen simultaneously.

Task Importance Test Results:

Test Case 1A: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)
Workload Management Flag set to **False**[1] (Nonessential data rows hidden)

In the absence of workload strategy (Figure 92), all simultaneously encountered tasks (48_172, 48_175 and 48_176) are taken on (Table 6). Workload well exceeds the 11.0 and 7.0 redlines (**Error! Reference source not found.**).

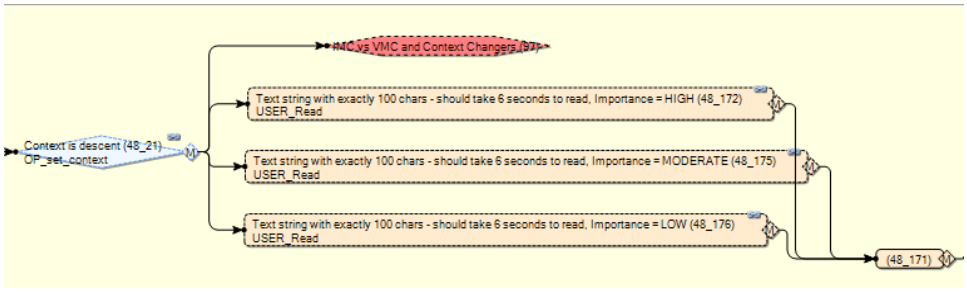


Figure 92. Task network of test case 1a, baseline model run with no workload management.

Table 6. Code output of test case 1a - Task begin and end times output to support workload management verification.

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name			
1	0	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH		
1	0	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE		
1	0	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW		
1	6	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH		
1	6	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE		
1	6	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW		

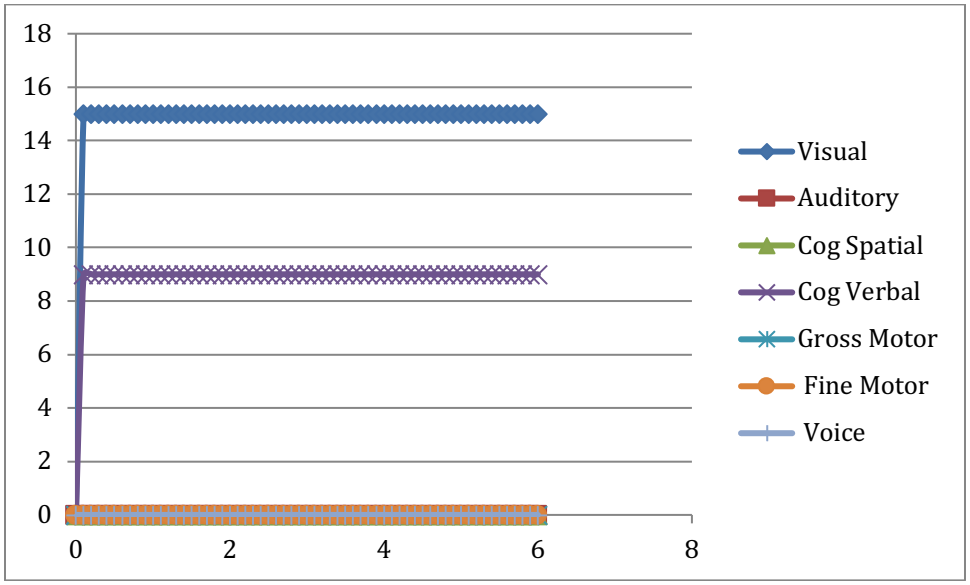


Figure 93. Workload timeline output of test case 1a to illustrate effect of workload management model (WM=false).

[1] This is a baseline run to test the absence of workload management strategies

Test Case 1B: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)
Workload Management Flag set to **True** (Nonessential data hidden).

In this test, the operator is allowed to perform up to two tasks simultaneously without being in overload (Figure 94). The Task Begin/End report excerpt below demonstrates this by the fact that the third primitive, 48_176, is held back while the first two primitives are still executing (Table 7).

Table 7. Code output of test case 1b - task begin and end times output to support workload management verification (WM=true).

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name		
1	0	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	0	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	6	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	6	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	6	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	
1	12	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	

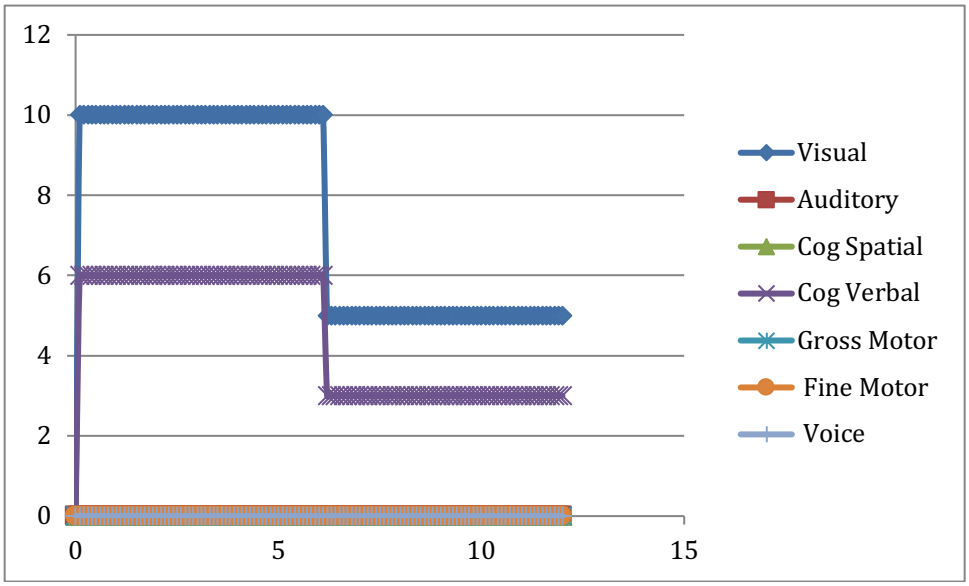


Figure 94. Workload timeline output of test case 1b to illustrate effect of workload management model (WM=true).

Test Case 2: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0)
Workload Management Flag set to **True** (Nonessential data hidden).

In this test, the operator is allowed to perform at most one primitive at a time without being in overload (Figure 95). This is demonstrated in the test below by the fact that no two primitives overlap in Clock time (Table 8). The corresponding workload output can be found in Figure 96.

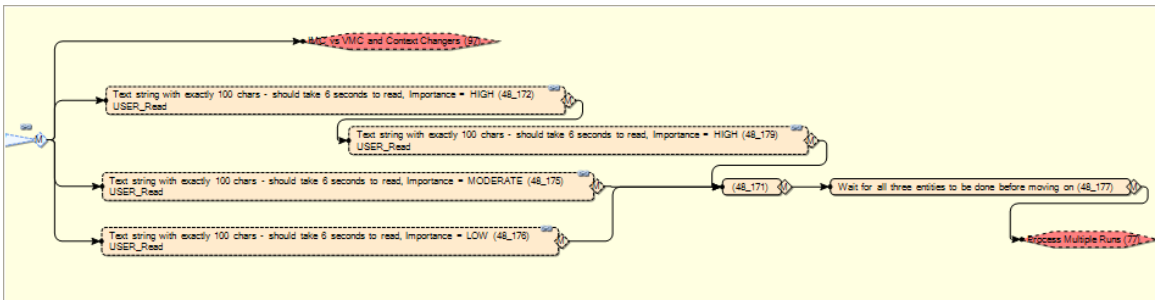


Figure 95. Task Network of test case 2 threshold equal to total, one task to be completed without being in overload (WM=true)

Table 8. Code output of test case 2 task begin and end times output to support workload management verification (WM=true).

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name		
1	0	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	6	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	6	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	12	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	12	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	18	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	18	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	
1	23.9	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	

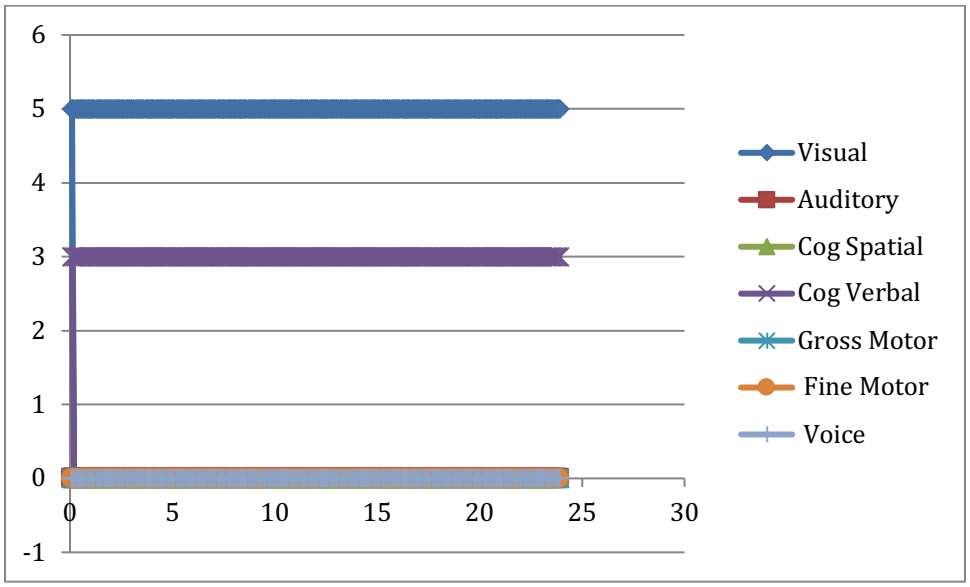


Figure 96. Workload timeline output of test case 2 to illustrate effect of workload management model with threshold of 1 (WM=true).

Test Case 3: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0), One Task Executes while Two Follow (three seconds later) in Overload
Workload Management Flag set to **True** (Nonessential data hidden).

In this test, the operator is allowed to perform at most one primitive at a time without being in overload (Figure 97). This is demonstrated in the test below by the fact that no two primitives overlap in Clock time (Table 9). This test also demonstrates that workload strategies operate as expected in the event the tasks are staggered in time (Figure 98).

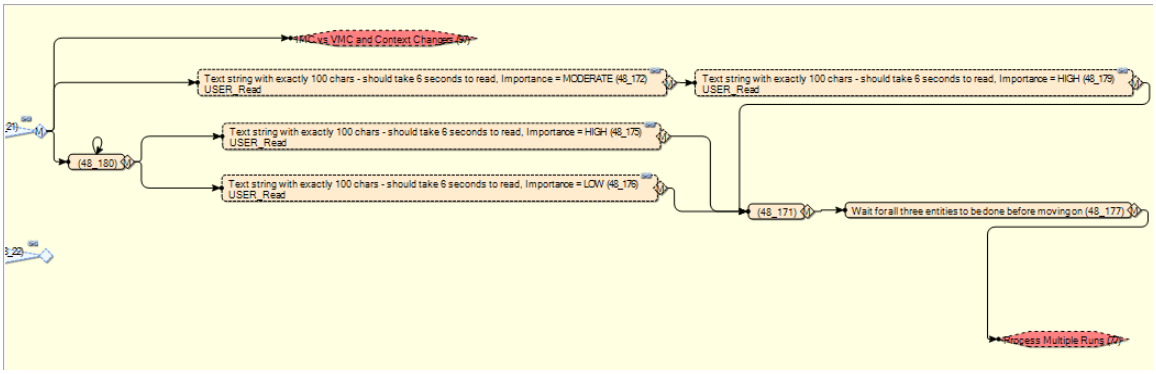


Figure 97 Task network of test case 3 - one task executing while two follow (three seconds later) in overload.

Table 9. Task begin and end times output of test case 3 to support workload management verification of one task executing while two follow (three seconds later) in overload.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name			
1	0	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE		
1	6	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE		
1	6	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH		
1	12	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH		
1	12	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH		
1	18	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH		
1	18	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW		
1	23.9	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW		

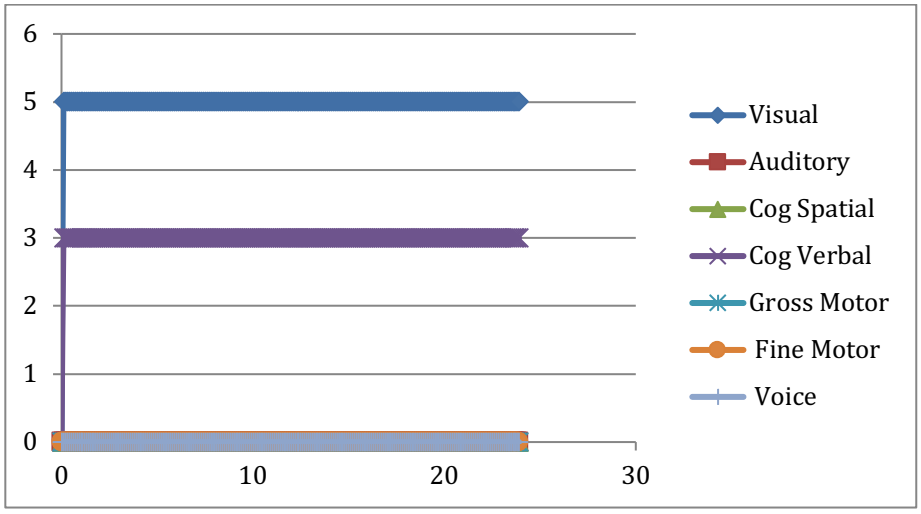


Figure 98. Workload timeline output of test case 3 - one task executing while two follow (three seconds later) in overload.

Test Case 4: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0), All Strategies Weighted

Workload Management Flag set to **True** (Non-transition times hidden).

In this test, the operator is allowed to perform at most one primitive at a time without being in overload (Figure 99). This is demonstrated in the test below by the fact that no two primitives overlap in Clock time (Table 10). This test also demonstrates that the

highest priority strategy still works as expected when all other strategies are weighted (Figure 100).

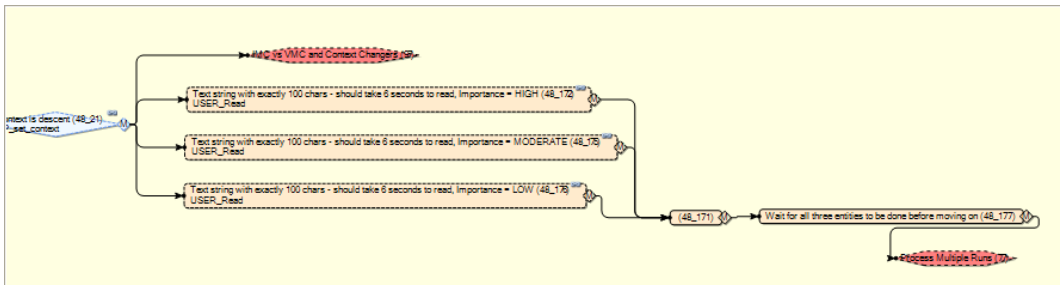


Figure 99. Take network of test case 4 - one primitive at a time without being in overload.

Table 10. Code output of test case 4 - one primitive at a time without being in overload.

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name		
1	0	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	6	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	6	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	12	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	12	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	
1	18	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	

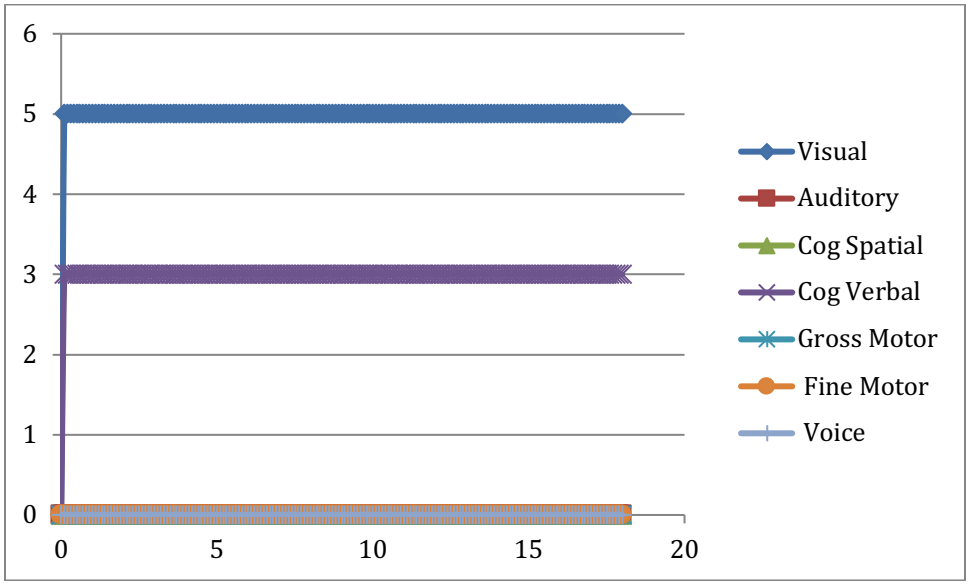


Figure 100. Workload timeline output of test case 4 - one primitive at a time without being in overload.

Test Case 5: Threshold Greater Than total (Visual: 11.0, Cog Verbal: 7.0), All Strategies Weighted

Workload Management Flag set to **True** (Non-transition times hidden).

In this test, the operator is allowed to perform up to two primitives at a time (Figure 101). This task also demonstrates that the highest priority strategy, in this case Task Importance, is working as expected since the primitive with the lowest importance, 48_176, executes last (Table 11), and workload becomes overloaded (Figure 102). This test also demonstrates that the highest priority strategy still works as expected when all other strategies are weighted.

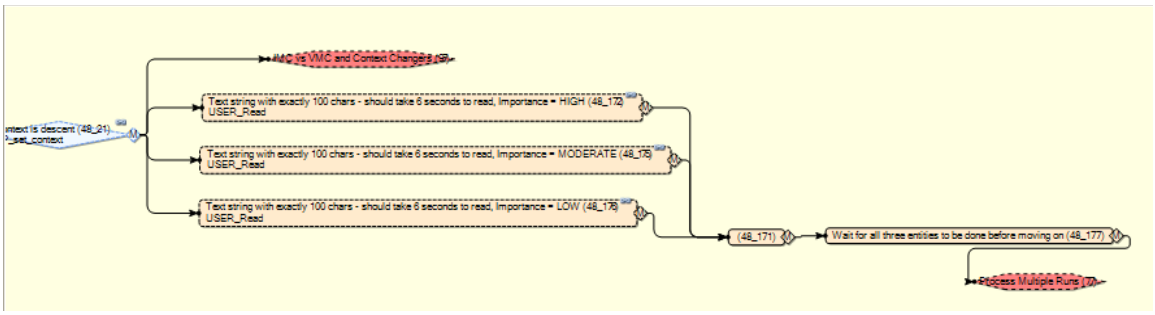


Figure 101. Take network of test case 5 - two primitive allowed at a time without being in overload.

Table 11. Code output of test case 5 - two primitive allowed at a time without being in overload.

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name		
1	0	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	0	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	6	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	6	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Importance = MODERATE	
1	6	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	
1	12	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Importance = LOW	

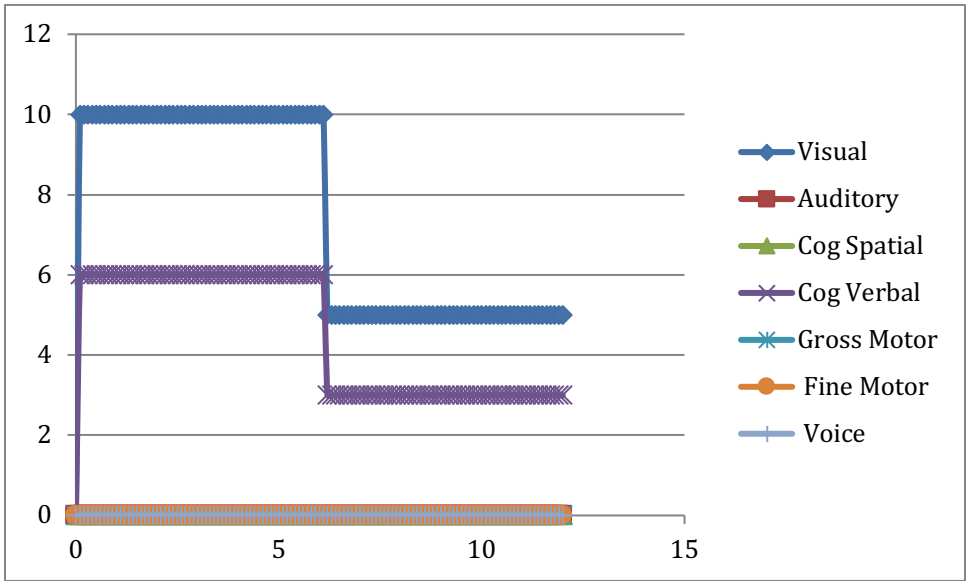


Figure 102. Workload timeline output of test case 5 - two primitives at a time without being in overload.

These verification test results illustrate that the workload management model operates verifiably and that all results are as expected. The next test that is required to evaluate whether the MIDAS workload management operates verifiably is a test of the task duration times.

2. Task Duration Test

The Task Duration Test uses several variations of the same User_Read primitive. This primitive involves workload in the Visual and Cognitive Verbal Channels. By itself a single instance of this primitive contributes a visual workload value of 5.0 and a cognitive verbal workload value of 3.0. Each additional instance of the primitive adds the same amount of workload. **The difference in each instance of the primitive,**

however, is the duration, which is a function of string length specified in the Text String field.

In the network below (Figure 103), all three primitives start at the exact same time. The duration of each primitive, however, is calculated as follows:

First task: (0.06) x (number of characters in the string) = 0.06 x 100 = 6 seconds.
Second task: (0.6) x (number of characters in the string) = 0.06 x 50 = 3 seconds.
Third task: (0.6) x (number of characters in the string) = 0.06 x 10 = 0.6 seconds.

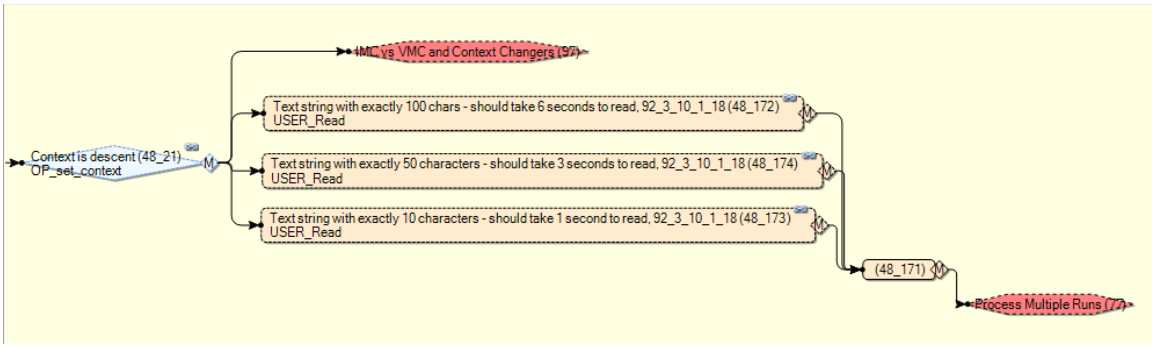


Figure 103. Task network of task duration test.

Each primitive is linked to a specific SEEV Start task whose importance value is then inherited (Figure 104). Because the importance is being held constant among all primitives in this test, all three primitives have been linked to the same SEEV Start task:

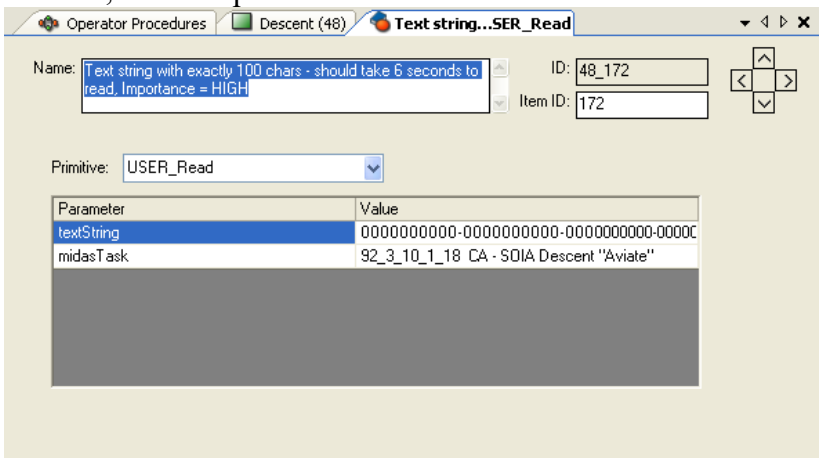


Figure 104. Step to link SEEV start primitive to MIDAS primitive, parameter input.

The single SEEV Task referenced by all the primitives above may be found in the Captain's SEEV Settings for the Descent context, SOIA Scenario and IMC Conditions (Figure 105):

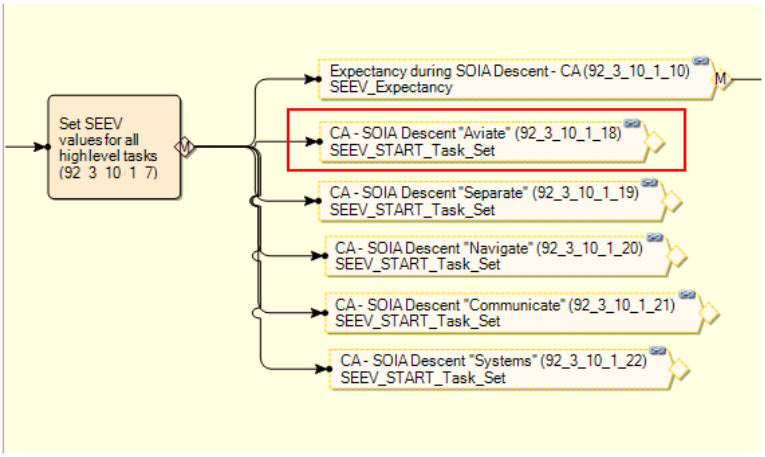


Figure 105. Task network example of SEEV start tasks.

In this test, the value of the Task Importance was set to **High** (Figure 106).

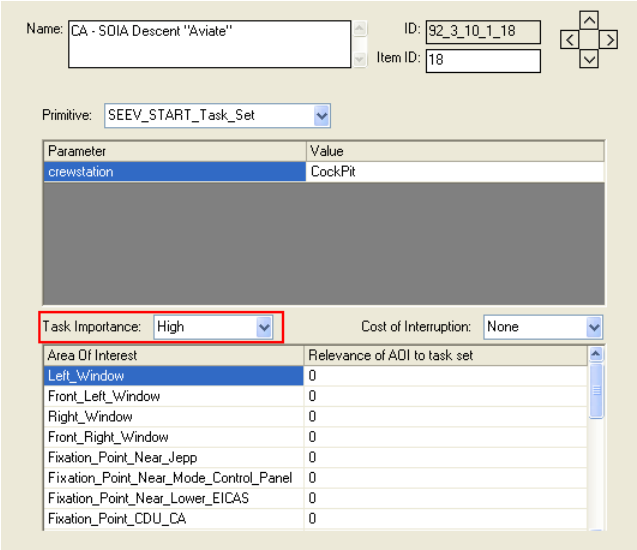


Figure 106. Task importance flag settings.

Next, the prioritization of the Workload/Task Management Strategies were set by accessing the User Simulation Settings in the Figure 107 (a) MIDAS tree, clicking on it, to show (b) its properties:

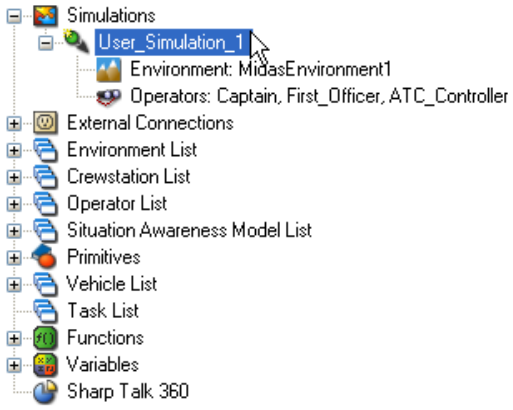


Figure 107a. Workload management strategies settings through the User simulation pull down menu in the properties window.

Workload Management	True
Simulation To Execute	
Is Primary Simulation	True
Workload Management Settings	
Priority Weighting Duration	4
Priority Weighting Importance	0
Priority Weighting Interrupt Cost	0
Priority Weighting Urgency	0
Redline Auditory	4
Redline Cognitive Spatial	4
Redline Cognitive Verbal	7
Redline Fine Motor	4
Redline Gross Motor	4
Redline Visual	11
Redline Vocal	4

Figure 107b. Workload management settings window.

- **Priority Weighting Duration**: set to the highest value of 4 in order that the task duration determine the priority of the next primitive to occur.
- **Redline Visual:** depending on the test being run, this value is set to 10 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 11 to allow a maximum of two User_Read primitives to happen simultaneously.
- **Redline Cognitive Verbal:** depending on the test being run, this value is set to 6 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 7 to allow a maximum of two User_Read primitives to happen simultaneously.

Task Duration Test Results:
Test Case 1: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)
Workload Management Flag set to **False**[2] (Nonessential data rows hidden).

The task network used in this test can be found in Figure 108. In the absence of workload strategy, all simultaneously encountered tasks(48_179, 48_180 and 48_181) are taken on (Table 12). Workload well exceeds the 11.0 and 7.0 redlines (Figure 109).

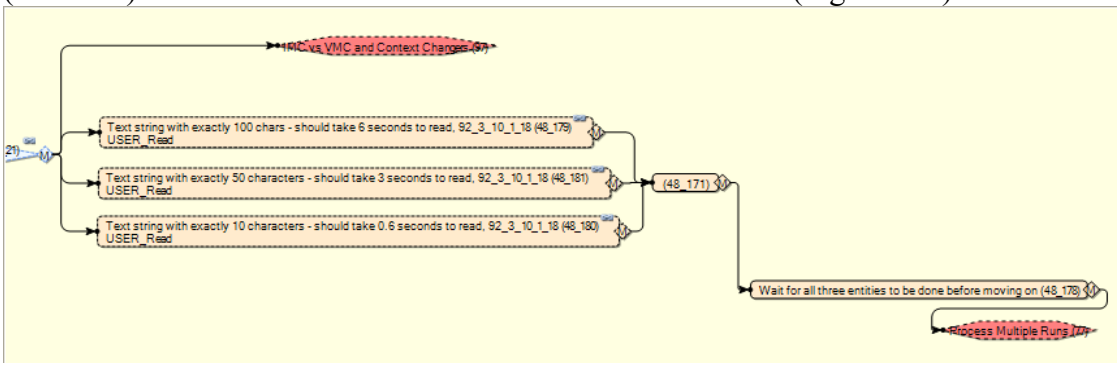


Figure 108. Task network for test case 1 for task duration; threshold > total, WM false.

[2] This is a baseline run to test the absence of workload management strategies.

Table 12. Code output for test case 1 for task duration; threshold > total, WM false.

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	0	descent	Captain	start	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	0	descent	Captain	start	48_180	Text string with exactly 10 characters - should take 0.6 seconds to read	92_3_10_1_18
1	0.5	descent	Captain	end	48_180	Text string with exactly 10 characters - should take 0.6 seconds to read	92_3_10_1_18
1	2.9	descent	Captain	end	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	6	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18

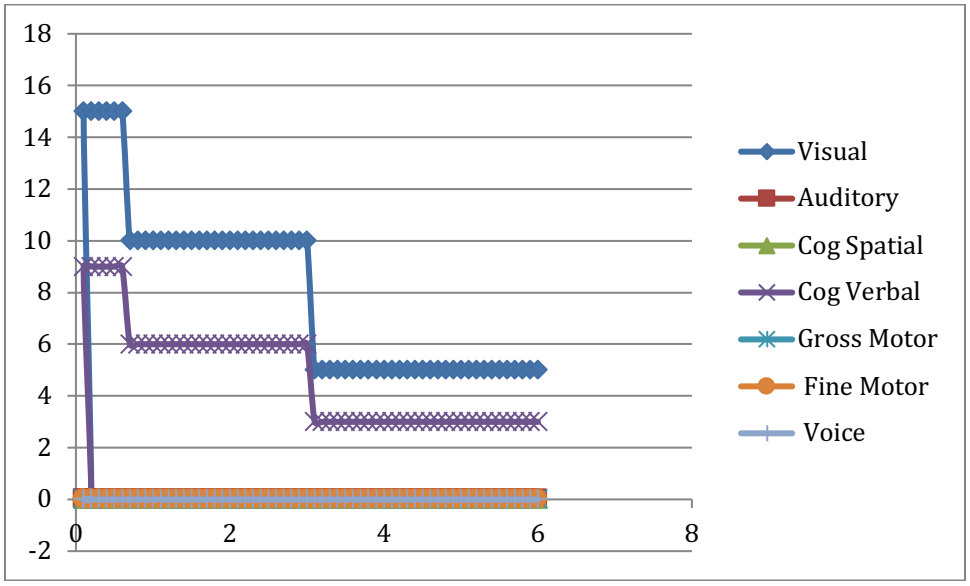


Figure 109. Workload timeline output for test case 1 task duration; threshold > total, WM false.

Test Case 1: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)
Workload Management Flag set to **True** (Nonessential data hidden).

In this test, the operator is allowed to perform up to two tasks simultaneously without being in overload. The Task Begin/End report excerpt below demonstrates this by the fact that the third primitive, 48_179, is held back while the first two primitives are still executing (Table 13; Figure 110)

Table 13. Code output for test case 1 for task duration; threshold > total, WM true (Visual: 11.0, Cog Verbal: 7.0).

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_180	Text string with exactly 10 characters - should take 0.6 seconds to read	92_3_10_1_18
1	0	descent	Captain	start	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	0.5	descent	Captain	end	48_180	Text string with exactly 10 characters - should take 0.6 seconds to read	92_3_10_1_18
1	0.6	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	2.9	descent	Captain	end	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	6.6	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18

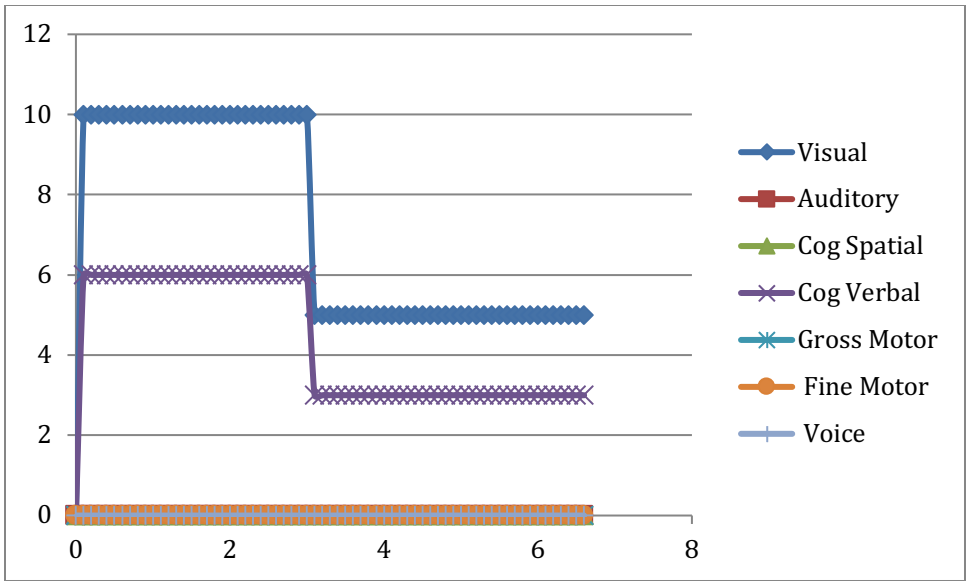


Figure 110. Workload timeline output for test case 1 task duration; threshold > total, WM true.

Test Case 2: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0)
Workload Management Flag set to **True** (Nonessential data hidden).

In the test below (Figure 111), the operator is limited to performing only one primitive at a time. The results in the Task Begin/End excerpt below clearly demonstrate that the primitive with the shortest duration is always performed first (Table 14, Figure 112).

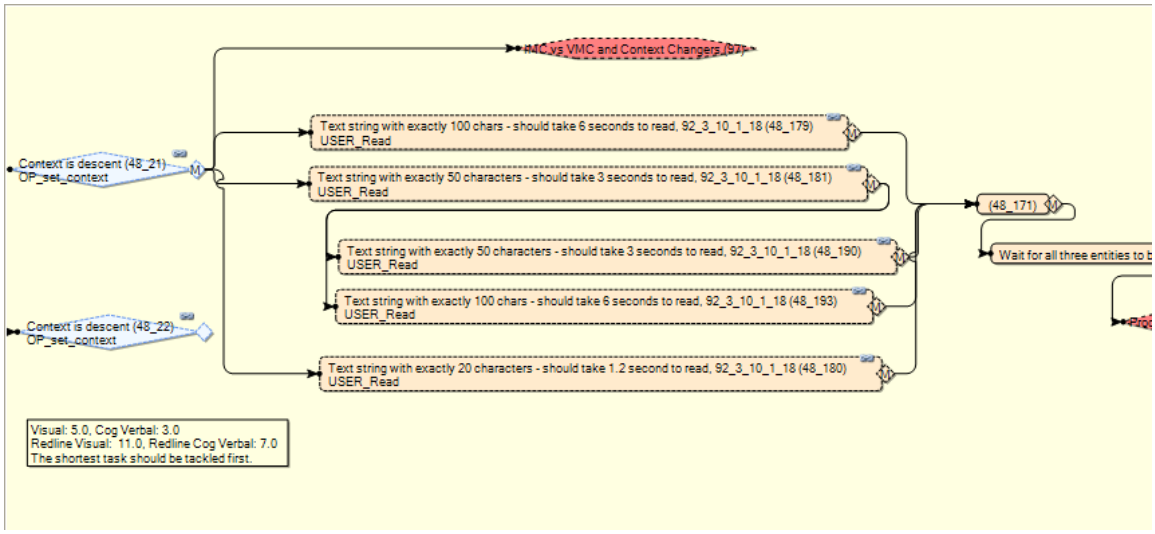


Figure 111. Task network for test case 2 for task duration; threshold = total, WM true, V is 10, CS is 6.

Table 14. Code output for test case 2 for task duration; threshold = total, WM true.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	1.1	descent	Captain	end	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	1.2	descent	Captain	start	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	4.1	descent	Captain	end	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	4.1	descent	Captain	start	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	7.2	descent	Captain	end	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	7.2	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	13.2	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	13.2	descent	Captain	start	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	19.1	descent	Captain	end	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18

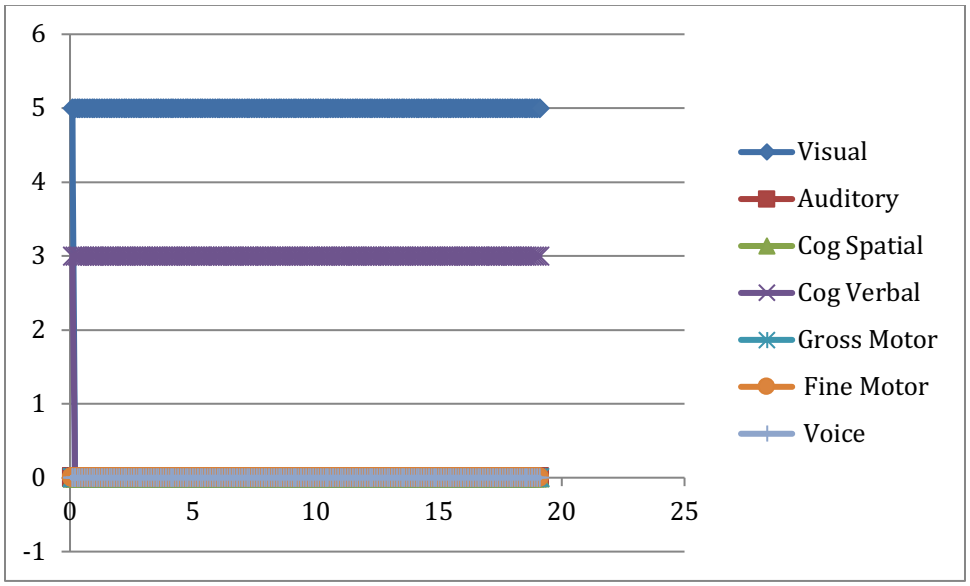


Figure 112. Workload timeline output for test case 2 task duration; threshold = total, WM true.

Test Case 3a: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0), One Task Executes while Two Follow in Overload

Workload Management Flag set to **True** (Nonessential data hidden).

In this test, the operator is allowed to perform at most one primitive at a time without being in overload (Figure 113). This is demonstrated in the test below as no two primitives overlap in Clock time (Table 15). This test also demonstrates that workload strategies operate as expected in the event the tasks are staggered in time (Figure 114).

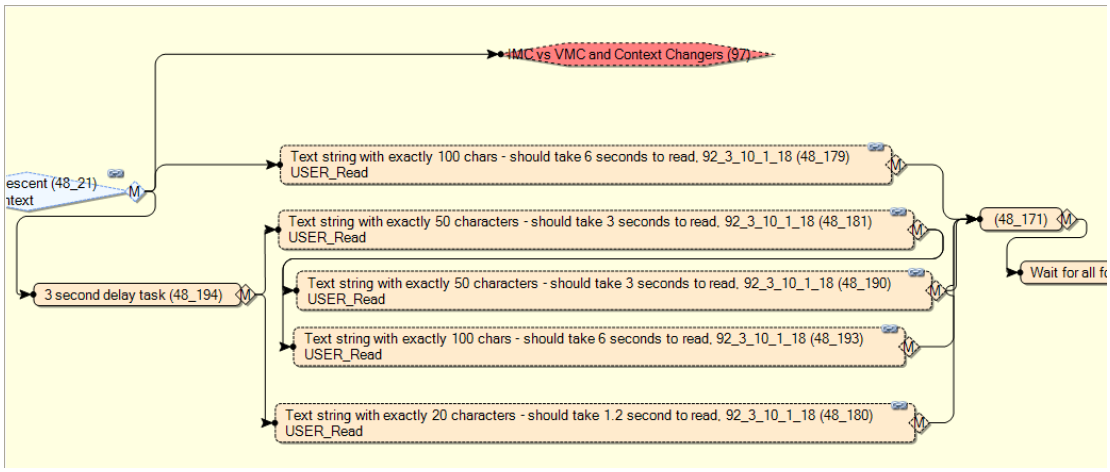


Figure 113. Task network for test case 3a for task duration; threshold = total, one task executes while two follow in overload, WM true.

Table 15. Code output for test case 2 for task duration; threshold = total, WM true.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	6	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	6	descent	Captain	start	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	7.2	descent	Captain	end	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	7.2	descent	Captain	start	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	10.2	descent	Captain	end	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	10.2	descent	Captain	start	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	13.2	descent	Captain	end	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	13.2	descent	Captain	start	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	19.1	descent	Captain	end	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18

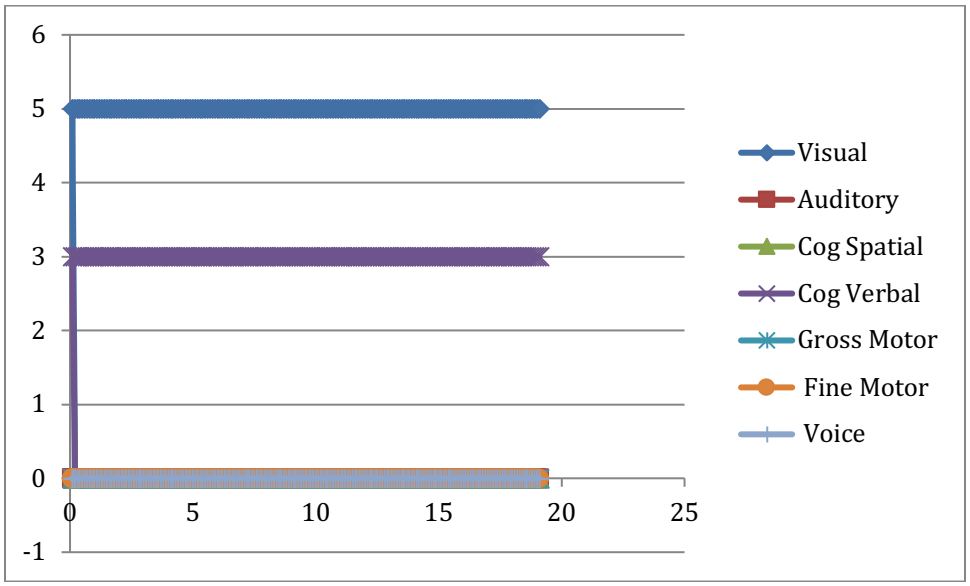


Figure 114. Workload timeline output for test case 3a for task duration; threshold = total, on task executes while two follow in overload, WM true.

Test Case 3b: Threshold Bumped Up (Visual: 11.0, Cog Verbal: 7.0

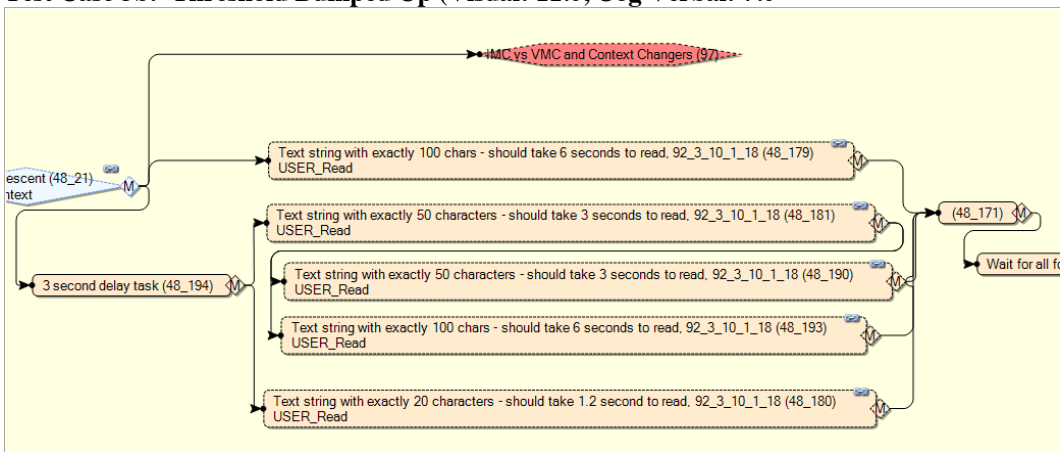


Figure 115. Task network for test case 3b for task duration; threshold = total, one task executes while two follow in overload, WM true, threshold for v 11, CV is 7.

Table 16. Code output for test case 3b for task duration; threshold = total, WM true threshold for v 11, CV is 7.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	2.9	descent	Captain	start	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	4.1	descent	Captain	end	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	4.1	descent	Captain	start	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	6	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	7.2	descent	Captain	end	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	7.2	descent	Captain	start	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	7.3	descent	Captain	start	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	10.2	descent	Captain	end	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	13.3	descent	Captain	end	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18

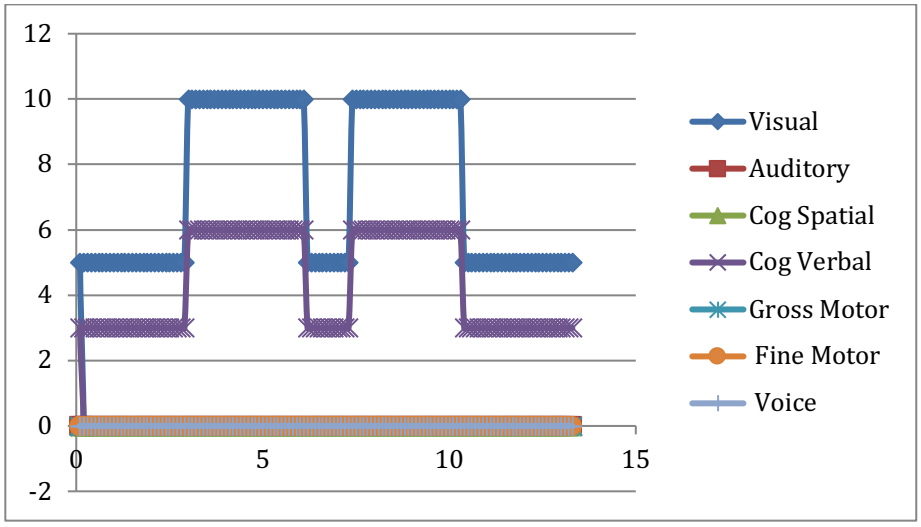


Figure 116. Workload timeline output for test case 3b for task duration; threshold = total, one task executes while two follow in overload, WM true.

Test Case 4: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0), All Strategies Weighted

Workload Management Flag set to **True** (Non-transition times hidden).

In test #4 illustrated in Figure 117, the operator is allowed to perform at most one primitive at a time without being in overload. This is demonstrated in the test below by the fact that no two primitives overlap in Clock time (Table 17). This test also demonstrates that the highest priority strategy, in this case duration, still works as expected when all other strategies are weighted (Figure 118).

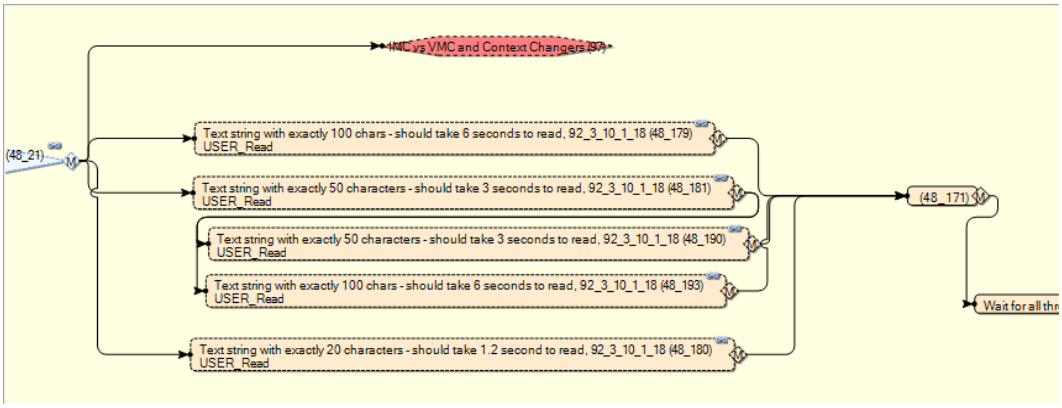


Figure 117. Task network for test case 4 for task duration; threshold = total, WM true, threshold for v is 10, CV is 6.

Table 17. Code output for test case 4 for task duration; threshold = total, WM true threshold for v is 10, CV is 6.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0.2	descent	Captain	start	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	1.3	descent	Captain	end	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18
1	1.4	descent	Captain	start	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	4.3	descent	Captain	end	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	4.3	descent	Captain	start	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	7.4	descent	Captain	end	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18
1	7.4	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	13.4	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18
1	13.4	descent	Captain	start	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18

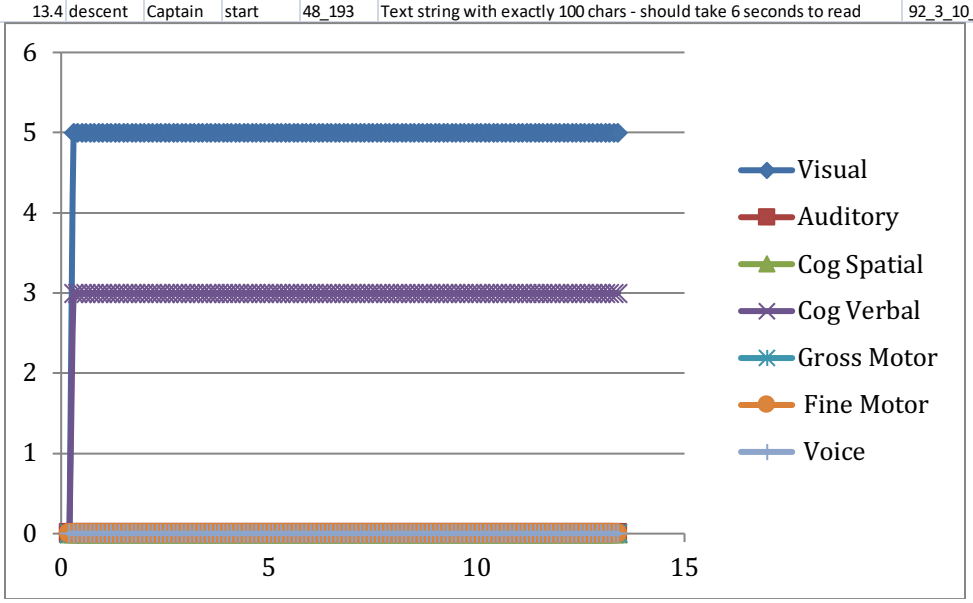


Figure 118. Workload timeline output for test case 4 for task duration; threshold = total, WM true, threshold for V is 10, CV is 6.

Test Case 5: Threshold Greater Than total (Visual: 11.0, Cog Verbal: 7.0), All Strategies Weighted

Workload Management Flag set to **True** (Non-transition times hidden)

In this test (task network shown in Figure 119), the operator is allowed to perform up to two tasks simultaneously without being in overload. The Task Begin/End report excerpt below demonstrates the Duration strategy is working by the fact that primitives 48_179, 48_190 and 48_193 (the primitives with the longest durations) are held back while the first two primitives are still executing (Table 18). This test also demonstrates that the highest priority strategy still works as expected when all other strategies are weighted (Figure 120).

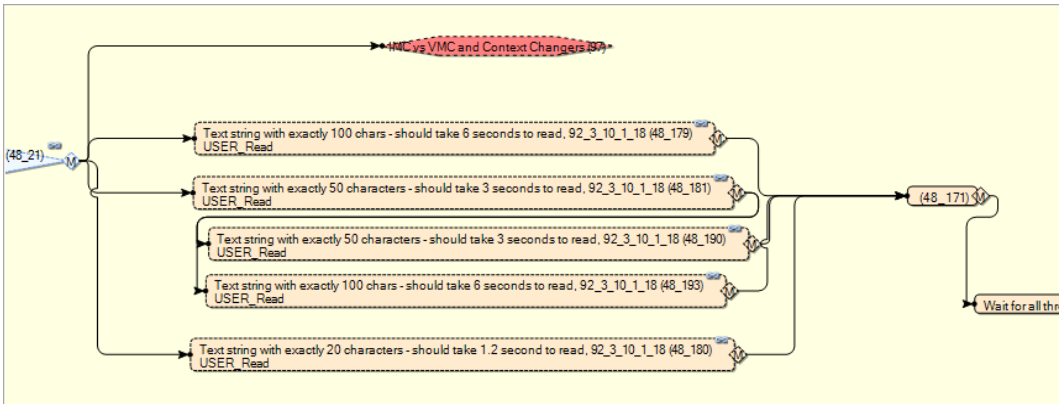


Figure 119. Task network for test case 5 for task duration; threshold > total, WM true, threshold for v is 11 CV is 7.

Table 18. Code output for test case 5 for task duration; threshold > total, WM true threshold for v is 11, CV is 7.

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name		
1	0	descent	Captain	start	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18	
1	0	descent	Captain	start	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18	
1	1.1	descent	Captain	end	48_180	Text string with exactly 20 characters - should take 1.2 second to read	92_3_10_1_18	
1	1.2	descent	Captain	start	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18	
1	2.9	descent	Captain	end	48_181	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18	
1	2.9	descent	Captain	start	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18	
1	6	descent	Captain	end	48_190	Text string with exactly 50 characters - should take 3 seconds to read	92_3_10_1_18	
1	6	descent	Captain	start	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18	
1	7.2	descent	Captain	end	48_179	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18	
1	12	descent	Captain	end	48_193	Text string with exactly 100 chars - should take 6 seconds to read	92_3_10_1_18	

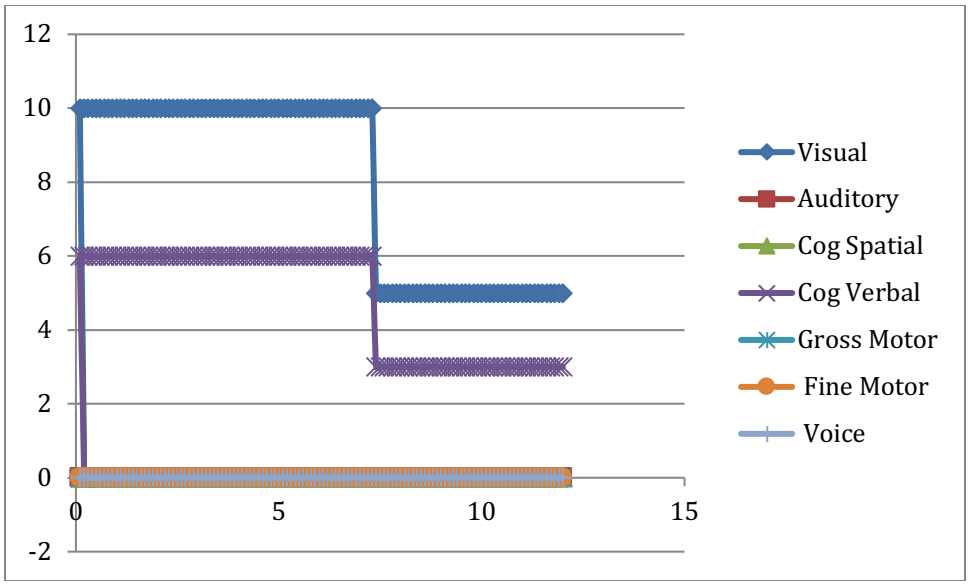


Figure 120. Workload timeline output for test case 5 for task duration; threshold > total, WM true threshold for v is 11, CV is 7.

3. Cost Interruption Test

The Cost Interruption Test uses several instances of the same User_Read primitive. This primitive involves workload in the Visual and Cognitive Verbal Channels. By itself a single instance of this primitive contributes a visual workload value of 5.0 and a cognitive verbal workload value of 3.0. Each additional instance of the primitive adds the same amount of workload.

In the network illustrated in Figure 121, all three primitives start at the exact same time and take the exact same amount of time: (0.06) x (number of characters in the string), which in this case is 0.06 x 100 = 6 seconds.

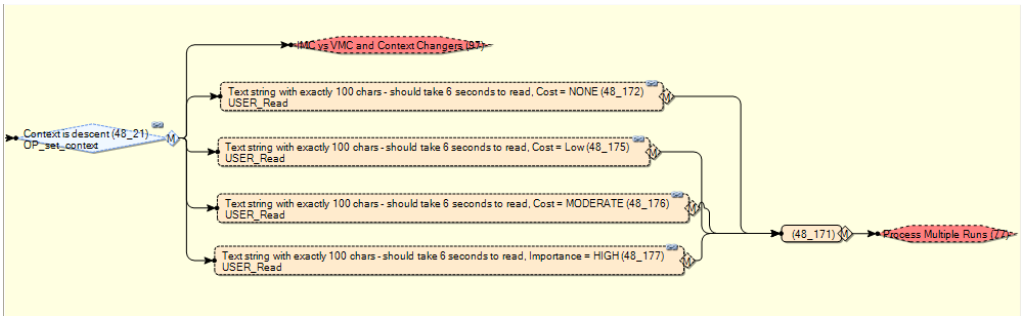


Figure 121. Task network of Cost interruption test.

Each primitive is linked to a specific SEEV Start task whose **Cost Importance** value is then inherited. The linked task is listed in the midasTask field for each primitive (Figure 122, Figure 123, Figure 124, Figure 125):

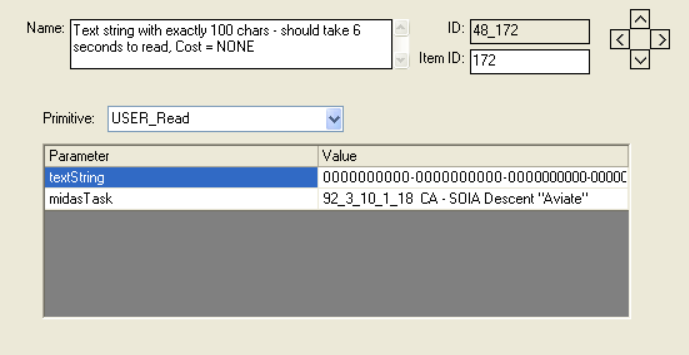


Figure 122. SEEV start primitive linked to MIDAS primitive, parameter input, cost importance is inherited.

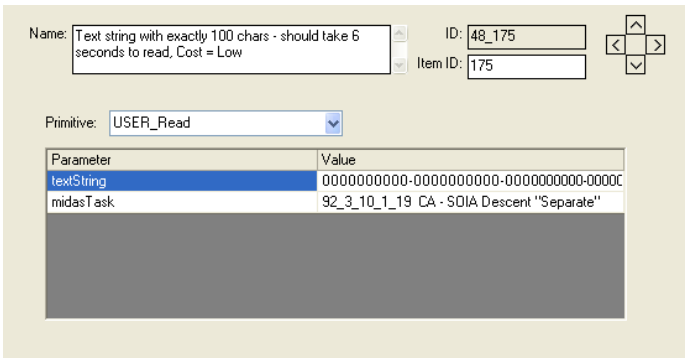


Figure 123. SEEV start primitive linked to MIDAS primitive, parameter input, cost importance is inherited.

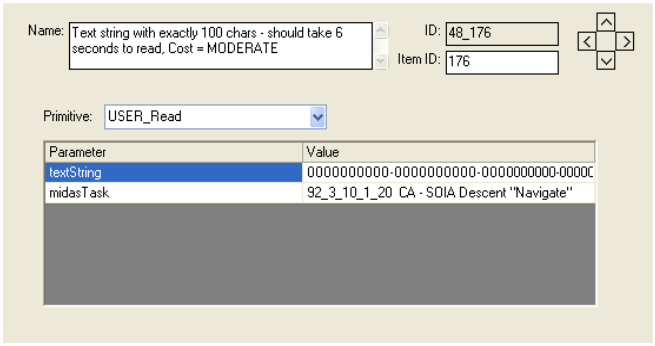


Figure 124. SEEV start primitive linked to MIDAS primitive, parameter input, cost importance is inherited.

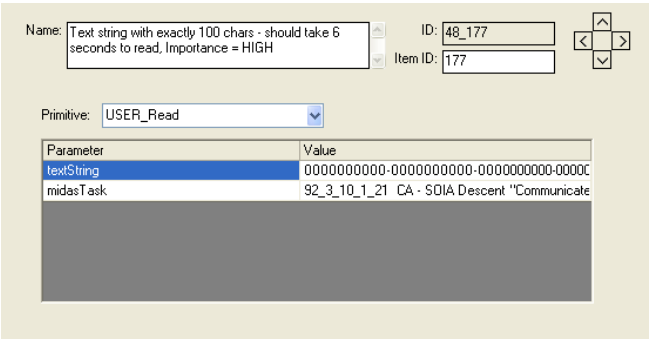


Figure 125. SEEV start primitive linked to MIDAS primitive, parameter input, cost importance is inherited.

The SEEV Tasks referenced in the primitives above may be found in the Captain's SEEV Settings for the Descent context, SOIA Scenario and under IMC Conditions (see Figure 126):

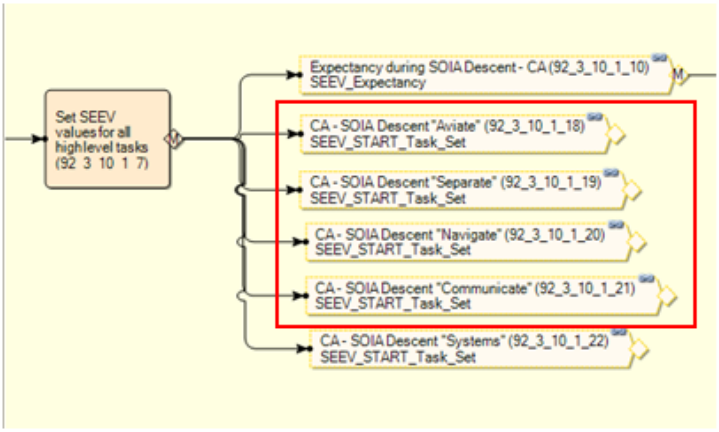


Figure 126. Task network example of SEEV start tasks.

In this test, the value of the **Interruption** Cost was modified while all other values (Importance and Duration) remained the same. The example in Figure 127 shows the Cost of Interruption being set to **None** displays:

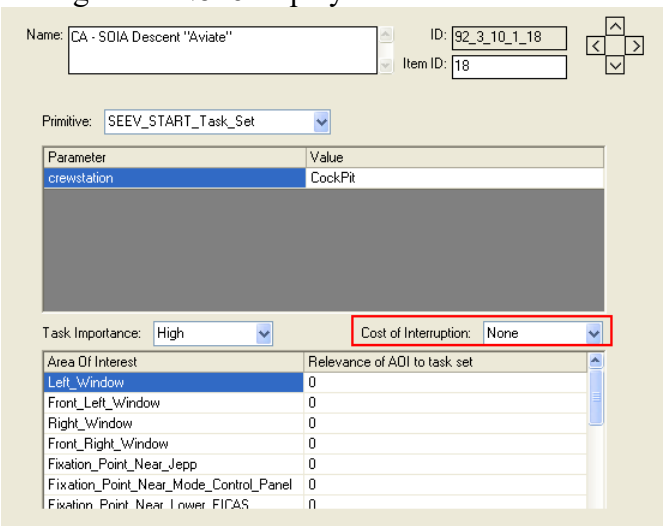


Figure 127. Cost of interruption setting in the SEEV start task set.

- Aviate Task: Cost of Interruption set to NONE
- Separate Task: Cost of Interruption set to LOW
- Navigate Task: Cost of Interruption set to Moderate
- Communicate Task: Cost of interruption set to HIGH

Next, the prioritization of the Workload/Task Management Strategies were set by accessing the User Simulation Settings in the Figure 128 (a) MIDAS tree, clicking on it, to show (b) its properties.

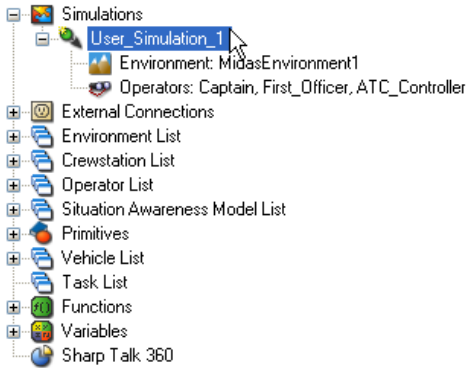
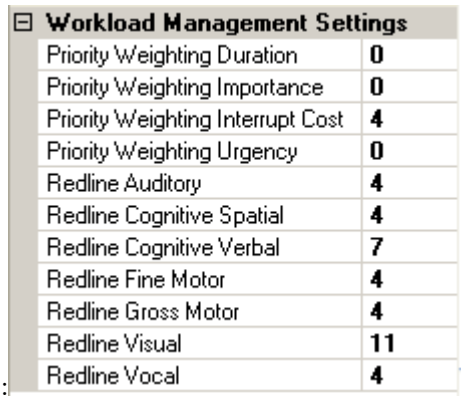


Figure 128a. Workload management strategies settings through the User simulation pull down menu in the properties window.



Workload Management Settings	
Priority Weighting Duration	0
Priority Weighting Importance	0
Priority Weighting Interrupt Cost	4
Priority Weighting Urgency	0
Redline Auditory	4
Redline Cognitive Spatial	4
Redline Cognitive Verbal	7
Redline Fine Motor	4
Redline Gross Motor	4
Redline Visual	11
Redline Vocal	4

Figure 128b. Workload management settings window.

- **Interrupt Cost:** set to the highest value of 4 in order that the **cost of interruption** determines the priority of the next primitive to occur. All other strategies' settings are set to zero.
- **Redline Visual:** depending on the test being run, this value is set to 10 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 11 to allow a maximum of two User_Read primitives to happen simultaneously.
- **Redline Cognitive Verbal:** depending on the test being run, this value is set to 6 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 7 to allow a maximum of two User_Read primitives to happen simultaneously.

Results:
Test Case 1: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)
Workload Management Flag set to **False (This is a baseline run to test the absence of workload management strategies)** (Nonessential data hidden)

In the absence of workload strategy, all simultaneously encountered tasks are taken on (Table 19). Workload well exceeds the 11.0 and 7.0 redlines (Figure 129).

Table 19. Code output for test case 1 for cost of interruption; threshold > total, WM false.

RunNumber	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Cost = NONE
1	0	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	0	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	0	descent	Captain	start	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Cost = NONE
1	6	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	6	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	6	descent	Captain	end	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH

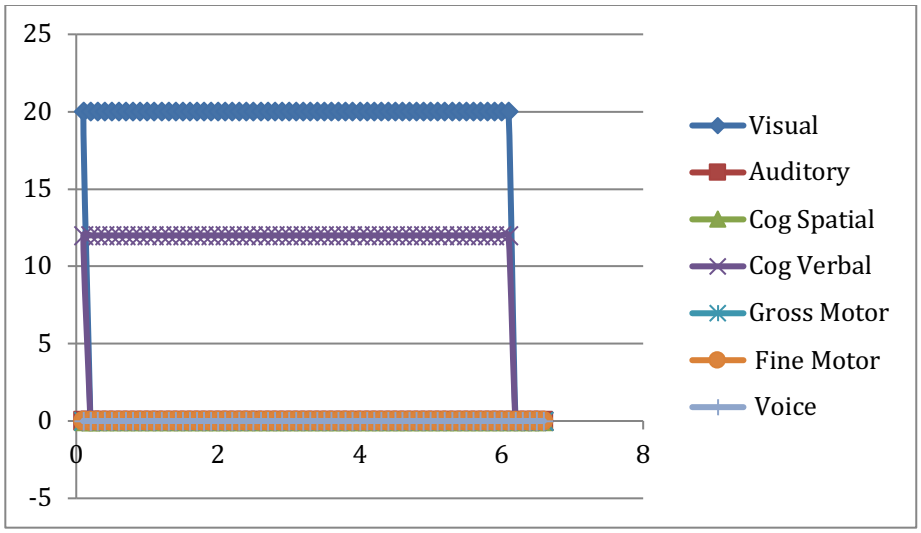


Figure 129. Workload timeline output for test case 1 cost of interruption; threshold > total, WM false.

Test Case 1: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)
Workload Management Flag set to **True** (Nonessential data hidden).

The operator is allowed to perform up to two tasks simultaneously without being in overload. The task network used can be found in Figure 130. The Task Begin/End report excerpt below demonstrates the Interruption Cost Strategy by the fact that the primitives with the lowest interruption cost values, 48_175 and 48_172, are held back while the primitives with the higher interruption cost still execute (Table 20). The effect on the runtime workload output can be found in Figure 131.

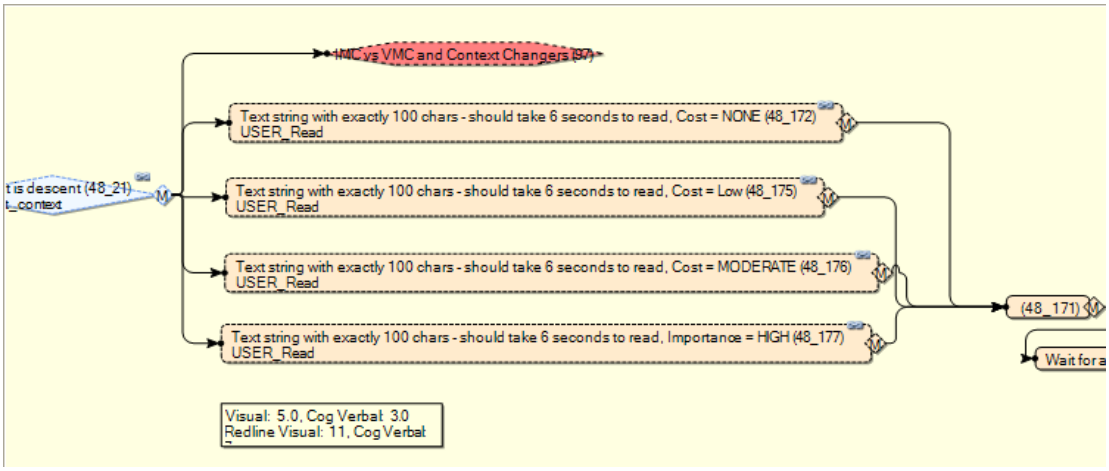


Figure 130. Task network for test case 5 for cost of interruption; threshold > total, WM true, threshold for v 11 CV is 7.

Table 20. Code output for test case 5 for cost of interruption; threshold > total, WM true threshold for v 11, CV is 7.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	0	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	6	descent	Captain	end	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	6	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	6	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Cost = NONE
1	12	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	12	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Cost = NONE

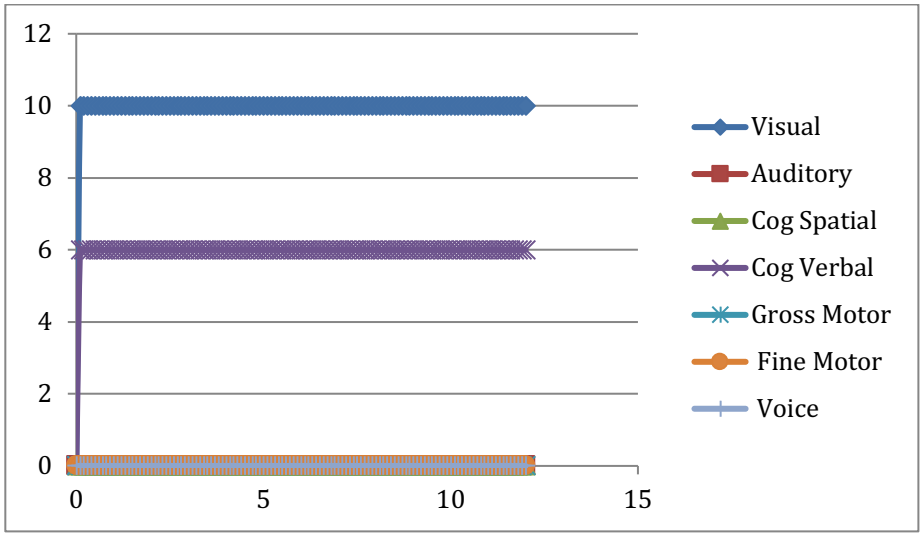


Figure 131. Workload timeline output for test case 1 cost of interruption; threshold > total, WM true., threshold for V is 11, CV is 7.

Test Case 2: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0)
Workload Management Flag set to **True** (Nonessential data hidden).

In the test below, the operator is limited to performing only one primitive at a time. The results in the Task Begin/End excerpt below clearly demonstrate that the primitive with the highest interruption cost is always performed first (Table 21). The workload timeline output confirms this as well (Figure 132).

Table 21. Code output for test case 2 for task duration; threshold = total, WM true, threshold for V is 10, CV is 6.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	end	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	12	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	12	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	18	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	18	descent	Captain	start	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Cost = NONE
1	23.9	descent	Captain	end	48_172	Text string with exactly 100 chars - should take 6 seconds to read	Cost = NONE

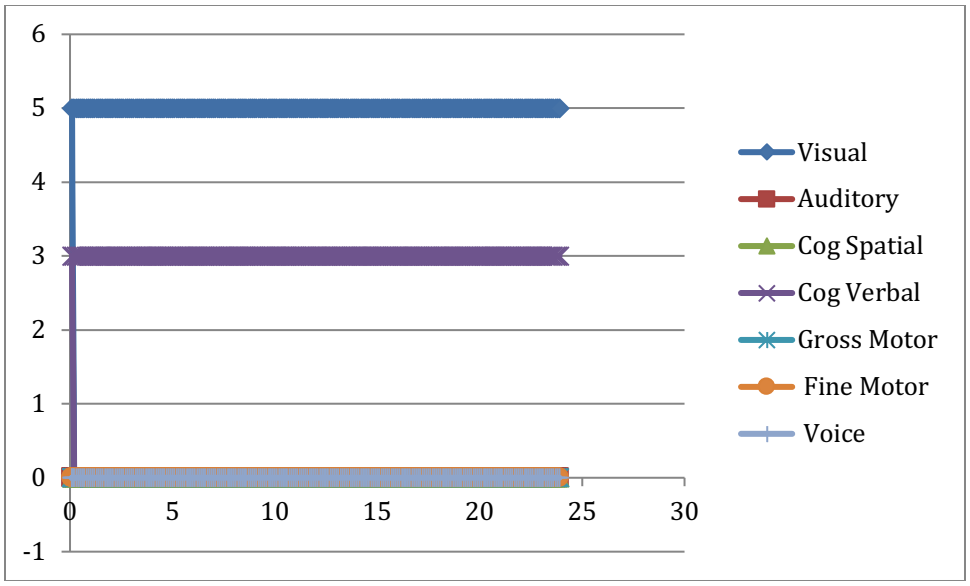


Figure 132. Workload timeline output for test case 2 cost of interruption; threshold = total, WM true, threshold for V is 10, CV is 6.

Test Case 3: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0), One Task Executes while Two Follow in Overload

Workload Management Flag set to **True** (Nonessential data hidden).

In this test, the operator is allowed to perform at most one primitive at a time without being in overload using the task network illustrated in Figure 133. This is demonstrated in the test below by the fact that no two primitives overlap in Clock time (Table 22). This test also demonstrates that the Interruption Cost strategy operate as expected in the event the tasks are staggered in time due to the fact that both primitives with cost set to High are executed before the primitive with cost set to Low (Figure 134).

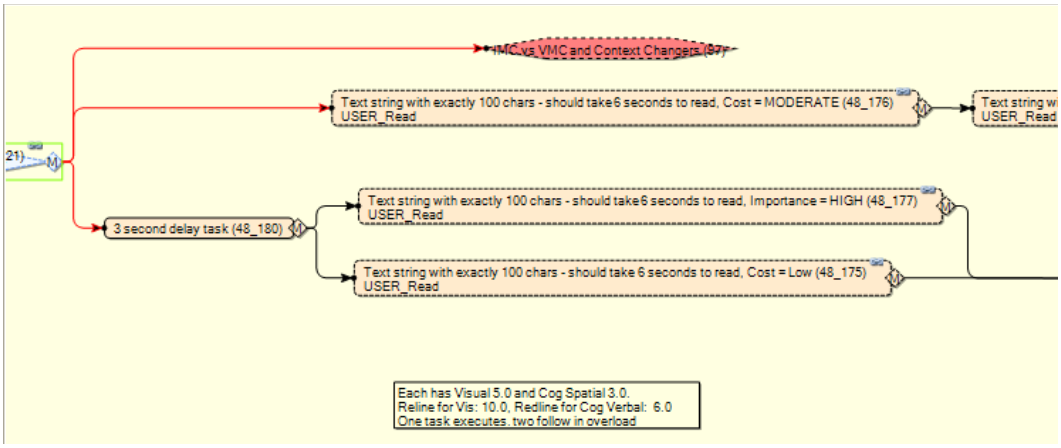


Figure 133. Task network for test case 3 for cost of interruption; threshold = total, one task executes while two follow in overload, WM true.

Table 22. Code output for test case 3 for cost of interruption; threshold = total, WM true, threshold for V is 10, CV is 6.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	6	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	6	descent	Captain	start	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	12	descent	Captain	end	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	12	descent	Captain	start	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	18	descent	Captain	end	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	18	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	23.9	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low

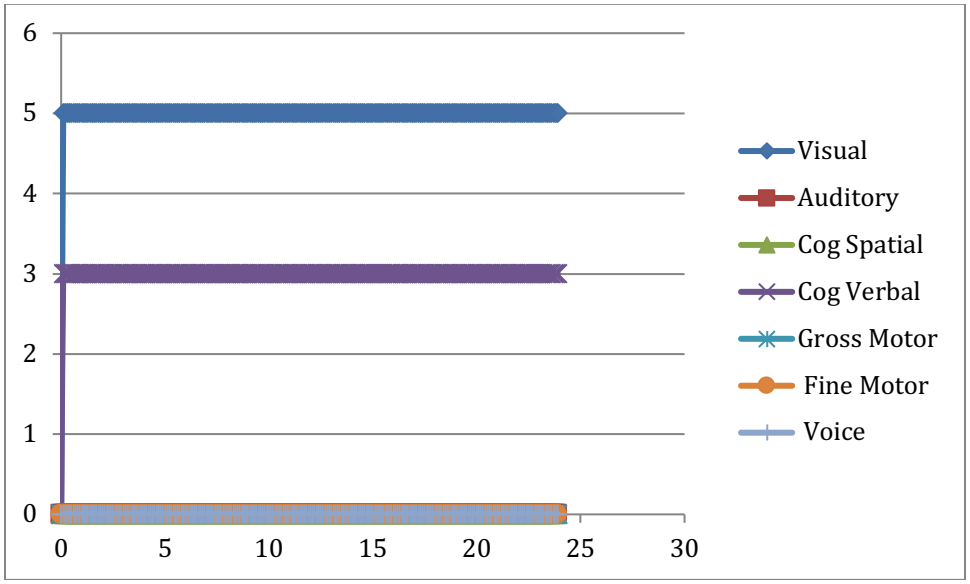


Figure 134. Workload timeline output for test case 3 for cost of interruption; threshold = total, one task executes while two follow in overload, WM true.

Test Case 4: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0), All Strategies Weighted

Workload Management Flag set to **True** (Non-transition times hidden).

In this test, the operator is allowed to perform at most one primitive at a time without being in overload. The workload management settings can be found in Figure 135. Figure 107 shows the task network used. In the current test no two primitives overlap in Clock time (Table 23). This test also demonstrates that the highest priority strategy, in this case Interruption Cost, still works as expected when all other strategies are weighted (Figure 136). The corresponding workload timeline output can be found in Figure 137.

Workload Management Settings	
Priority Weighting Duration	2
Priority Weighting Importance	1
Priority Weighting Interrupt Cost	4
Priority Weighting Urgency	3
Redline Auditory	4
Redline Cognitive Spatial	4
Redline Cognitive Verbal	6
Redline Fine Motor	4
Redline Gross Motor	4
Redline Visual	10
Redline Vocal	4

Figure 135. Workload management settings window for Threshold = total (Visual: 10.0, Cog Verbal: 6.0),

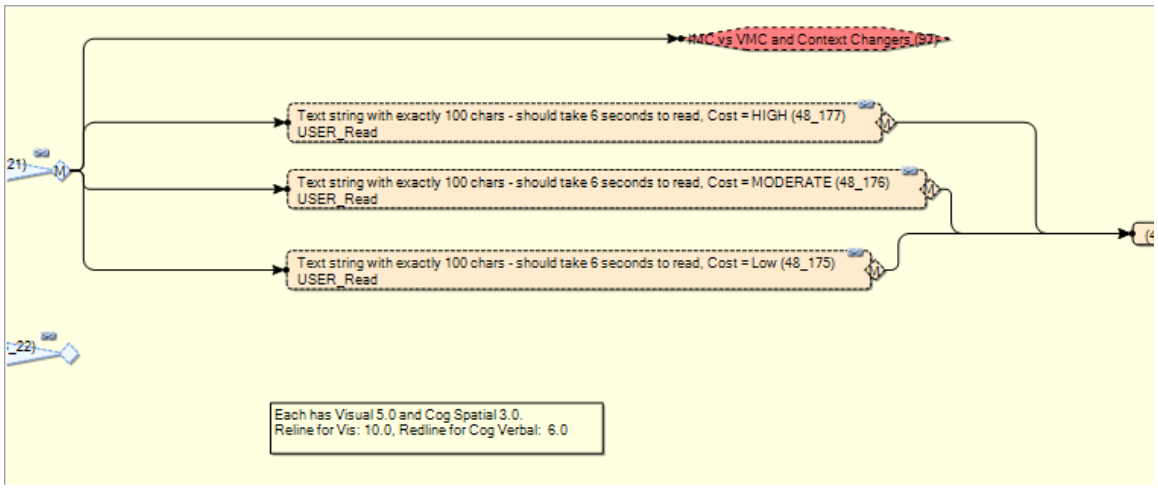


Figure 136. Task network for test case 4 for cost of interruption; threshold = total, WM true (Visual: 10.0, Cog Verbal: 6.0).

Table 23. Code output for test case 4 for cost of interruption; threshold = total, WM true (Visual: 10.0, Cog Verbal: 6.0).

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Cost = HIGH
1	6	descent	Captain	end	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Cost = HIGH
1	6	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	12	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	12	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	18	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low

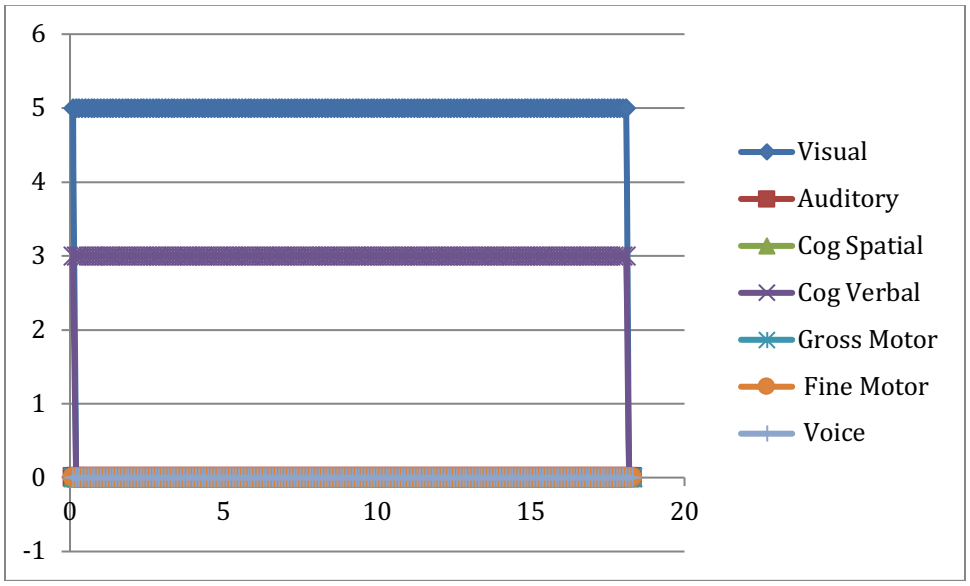


Figure 137. Workload timeline output for test case 4 cost of interruption; threshold = total, WM true (Visual: 10.0, Cog Verbal: 6.0).

Test Case 5: Threshold Greater Than total (Visual: 11.0, Cog Verbal: 7.0), All Strategies Weighted
Workload Management Flag set to **True** (Non-transition times hidden).

In this test, the operator is allowed to perform up to two tasks simultaneously without being in overload. The Task Begin/End report excerpt below demonstrates the Cost of Interruption strategy in use by the fact that the primitive with the lowest cost value, 48_175, is held back while the first two primitives execute (Table 24). This test also demonstrates that this strategy still works as expected when all other strategies are weighted (Figure 138).

Table 24. Code output for test case 5 for cost of interruption; threshold > total, WM true (Visual: 11.0, Cog Verbal: 7.0).

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Cost = HIGH
1	0	descent	Captain	start	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	6	descent	Captain	end	48_177	Text string with exactly 100 chars - should take 6 seconds to read	Cost = HIGH
1	6	descent	Captain	end	48_176	Text string with exactly 100 chars - should take 6 seconds to read	Cost = MODERATE
1	6	descent	Captain	start	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low
1	12	descent	Captain	end	48_175	Text string with exactly 100 chars - should take 6 seconds to read	Cost = Low

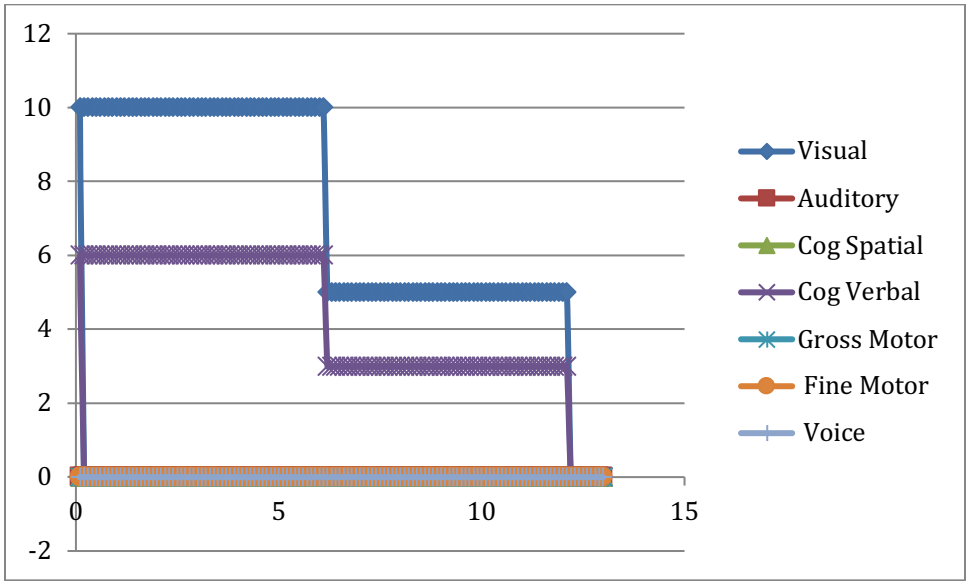


Figure 138. Workload timeline output for test case 5 cost of interruption; threshold = total, WM true.

4. Task Urgency/Delay Test

The Task Urgency Test uses several instances of the same exact User_Read primitive. All instances are identical in Importance, Duration and Cost Interruption. This primitive involves workload in the Visual and Cognitive Verbal Channels. By itself a single instance of this primitive contributes a visual workload value of 5.0 and a cognitive verbal workload value of 3.0. Each additional instance of the primitive adds the same amount of workload.

In Figure 139, all three primitives start at the exact same time and take the exact same amount of time: (0.06) x (number of characters in the string), which in this case is 0.06 x 100 = 6 seconds.

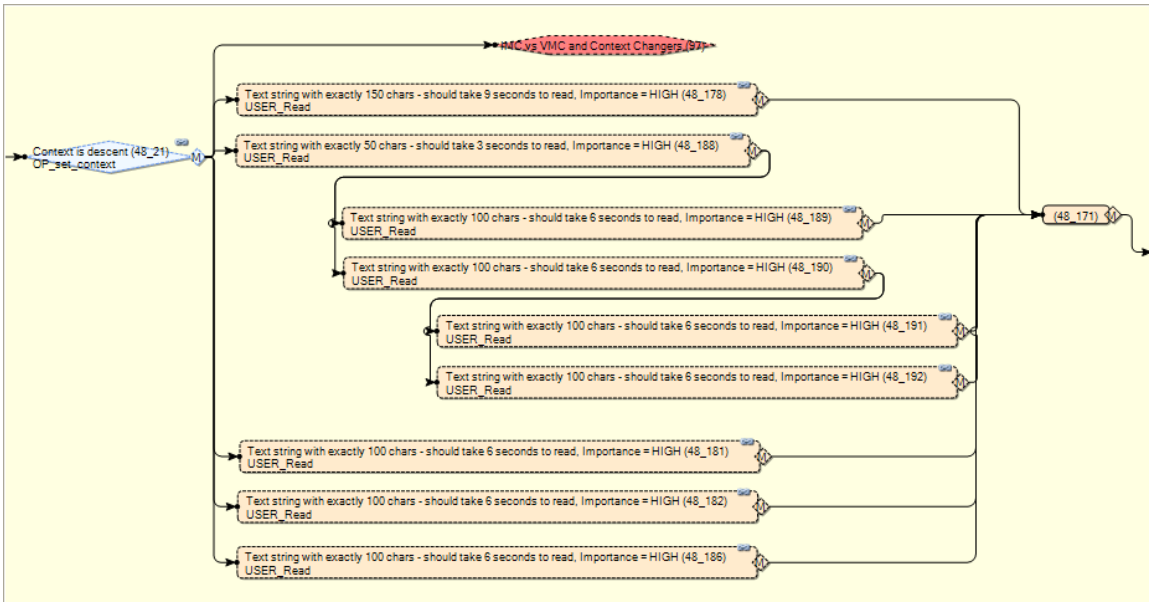


Figure 139. Task network for task urgency test.

Each primitive is linked to a specific SEEV Start task whose **Cost of Interruption** and **Importance** values are then inherited (Figure 140). The linked task is listed in the midasTask field for each primitive:

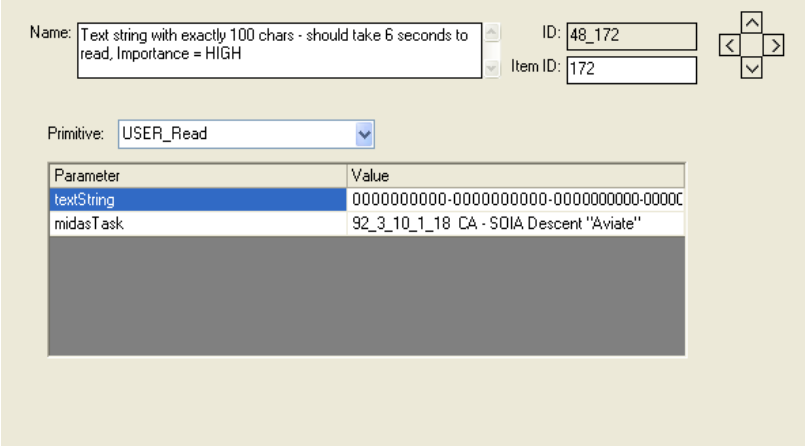


Figure 140. Step to link SEEV start primitive to MIDAS primitive, parameter input.

The SEEV Tasks referenced in the primitives above may be found in the Captain's SEEV Settings for the Descent context, SOIA Scenario and under IMC Conditions (Figure 141):

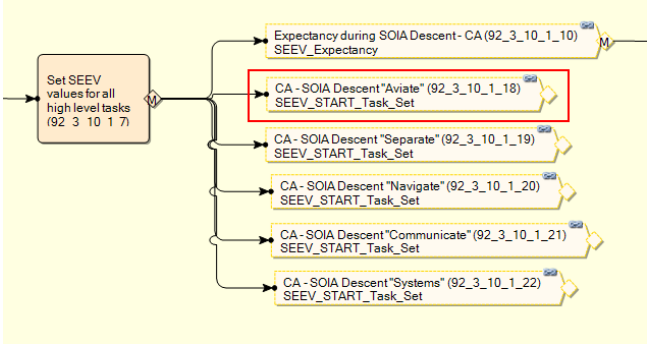


Figure 141. Task network example of SEEV start tasks.

In this test, the value of the **Interruption Cost** and **Task Importance** are held constant for all instances of the primitive (Figure 142).

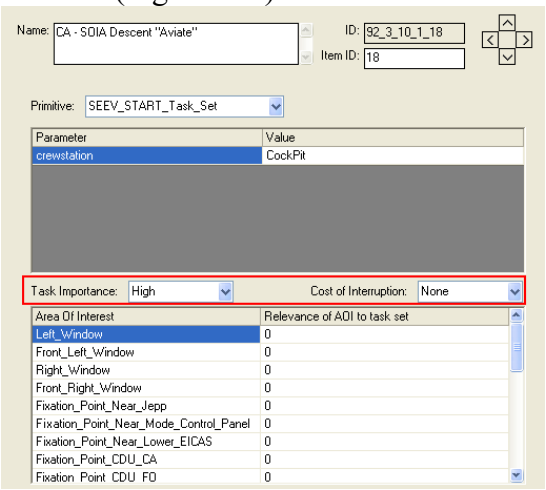
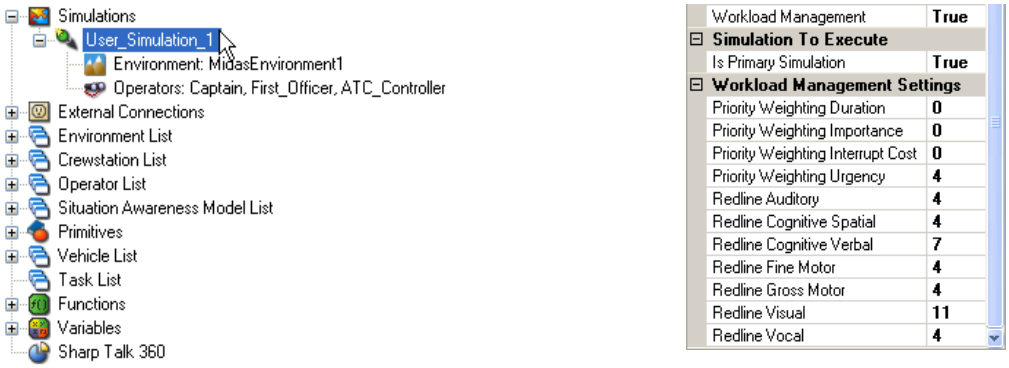


Figure 142. Task importance flag and cost of interruption settings.

Next, the prioritization of the Workload/Task Management Strategies were set through the User Simulation Settings in the MIDAS tree (Figure 143a) and the Properties for the Workload Management (Figure 145b).



(a) (b)

Figure 143. (a) Workload management settings window for task urgency and (b) workload management values.

- **Priority Weighting Urgency:** set to the highest value of 4 in order that the **Urgency** determines the priority of the next primitive to occur.
- **Redline Visual:** depending on the test being run, this value is set to 10 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 11 to allow a maximum of two User_Read primitives to happen simultaneously.
- **Redline Cognitive Verbal:** depending on the test being run, this value is set to 6 in order to cause an overload condition when two or more of the User_Read primitives are encountered simultaneously or to 7 to allow a maximum of two User_Read primitives to happen simultaneously.

Results:

Test Case 1: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)

Workload Management Flag set to **False**[3] (Nonessential data rows hidden).

The task network model for the Test Case 1 can be found in Figure 144. In the absence of workload strategy, all simultaneously encountered tasks are taken on (Table 24). Workload well exceeds the 11.0 and 7.0 redlines (Figure 145).

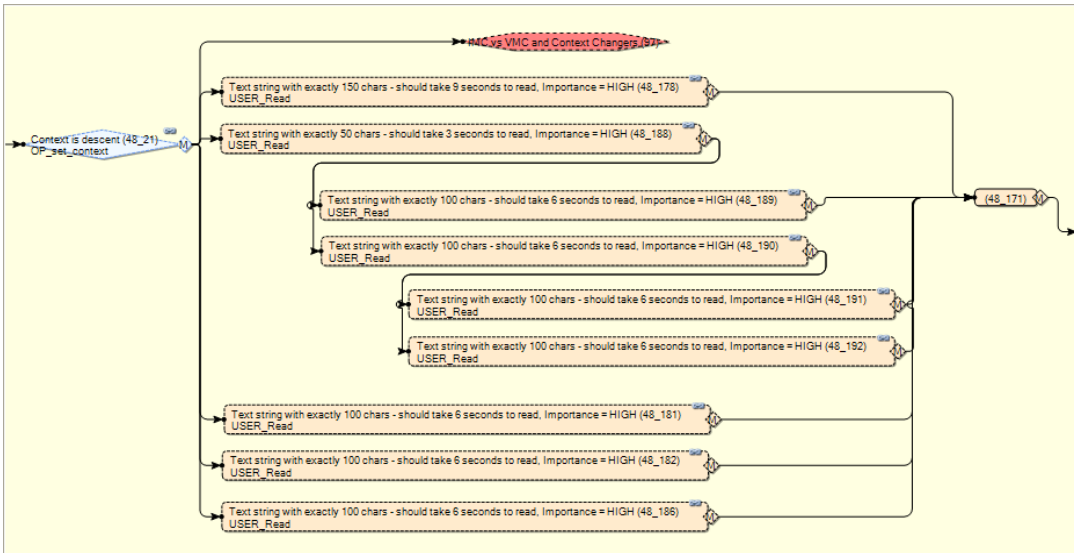


Figure 144. Task network for test case 1 for task urgency; threshold > total, WM false (Visual: 11.0, Cog Verbal: 7.0).

[3] This is a baseline run to test the absence of workload management strategies

Table 25. Code output for test case 1 for task urgency; threshold > total, WM false (Visual: 11.0, Cog Verbal: 7.0).

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	0	descent	Captain	start	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	0	descent	Captain	start	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	0	descent	Captain	start	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	0	descent	Captain	start	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	2.9	descent	Captain	end	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	2.9	descent	Captain	start	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	2.9	descent	Captain	start	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	end	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	end	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	end	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	end	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	end	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	start	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	start	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9.7	descent	Captain	end	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	15	descent	Captain	end	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	15	descent	Captain	end	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH

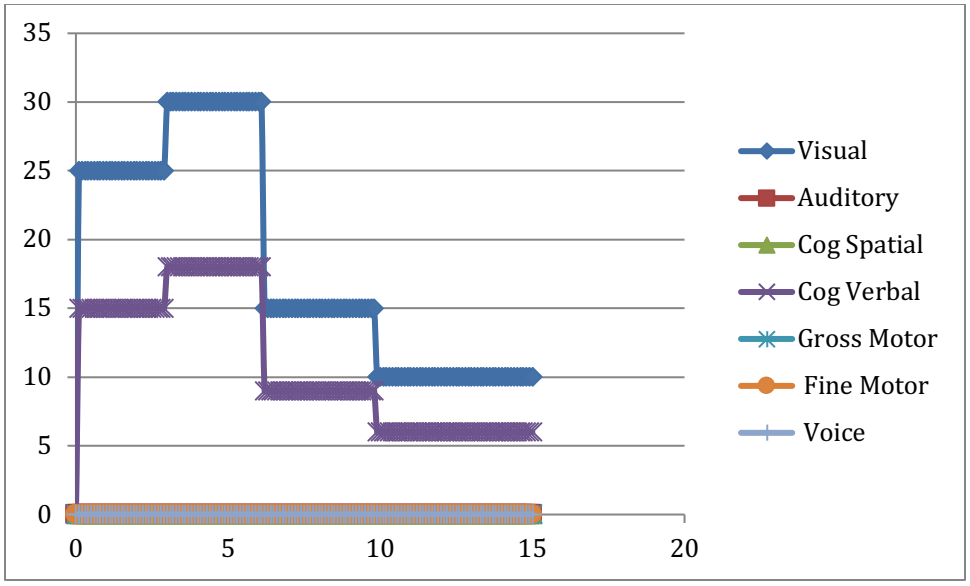


Figure 145. Workload timeline output for test case 1b task urgency; threshold > total, WM false (Visual: 11.0, Cog Verbal: 7.0).

Test Case 1b: Threshold Greater than total (Visual: 11.0, Cog Verbal: 7.0)
Workload Management Flag set to **True** (Nonessential data hidden).

The measure of interest in the following test is termed the delay time measure. The delay time measure indicates the point at which the delay time counter starts. Early delay times will be started before later delay times. In the test below, the operator is allowed to perform up to two primitives simultaneously (See Table 26). The primitives 48_181, 48_182 and 48_186 are clearly given priority over tasks 48_189, 48_190, 48_191 and 48_192. This is shown by the *earlier* onset of the "delay time" measure. The impact on the workload model can be found in Figure 146.

Table 26. Code output for test case 1b for task urgency; threshold > total, WM true (Visual: 11.0, Cog Verbal: 7.0).

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	0	descent	Captain	start	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	2.9	descent	Captain	end	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	2.9	descent	Captain	start	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	end	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	start	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9.7	descent	Captain	end	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	9.7	descent	Captain	start	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	15	descent	Captain	end	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	15	descent	Captain	start	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	15.7	descent	Captain	end	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	15.7	descent	Captain	start	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	20.9	descent	Captain	end	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	20.9	descent	Captain	start	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	21.7	descent	Captain	end	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	21.7	descent	Captain	start	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	26.9	descent	Captain	end	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	27.7	descent	Captain	end	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH

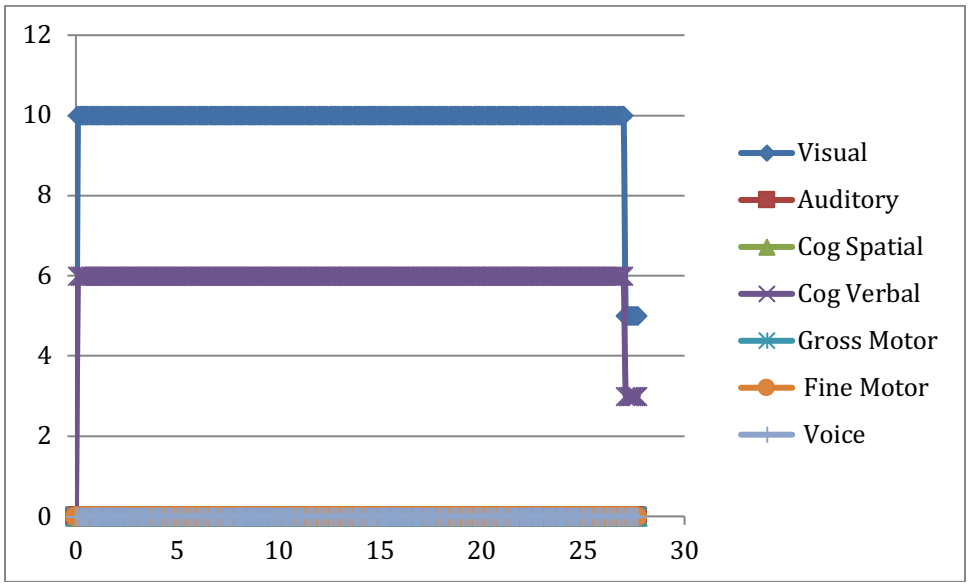


Figure 146. Workload timeline output for test case 1b task urgency; threshold > total, WM true (Visual: 11.0, Cog Verbal: 7.0).

Test Case 2: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0)
Workload Management Flag set to **True** (nonessential data hidden).

The measure of interest in the following test is termed the delay time measure. The delay time measure indicates the point at which the delay time counter starts. Early delay times will be started before later delay times. In this test, the operator is allowed to perform up to one primitive at a time. Once again primitives 48_181, 48_182 and 48_186 are clearly given priority over tasks 48_189, 48_190, 48_191 and 48_192 due to their early delay (see Table 23). This is shown by the *earlier* onset of the "delay time" measure. The workload timeline produces the expected constant workload (Figure 147).

Table 27. Code output for test case 2 for task urgency; threshold = total, WM true (Visual: 10.0, Cog Verbal: 6.0).

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name		
1	0	descent	Captain	start	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH	
1	2.9	descent	Captain	end	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH	
1	2.9	descent	Captain	start	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH	
1	12.7	descent	Captain	end	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH	
1	12.7	descent	Captain	start	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	18.7	descent	Captain	end	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	18.7	descent	Captain	start	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	24.7	descent	Captain	end	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	24.7	descent	Captain	start	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	30.7	descent	Captain	end	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	30.7	descent	Captain	start	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	36.7	descent	Captain	end	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	36.7	descent	Captain	start	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	42.7	descent	Captain	end	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	42.7	descent	Captain	start	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	48.7	descent	Captain	end	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	48.7	descent	Captain	start	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	
1	54.7	descent	Captain	end	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH	

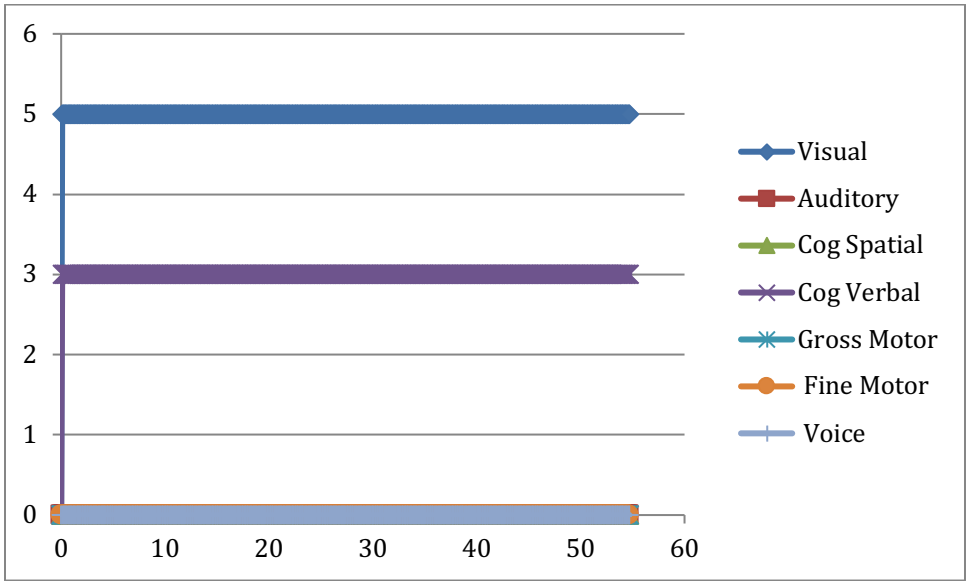


Figure 147. Workload timeline output for test case 2 task urgency; threshold = total, WM true (Visual: 10.0, Cog Verbal: 6.0).

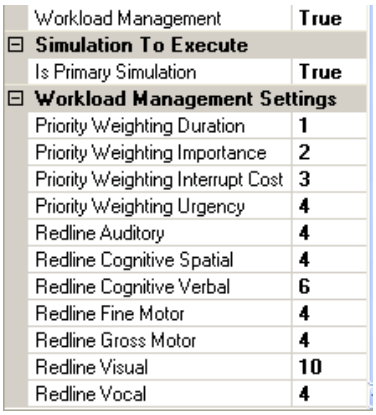
Test Case 3: the corresponding Test 3 scenario for the Urgency strategy has been omitted due to the fact that Urgency already requires a task in progress.

Test Case 4: Threshold Equal to total (Visual: 10.0, Cog Verbal: 6.0), All Strategies Weighted

Workload Management Flag set to **True** (Non-transition times hidden).

In test case 4 whose settings can be found in Figure 148, the operator is allowed to perform at most one primitive at a time without being in overload. This is demonstrated in the test below by the fact that no two primitives overlap in Clock time (Table 28). As with the earlier tests in this section, the measure of interest in the following test is termed the delay time measure. The delay time measure indicates the point at which the delay

time counter starts. Early delay times will be started before later delay times. The Urgency priority is clearly demonstrated by the fact that primitives 48_181, 48_182 and 48_186 are given priority over tasks 48_189, 48_190, 48_191 and 48_192. This test also demonstrates that the highest priority strategy, in this case Urgency, still works as expected when all other strategies are weighted. This is shown by the *earlier* onset of the "delay time" measure (see Table 28). The workload timeline produces the expected constant workload (see Figure 149).



Workload Management	**True**
Simulation To Execute	
Is Primary Simulation	**True**
Workload Management Settings	
Priority Weighting Duration	**1**
Priority Weighting Importance	**2**
Priority Weighting Interrupt Cost	**3**
Priority Weighting Urgency	**4**
Redline Auditory	**4**
Redline Cognitive Spatial	**4**
Redline Cognitive Verbal	**6**
Redline Fine Motor	**4**
Redline Gross Motor	**4**
Redline Visual	**10**
Redline Vocal	**4**

Figure 148. Workload management settings window.

Table 28. Code output for test case 4 for task urgency; threshold = total, WM true (Visual: 10.0, Cog Verbal: 6.0).

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	2.9	descent	Captain	end	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	2.9	descent	Captain	start	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	end	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	start	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	15	descent	Captain	end	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	15	descent	Captain	start	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	20.9	descent	Captain	end	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	20.9	descent	Captain	start	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	30.7	descent	Captain	end	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	30.7	descent	Captain	start	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	36.7	descent	Captain	end	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	36.7	descent	Captain	start	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	42.7	descent	Captain	end	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	42.7	descent	Captain	start	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	48.7	descent	Captain	end	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	48.7	descent	Captain	start	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	54.7	descent	Captain	end	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH

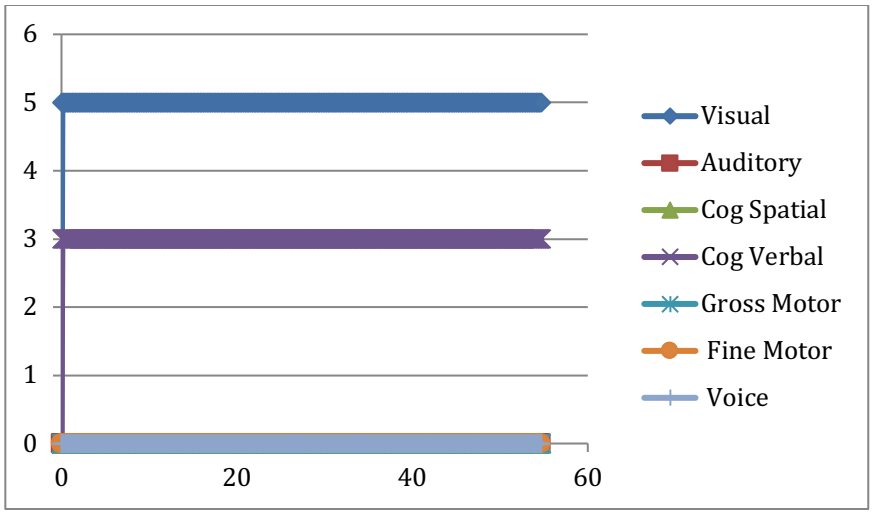


Figure 149. Workload timeline output for test case 4 for task urgency; threshold = total, WM true, threshold for V is 10, CV is 6.

Test Case 5: Threshold Greater Than total (Visual: 11.0, Cog Verbal: 7.0), All Strategies Weighted

Workload Management Flag set to **True** (Non-transition times hidden).

As with the earlier tests, the measure of interest in the following test is termed the delay time measure. The delay time measure indicates the point at which the delay time counter starts. Early delay times will be started before later delay times. In this test, the operator is allowed to perform up to two primitives at a time. Primitives 48_181, 48_182 and 48_186 are given priority over tasks 48_189, 48_190, 48_191 and 48_192. This is shown by the *earlier* onset of the "delay time" measure (as seen in Table 29). The addition of other weights to other strategies has no impact on this model. The workload timeline output can be found in Figure 150, and illustrates the expected drop given the thresholds in case 5.

Table 29. Code output for test case 5 for task urgency; threshold > total, WM true threshold for v 11, CV is 7.

RunNumb	Time	Context	Operator	start/end	Task ID	Task Name	
1	0	descent	Captain	start	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	0	descent	Captain	start	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	2.9	descent	Captain	end	48_188	Text string with exactly 50 chars - should take 3 seconds to read	Importance = HIGH
1	2.9	descent	Captain	start	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	end	48_186	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	6	descent	Captain	start	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	end	48_182	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	9	descent	Captain	start	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	12	descent	Captain	end	48_181	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	12	descent	Captain	start	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	18	descent	Captain	end	48_190	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	18	descent	Captain	start	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	18.7	descent	Captain	end	48_178	Text string with exactly 150 chars - should take 9 seconds to read	Importance = HIGH
1	18.7	descent	Captain	start	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	23.9	descent	Captain	end	48_189	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	23.9	descent	Captain	start	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	24.7	descent	Captain	end	48_191	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH
1	29.9	descent	Captain	end	48_192	Text string with exactly 100 chars - should take 6 seconds to read	Importance = HIGH

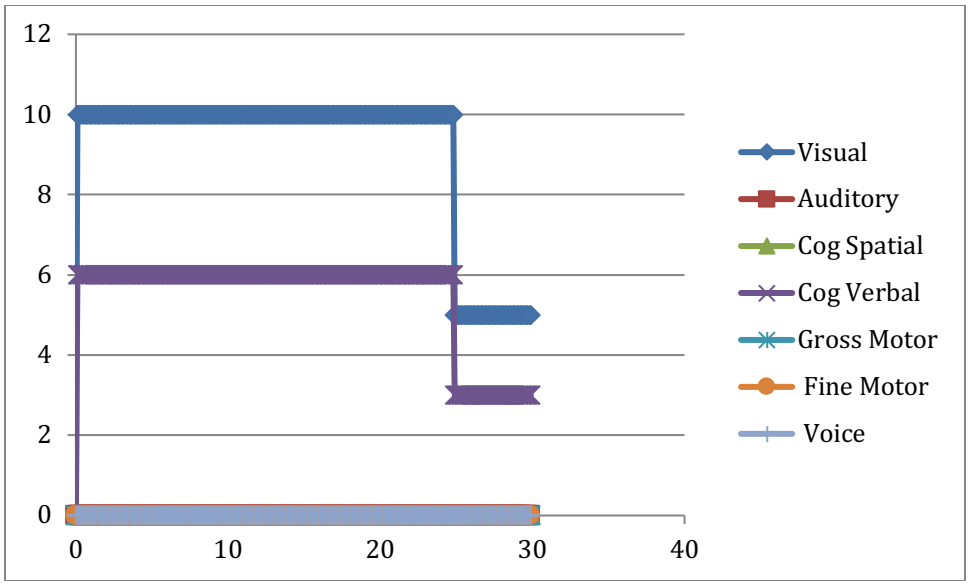


Figure 150. Workload timeline output for test case 5 for task duration; threshold > total, WM true. threshold for v 11, CV is 7.

Discussion and Conclusion

This document explains the rationale used to generate a computational model of operator workload for MIDAS v5 and to verify its performance. As outlined in the introduction, MIDAS possesses two distinct approaches to represent workload and its impact on operator performance both of which rely on the Multiple Resource Theory (MRT) operating behind the scenes to impact performance. The first step was to verify the operation of the MRT and the conflict matrix. This was completed in the first test that illustrated the workload spikes in performance that exceeded the threshold of operator performance. This baseline output was then used as the benchmark to examine the impact of the workload management model manipulations. The four workload management manipulations evaluated the sensitivity of the parameters that make up MIDAS's priority schedule, namely the task importance, urgency, duration, and the interrupt cost. Success of the model manipulations was revealed by (1) the difference in the workload timeline output when compared to the baseline workload output that contained no workload management model, and (2) the combination of the output string generated by the MIDAS BeginEnd output file and the channel-specific workload timeline. The workload management model output illustrated that the two measures (timeline and the task order) are needed to correctly evaluate the output from the MIDAS workload management model

References -SA

Ericsson, K. A. & KIintch, W. (1995). Long term working memory. Psychological Review, 102, 211-245.

Wickens, C.D. (2008). Situation awareness. Review of Mica Endsley's articles on situation awareness. Human Factors, Golden Anniversary Special Issue, 50, 397-403.

Durso, F., Rawson, K., & Girotto, S (2007). Comprehension and Situation Awareness. In F. Durso et al (Eds). Handbook of Applied Cognition. John Wiley & Sons.

Gugerty, L. (1998) Evidence from a partial report task of forgetting from dynamic spatial memory. Human Factors, 40, 498-508.

McCracken, J.H. & Aldrich, T.B. (1984). Analysis of Selected LHX Mission Functions: Implications for Operator Workload and System Automation Goals (Technical note ASI 479-024-84(b)). Anacapa Sciences, Inc.

Hamilton, D.B. Bierbaum, C.R. & Fulford, L.A. (1990). Task Analysis/Workload (TAWL) User's Guide Version 4.0. U.S. Army Research Institute, Aviation Research and Development Activity, Fort Rucker, AL: Anacapa Sciences, Inc.

Mitchell, D. K. (2000). Mental Workload and ARL Workload Modeling Tools. ARL-TN-161. Aberdeen Proving Ground, M.D., Army Research Laboratory.

Gore, B.F., Hooey, B.L., Wickens, C.D., & Scott-Nash, S. (2009). A computational implementation of a human attention guiding mechanism in MIDAS v5. In V.G. Duffy (ed.): Digital Human Modeling, HCII 2009, LNCS 5620, pp. 237-246, 2009.

Norman, D. A., & Bobrow, D. G. (1975). On data-limited and resource-limited processes. *Cognitive Psychology, 7,* 44-64.

Gore, B.F. (2011). Man-machine Integration Design and Analysis System (MIDAS) v5: augmentation, motivations, and directions for aeronautics applications. In P.C. Cacciabu, M. Hjalmdahl, A. Luedtke, and C. Roccioli (eds) *Human Modeling in Assisted Transportation 2011*, *207-213*, Heidleberg: Springer-Verlag.

References - Workload

Freed, M. (2000). *Reactive prioritization.* Paper presented at the 2nd NASA International Workshop on Planning and Scheduling in Space, San Francisco, CA.

Gore, B. F. (2010). Man-machine integration design and analysis system (MIDAS) v5: Augmentations, motivations, and directions for aeronautics applications. In P. C. Cacciabu, M. Hjalmdahl, A. Luedtke & C. Riccioli (Eds.), *Human Modelling in Assisted Transportation*. Heidelberg: Springer.

Gore, B. F., Hooey, B. L., Socash, C., Haan, N., Mahlsted, E., Bakowski, D. L., et al. (2011). Evaluating NextGen closely spaced parallel operations concepts with human performance models, *HCSL Technical Report (HCSL-11-01)*. Moffett Field, CA: NASA Ames Research Center.

Gore, B. F., Hooey, B. L., Wickens, C. D., & Scott-Nash, S. (2009). A computational implementation of a human attention guiding mechanism in MIDAS v5. In V. G. Duffy (Ed.), *Digital Human Modeling* (Vol. 5620/2009, pp. 237-246). Berlin / Heidelberg: Springer.

Gore, B. F., & Smith, J. D. (2006). Risk assessment and human performance modeling: the need for an integrated approach. *International Journal of Human Factors of Modeling and Simulation, 1*(1), 119-139.

Hamilton, D. B., & Bierbaum, C. R. (1992). *Operator Workload Predictions for the Revised AH-64A Workload Prediction Model: Volume I: Summary Report (No. AD-254 198)*. Alabama: Anacapa Sciences, Inc.

Hooey, B. L., Gore, B. F., Wickens, C. D., Salud, E., Scott-Nash, S., Socash, C., et al. (2010). *Modeling pilot situation awareness*. Paper presented at the Human Modelling of Assisted Technologies Workshop.

McCracken, J. H., & Aldrich, T. B. (1984). *Analysis of selected LHX mission functions: Implications for operator workload and system automation goals (Technical note ASI 479-024-84[b])*. Fort Rucker, AL: Anacapa Sciences, Inc.

Mitchell, D. K. (2000). *Mental workload and ARL workload modeling tools*. Aberdeen Proving Ground, MD: U.S. Army Research Laboratory.

Wickens, C. D. (1984). *The multiple resource model of human performance: Implications for display design*. Williamsburg, VA: AGARD/NATO Proceedings.

Wickens, C. D. (1991). Processing resources and attention. In D. Damos (Ed.), *Multiple-task performance* (pp. 3-34). London: Taylor & Francis.

Wickens, C. D. (2002). Multiple resources and performance prediction. *Theoretical Issues in Ergonomics Science, 3*(2), 158-177.

Wickens, C. D., & McCarley, J. S. (2008). *Applied Attention Theory*. Boca Raton, Fla: CRC Press, Taylor and Francis Group.

Chapter 5: The Simulation Scenario Being Modeled in this Training Effort

It is necessary to outline the simulation scenario that you will be building at the present time to give some of the instructions the proper context. The scenario you will be building is meant to exercise the MIDAS model development environment, the procedure development environment, and the Jack CAD simulation environment. Made up of a human operator in a control room, with 5 monitors and 3 wall screens that reflect the displays of the main monitors, a keyboard & 5 trackballs to interface with the monitors. You will be defining a simulation of a Process Control Room operator conducting an normal internal scan of some of the pieces of equipment in his world. Specifically, the Control Room Operator's goals are to:

1) Monitor the five displays and responding to both visual and auditory alerts in the simulation.
2) Begin with a nominal scan of the monitors
3) Respond to an auditory alarm by reaching the keypad with his left hand and acknowledging the alarm -- note this causes a change one of the monitors to display emergency information, which is also displayed on the wall screen.
4) Receive a request from his supervisor to check pressure values. He looks up with an emergency scan pattern and checks for a pressure value over 80. He comprehends the information displayed on the screen extracting required information. During his scan, the far left monitor gets into hibernation mode hence lowering his situation awareness and workload.

Setting up Jack™

Start Jack™

- MIDAS operates with Jack™ version 4.1 to 7.0
- Double click the Jack™ icon
- You should see an empty Jack™ environment

Create a Jack™ human

- Click on the Human menu
- Select Create -> Default Male
- Verify the male is now available in your Jack™ window

Load in the Jack™ objects

- Click on the File menu
- Click on Import
- Navigate to the directory containing the image objects
- Select the chair.pss image
- In the window that appears, click the Translate button
- A new window will open, wait until some text scrolls through it until you see "SUCCESS"
- Click the Close button to remove that new window that came up
- Verify the object is now available in your Jack™ window

- Repeat the process for the following images
  - far_right_monitor.fig
  - right_center_monitor.fig
  - primary_center_monitor.fig
  - left_center_monitor.fig
  - far_left_monitor.fig
  - Wall_Screen_Left.fig (this is a figure file that is sitting just on top of the Wall_Screen1_0
  - Wall_Screen_center.fig
  - Wall_Screen_center_right.fig
  - keyboard.fig
  - Wall.fig (this is the wall on the right side of the plane)
  - Wall1.fig (this is the wall on the left side of the plane)
  - Wall-Screen1.fig (this is the wall that has the 3 screens on it)
  - Wall0.fig (this is the ceiling)
  - trackpad_border_1_0.fig
  - trackball_1_0.fig
  - trackpad_border_1.fig
  - trackball_1.fig
  - trackpad_border_10.fig
  - trackball_1_1.fig
  - trackpad_border_10_1.fig
  - primary_trackball.fig
  - trackpad_border_10_0.fig
  - trackball_10.fig
  - office_chair.fig
  - joe.fig
  - office_table_large.fig
  - office_table_large0.fig
  - office_table_large1.fig

- Create a number of CAD objects in jack (for this step you will be creating a table with two monitors on its surface and these figure files will be located in front of the Jack™ anthropometrical figure that is sitting in the chair)
  - Go to >Object >> Create Figure from Library
  - Select the Furniture option
  - Load the office-table-large.pss file.
    - Move the table and place it at –15.75, 0, -22.16
  - Load a monitor from the furniture option (select the name of the figure file and press the load icon)
    - Select the object with the finger toolbar
    - This will load a piece of furniture and will locate it at the 0,0,0 location with 0,0,0 rotation.
    - Place the monitor on the left side of the large table
    - E.g. the monitor should be at –3, 66.1, 58.73 with rotation of 0, 90.2, 0.

        - Save the scene file
    - Attach the monitor to the table
        - Move your mouse over the monitor, right click, select the "Attach" option
        - Unattach the monitor from the world, attach the monitor to the table
        - the monitor to the table
        - Repeat this step
    - Add another monitor to the right of the first monitor you placed on the table
        - Complete the same steps as above but place the monitor at -1.31, 67.38, -8.1, rotation 0,-90.2,0
        - Attach the monitor to the table as completed above
    - NOTE – attaching the monitor to the table will allow you to move the table with the monitors on its surface as one unit.
    - Create a second large table and place this on the right side of the first table, at an angle of 16 degrees
    - Place one monitor in the center of this table
    - Add a third table, this one to the right of the second table at an angle of –36 degrees
        - This will serve to make a bank of tables with a bank of computer terminals which the jack operator will be set to interact with.

- Load a number of *.pss files into the Jack™ environment.
    - Select the import option from the File pull-down menu in Jack
    - Import the relevant jack *.pss file.
    - See Figure 151.

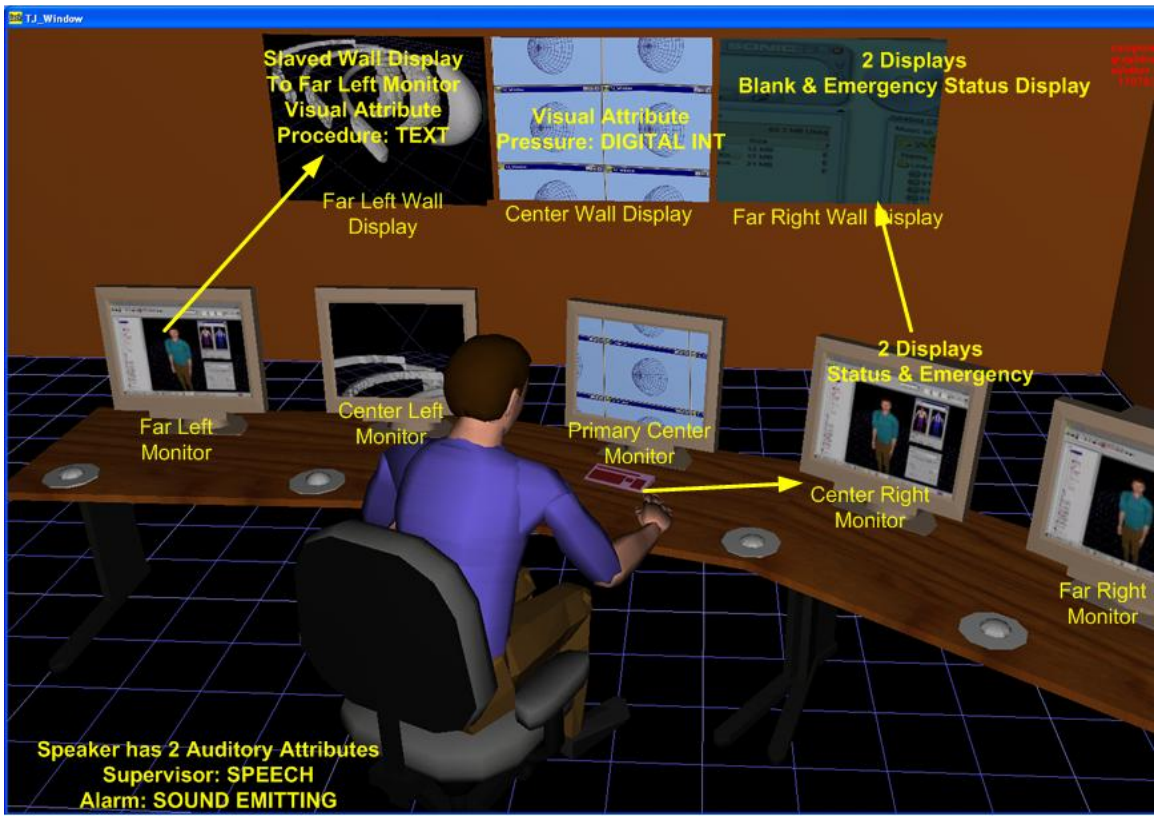


Figure 151. Jack and the environment being modeled in the training scenario.

Position the Jack™ objects

- Create a CAD object using the Jack™ software
- Move the chair to location cm = 107.52, -2.75, -40.17; .deg = 2 155.4, -0.1
- Sit Jack™ in the chair – use one of the postures that are contained within the jack software
    - Step 1 create a jack™ figure
    - Save the figure file as a new name – change it from “Human” to “Joe”
    - Hover your mouse over the mannequin and right click. This will bring up a series of actions possible on the mannequin. Move your mouse to the postures choice and select a sitting posture for “Joe”
    - Move Joe (the seated posture of the jack™ human figure) to the desired location
    - Edit the posture by selecting the human control/manipulation option in jack to position the posture given the other CAD objects in the environment
    - Save the posture under a new name (e.g. “Joe_4_training)
- Move the primary_center_monitor in front of Jack™, to his right (get some relative X, Y, Z coordinate – the training simulation has this monitor placed at 10.56, 65.85, -82.26; 0, -110.2, 0)
- Move the active monitor to the same location as the blank monitor
- Set the blank and active monitors to active and inactive

- Move the center_left_monitor in front of Jack™, to his left (the training simulation has –1.31, 67.38, -8.1; deg 0, -90.1, 0)
- Move the center_right_monitor in front of Jack™, to his right (the training simulation has 43..73, 65.85, -148.78; deg 0, -124.9, 0)
- Move the far_left_monitor in front of Jack™, to his left (the training simulation has –3.00, 66.1, 58.73; deg 0, -90.2, 0)
- Move the far_right_monitor in front of Jack™, to his right (the training simulation has 88.59, 67.58, -198.48; deg 5.2, -134.1, 3.6)
- Move the keyboard below the main monitor (29.22, 65.93, -74.75; deg 0, -114.5, 0)

Attach the Jack™ objects

- Attach steps
- Identify the figures that you would like to attach together
- Upon initially creating a figure, it is attached to the world database. You need to un-attach the figure from the world database and attach it to the desired figure file and most importantly, the site location of the figure file. For example, if you are trying to attach jack™'s hand to a hammer, you need to right mouse click on the figure of interest (in this case the hammer), specify attach. This will bring up a properties window of the hammer and this will note whether the figure is attached to anything.
- select SITE in the pull down menu in the jack software, select the finger tool of the jack software, navigate to the jack™'s hand (specifically the palm), click on this location and specify "attach". This will bring up a window that should allow you to select the one
- Use the finger tool to select the figure of interest. This will highlight the figure and will list the XYZ location in centimeters along the toolbar. These are the values that need to go into MIDAS so that MIDAS has an awareness of the end effectors of jack™'s objects and relative locations (the crewstation import function in MIDAS should bring all of the XYZ's over from Jack but in the event that the simulation fails in some regard, double check the XYZ locations.

Create a camera

- Create an object to attach camera to
    - Go through create object steps
    - Position the object
    - Rotate the object
- In the View Control window, click on the hand icon that's to the right of the Attach Object textbox
- Click on the camera object created above
- Note: The attach point is not saved, so this section will have to be repeated each time the environment is loaded

Create a Texture object in Jack™

- Details associated with creating textures will come when the Jack software is able to better handle texture swapping.

Now that you have the operational environment defined and a rough idea of the look of the simulation scenario and the components in the scenario, you can start to define the elements and their interaction on the MIDAS side.

Setting up the MIDAS Crew Station List

- Right Click, on CrewStation List in the Tree View and select Add Crewstation from the context menu (Figure 152).
    - Select the newly created Crewstation named Crewstation 1
    - Type “Control Room” in the Name field

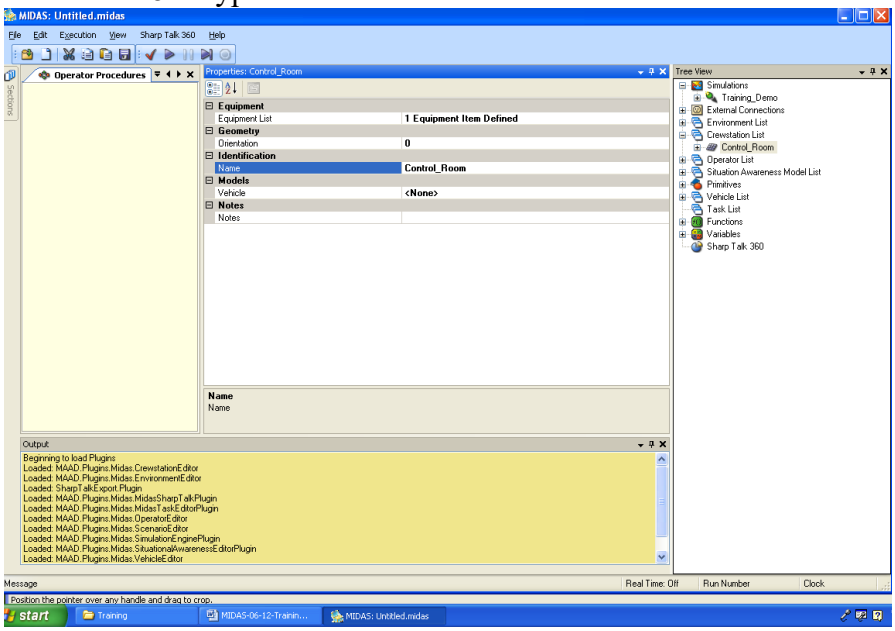


Figure 152. Setting up the crewstation list.

- The “Control Room” automatically adds a Default component into its list, select the + sign to expose the default component (see Figure 153).

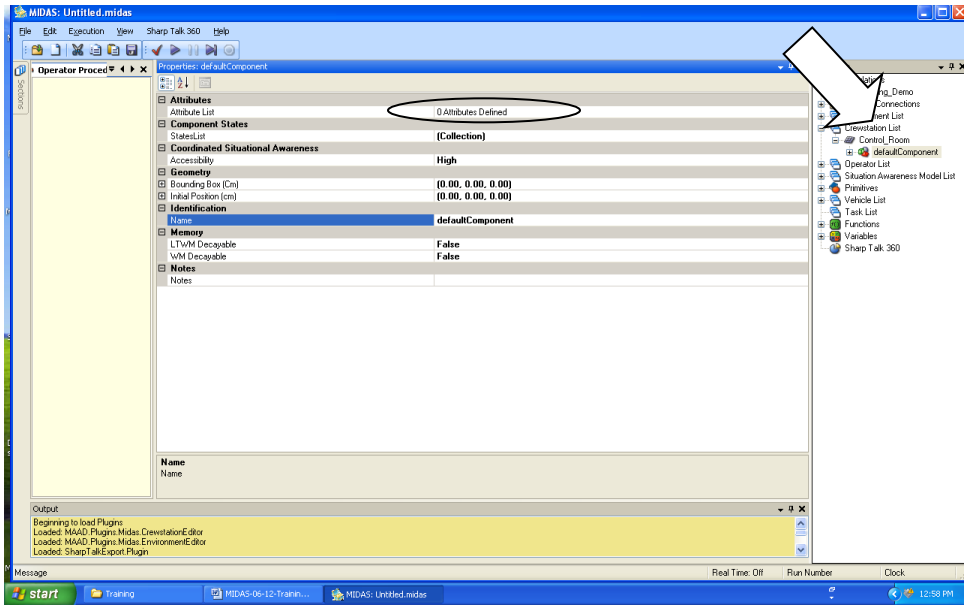


Figure 153. Create component states.

- Type “keyboard” in the Name Field
- Right click the Component States ‘…’ button to set the discrete state component definitions (see Figure 154).
- Right click on keyboard. This defines the states, attributes associated with the keyboard (Figure 154).

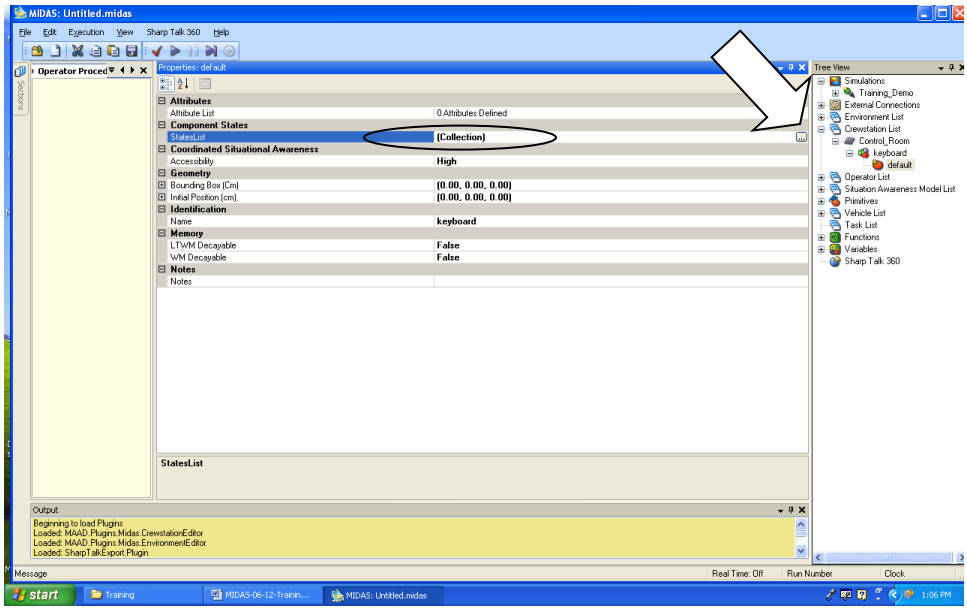


Figure 154. Define component states.

- Right Click on Control Room, select Add Component,
  - Add Discrete State Component (DSC)
  - Type “Primary Center Monitor” in the Name field (see Figure 155).
  - Add a state onto the primary center monitor (see Figure 156).

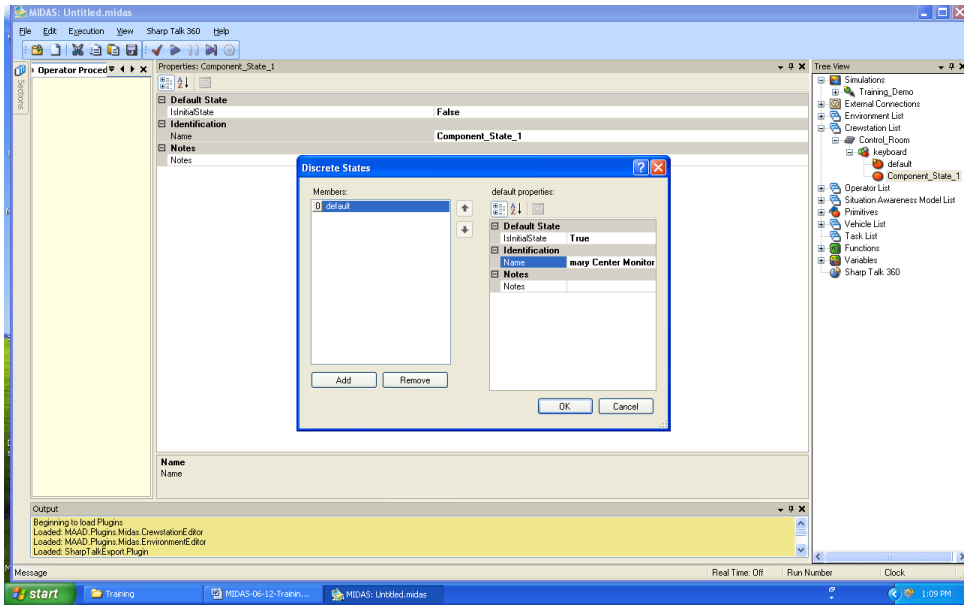


Figure 155. Add primary center monitor component in the control room.

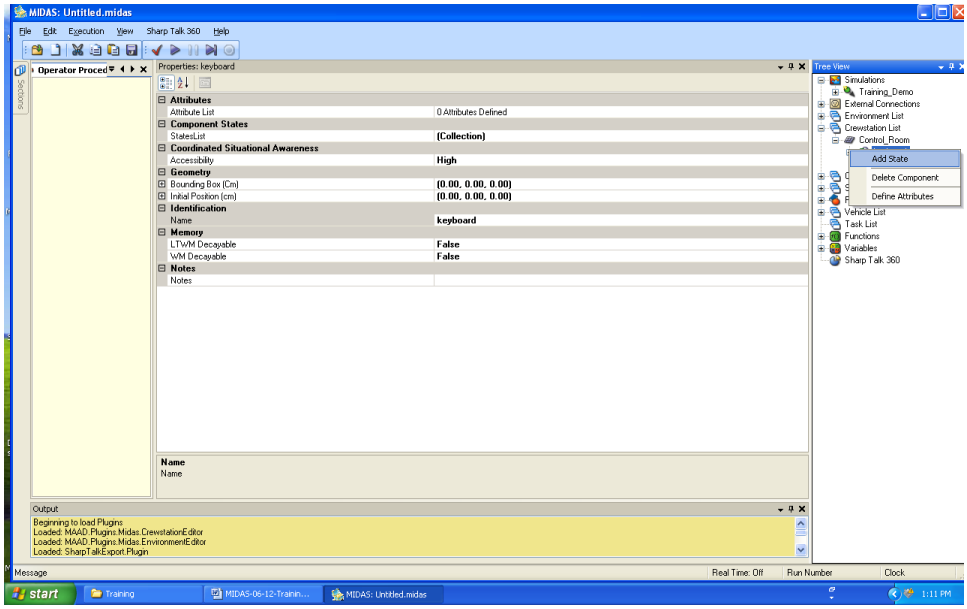


Figure 156. Add state onto the primary center monitor in the control room.

Right click to define the attribute (see Figure 157) below.

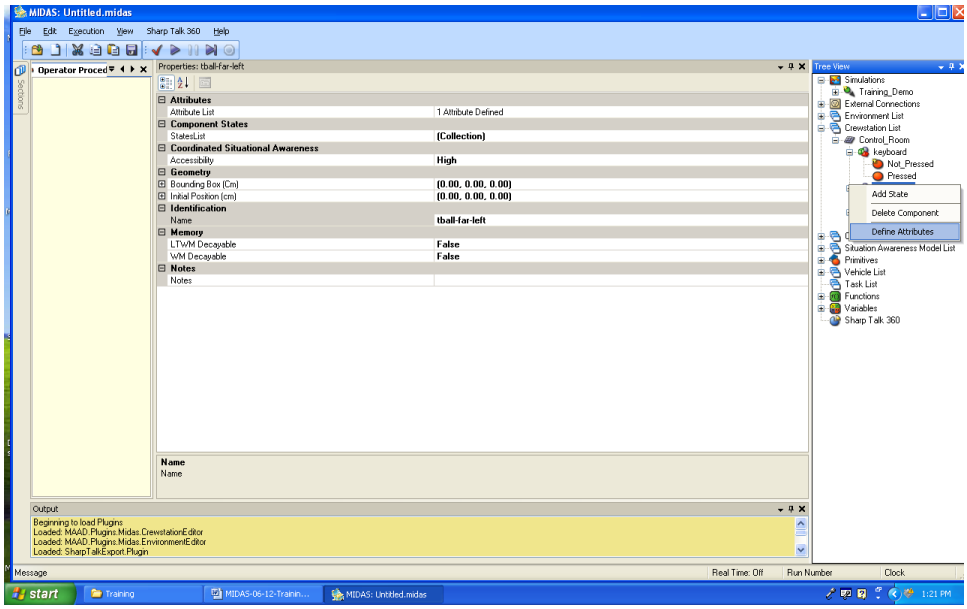


Figure 157. Define attributes on the primary center monitor.

NOTE: You will need to create a monitor that looks for the display attribute to change. To do this, you will be using the scanning tasks that will cycle through all of the attributes to determine their states. When all is normal, then it will just continue to scan. If it is in an emergency state, then the scan will go to the emergency scan.

- Right Click on Control Room, select Add Attribute on the DSC, and add a routing /repeating task on the attribute to update it. You create 2 primitives and 1 logic piece to the mini-network (this mini-network will cycle upon itself continually).

o Add DSC Component, add attribute onto this DSC
o Name the attribute tball-far-left and define as being a rotational component with spatial symbolic properties (see Figure 158).

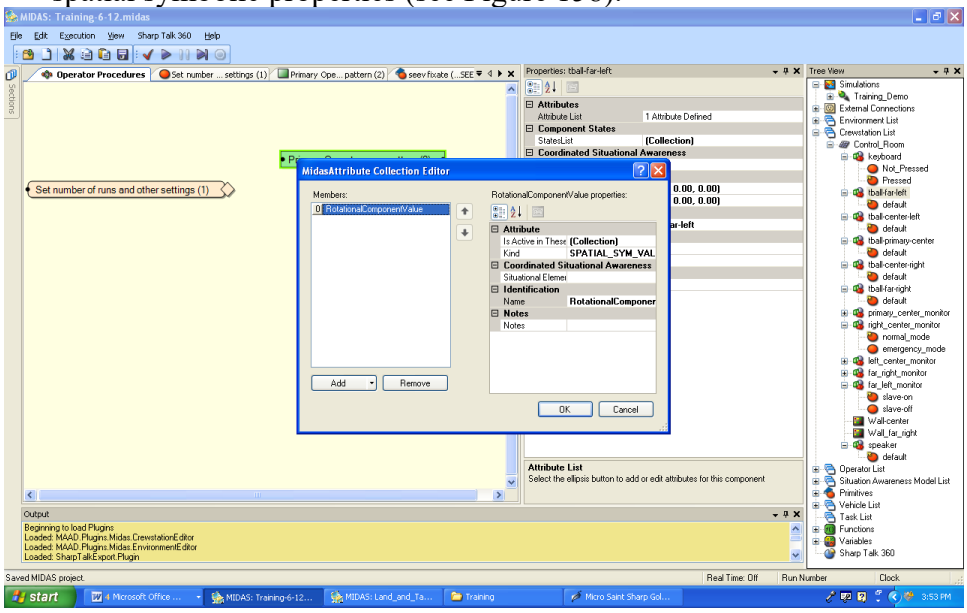


Figure 158. Defining the trackball entry device for the scenario.

- Repeat for tball-center-right, tball-primary-center, tball-center-right, tball-far-right

- Right Click on Control Room, select Add Component,
  - Add DSC, name it “primary-center-monitor”
  - Add a normal and emergency state to this display
  - Assign parameters to this piece of equipment, assign the primitive equipment_set_component_state (this defines the base state of the piece of equipment, see Figure 159).

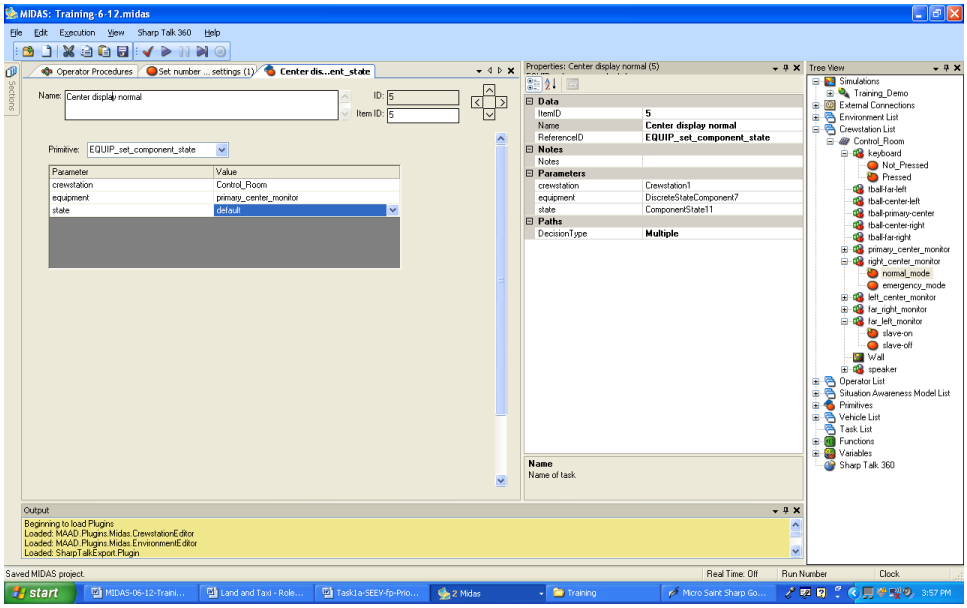


Figure 159. Primitive and parameters assigned to equipment state.

- Right Click on Control Room, select Add Component,
  - Add DSC
  - Name it “right-center-monitor”
  - Add a normal and emergency state to this display
  - Define the states for the right center monitor in the task network model as illustrated in Figure 160.

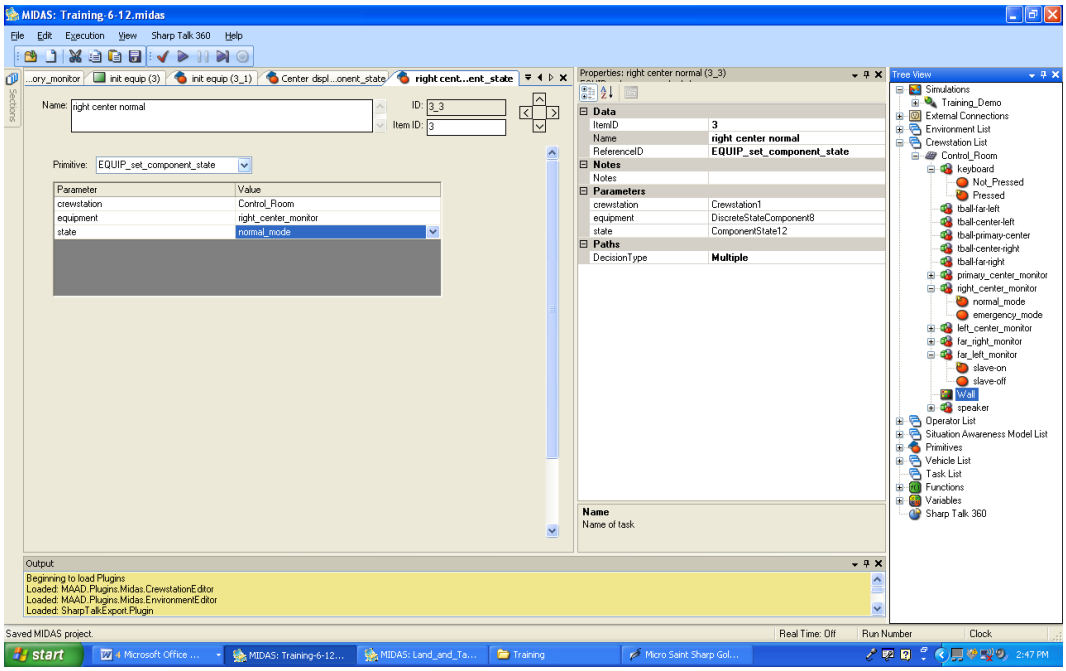


Figure 160. Define states for the right center monitor.

- Right Click on Control Room, select Add Component,
  - Add DSC
  - Name it “left-center-monitor”
  - Add a normal and emergency state to this display
- Right Click on Control Room, select Add Component,
  - Add DSC
  - Name it “far-left-monitor_position”
  - Add a normal and emergency state to this display
- Right Click on Control Room, select Add a Fixation point,
  - Add Fixation point
  - Name it “Wall”
  - Add 2 fixation points (“center”, and “far right”) to the Wall
- Right Click on Control Room, select add Component,
  - Add DSC “Speaker”
  - Add the attribute “auditory” and define this as being ‘sound emitting’ (form pull down menu) and name it ‘alarm’ (see Figure 161).

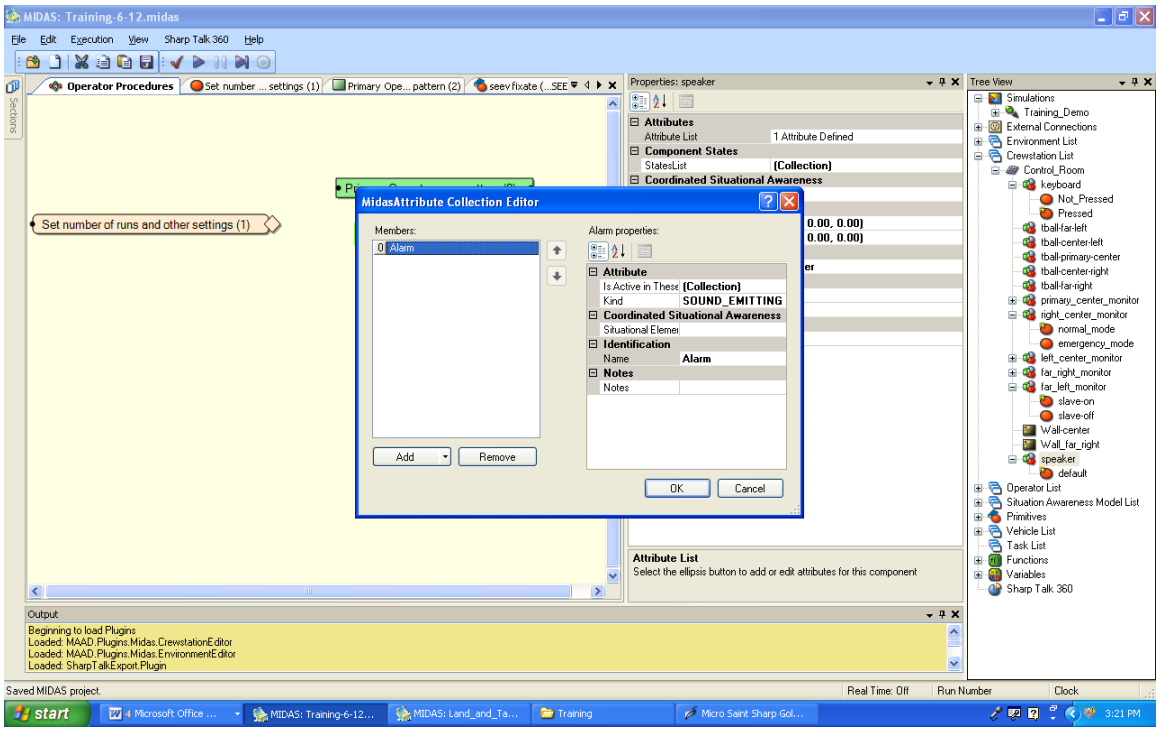


Figure 161. Defining the speaker component.

Important Note: if you want to delete component or any crewstation piece of information from the MIDAS window, right click your mouse on top of the item of interest. This will bring up a list of possibilities including add and delete among others that are contextually determined. Choose the delete option and this will remove the component from the list.

Configuring Component States

When the component states are configured to work together, they will link the Jack file names to the item in the task network model. When this happens, the information becomes available to the perception model.

- Right Click on the Keyboard and select Add State
- Name the state “pressed”.
    - Set the state to true
- Add another State to Keyboard named “not pressed”
    - Set the initial state to false
- Assign a transition state to the attribute in the attribute collection editor of the component “keyboard” – the transition states are defined under the “is active in these” portion of the attributes pull down menu (see Figure 165).
        - Click on the pull down attributes list (top ellipse in Figure 162) menu and select “visualattribute 1” and assign this to pressed, “visualattribute 2” and assign this to not-pressed.
        - Define the transition by going to the component states list and defining the attributes that are configured with the state (visual or auditory; see Figure 162).

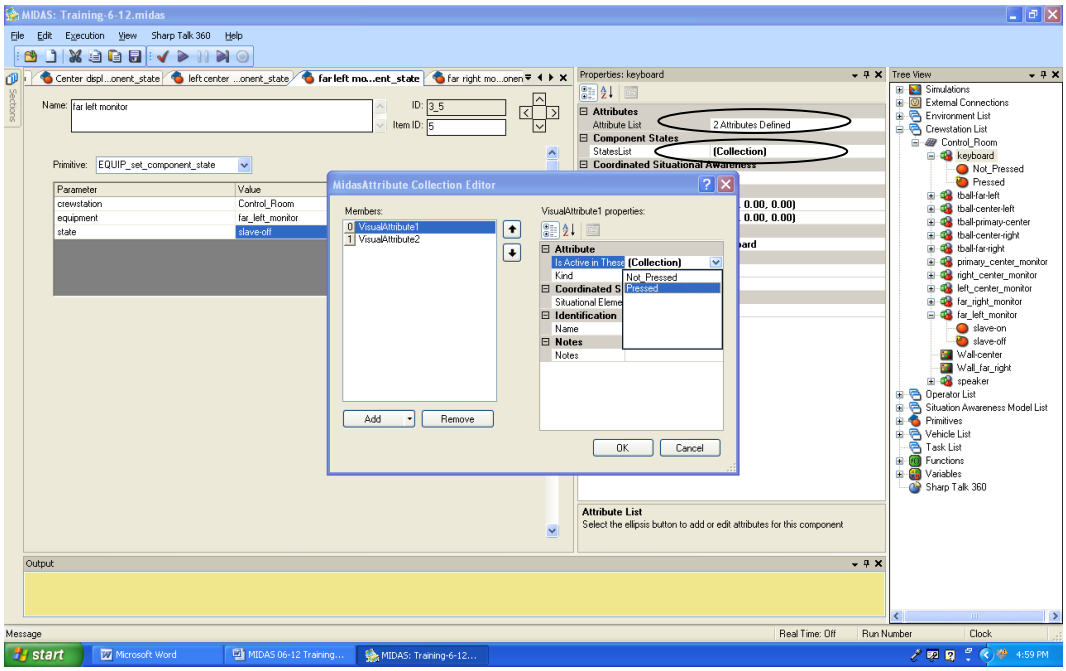


Figure 162. Configuring component states.

- Cause the "right-center-monitor" to change to the Emergency State when the Keyboard is pressed.
    - Set up the task network operator procedures to include a reach arm (uses the Welford variant of Fitt's Law - ballistic movements) followed by a press & release primitive (Shannon variant of Fitt's law; see Figure 163)
    - The equipment set component state primitive causes the item named 'emergency mode' to be visible while the 'normal mode' is not visible

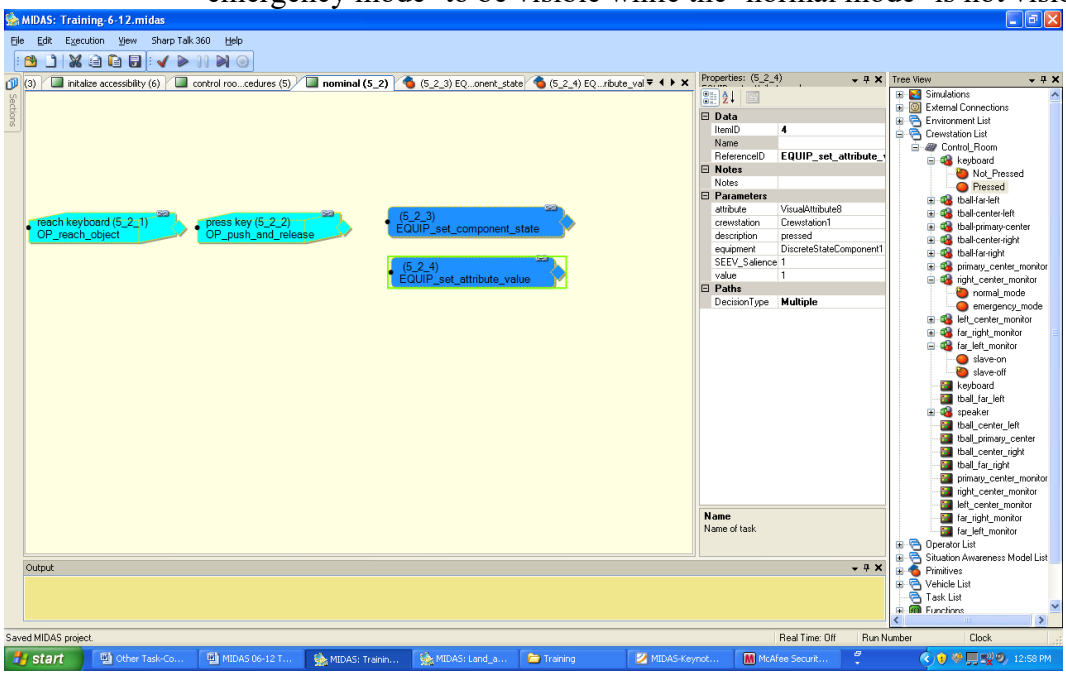


Figure 163. Procedures required to cause link the keyboard to the monitor change.

- The component models will be linked to the Jack CAD models and at that point, the displays will turn on or off depending on the state definitions completed in the DSC tree view (Figure 164).
- The attributes of the keyboard are defined at the keyboard level and not at the lower state level.

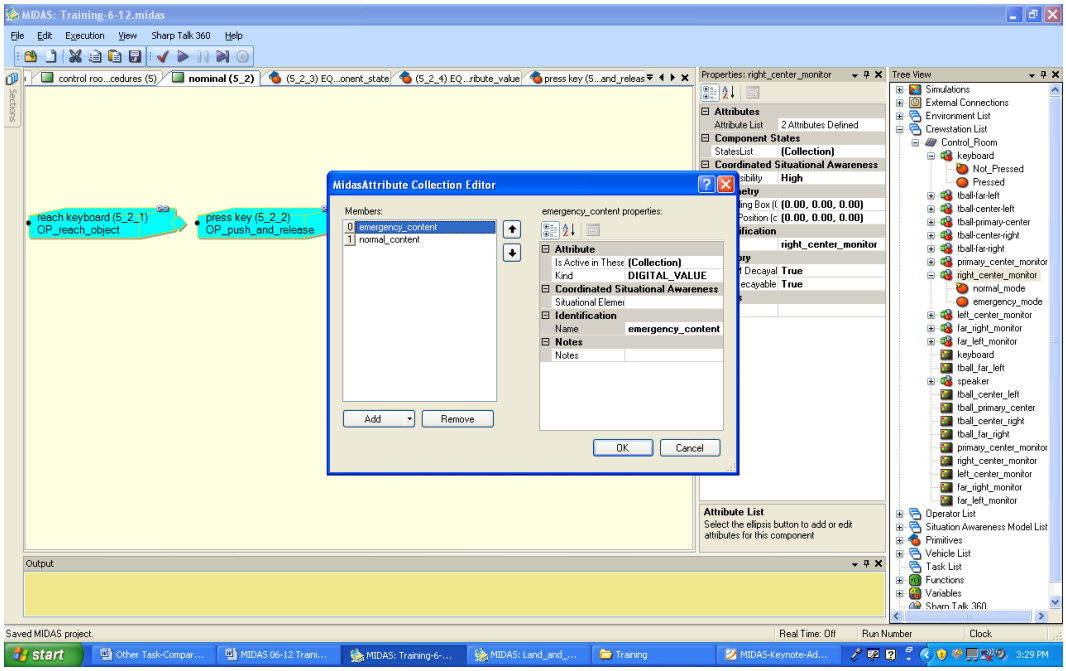


Figure 164. Defining attributes of the component models to be linked in Jack/CAD software.

- Define the “pressed” attribute of the keyboard state under Keyboard (Figure 165)
- Define the “not pressed” attribute of the keyboard state under Keyboard (Figure 166)

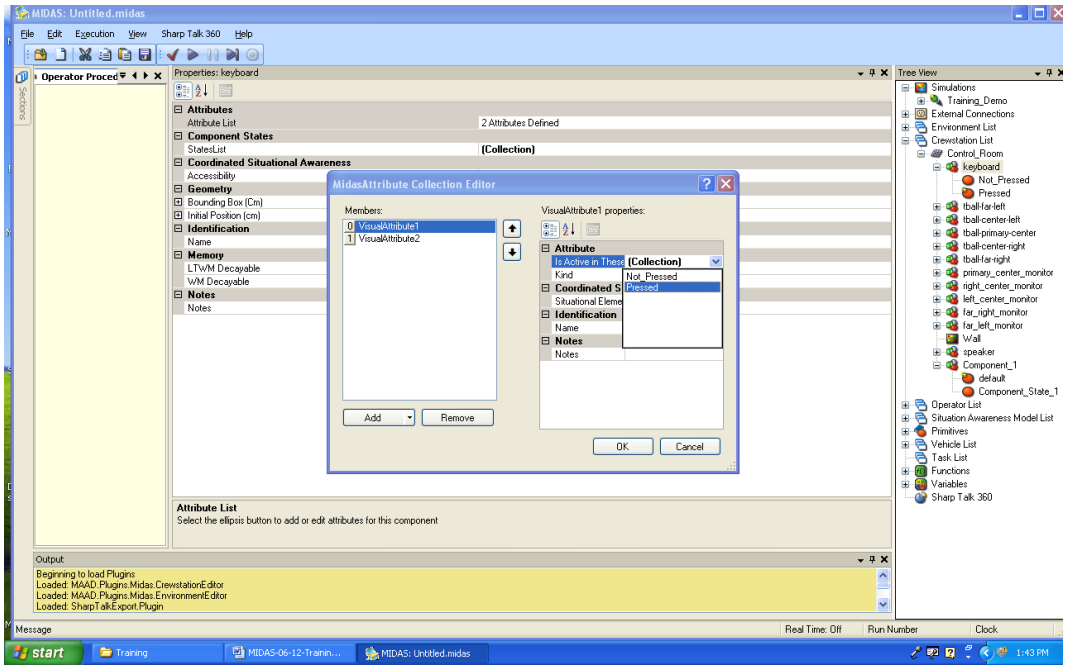


Figure 165. Visual attribute 1 defined of state transition defined.

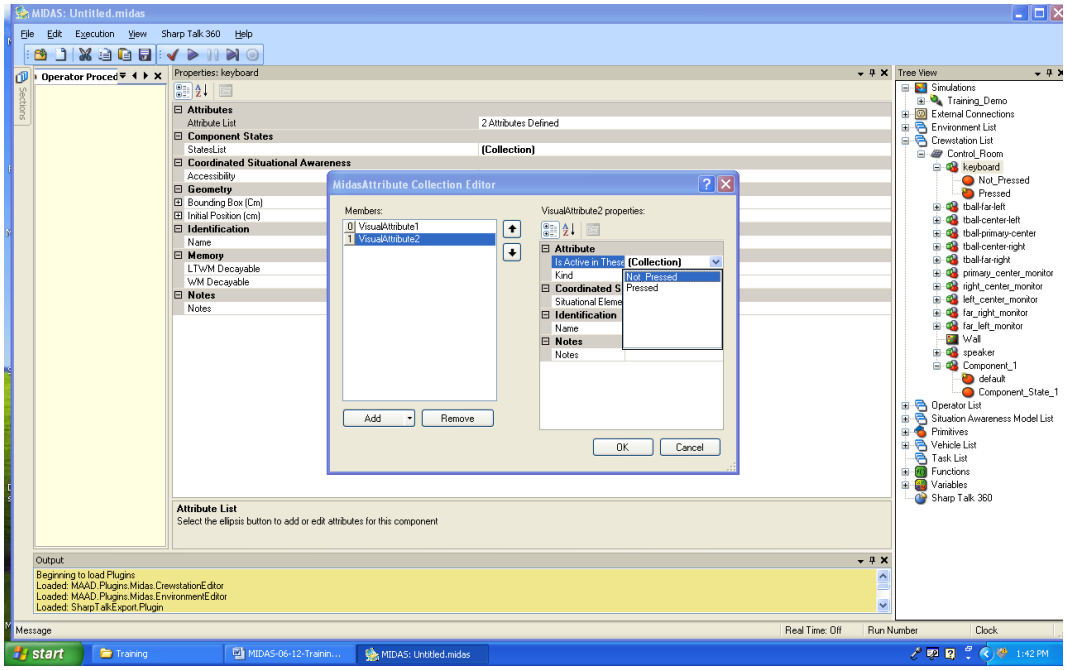


Figure 166. Visual attribute 2 of state transition defined.

- To set the transition states – link the attributes to their active state. The normal mode is active when the attributes are normal content while the emergency model is active when the attributes are emergency content (Figure 167)
- Define the equipment states in the task network model as illustrated in (Figure 168, Figure 169)

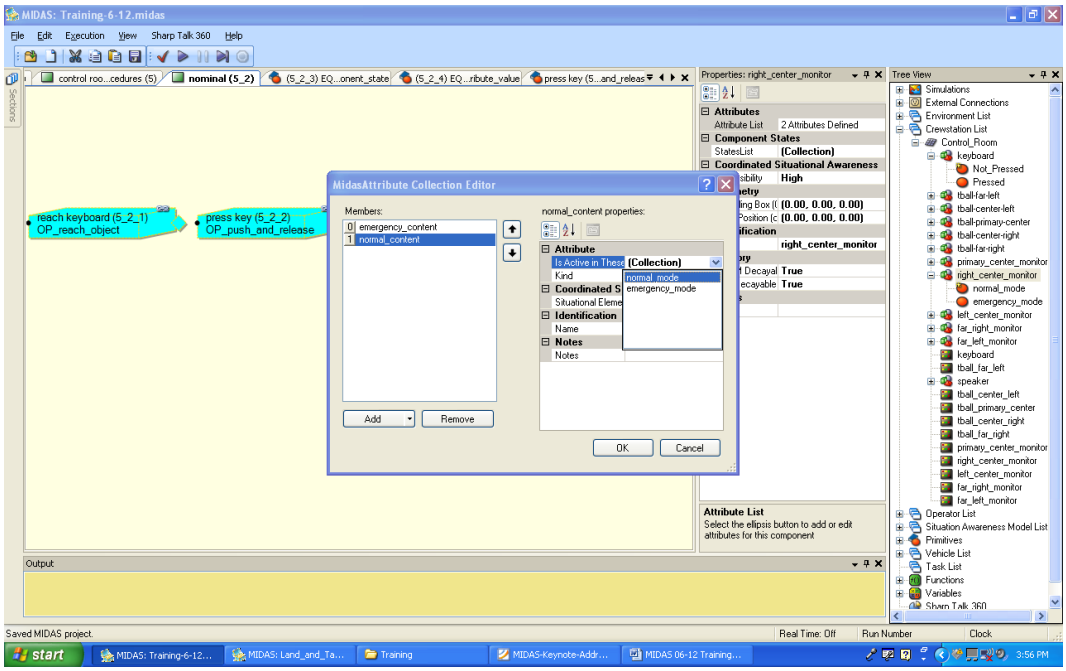


Figure 167. Define attributes to state components.

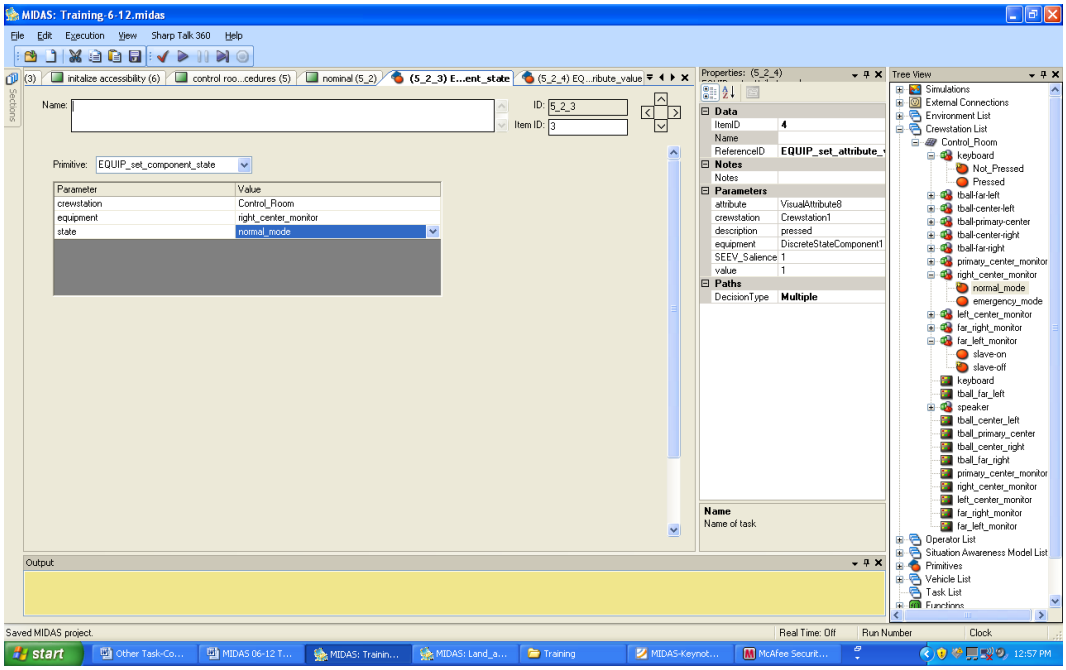


Figure 168. Define equipment states in the operator procedures window,

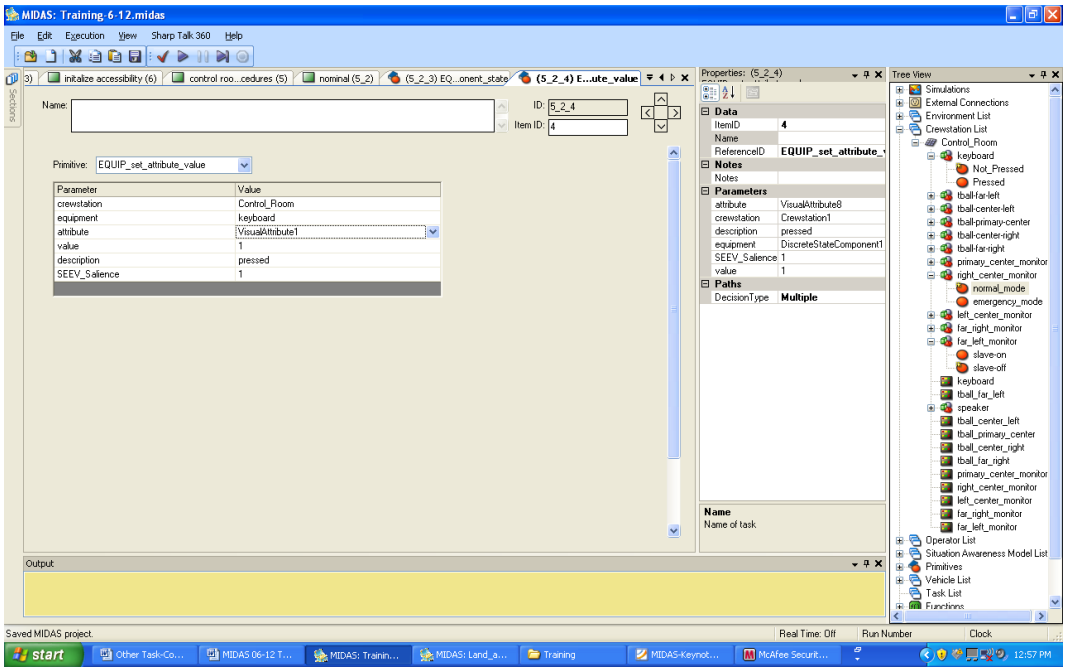


Figure 169. Defining equipment values in the operator procedures window.

- Add 2 States to Right Monitor (Figure 170).
  - Call them “normal_mode” and “emergency_mode”
  - Set “normal_mode” as the initial state.

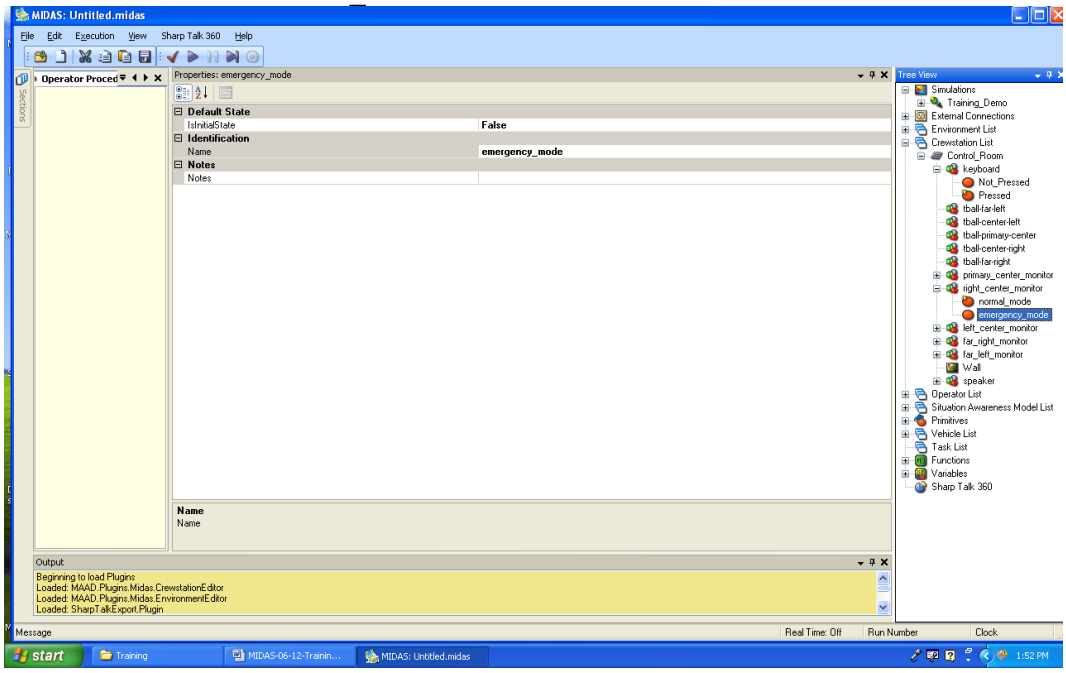


Figure 170. State definition of the right monitor.

Click back up one level to the right center monitor and define attribute states (that you will then tie to state definitions of the displays; see Figure 171).

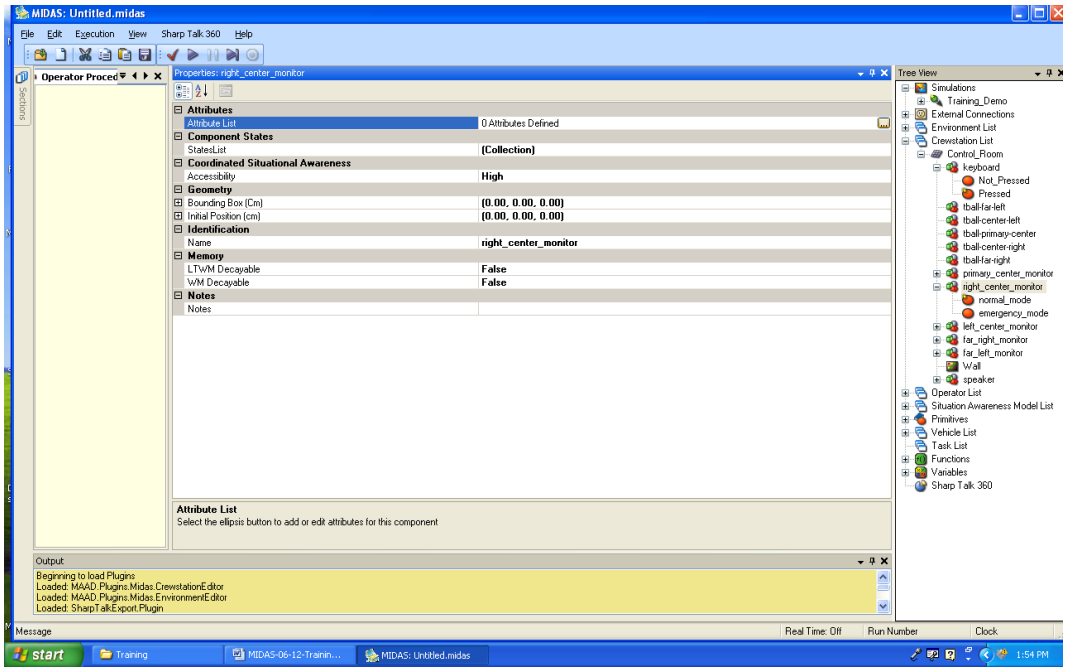


Figure 171. Attribute definition on display states.

- Add 2 States to Right Wall
  - Call them “Blank” and “Emergency”
  - Set Blank as the initial state by tying it back to the initial state of the display (Figure 172).

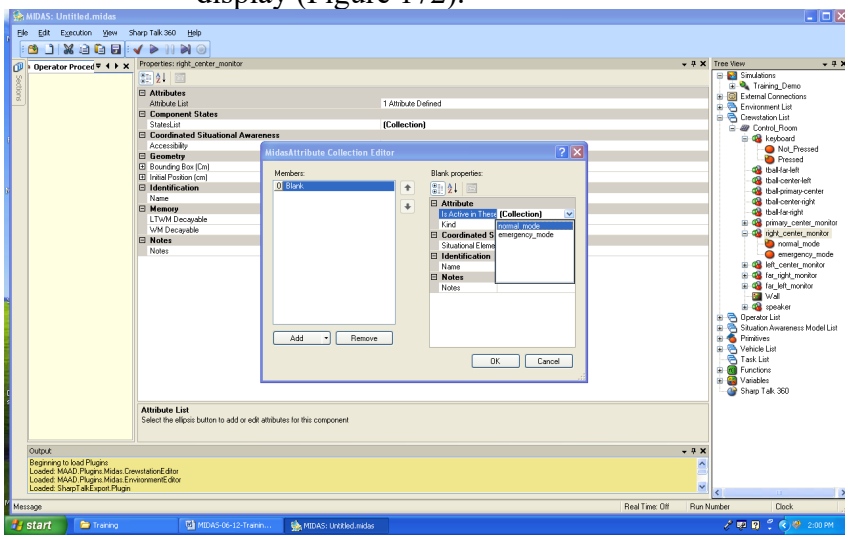


Figure 172. State definition of wall-right attributes.

- Add 2 States to “far-left-monitor_position” (Figure 173).
  - Name them “slave-on” and “slave-off”
  - Set “slave-on” on as the initial state

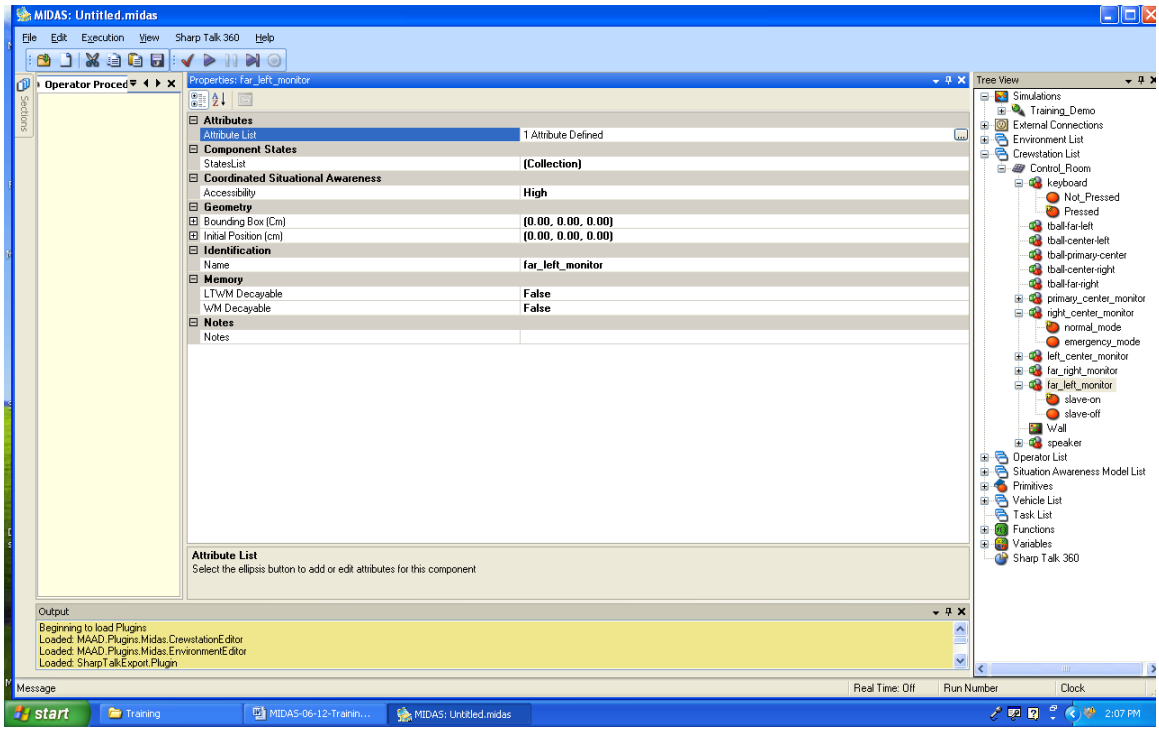


Figure 173. State definition of far left monitor.

Configuring Component Interactions

There are occasions in the model when pieces of equipment are tied together, for instance, when a keyboard is pressed there will be an impact on another model in the simulation. To configure these equipment components together, equipment definitions will need to be created in the procedures window (see Figure 174). These equipment definitions are tied to the DSC models in the tree view. If the DSC has defined states such as on and off as in the slave on and off example above, then these states will be available and can be tied together. If left as the default, then this will be the configuration that will be possible through the pull down menu (see Figure 175).

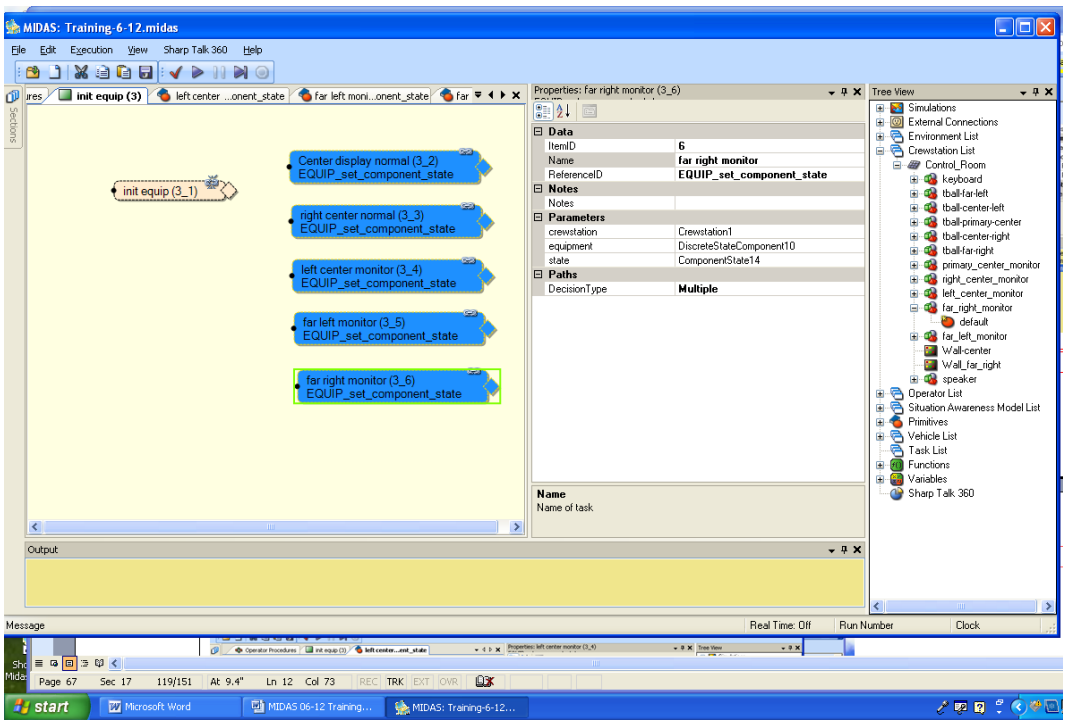


Figure 174. Configure equipment components.

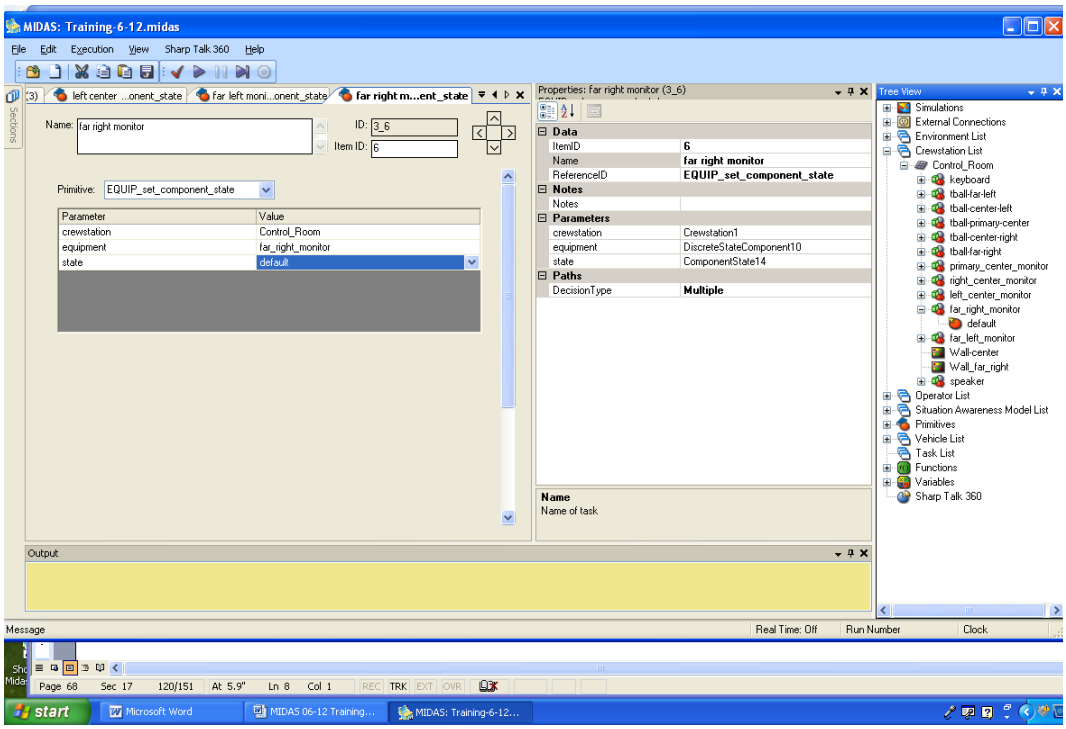


Figure 175. Example of a Figure being defined with default values.

Note: for best tracking between these elements in a model, it is suggested that the equipment states in the operator procedures window be named similarly to the tree view names. The states in the pull down menu of the equipment set component come from the states that are defined in the crewstation components in the tree view.

To configure the far left monitor to the states “on” and “off” which have been defined in the DSC tree view, tie the far left monitor equipment definition to the relevant parameters (crewstation 1, far left monitor, and slave on/off; Figure 176). If a definition has been accidentally omitted, simply go into the tree view and define the states (e.g. on or off). Then return to the equipment primitive and the option will be available in the pull down menu.

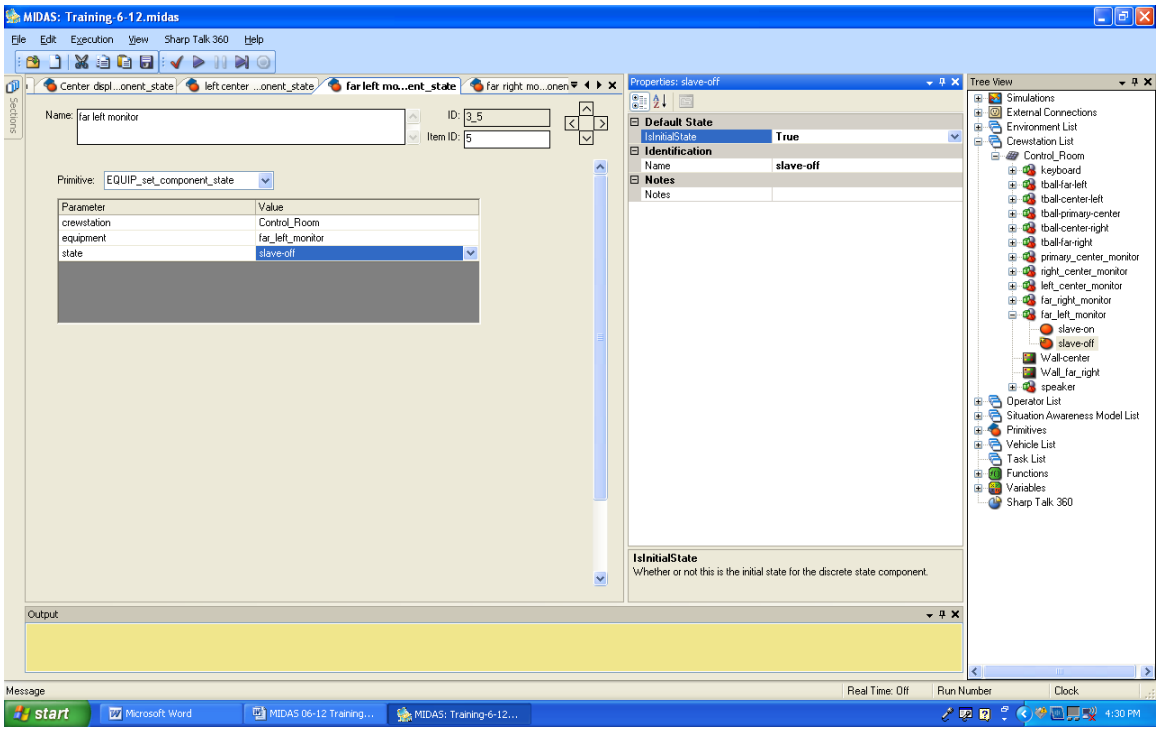


Figure 176. Configuring component interactions.

Setting up the Operator Procedures

To configure the tree view components to interact with each other, it is recommended that the model be set up to launch effects from the operator procedures portion of the software. All of the starting positions of the components in the model will need to be defined –the routing task will be required as the first step.

Add a routing task and Name it Set Number of runs and other settings (Figure 177).

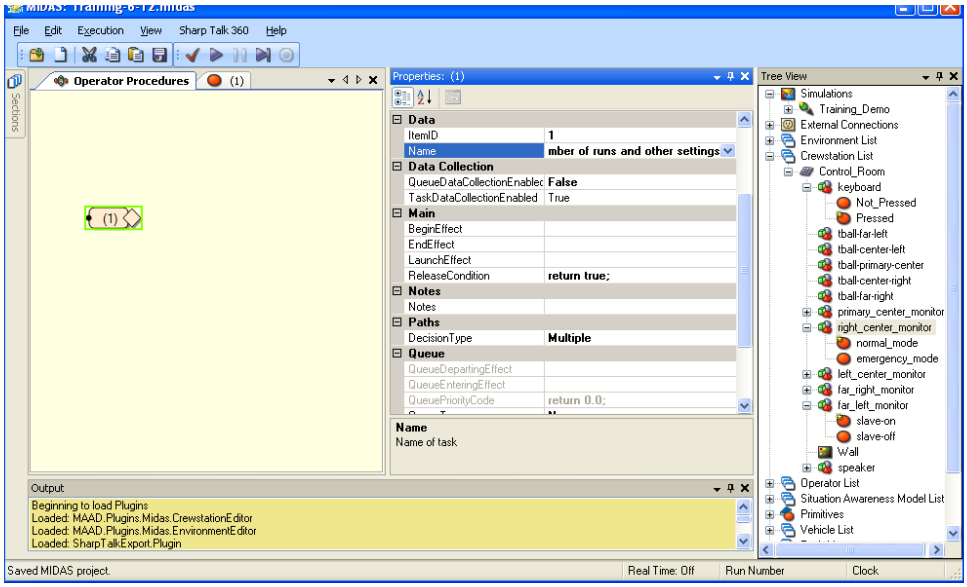


Figure 177. Setting up the operator procedures and scenario tasks.

In the Main Properties, put in the beginning effect, NumberofRuns =1. This defines the number of runs for the simulation. If you desire multiple runs (MC mode) then type the number of runs you would like here (Figure 178).

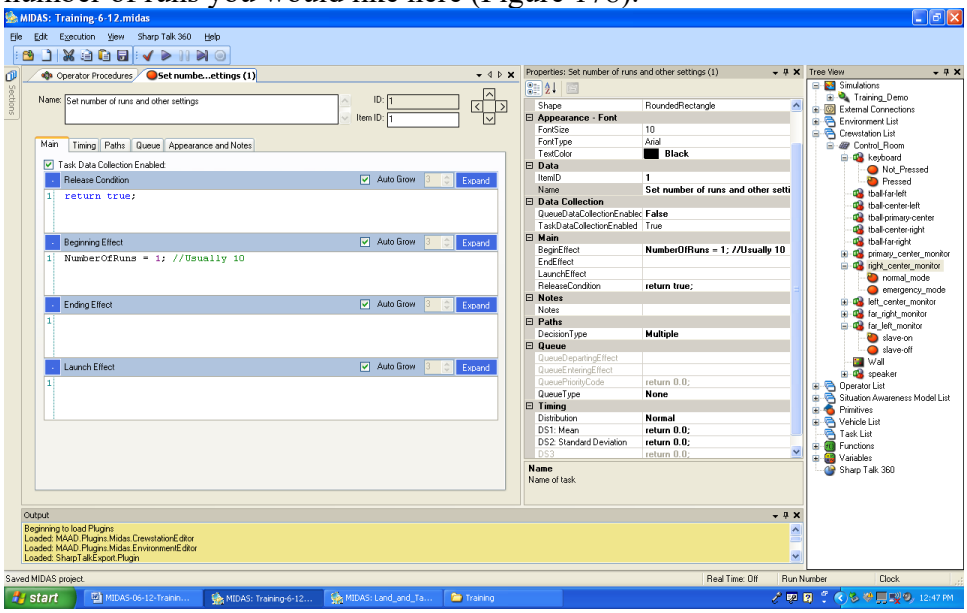


Figure 178. Settting up the model simulation settings.

To start creating the scenario and interacting with MIDAS, begin by entering some very high level information into the task network as illustrated in Figure 179.

Create Three initialize model networks; one for the scan pattern, one for the equipment states, and one for the Jack initial states (this last network will reset Jack's anthropometry so that the start point for the operator will be consistent throughout the various scenarios and simulation runs (Figure 179).

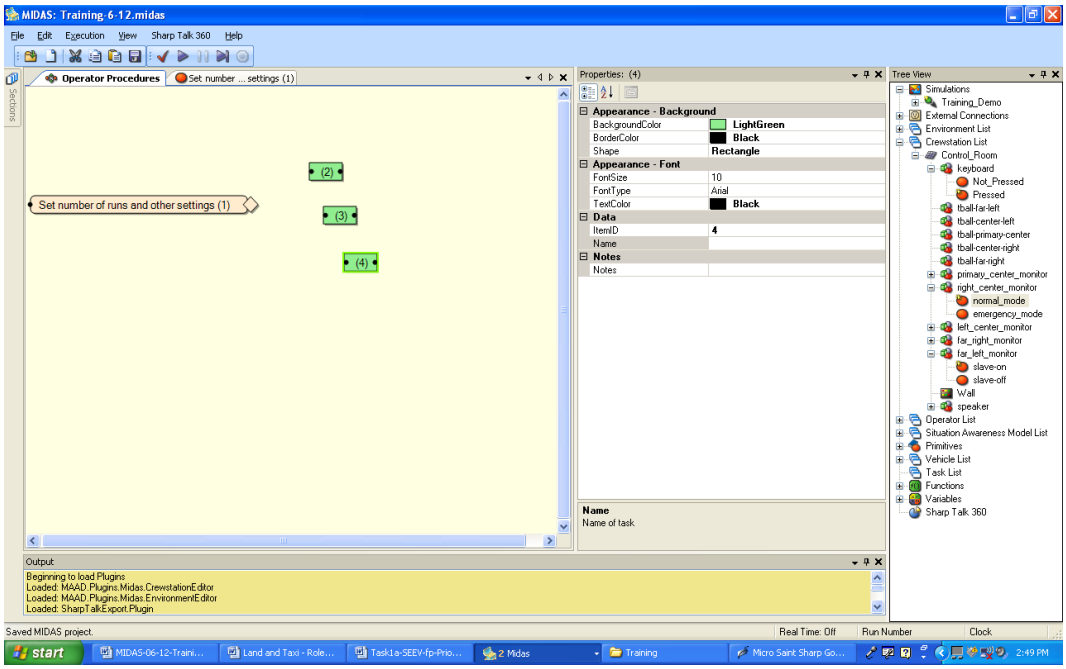


Figure 179. Exemplar of one settings node and three task networks that can be used to start the model.

Once the three operator procedures have been created as in Figure 179, define them as in Figure 180.

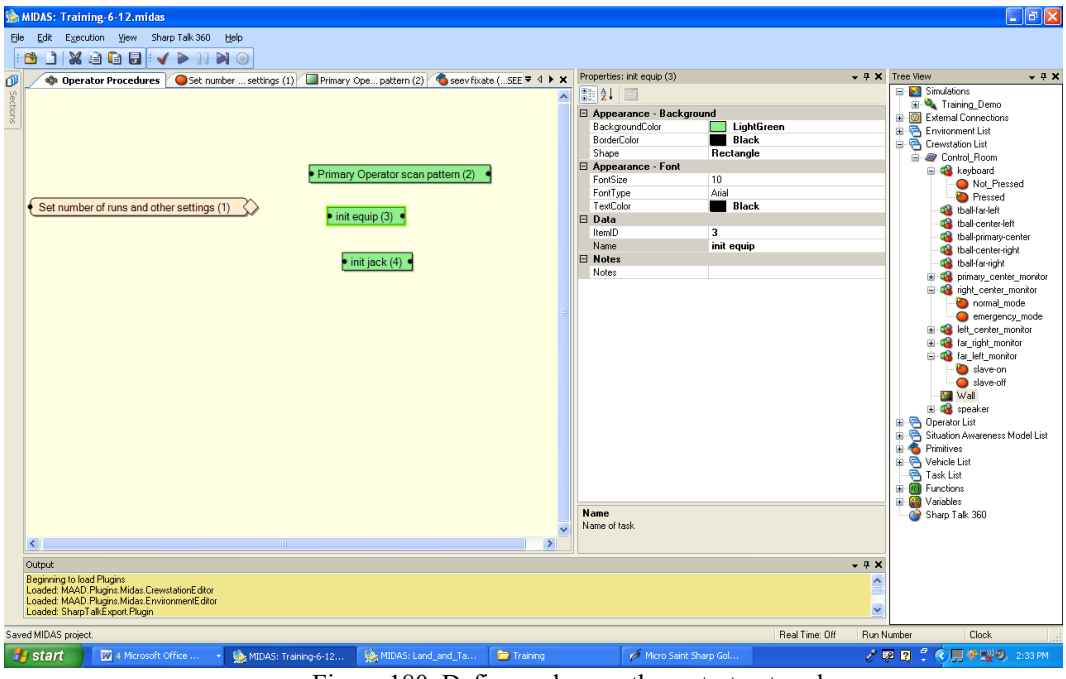


Figure 180. Define and name three start networks.

Double click the scan pattern network (this will bring you to screen as illustrated in Figure 181 and add a routing task in the primary operator scan pattern; one to go to visual and one to go to auditory. This will be used when the scenario requires the response to the alarm.

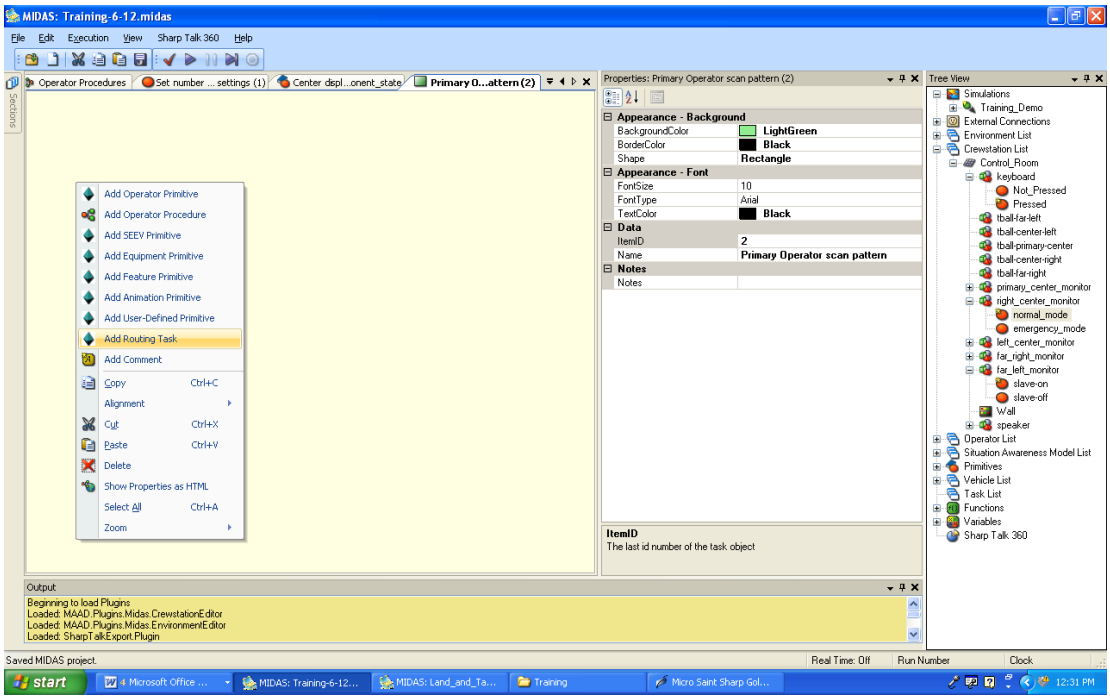


Figure 181. Define scan pattern network.

Add a seev primitive for the visual fixations, and add an operator model for the auditory monitoring task. Type in the desired name into the name field, for e.g. seev fixate for the visual and op auditory monitor for the auditory (Figure 182; Figure 183; Figure 184; Figure 185).

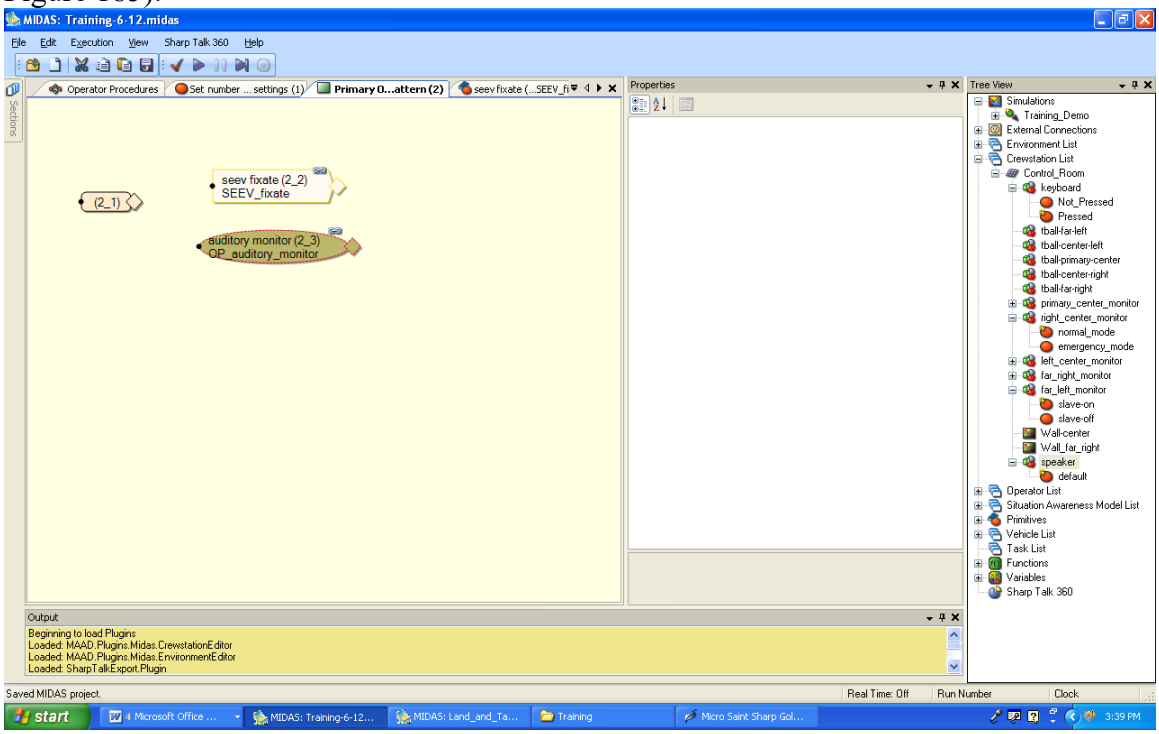


Figure 182. Create operator attention networks.

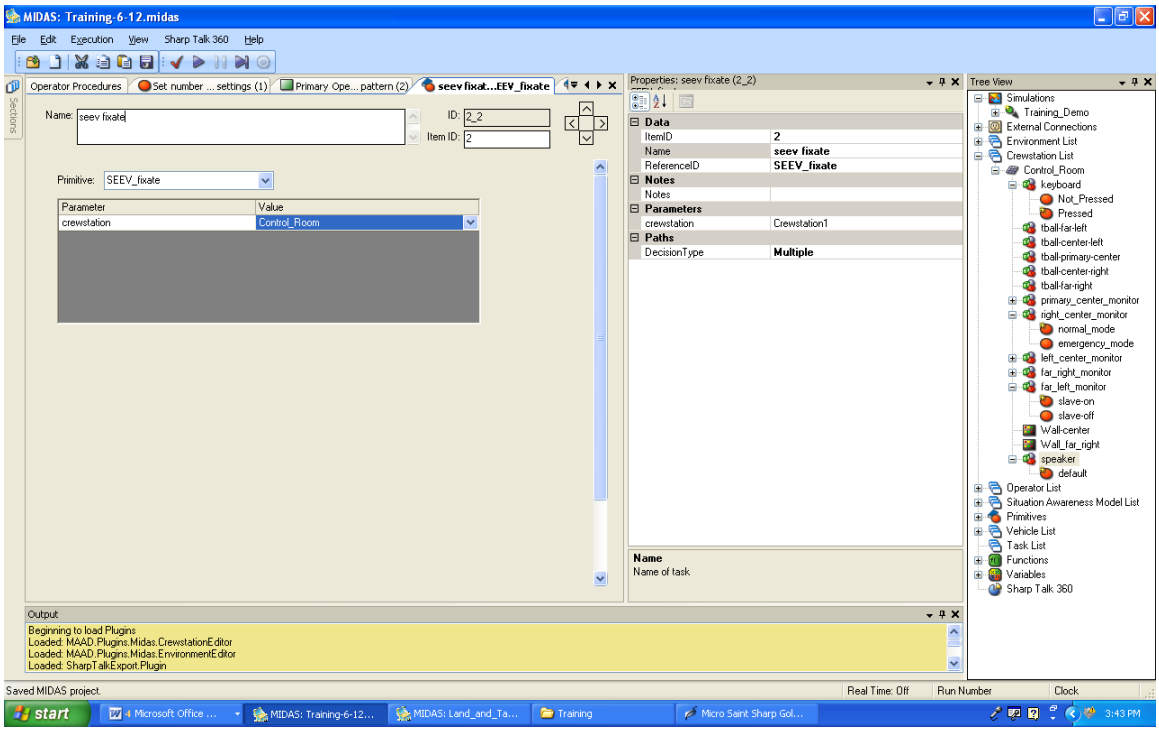


Figure 183. Assign SEEV Fixate to the crewstation used in the scenario.

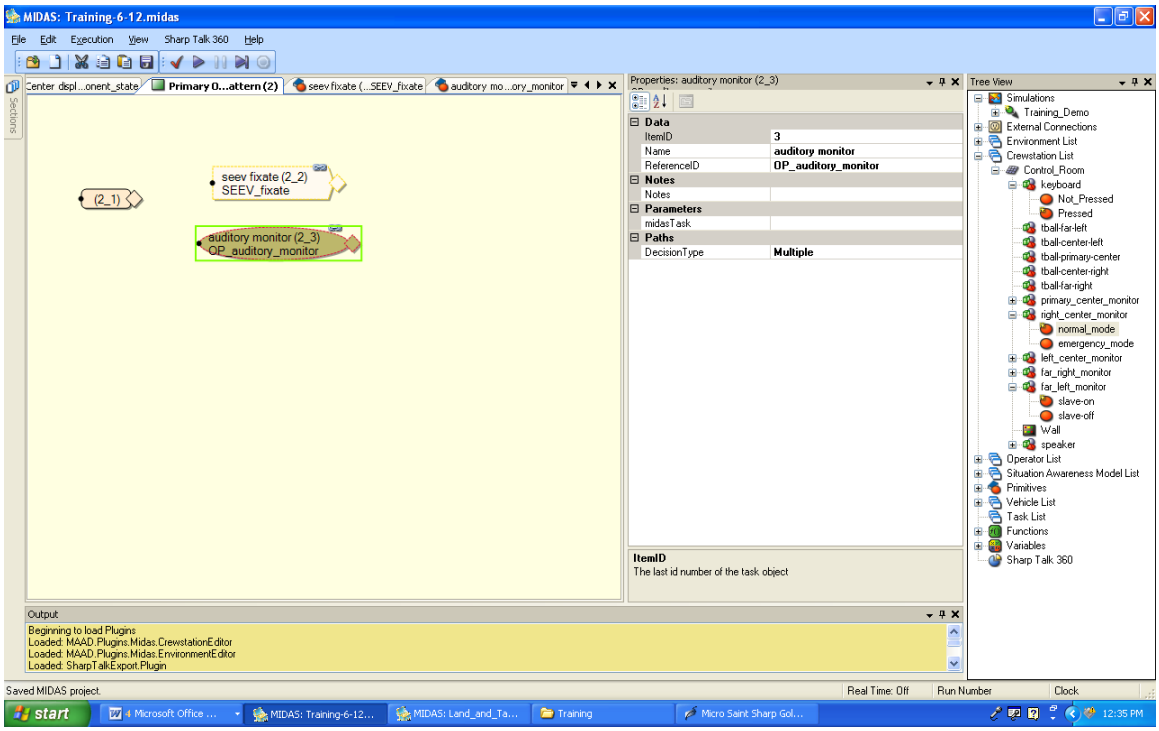


Figure 184. Defining the auditory monitor network.

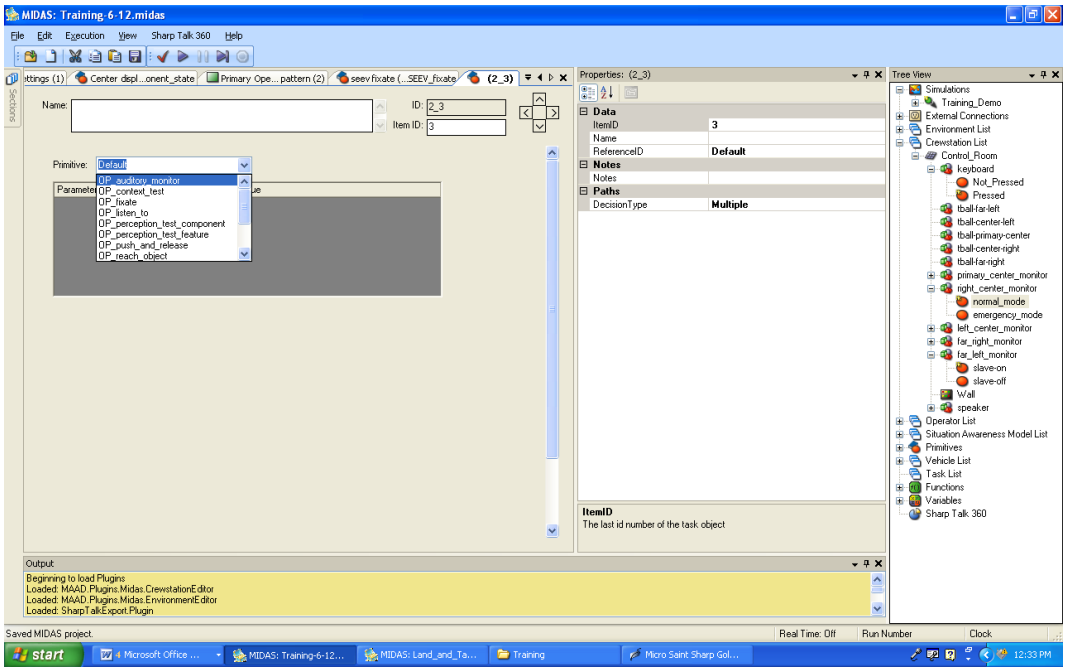


Figure 185. Assigning and defining the auditory model in the training scenario.

Import Crew Station Component size and position from Jack™

In this step we will map the names of component in MIDAS to the corresponding names in Jack™ (Figure 186). The locations of component in MIDAS will be populated with the locations from Jack™.

- Right Click on Control Room and select Import Crew Station from Jack™.

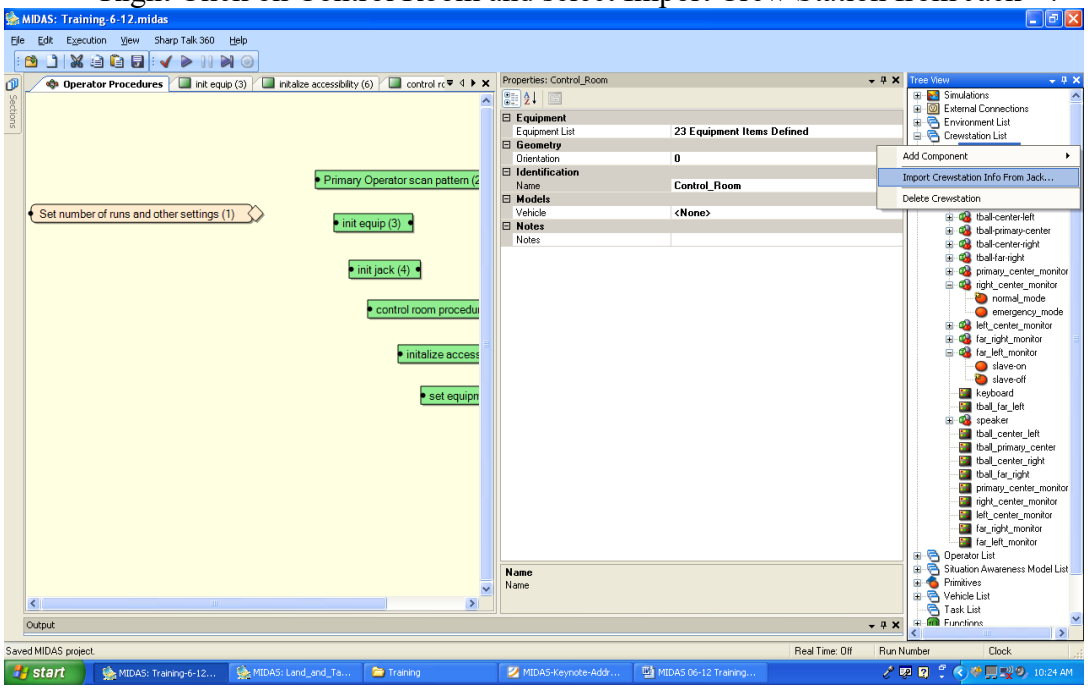


Figure 186. Importing crewstation components from Jack and mapping them to MIDAS component models and states.

- At this stage, you will want to map the Jack™ figures to the Crewstation components (Figure 187).

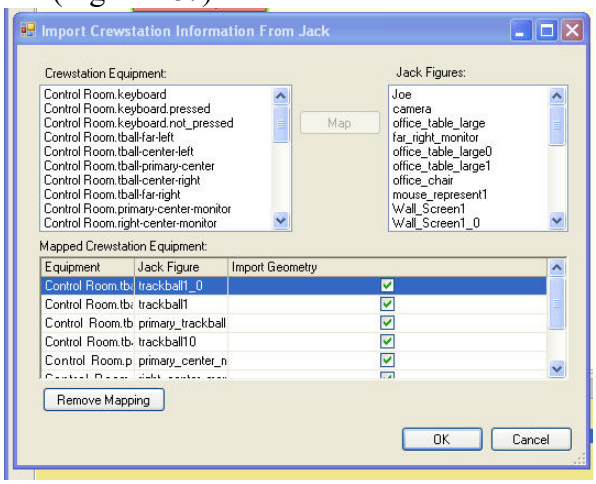


Figure 187. Map Jack figures to MIDAS crewstation component models.

- Note the MIDAS component names in left panel.
- Note that not all the figures in Jack™ appear in the right hand panel. *The set of names from Jack™ is too long causing a prematurely terminated communication from Jack™*. To work around this, do the following:
  - Close the import crewstation dialog.
  - In Jack, right click on all the track balls, the track ball bases, the tables and the chair.
  - Delete those objects.
  - Do **not** save the scene. If you do save the scene when you exit, you risk overwriting your original scene with the end state of the simulation.
- Open the Import Crew Station from Jack dialog
  - Map the components from MIDAS to Jack as follows
  - Select the Import button
  - Reset Jack by deleting the Scene and reopening the .env file without saving your changes from the scene with the deleted components.

Side bar: If you move an object in Jack you can update the location by re-doing the above sequence, mapping only the objects you've moved.

- Note: when you bring over the Jack™ files to MIDAS, you will see the files' positional information listed in the MIDAS window as exemplified with the keyboard below (Figure 188).

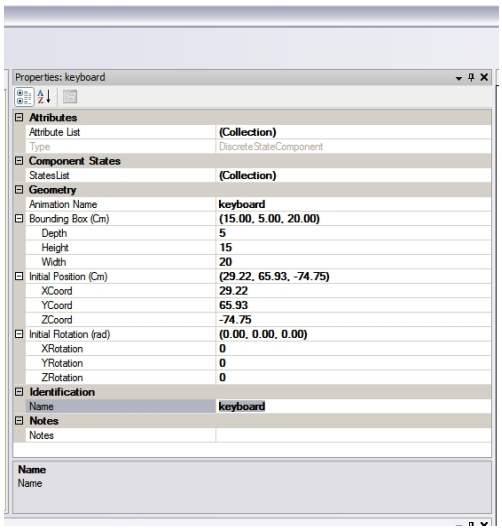


Figure 188. Positional information of the model's components.

Modeling Side bar: You can change the XYZ location in MIDAS after having imported the XYZ locations from Jack by typing over the respective X,Y,Z value. This will cause Jack to look at the new X,Y,Z value that you inserted. This is a workaround step to get Jack™ to fixate on the correct object site location.

The modeler will have to set the initial states of the pieces of equipment in the simulation by creating a task network entitled set equipment states as exemplified in Figure 189 (task network 7). Figure 190 and Figure 191 illustrate the settings needed to cause the displays to change, and then to be used by the modeled operator in the scenario.

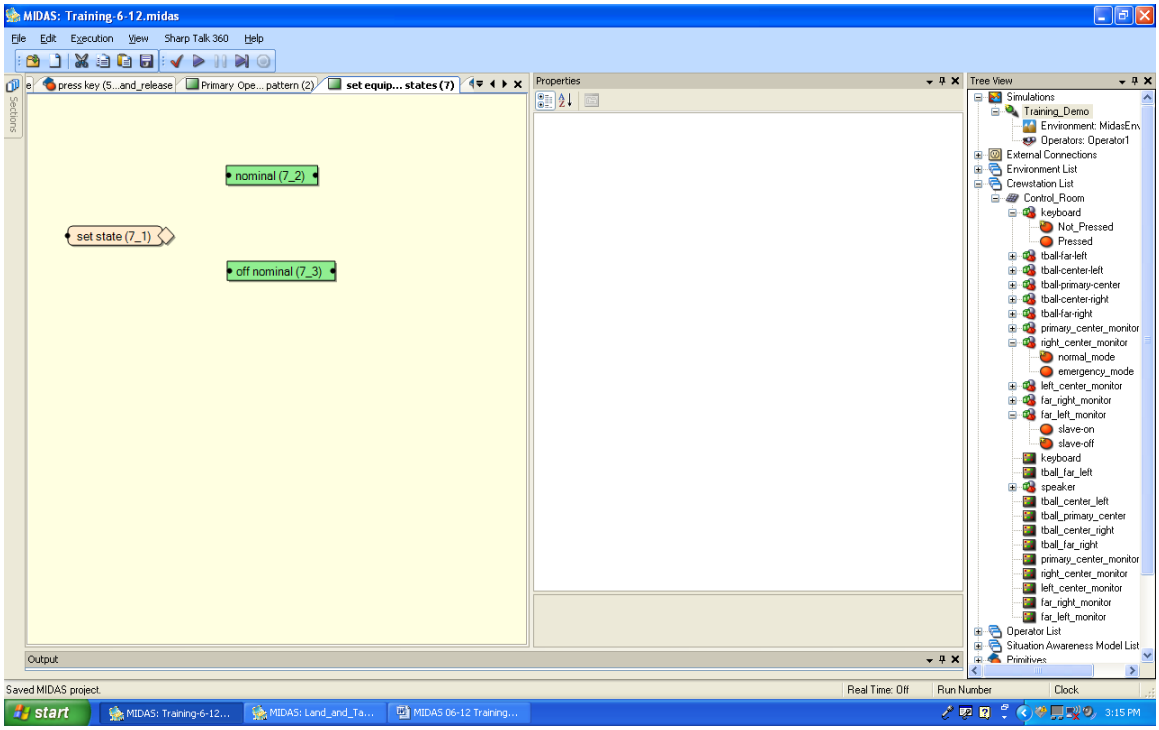


Figure 189. Task network to set equipment states that will change throughout the simulation.

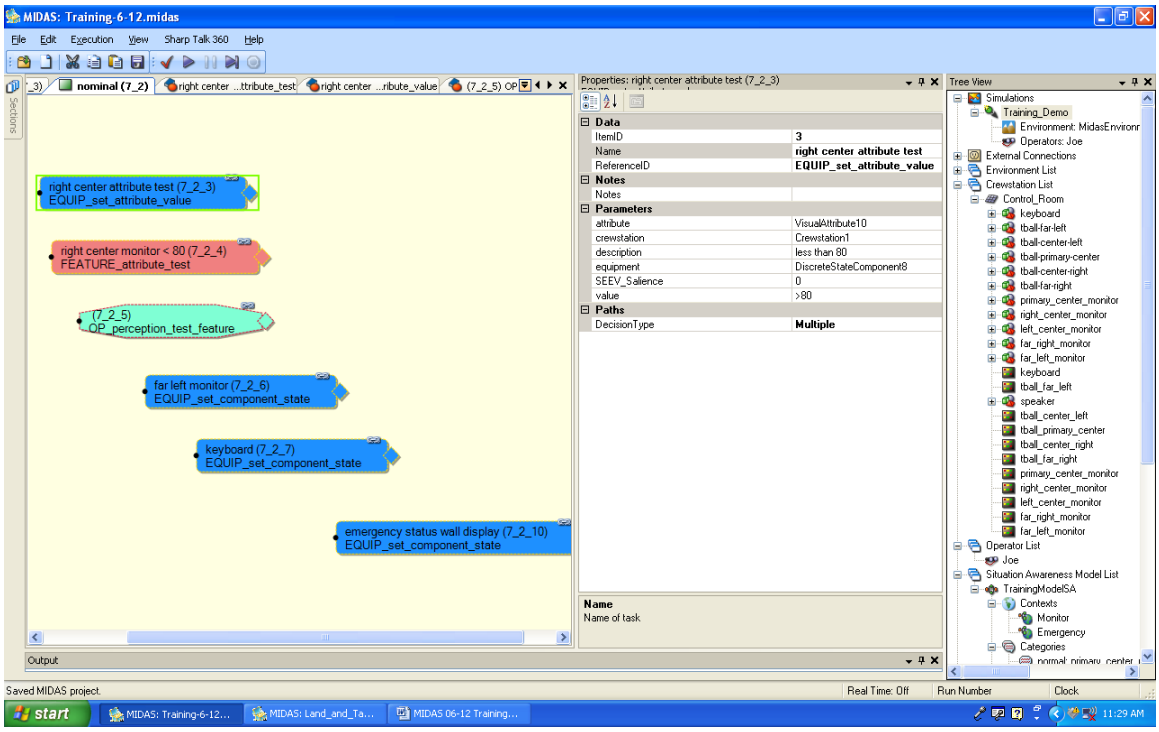


Figure 190. Equipment states - Nominal.

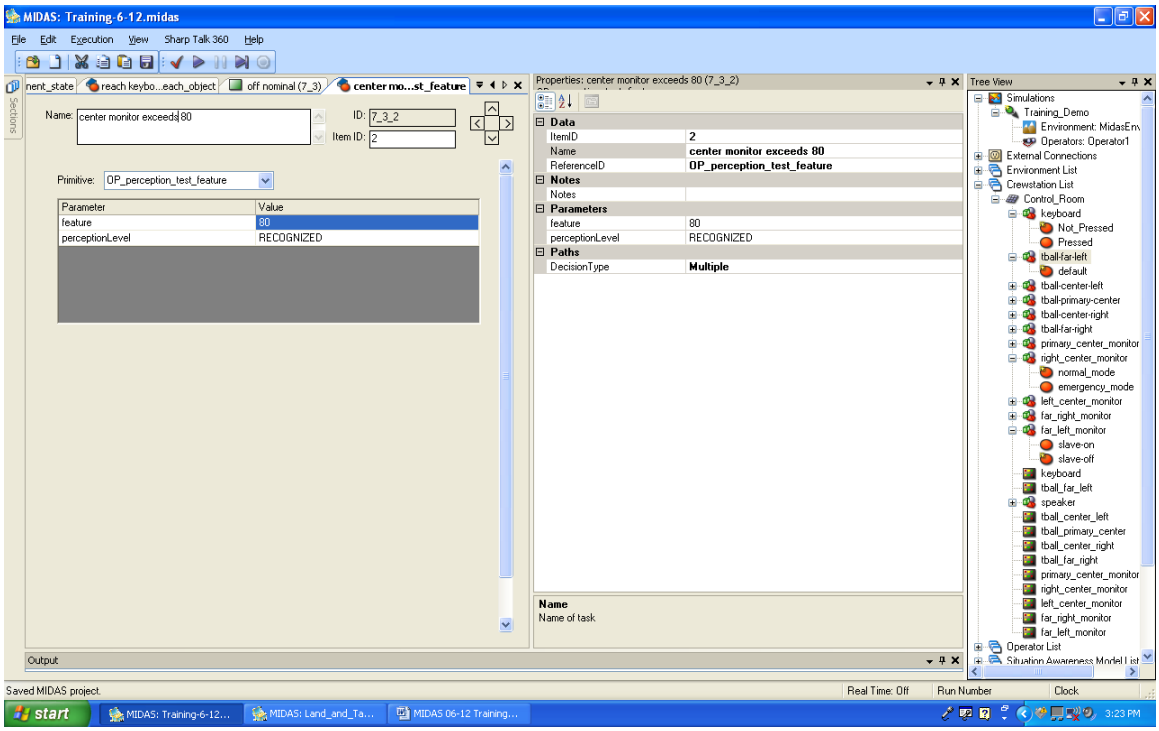


Figure 191. Definition state associated with the center monitor exceeding pressure of 80.

Adding a Component State, a second component, a Display state

- Add a new component state
- Select the component state by left clicking the mouse on the state of interest

- Right click the state to add a component to the state (can be either a discrete state, a base, a digital, rotational or textual component)
- In the case of a DSC, begin in the properties window and proceed to define the Geometry information (as in Figure 192).
    - Mouse click in the Animation Name field of the "Properties:" window (the middle pane).
    - Animation name – select the default identification name ("discrete state component") and insert the name you would like to call this display (e.g. "normal_content").
    - Set the Auto Show/Hide Components to "True"
    - Set the initialState to "True"
    - Select the "normal_content" in the tree view
    - Add a DSC to this "normal_content" and call it "status_display"
    - Type in the Jack file name to which you want this component tied in the "Animation Name" (e.g. "right_center_monitor") (see Figure 193).
    - This will bring in all of the XYZ values from Jack when the crewstation information is mapped between MIDAS and Jack

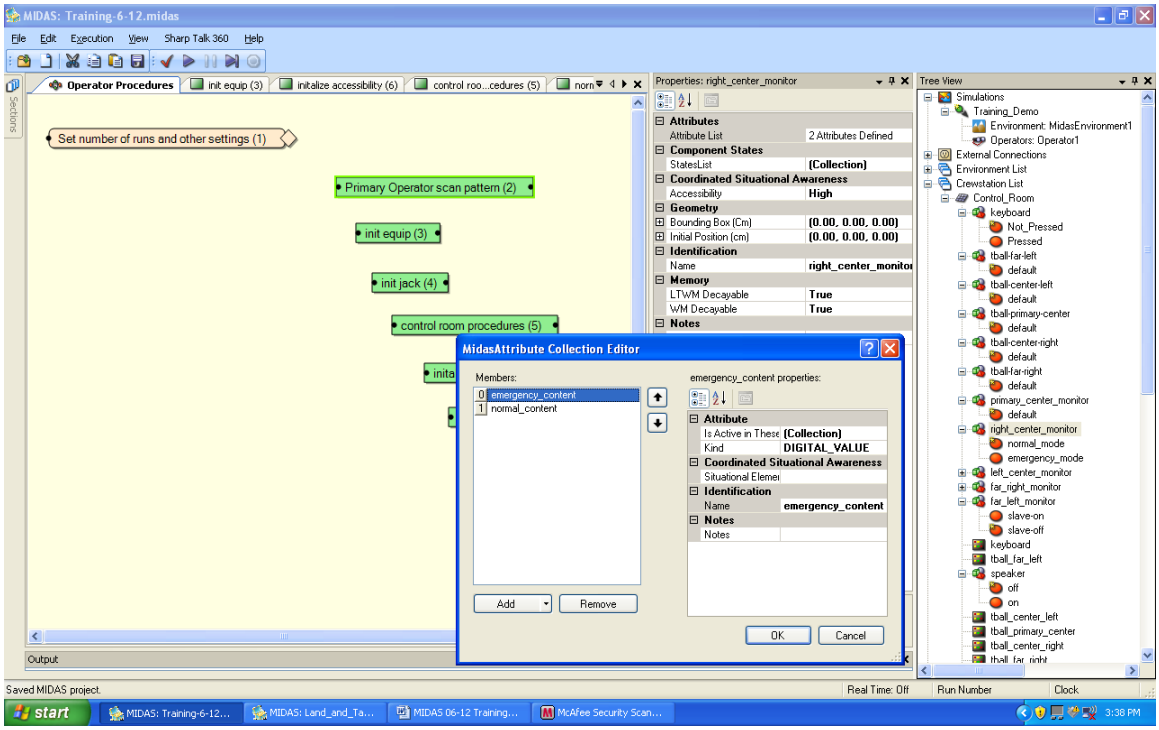


Figure 192. Example of Adding a Component State, a second component, a Display state through the attribute collection editor.

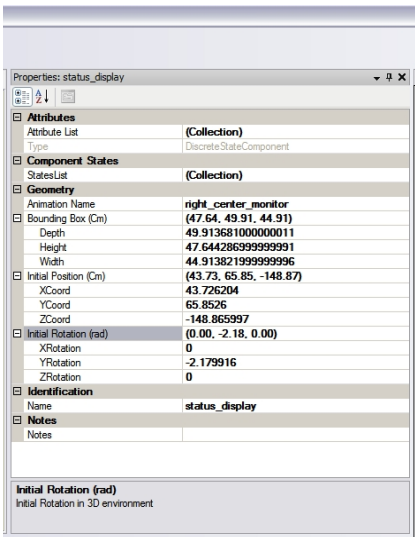


Figure 193. Screen snaphot of the DSC matching module between MIDAS and the CAD software.

- Add another Component state called “emergency_mode”
  - In the Properties window, Geometry section, type the Animation Name “blank-to-on” in the Animation Name’s field.
  - Set initial state to false (although it likely already is set to false because it is dependent on the “normal_content” initial state.
- Add a state called “emergency_status” (this will be defined when we define the wall components).
  - Add a DSC onto “emergency_status” called “emergency_display”
  - Type “right_center_monitor_emergency_display” in the Animation Name Field.

Adding Attributes to Component States

- Select the tball component and define it as a rotational component attribute with spatial float values
- To do this click on add, then define the values as needed in the “RotationalComponentproperties” along the right side of the Attribute collection editor (see Figure 194).

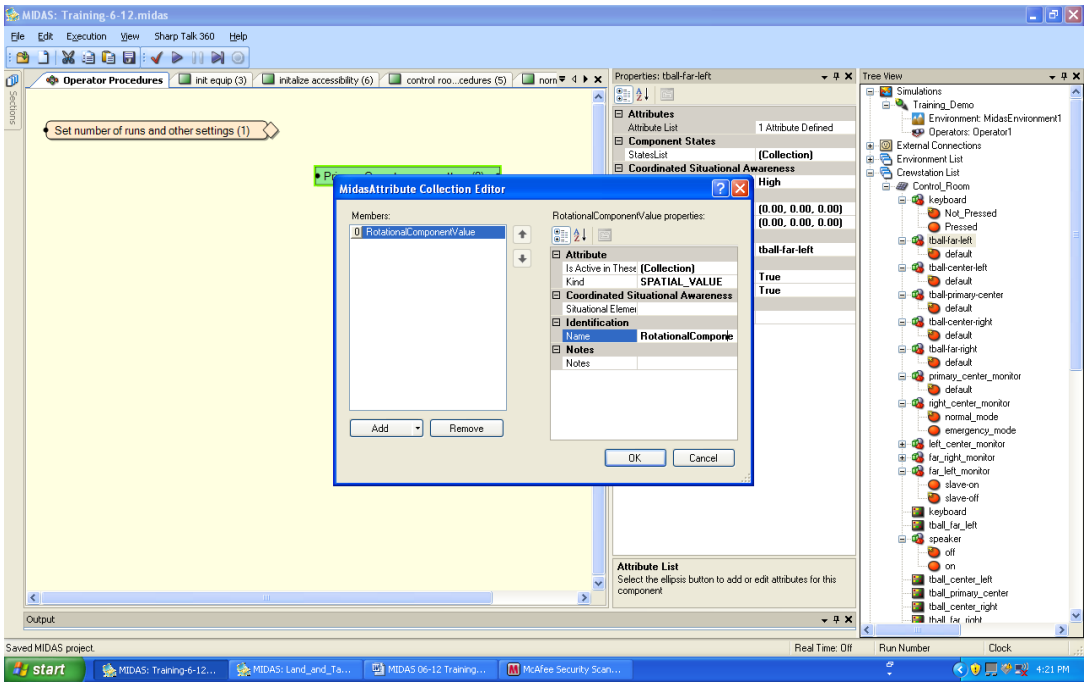


Figure 194.Screen snapshot of location to add attributes to component states.

Adding Attributes to Displays and Sound Emitters

- Select the far_left_monitor_position" DSC (Figure 195).
- Select the state "slave-on"
  - Add DSC "far_left_Wall_display"
  - Add DSC "far_left_monitor_on"
  - Select "far_left_Wall_display"
    - Click on the attribute field to show the ellipse
    - Select the ellipses (…) in attribute field and select the add button. This will add a Visual Attribute.
    - Add "emergency procedure" Attribute to "far_left_Wall_display"
  - Name it "emergency_procedure"
  - Set the Kind to Text
  - Type "procedure steps" in the text field

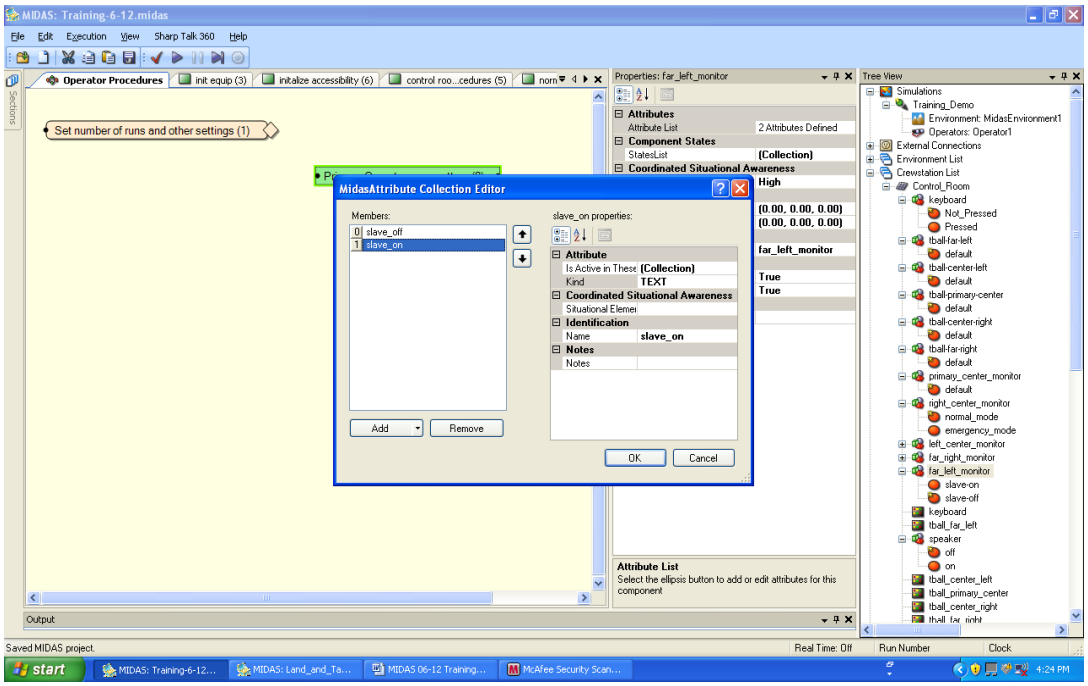


Figure 195. Exemplar of adding attributes and sound emitters to displays.

- Return to the far right wall that was added under the Wall base component
  - Add 2 States “blank” and “emergency_status” with blank as the initial state.
  - Add an attribute to “emergency_status” named “emergency_status_wall_display” (Figure 196)

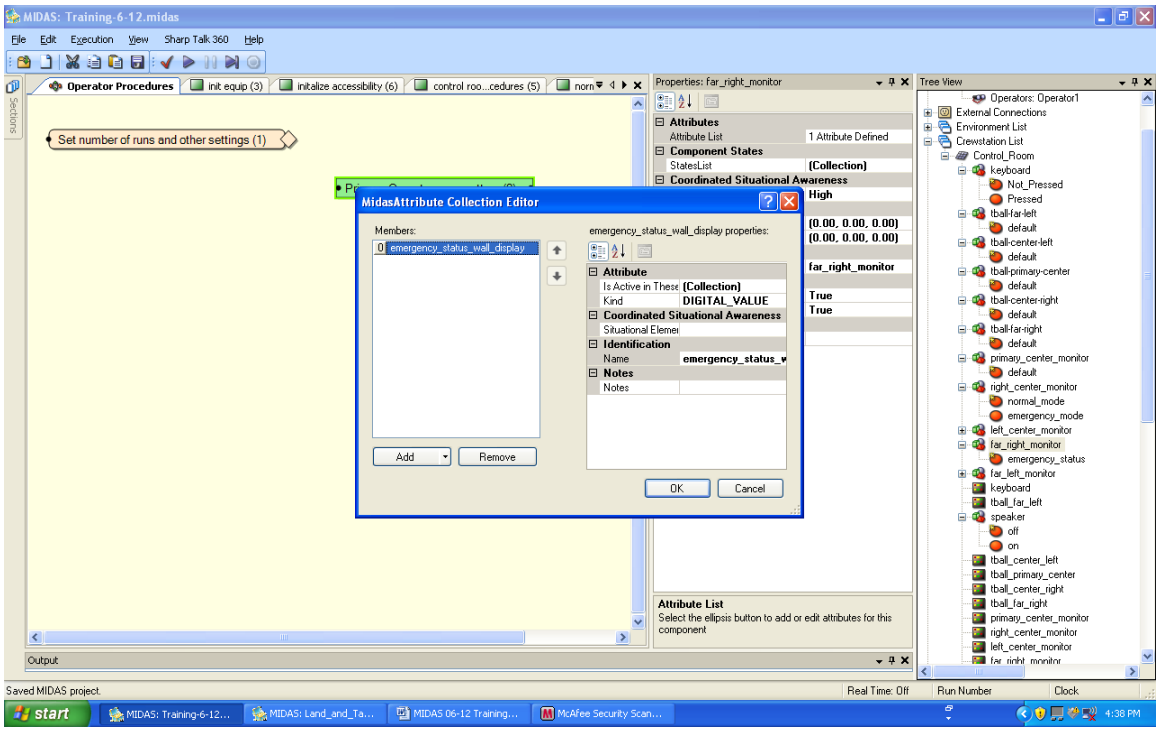


Figure 196. Add attribute to emergency status of the wall display.

- Now go to the center wall display
  - Add a visual attribute named “pressure”. It is a DIGITAL_INT. Give it an initial value of 10 psi (Figure 197, Figure 198).
  - Hint: Select the down arrow of the New Attribute Value field to type in it.

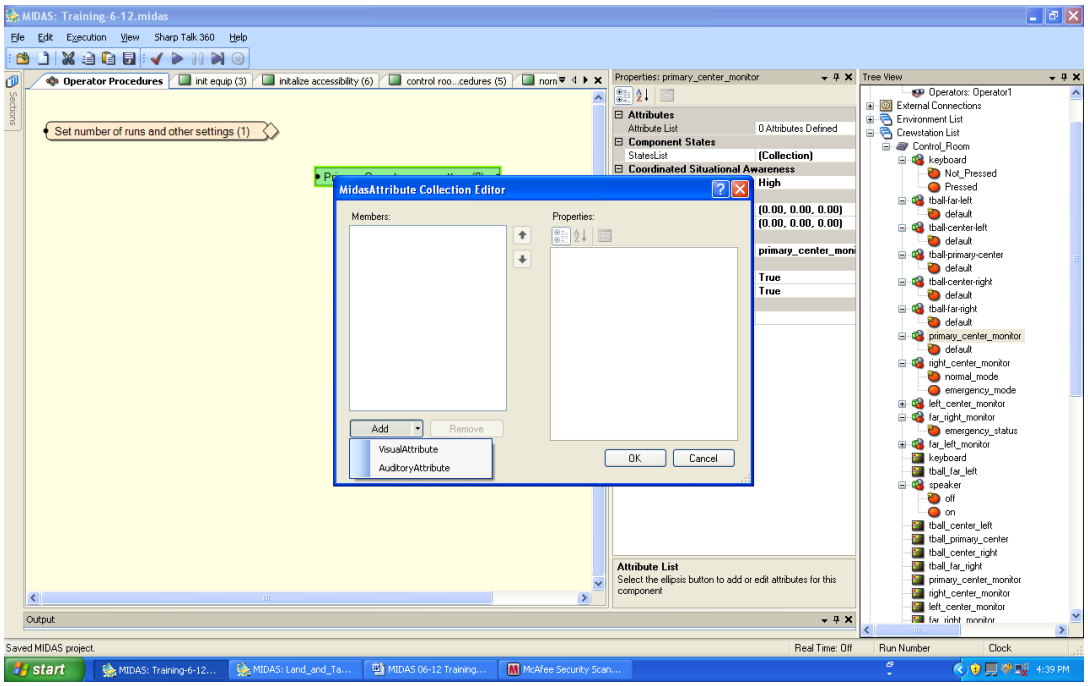


Figure 197. Exemplar of adding a visual attribute to the wall display.

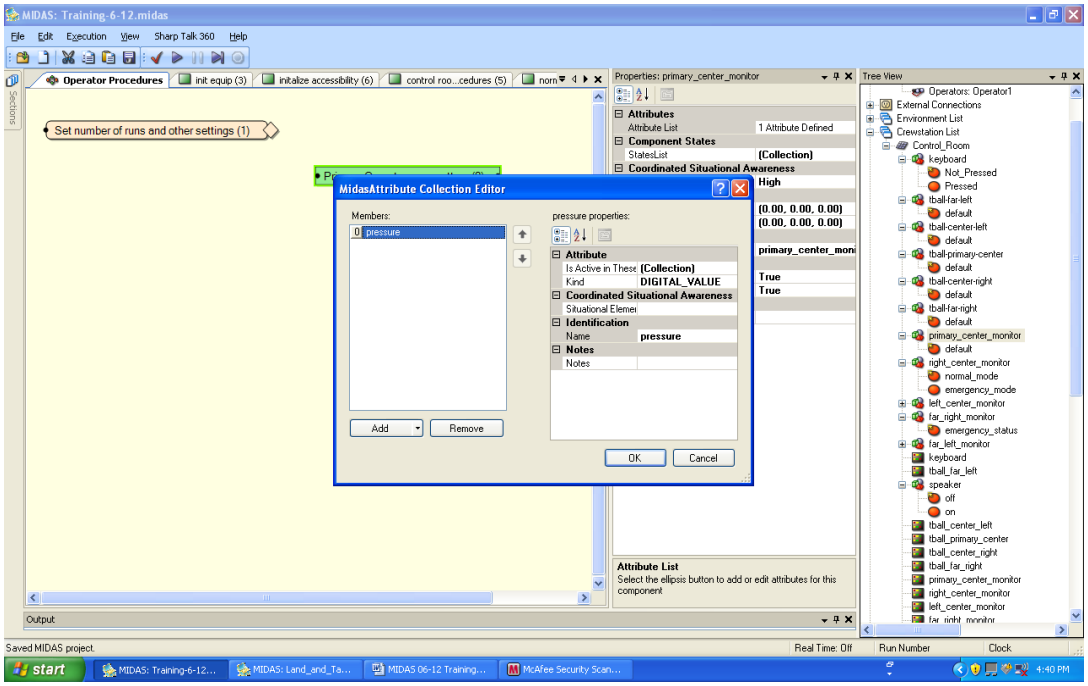


Figure 198. Example of visual attribute definition window.

- Finally, add 2 Auditory attributes to the Speaker
  - One is “auditory alarm tone” that is Sound Emitting. Set the initial state to False (Figure 199).

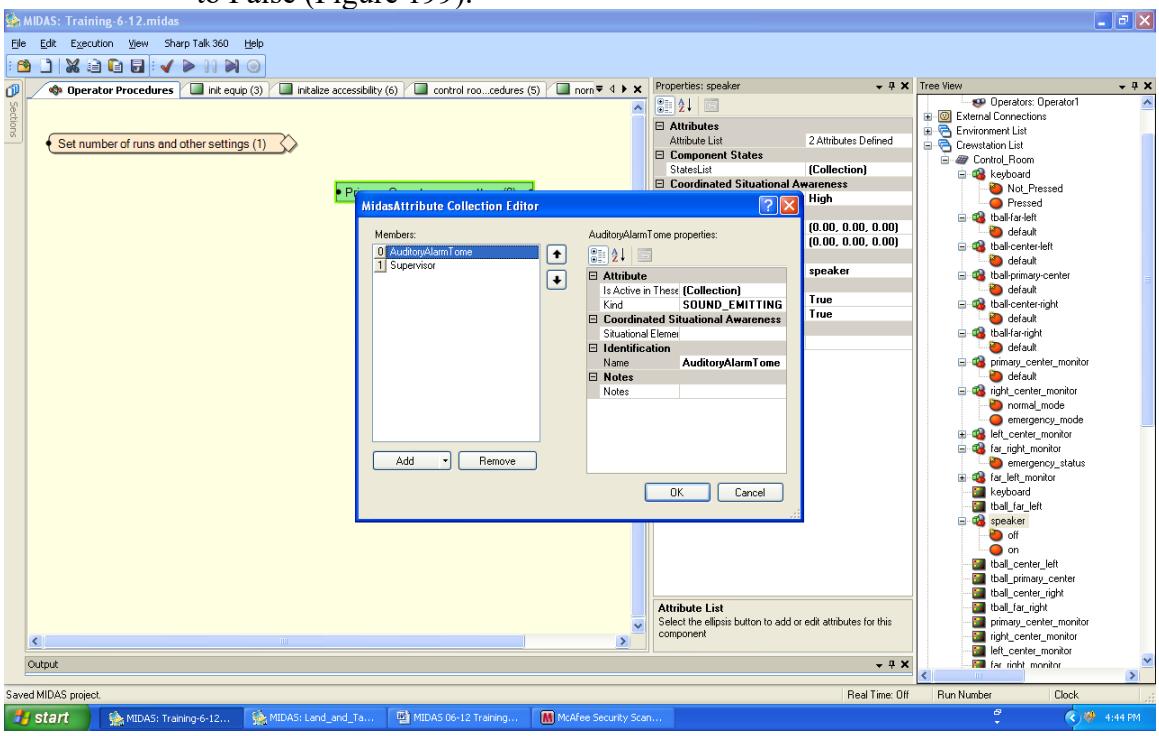


Figure 199. Defining/Setting the attributes on the speaker DSC.

  - The other is “Supervisor”, a Speech attribute (Figure 200).
    - The initial value is set later so you don’t need to type in a string.

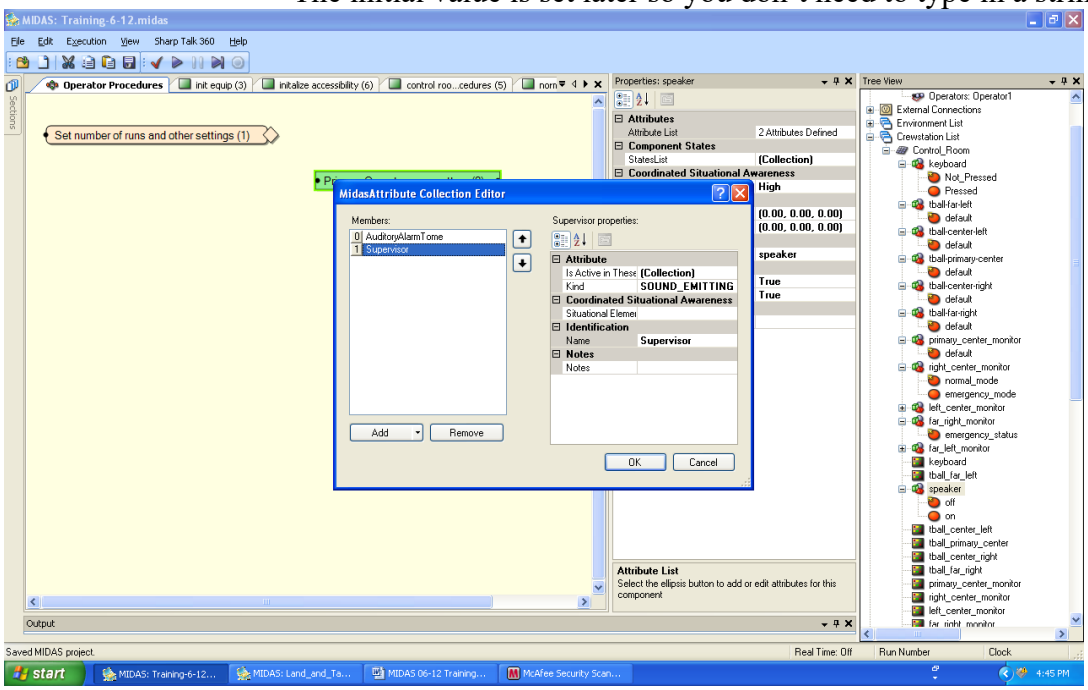


Figure 200. Defining attributes on the supervisor response to the speaker.

Create Scan Patterns

- Select the "Control Room Crewstation" from the Crewstation list
- Create two new seev primitive tasks; one for visual monitoring, and one for auditory.
- Name the first "monitor scan"
    - Double click the SEEV scan icon. This will bring all the defined DSCs into the scan pattern and will assign default attention values for the DSC's.
    - You are able to edit any AOI to reflect the demands of the context you are defining. In this case, leave them at the default values (Figure 201).

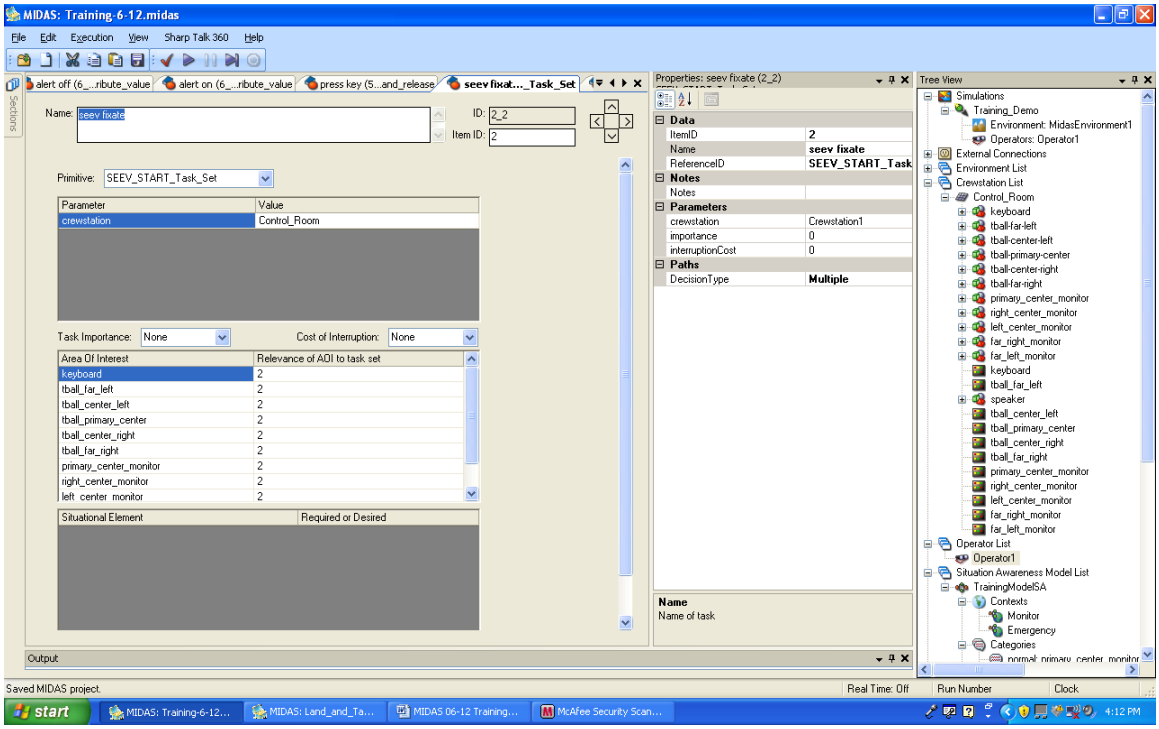


Figure 201. Setting SEEV visual attention model definitions.

- Name the second "emergency scan" (see Figure 202)
    - Create a SEEV_Fixate primitive, "visual" and assign all the wall displays to the "emergency" scan pattern.
    - Create a SEEV_Fixate primitive, "auditory" and assign all displays to the "emergency" scan pattern.

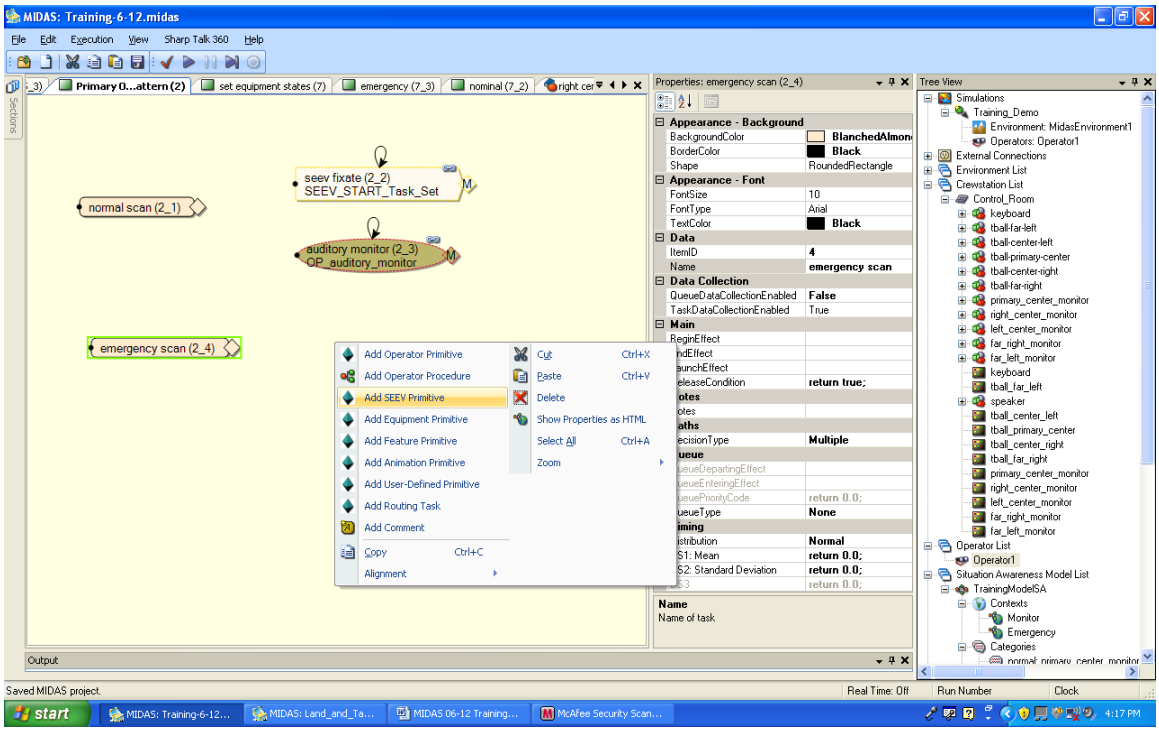


Figure 202. Defining SEEV emergency scan model.

Creating Simulation Events

- Create a new operator procedures task network
- Call it “events”
- Create equipment attribute event networks (these will serve as the triggers for the model to respond to)
  - Add 3 Attribute Events (Figure 203)
  - Name the first one “Alarm Tone”. Set the auditory alarm tone attribute of the Speaker to True. Set the Time to 5000 msec .

    - Hint: Select the check box before the false.

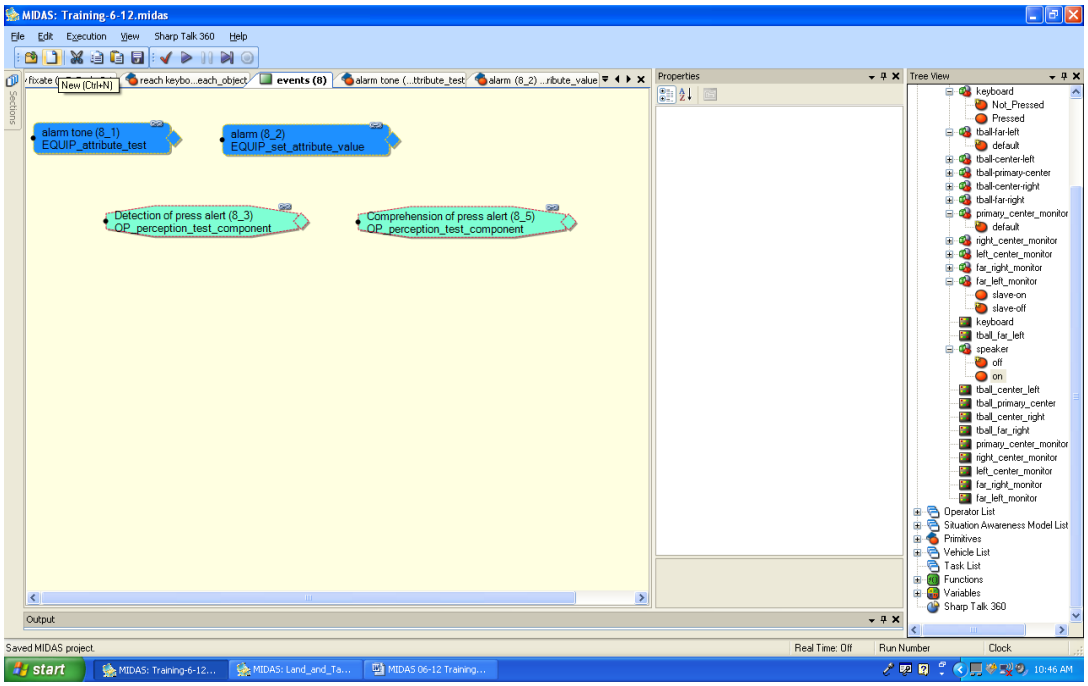


Figure 203. Creating simulation events and the necessary equipment and operator settings.

- Name the second attribute event "Supervisor Request". Select the Speaker once again. This time set the SPEECH attribute to "Check the pressure!" (Figure 204, Figure 205). The time should be set to 5700 msec.

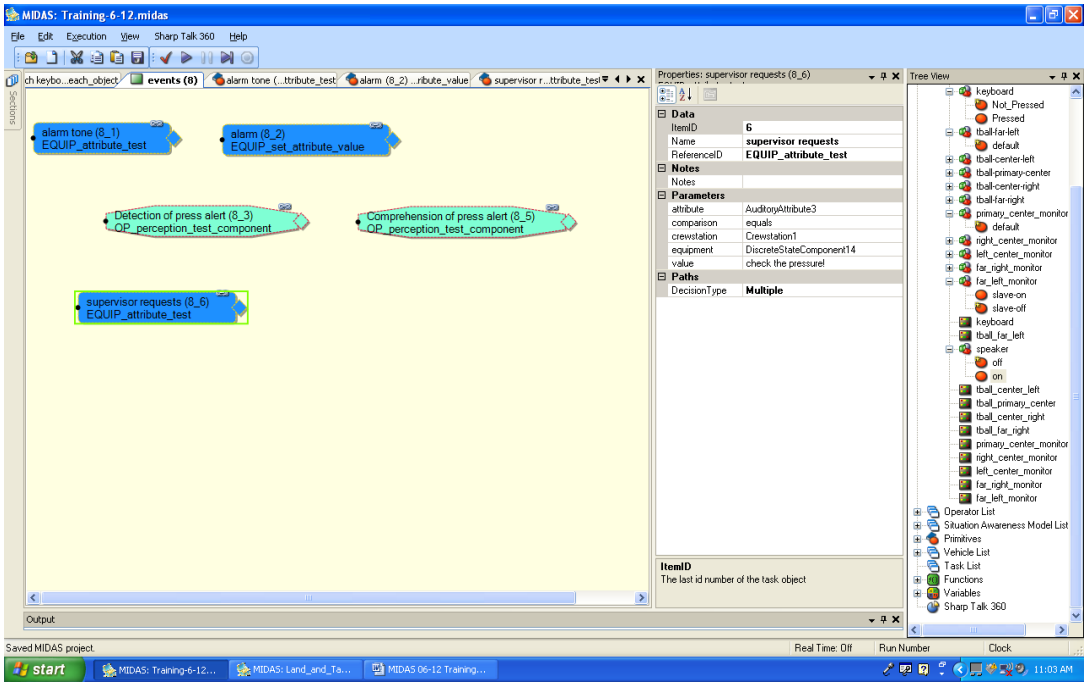


Figure 204. Defining response to the alarm tone.

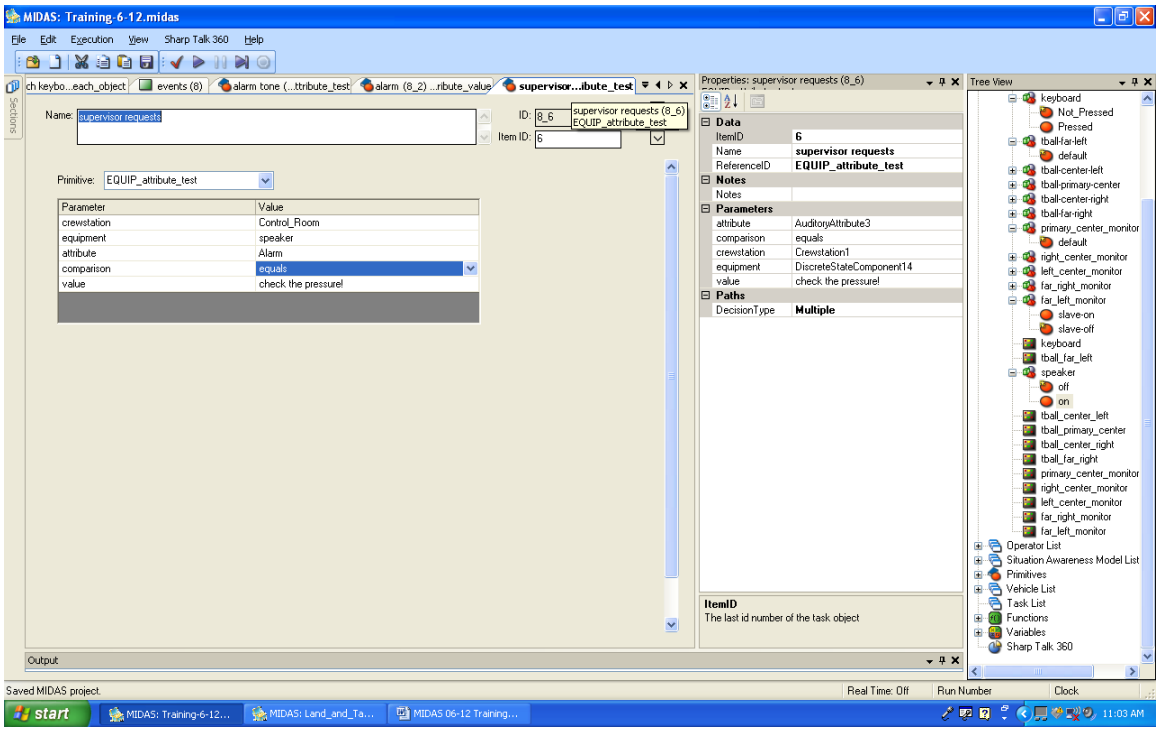


Figure 205. Definition states of the alarm tone when exceedances experienced.

- Name the third attribute event “pressure rises”. The attribute owner is “center wall” and the new value is 100 psi. The Attribute is pressure (Figure 206).

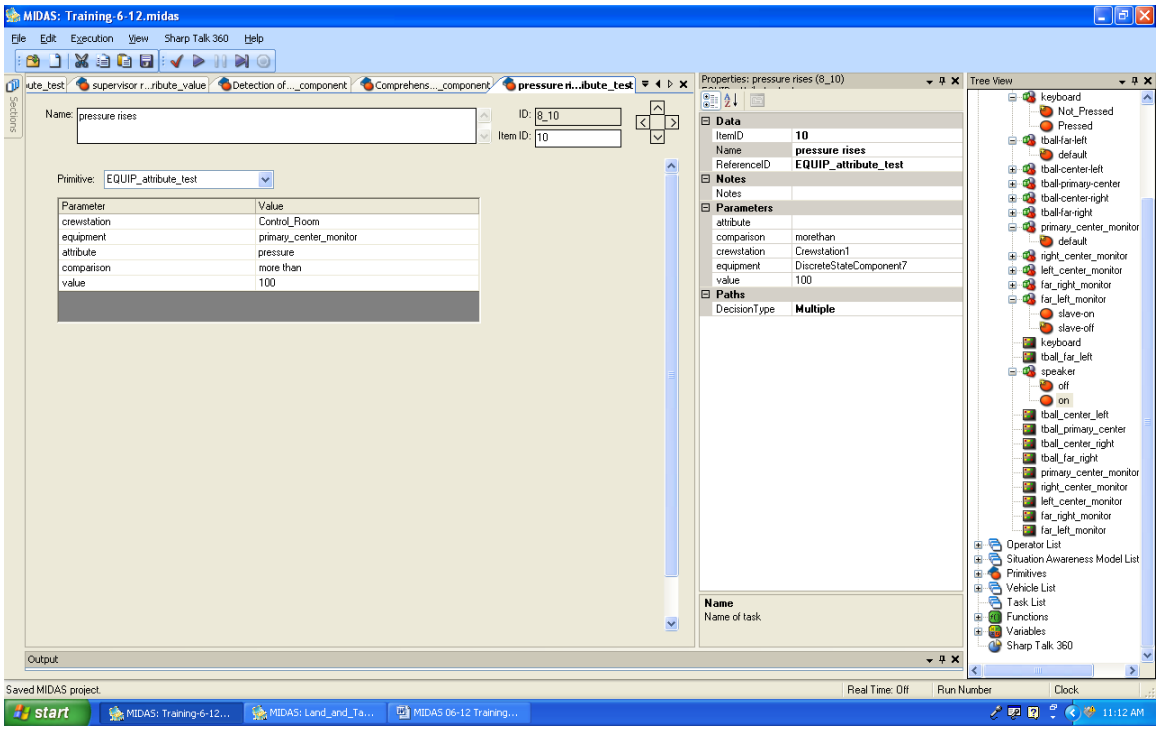


Figure 206. Definition of the attributes needed when events are used in a simulation.

- Add operator primitive, name it comprehension of pressure rises and tie this to the OP_perception_test_component primitive, COMPREHENDED state (see Figure 207, Figure 208).

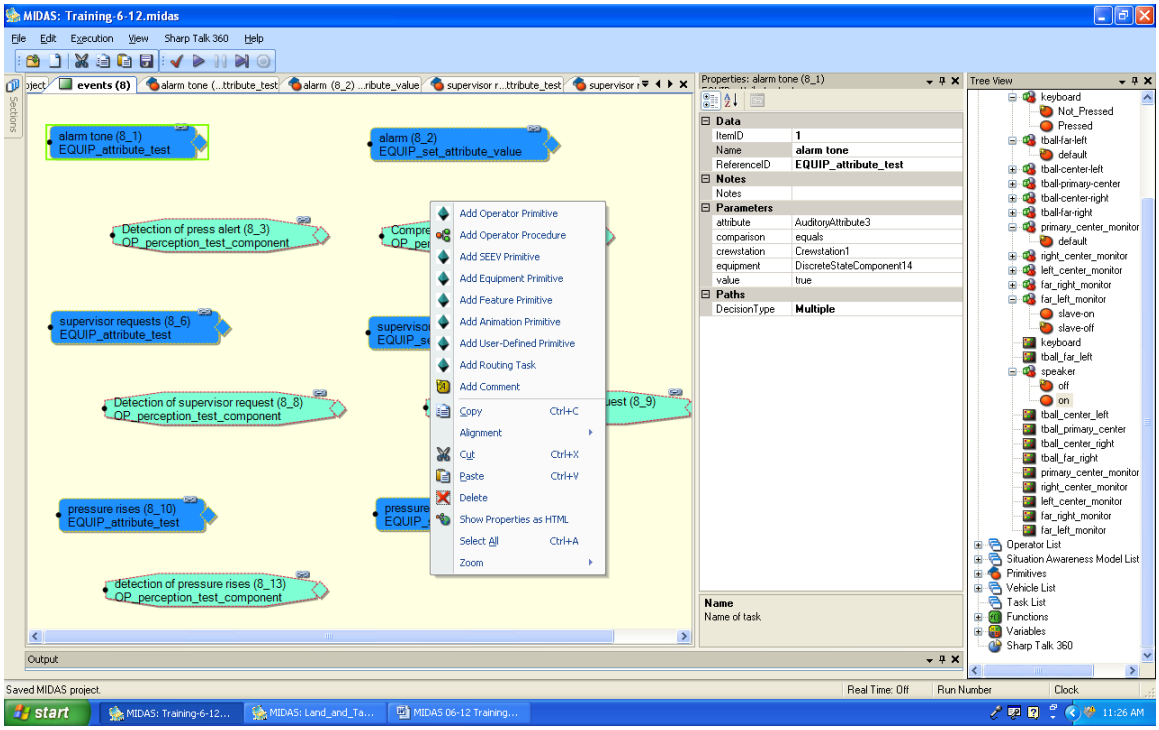


Figure 207. Defining operator primitives in response to equipment states.

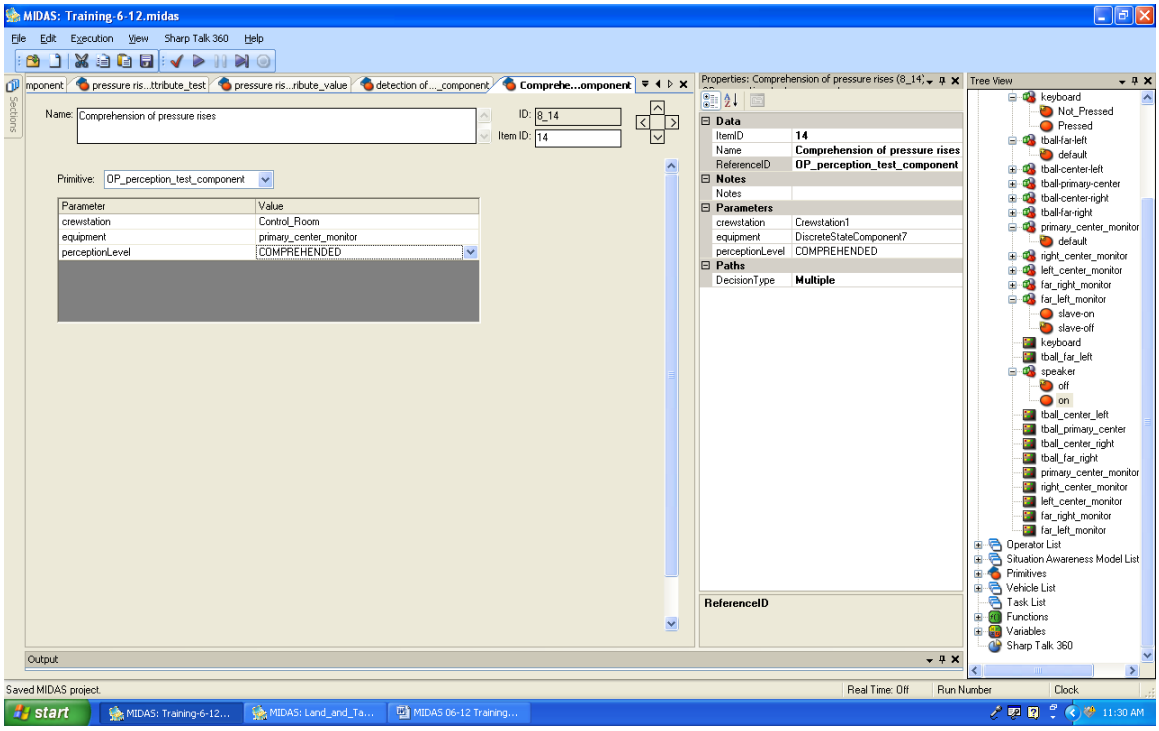


Figure 208. Defining comprehension state of the event.

- Add Context Changes
  - Go to the top level network and create a new network
  - Name it “context specific task settings”
  - Clear the ongoing SEEV scan process with an operator primitive (see Figure 209task 9_2)
  - Create routing task going to two operator procedures, one path going to normal and one going to emergency
  - Add one routing task to link the normal and emergency networks together.
  - We’ll return here after we create the SA model.

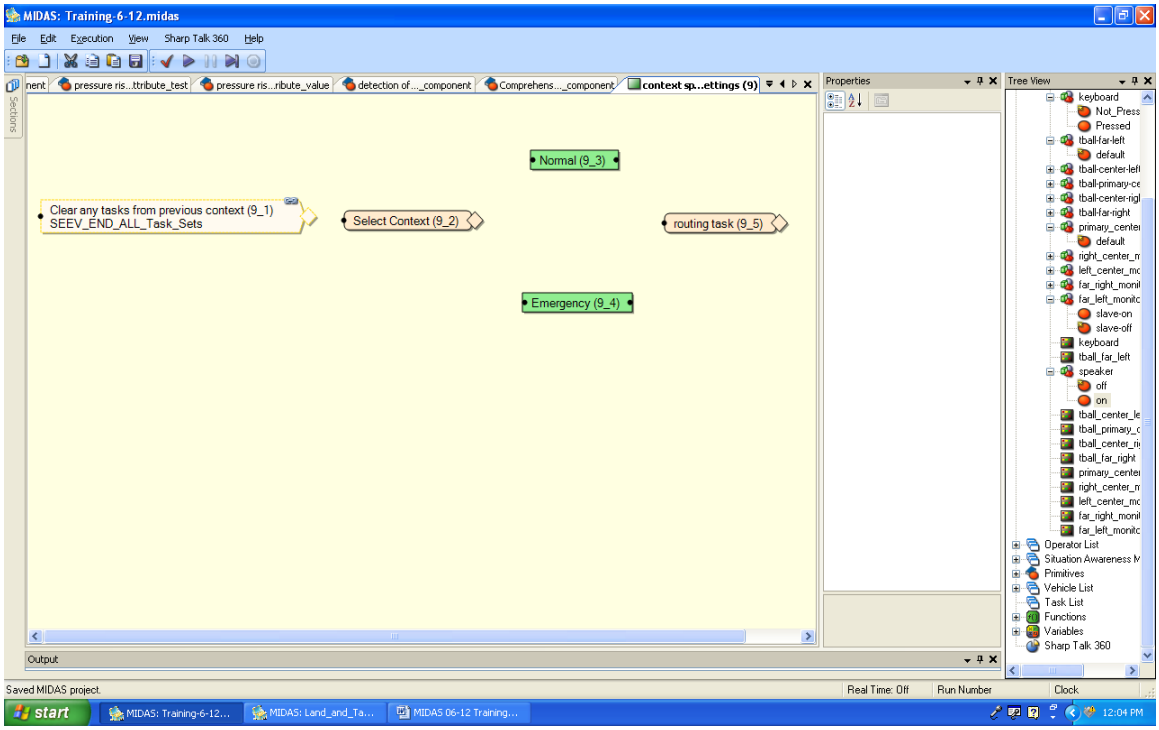


Figure 209. Defining context changes.

- Add One State Change Event
  - Name it “far left monitor hibernates” (Figure 210)
  - The time should be 15000 msec. Select the far left monitor and set the new state to slave off.

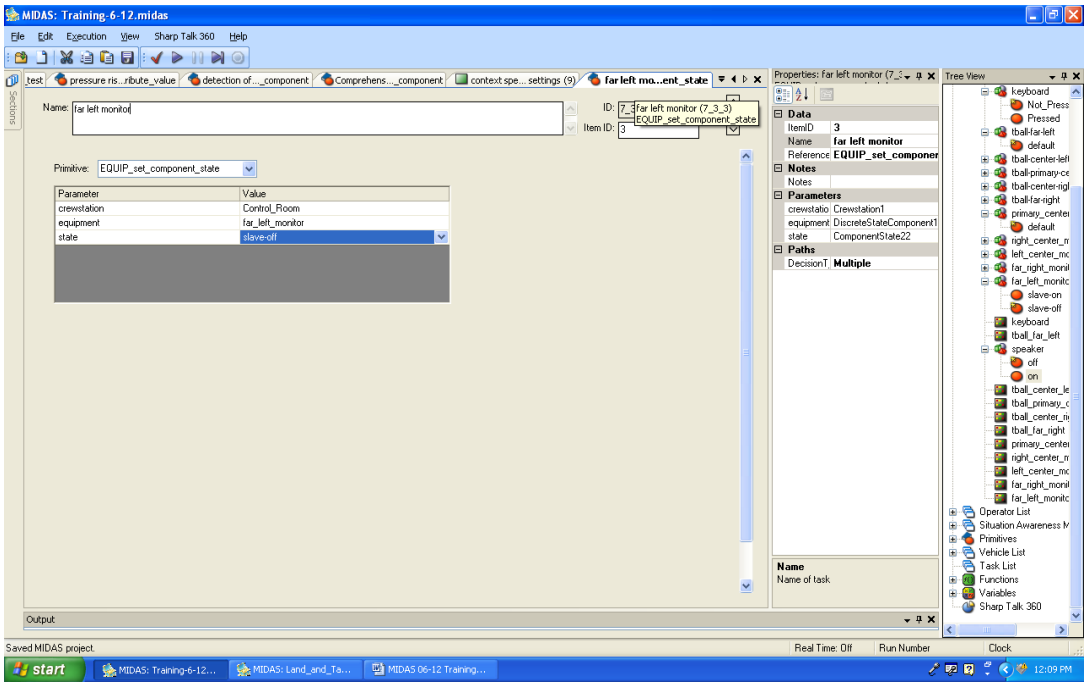


Figure 210. Setting state change events.

Setting up the Operator List

- Select Operator List in the Tree View
- Right Click and select Add Operator from the context menu
- Type “Joe” in the Name field
- Select the Control Room as the Crewstation model.
- We’ll return here after the procedures and SA are defined.

Setting up the Vehicle List

- Right Click on the Vehicle List
- Add Vehicle
- Use the ellipses in the Waypoints field under Guidance model to add one waypoint. The default values are fine since we are not animating the vehicle in this exercise.

Setting up the Situational Awareness Model List

- Right Click on SA model list and add an SA model, named “Situational-Awareness-Model-Training” (see Figure 211).
- This will automatically load the three components necessary in MIDAS’ Situation Awareness model, the contexts, the categories, and the situational elements.
- Add 5 situational elements (SE) corresponding to the following displays (see Figure 211)
    - The center and far right wall displays. There are 3 of these.
        - One center display
        - Two right displays corresponding to the blank and emergency states.
    - The far left wall emergency display

- The primary center monitor

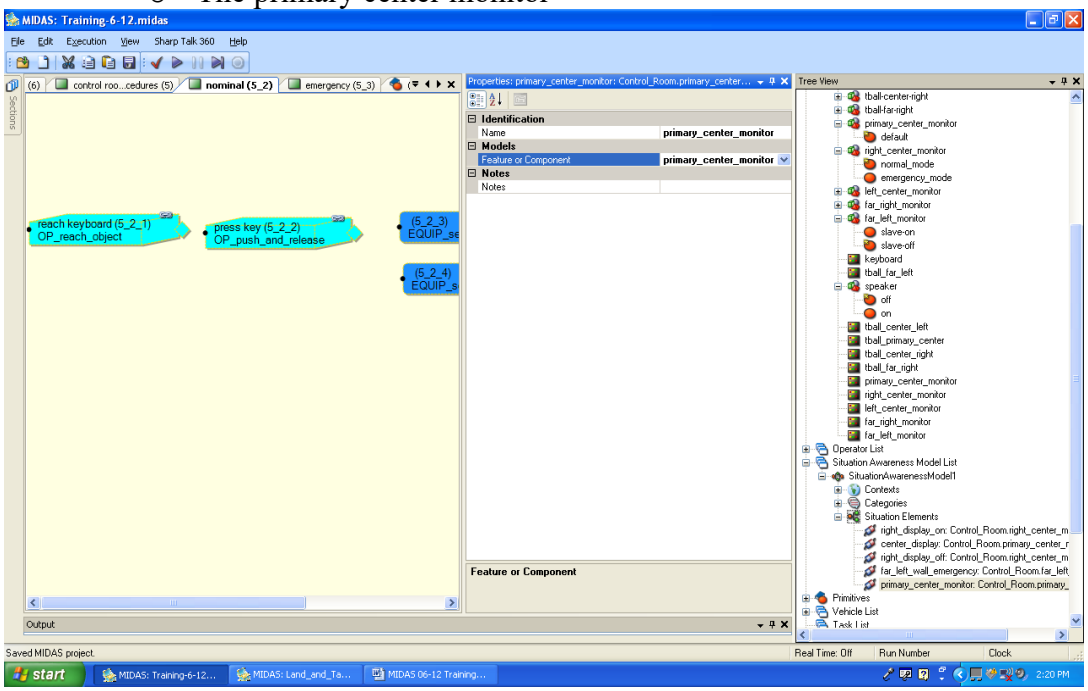


Figure 211. Defining the situation awareness model.

- Create 2 Categories (Figure 212). Nominal monitors and Emergency
  - Assign the primary center monitor SE to the Nominal-monitor Category
  - Assign all the rest, the Wall Displays, to the Emergency Category

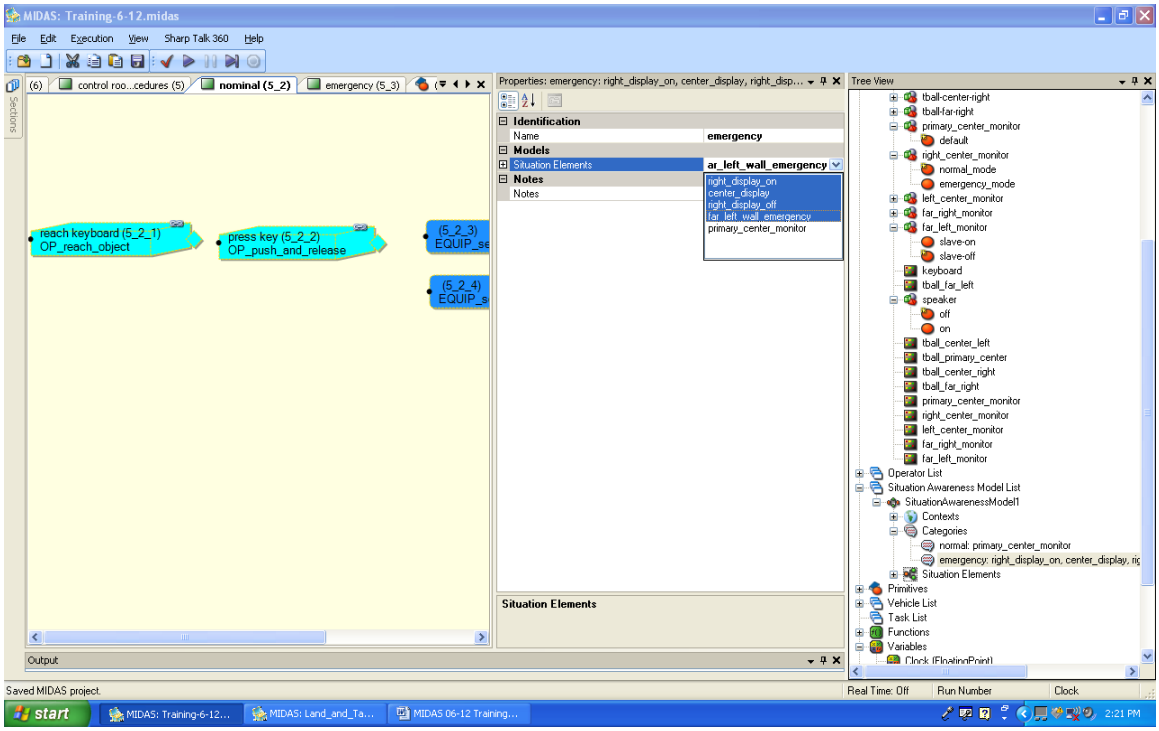


Figure 212. Creating situation awareness categories necessary to drive the SA model..

- Create 2 Contexts (Figure 213)
  - Name them Monitor and Emergency
  - Select the Monitor Context and weight the monitor category a full weight of "1".
  - Select the Emergency Context. Weight the emergency category .8 and the monitor category .2

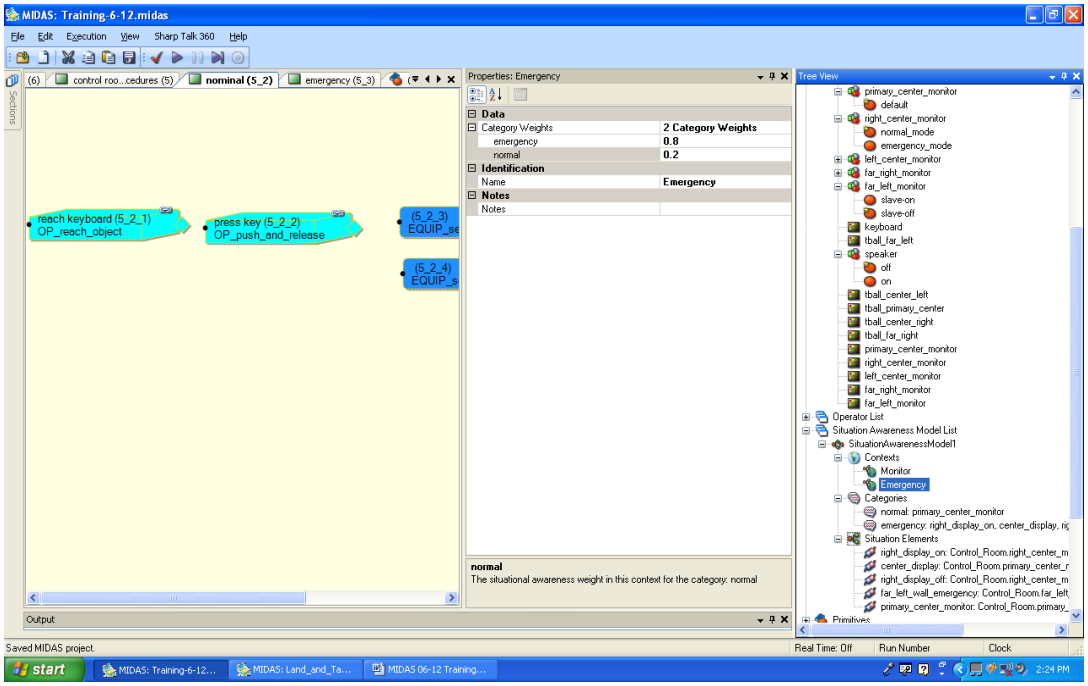


Figure 213. Defining contexts in the MIDAS situation awareness model.

- Return to the operator model in the tree view (Figure 214).
  - Select "Training" as the SA model
  - Select monitor as the initial context for the operator.

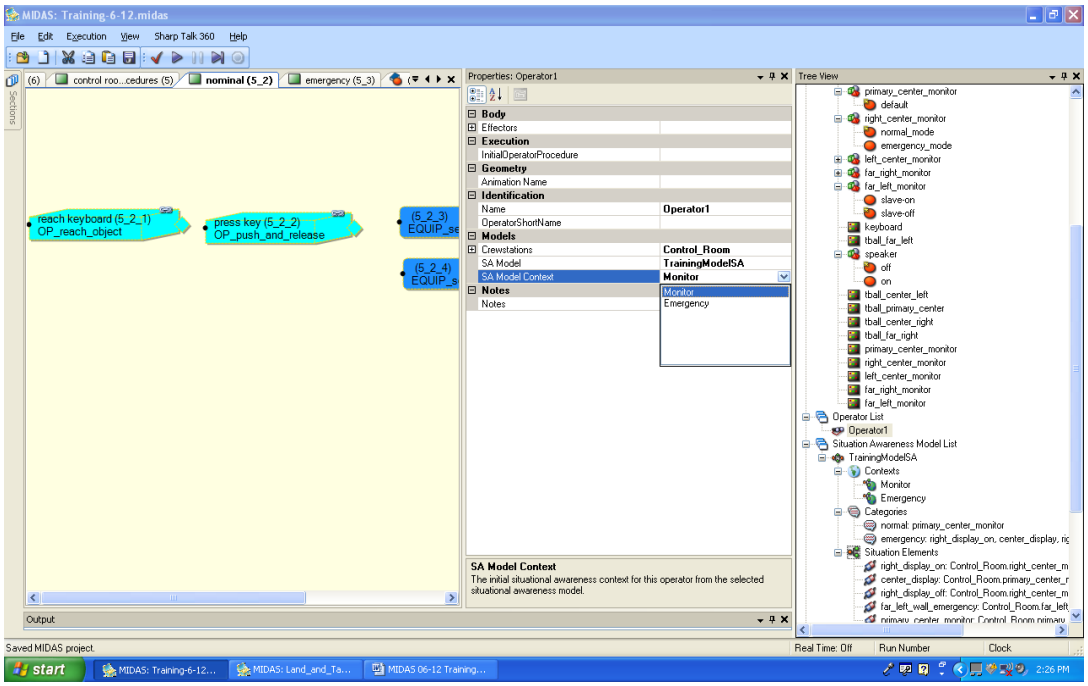


Figure 214. Setting the situation awareness model contexts.

Creating Operator Procedures

- Create 9 task networks in the MIDAS operator procedures tab (see Figure 215). Name them set number of runs and other settings (make this network a routing network), Primary operator scan pattern (operator procedure), init equip, init jack, control room procedures, initialize accessibility, set equipment states, events, context specific settings.

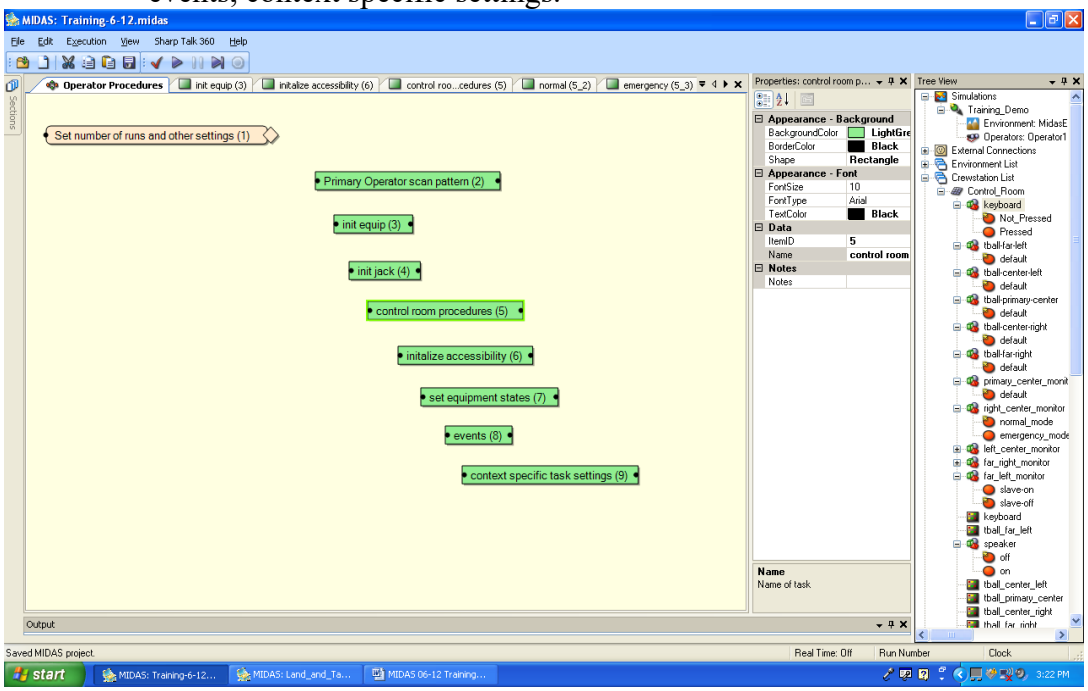


Figure 215. Setting up operator procedures

- Drill down into Primary operator scan pattern (double click left mouse button on the monitor icon)
- Right Click to add the operator primitive “SEEV fixate”.
    - double click the “SEEV fixate primitive”. This will bring up a window where the parameters associated with the scan pattern can be defined (Figure 216).

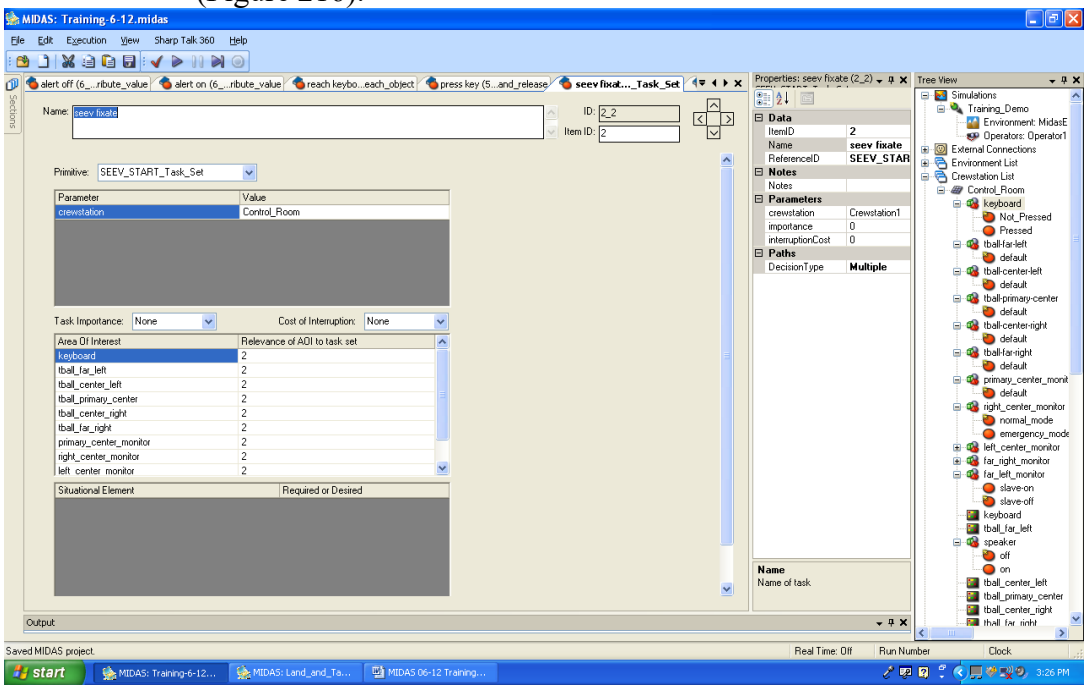


Figure 216. Defining SEEV visual fixate parameters.

- Add an auditory monitoring primitive that will cycle until it reaches a value as defined by the scenario events as needing a response. Associate the auditory monitor with the OP_Auditory_monitor behavioral primitive (Figure 217).

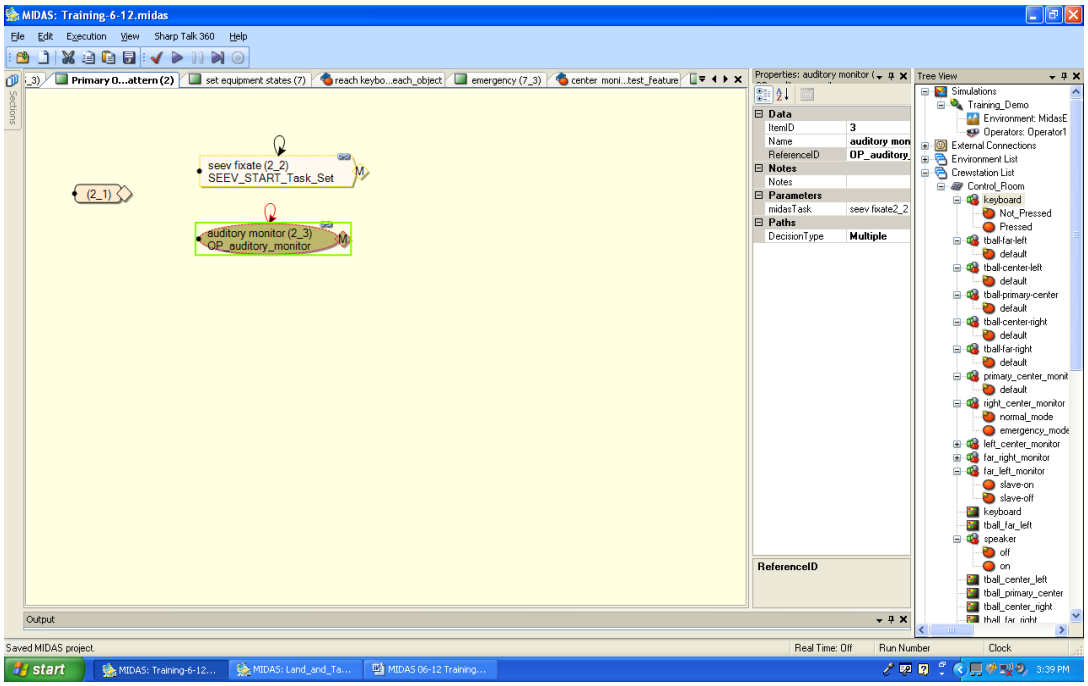


Figure 217. Creating Auditory Monitor primitive.

- Add another task (right click in the Author window). Set the task description to Emergency by typing "emergency" in the name text field
- Add an Attribute Test (right click in the MIDAS procedure task network window)
    - Select Speaker as the object to check, the attribute to check to auditory alarm tone and the value to True.
    - Then go to the "Attribute to Check" and select the "Sound-emittingAuditoryAlarmTone" from the pull down menu
    - Set the attribute test as a dependency to the Emergency step.
    - Hint: Hover the pointer near the right far side of the attribute step until the hand cursor displays. Click and drag to connect the arrow to the left side of the Emergency Step.
- Return to the Emergency procedure in the MIDAS procedure window (Figure 218)
    - Add a check speaker state task and a check right center monitor task, if either are true, then continue to "reach-object" with "right hand" to keyboard.
    - Add an operator primitive "reach-object" with "RightHand" to keyboard.
    - Add an operator primitive "push-and-release" keyboard with right hand
        - Set the state of the keyboard to "pressed"
        - when the keyboard is put into the pressed state, the display on the right center screen will go into hibernation mode.

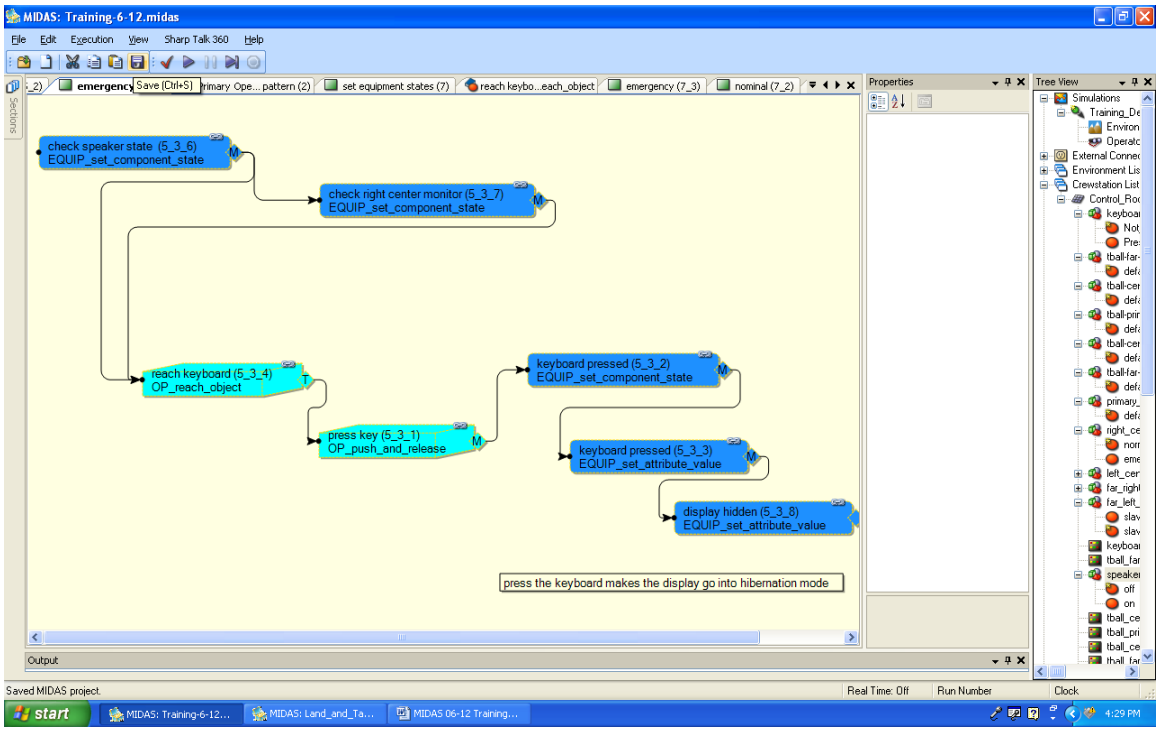


Figure 218. Setting the emergency procedures and states in the simulation context.

- Return to the control room procedures tab and drill down into "emergency" (Figure 219)
  - Add a "say-message" task. Set the argument to "Confirm Pressure is too High" (Figure 220)
  - Add an attribute test checking for the pressure displayed on the center wall screen to be greater than 80psi (Figure 221)
  - Add a perception level test checking for the Speaker to reach Comprehension (Figure 222)
  - Add one more perception test checking for the center wall display to have reached EXACT_READ (Figure 223, Figure 224).
  - Make all three tests dependencies of the "say-message" step.

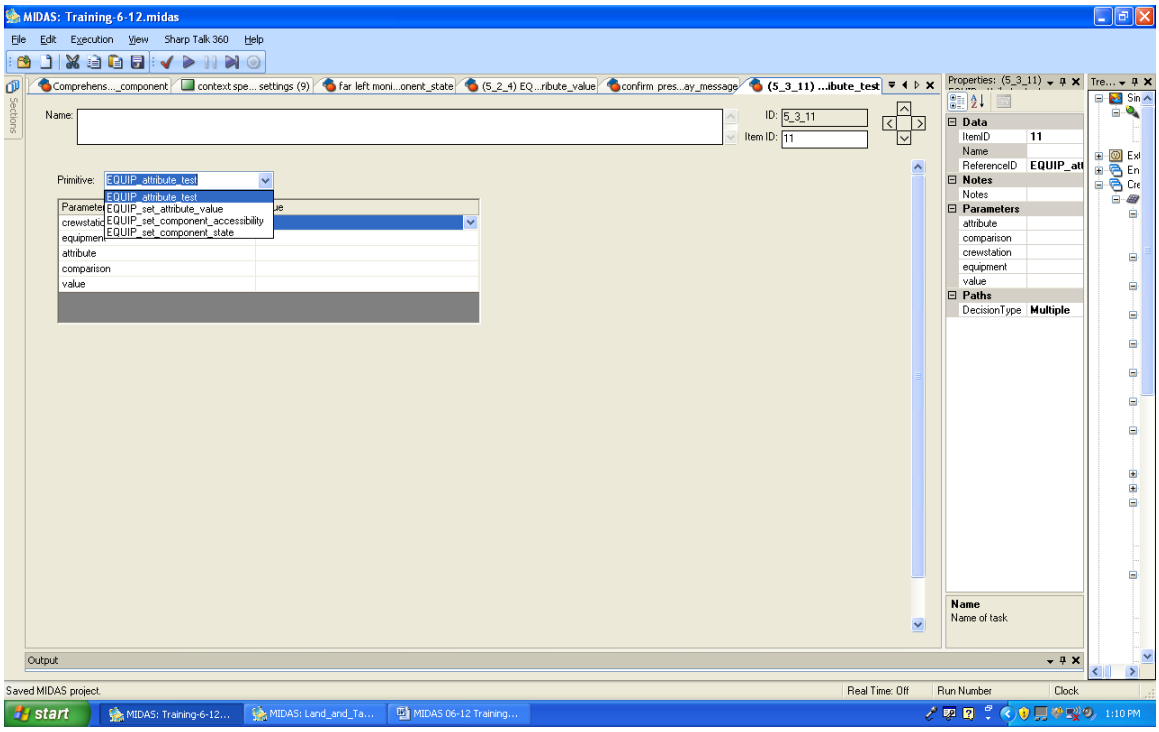


Figure 219. Emergency control room procedures.

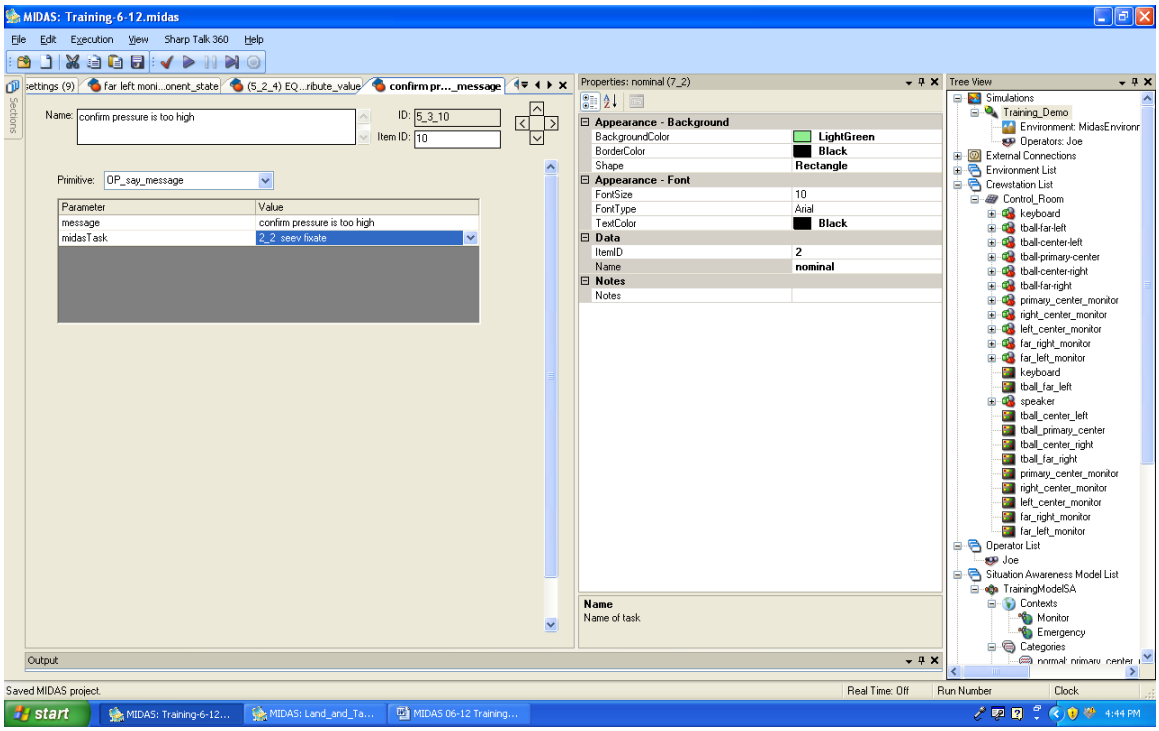


Figure 220. Confirm pressure is too high.

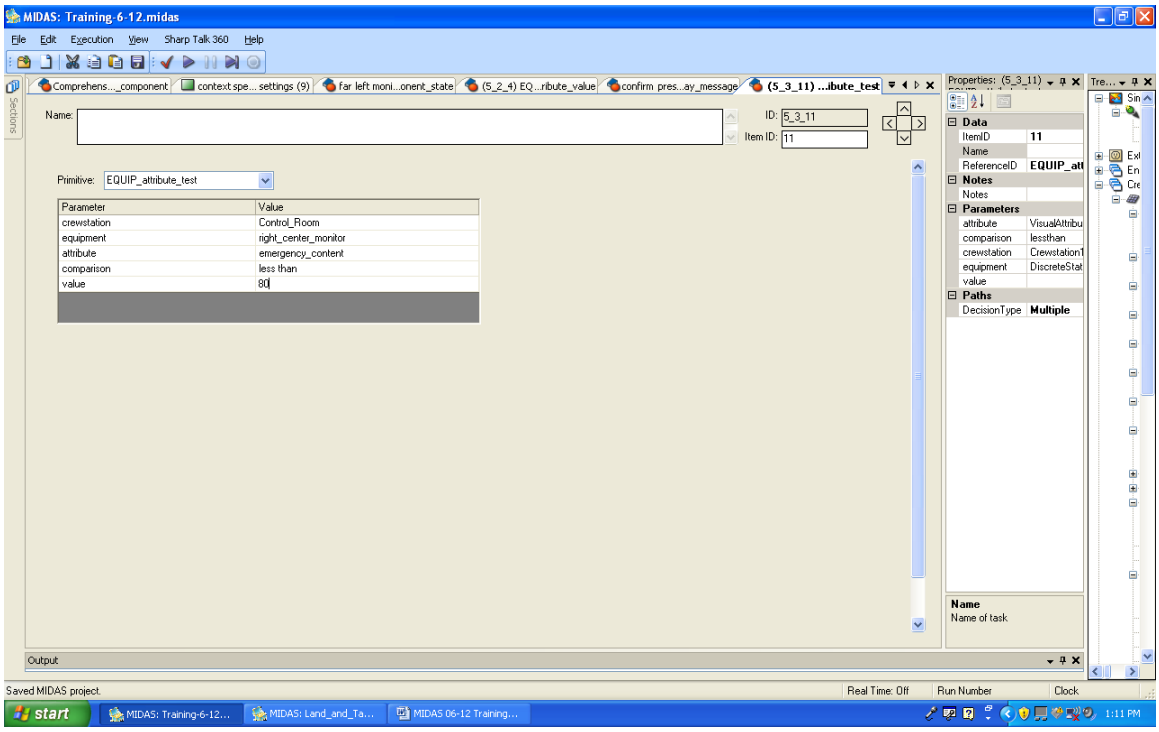


Figure 221. Example of attribute Test of the primary center monitor in the emergency context.

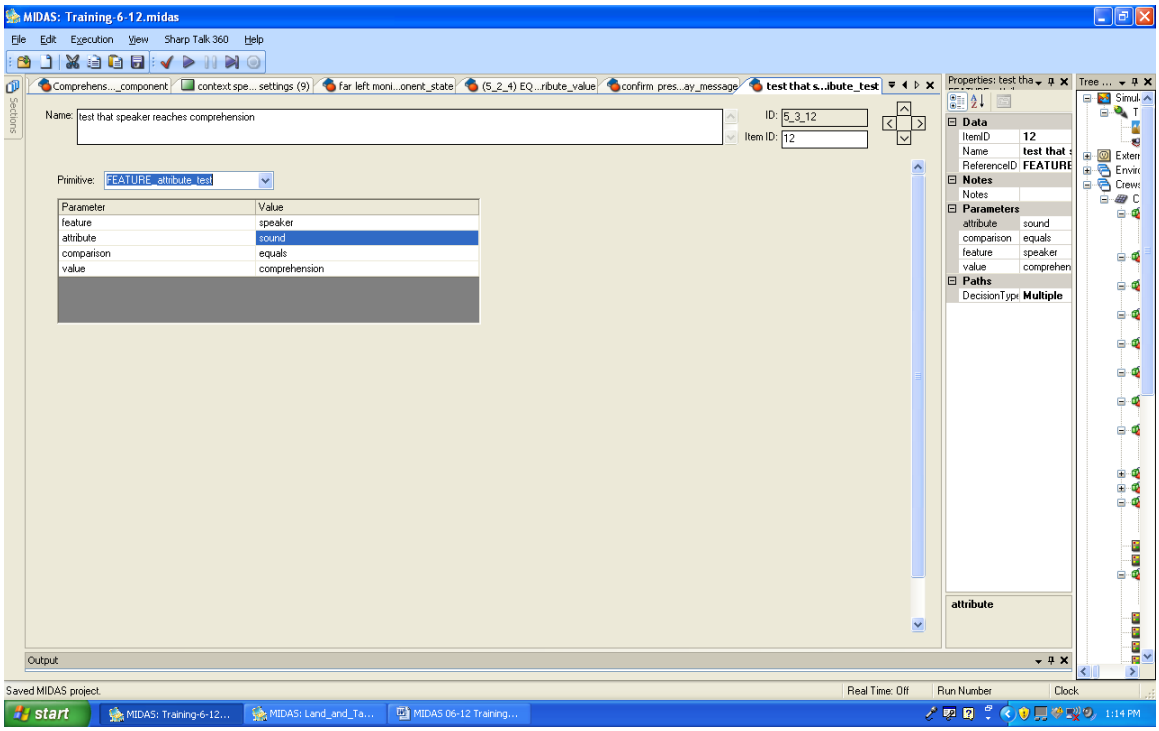


Figure 222. Test for speaker to reach comprehension.

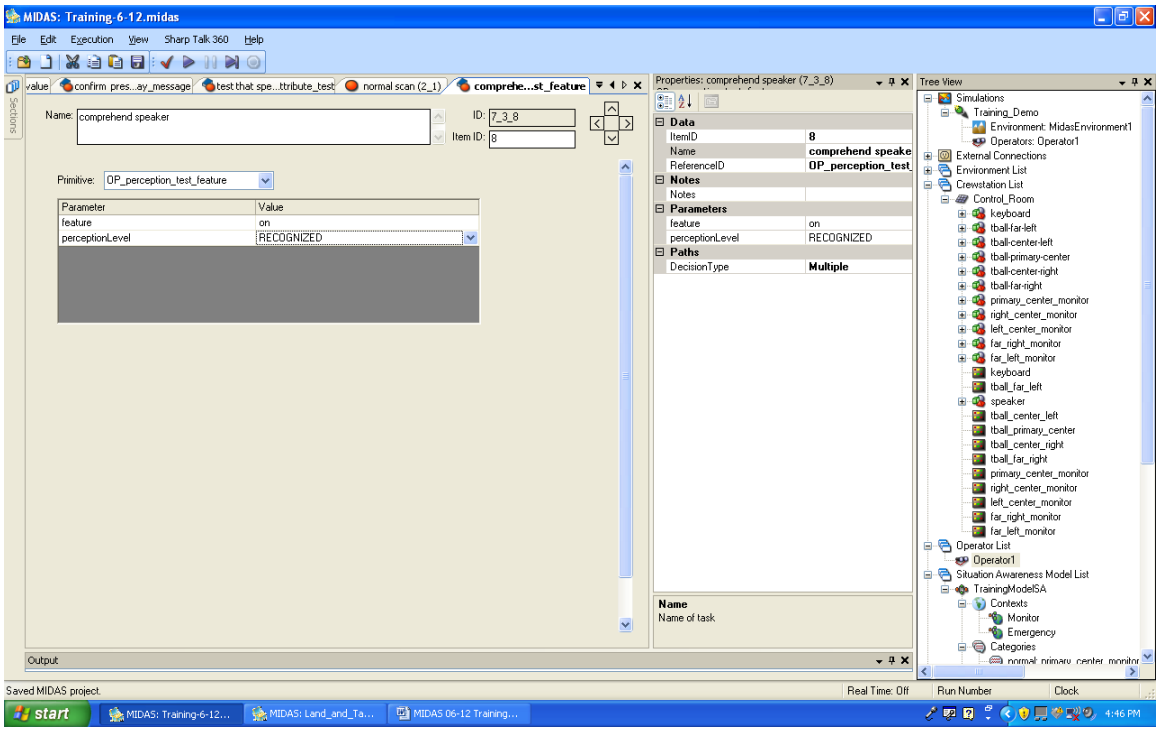


Figure 223. Perception test for the speaker feature.

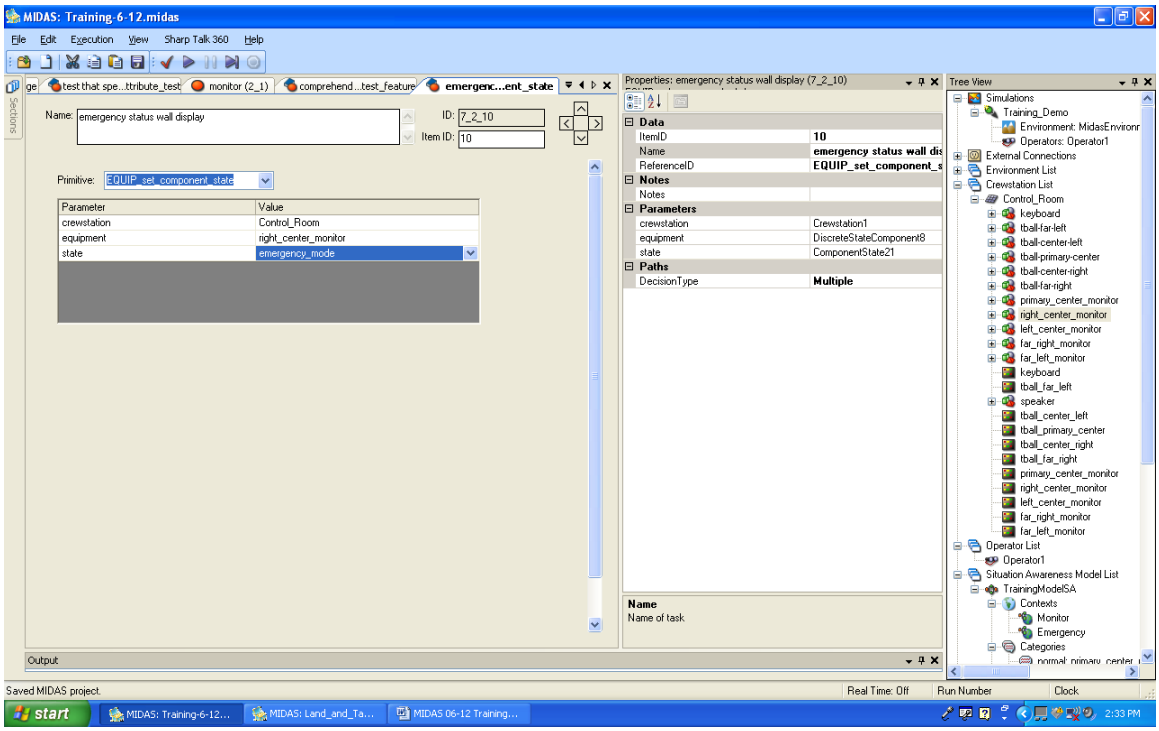


Figure 224. Emergency status wall display.

- Return to the operator, “Joe”
  - Set Monitor as the initial procedure that “Joe” will begin with when the simulation begins (Figure 225).

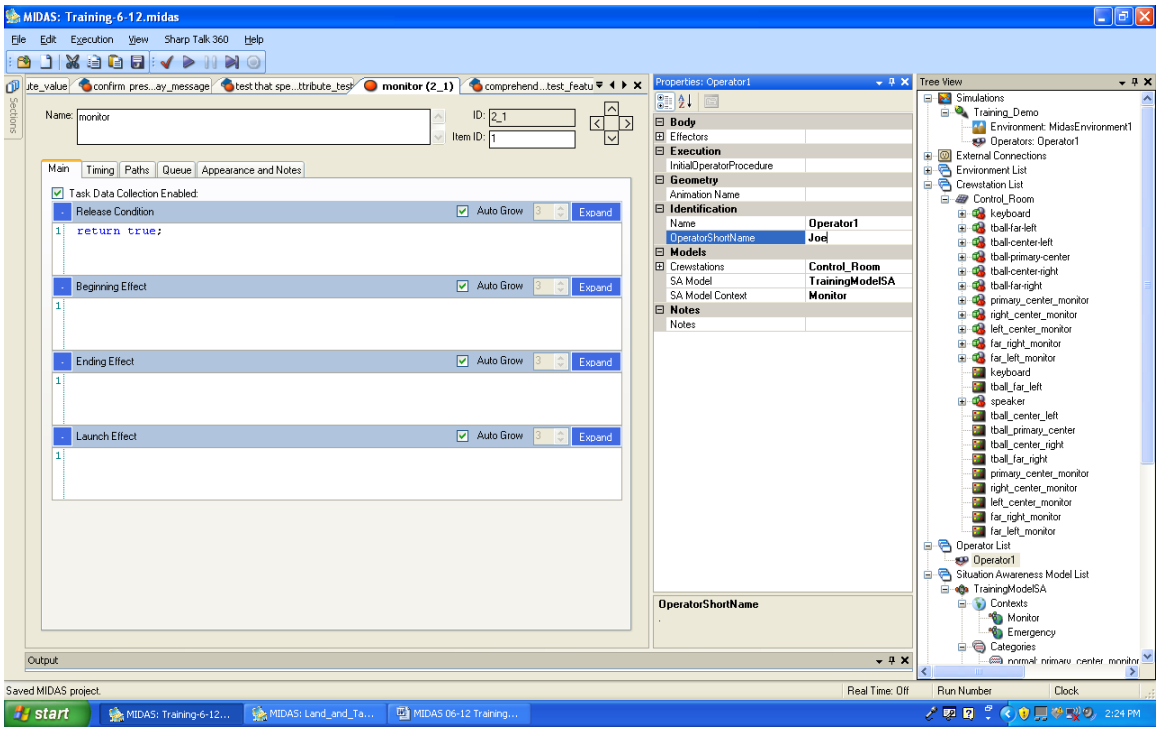


Figure 225. Example of setting the SA model context to Monitor.

Assigning Models to a Simulation

- Return to the Simulation, “TrainingDemo”
- Assign Environment to Environment 1
- Assign “Joe” as the operator
- Assign Vehicle to Vehicle 1

Chapter 6: General MIDAS Software Runtime Functions

Check the model for errors

- Click the red checkmark icon in the MIDAS toolbar at the top of the screen. When the mouse cursor is over the icon it reads Check for Errors
- This will check the model you created for errors that must be fixed before you can begin running the model
- Note the results in the Output window

Play

- Click the forward arrow icon in the MIDAS toolbar at the top of the screen. When the mouse cursor is over the icon it reads Begin Simulation
- Note the Output window scrolling the information, and in the bottom righthand corner of the status bar the Clock ticks off the time. Later we'll look at the Output window in detail
- Verify that the Jack™ Command Window status has changed to a green Connected

Halt

- If the model is still running, click the stop sign icon in the MIDAS toolbar at the top of the screen. When the mouse cursor is over the icon it reads Halt Simulation.
- This causes the model to stop running no matter where it is in its execution.

Pause

- This causes the model to pause. Depress the pause again to release the pause.

Continue

Step

- This causes the model to step 100ms at a time through its execution.

To Run an Existing MIDAS Scenario

1. Open Midas and select your desired .midas file (the current example comes from a model that was developed for the Federal Aviation Authority (FAA) in 2010).
2. Open MicroSaint Sharp and select your desired .saint file (the current example comes from a model that was developed for the Federal Aviation Authority (FAA) in 2010).
3. Press the play button (Control+G) in Midas. This will open Sharp Talk 360 (Figure 226).

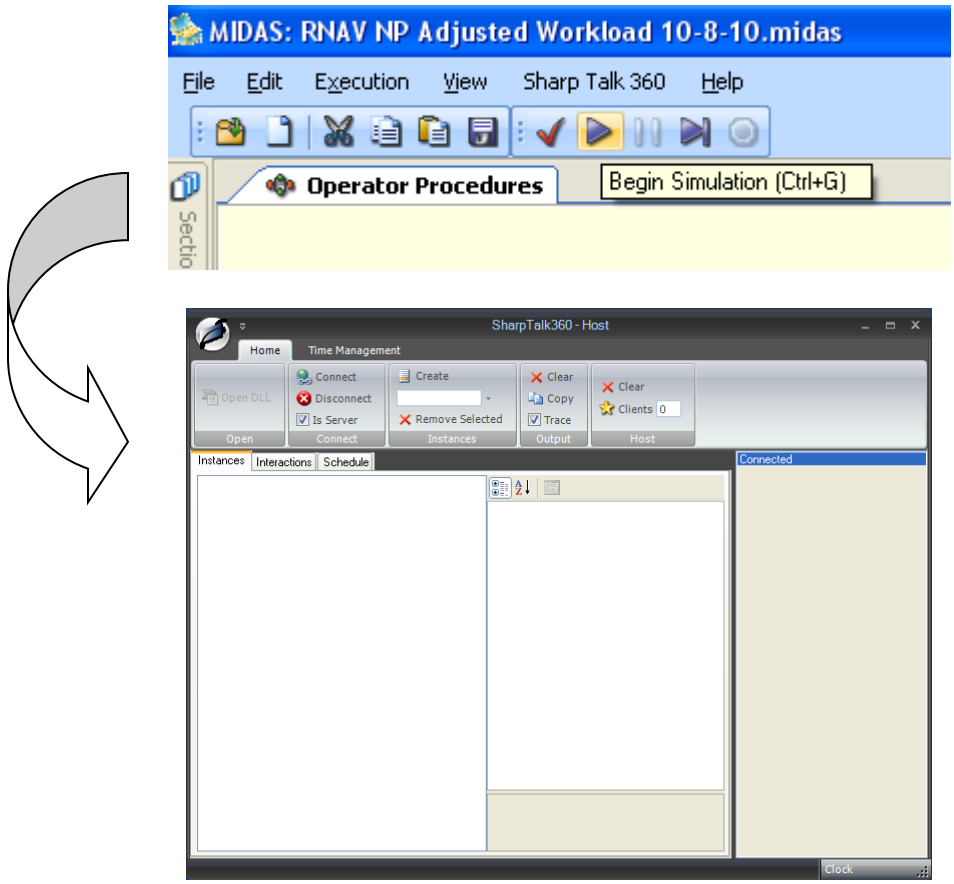


Figure 226. Launching MIDAS and Sharp Talk 360 (the controller of the MIDAS Software).

4. In the Midas toolbar, select Sharp Talk 360 drop down menu > Connect (Figure 227)

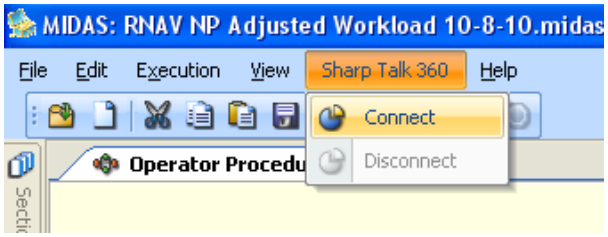


Figure 227. MicroSaint Sharp Talk 360 in the MIDAS software - Connecting Sharp Talk.

5. In the MicroSaint Sharp toolbar, select Sharp Talk 360 drop down menu > Connect (Figure 228)

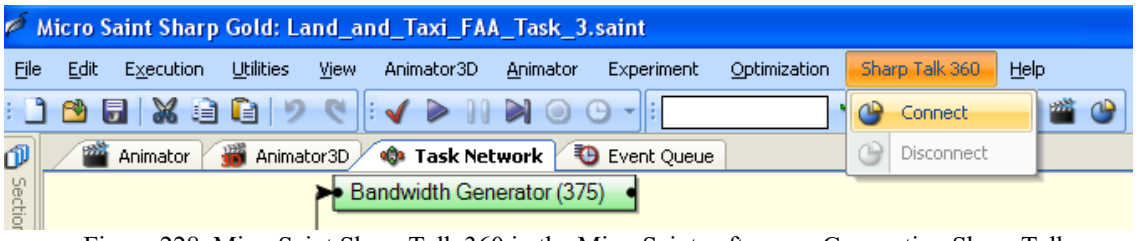


Figure 228. MicroSaint Sharp Talk 360 in the MicroSaint software – Connecting Sharp Talk.

6. You should now see that both Midas and MicroSaint Sharp are connected in the output box of Sharp Talk 360 – Host (Figure 229)

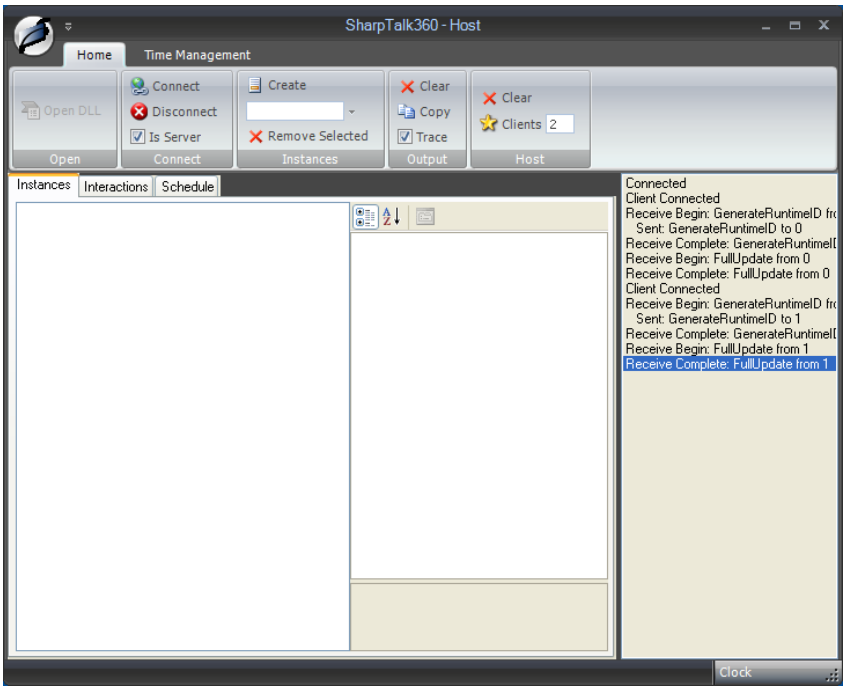


Figure 229. Screen appearance - MIDAS-MicroSaint successfully communicating.

7. Select the time management upper tab in Sharp Talk 360 – Host and Press the blue play button in the control box on the top left (Figure 230)

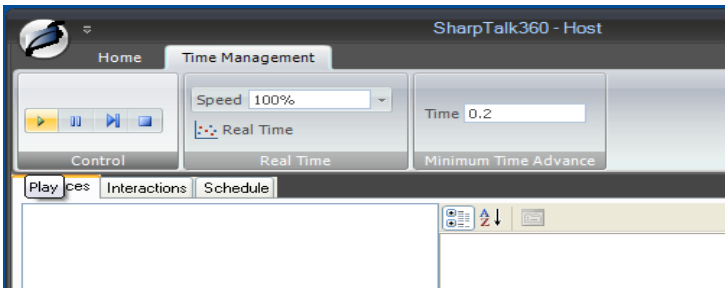


Figure 230. Time Management tab in Sharp Talk 360.

8. The simulation has now been initiated. It usually takes about 30 seconds before you can see the runtime processes in the MicroSaint Sharp Window (Figure 231).

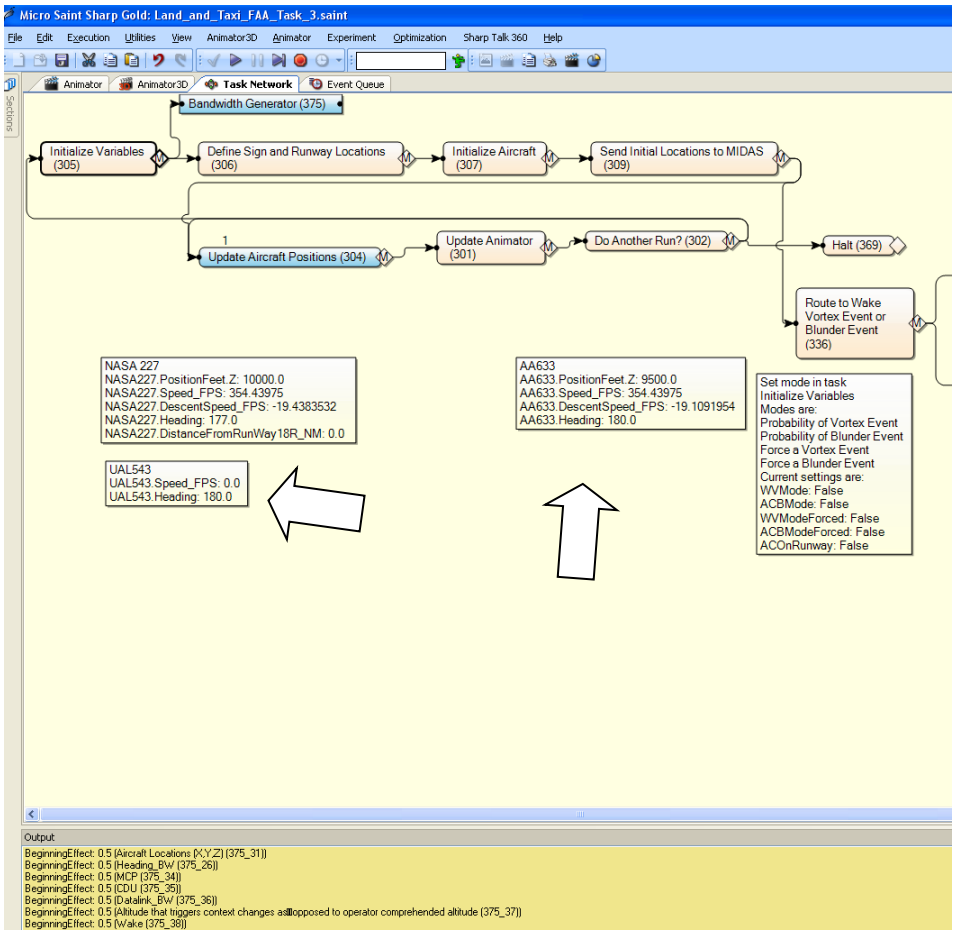


Figure 231. A Runtime example of a successfully linked simulation.

SharpTalk 360 – Host is used to control the simulation. You can pause it, stop it, step through it, speed it up, or slow it down. If the trace box is checked in the output box of SharpTalk 360, you will see a dynamic list of tasks on the right side (above the clock). Depending on your model, typical time from start to finish is about 15 minutes.

Note: don't leave folder open where your Midas file is during simulation because it can cause the simulation to crash sometimes. Also, the simulation will run about 50% faster if you minimize the MIDAS and SAINT windows because it doesn't have to load the GUI's.

Running a Monte Carlo Simulation

1. Open MIDAS and select Module: Set Number of Runs & Other Settings (75) (Figure 232).

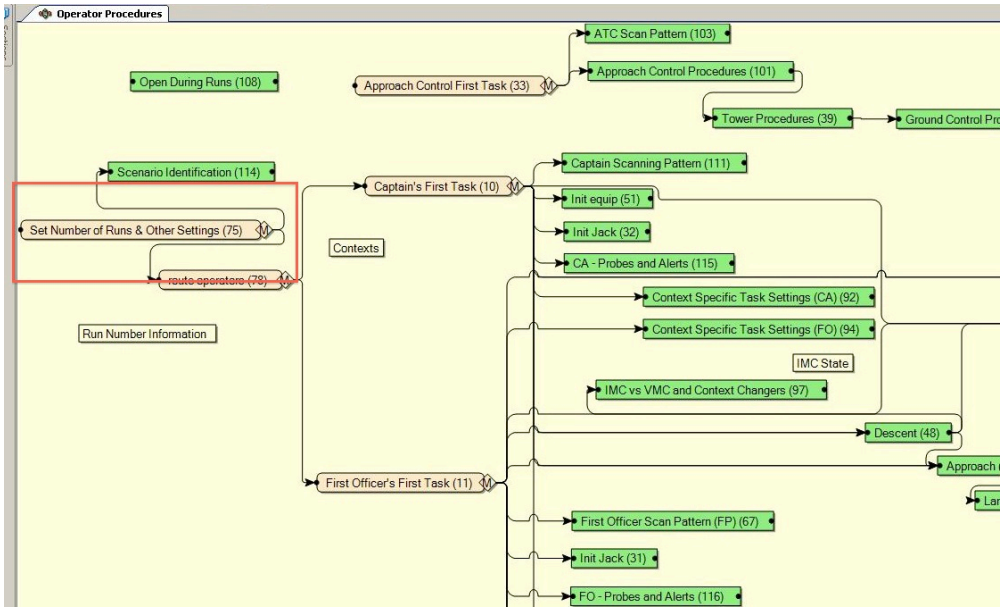


Figure 232. Adjusting the number of runs for a Monte Carlo simulation.

2. Adjust NumberOfRuns by retyping your desired number of runs and change comment to usually 1. A typical number for monte carlo runs is 10 (Figure 233). Note: do not adjust this anywhere else in MIDAS.

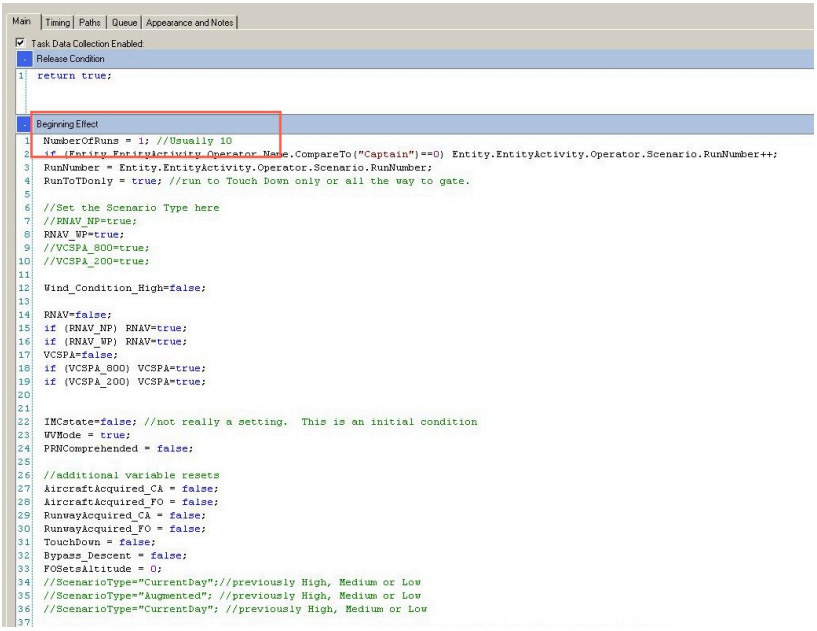


Figure 233. Adjusting the number of runs in MIDAS.

Manipulating Existing MIDAS Scenarios

1. Open MIDAS and select Module: Set Number of Runs & Other Settings (75) (Figure 234).

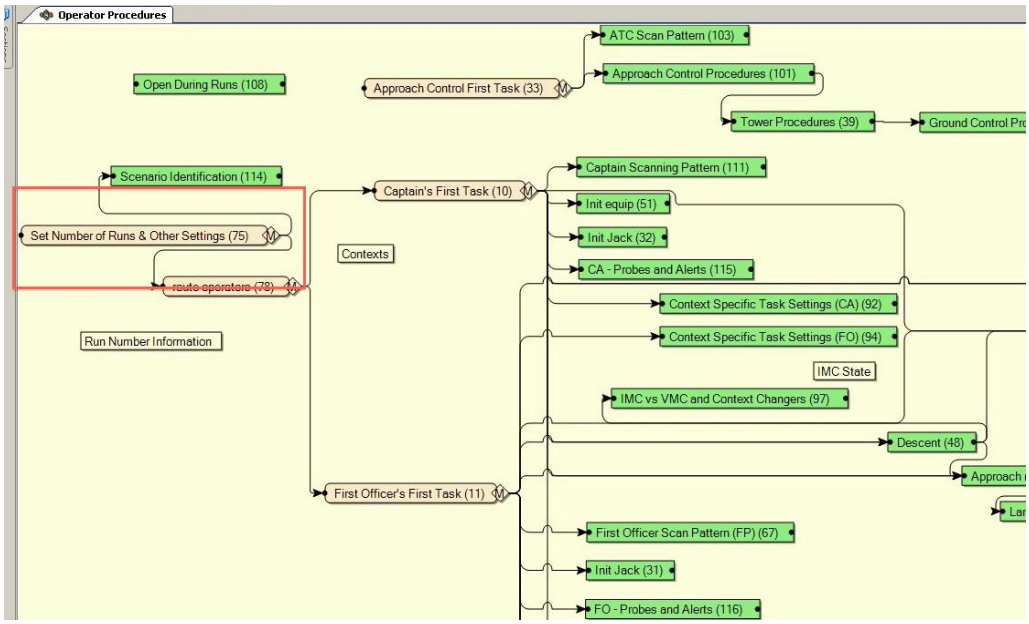


Figure 234. Manipulating MIDAS settings in an existing scenario.

2. Adjust code in beginning effect box. To turn something off, type two "/" marks consecutively to comment it out. To turn something on, delete the two "/" marks (Figure 235).

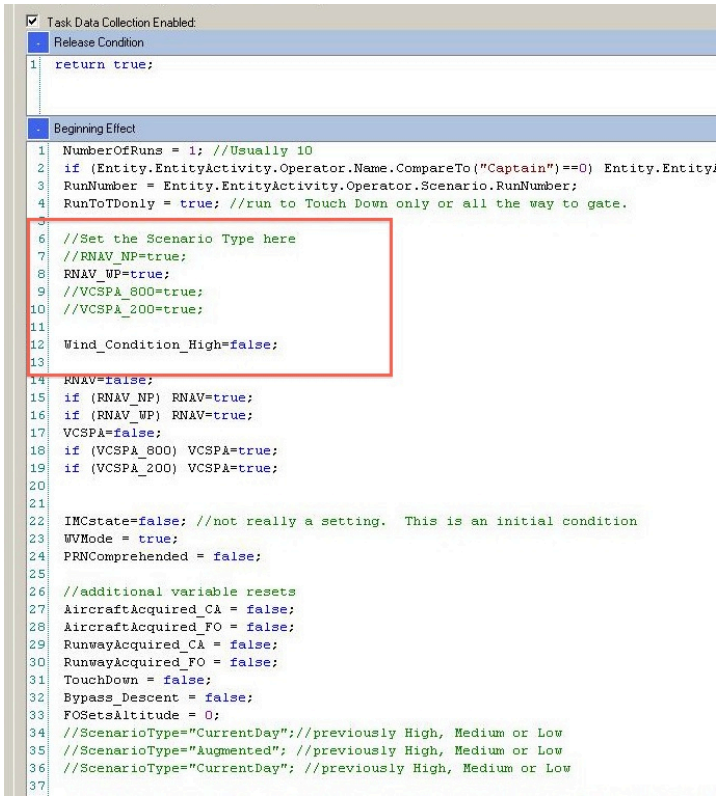


Figure 235. Commenting out scenario settings in MIDAS.

Modifying Operator Phraseology

1. When modifying operator phraseology, both the name and parameter values must match (Figure 236).

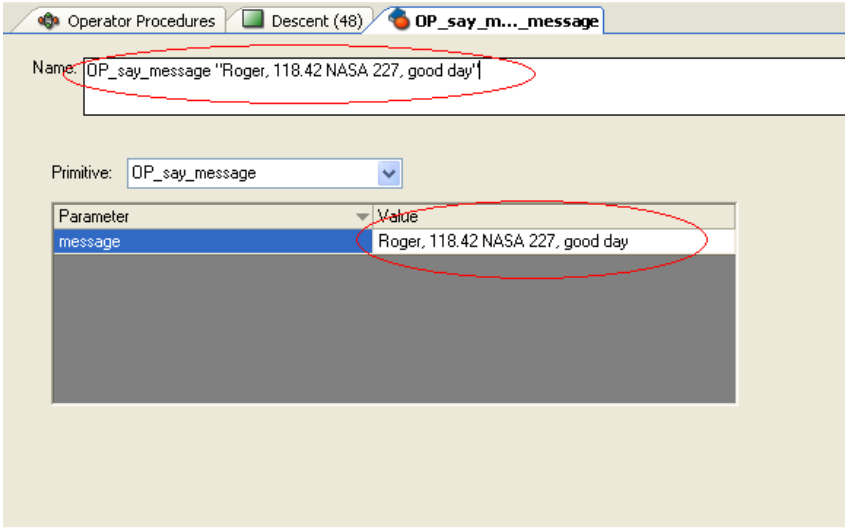


Figure 236. Operator phraseology must match in name and parameter locations.

2. After you do this, make sure that the op 'say message' values match the op 'listen to' and any other tasks tied to it (e.g. wait for) (Figure 237).

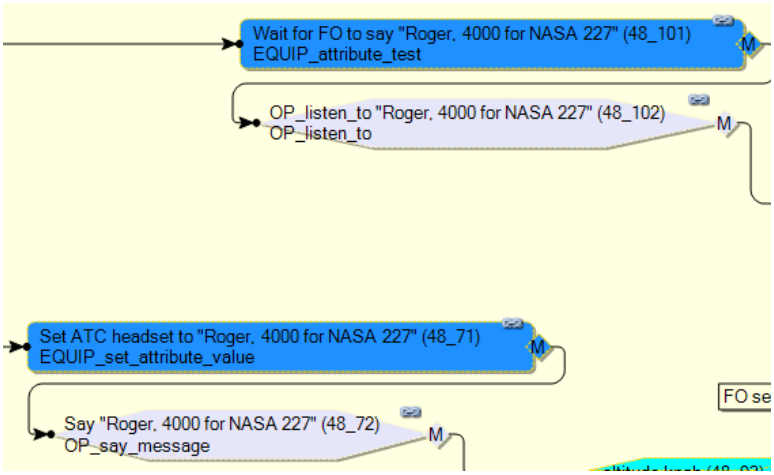


Figure 237. Ensure that the OP say and listen to match.

In this example, all the text within quotations is the same for the wait for, op listen to, and op say message primitives "Roger, 4000 for NASA 227"

Modifying Workload Management Strategies

Method One

1. Open up a primitive (Figure 238)
2. Tab over to Parameters tab
3. Look for fourth parameter called Midas task (will receive in new build)
4. Select one of 5 tasks (Aviate, Separate, Navigate, Communicate, Systems) and make further adjustments here

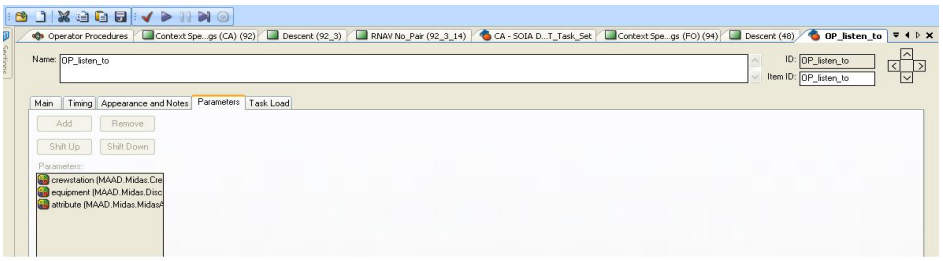


Figure 238. Method 1 to modify the workload management model.

Method Two

1. Reset screen layout so you have access to tree view and properties window
2. In the tree view window select simulations
3. From the simulations drop down, single click on User_Simulation_1
4. Locate the Properties Window and Under Runtime Settings subheading, change workload management box from false to true (see Figure 239)
5. A Workload Management Settings subheading will open below this in the properties settings and make necessary adjustments (especially redline numbers)

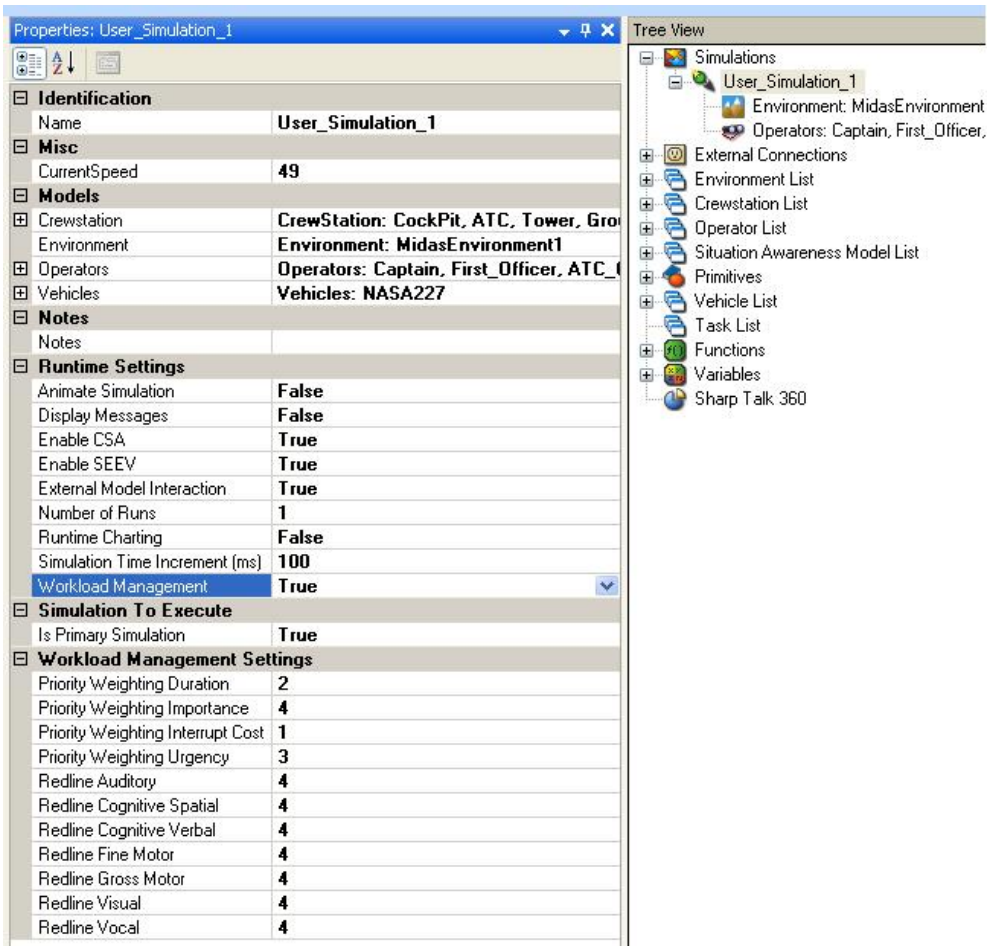


Figure 239. Method 2 to modify the workload management strategies in MIDAS.

Modifying Information Relevance Values

Information relevance is manipulated through the expectancy representation of information. Expectancy is the rate at which information changes, or bandwidth. The start task is the trigger for information relevance. To modify the information relevance in a scenario:

1. Select Context Specific Task Settings (CA) (92) or (FO)(94) from task network (Figure 240)
2. Select phase of flight (e.g. Descent (92_3), Land Initial (92_35)
3. Select Approach Mode (e.g. RNAV No_Pair (92_3_14)
4. Select Aviate, Separate, Navigate, Communicate, Systems or all tasks

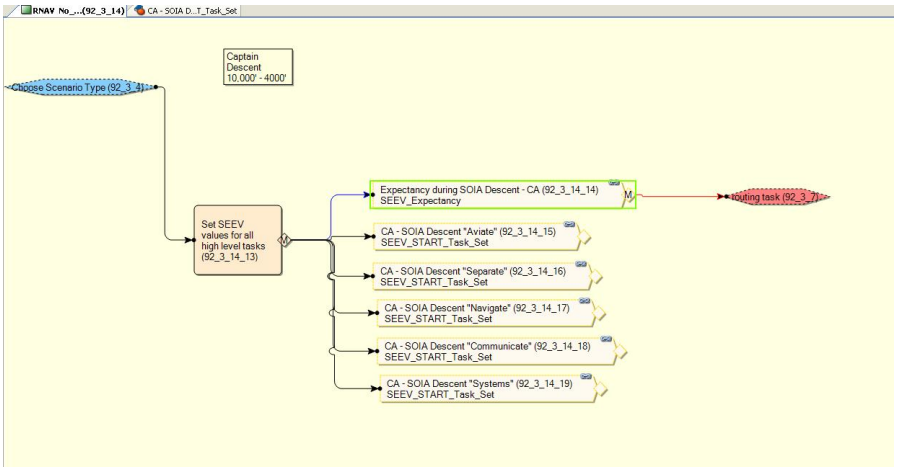


Figure 240. Setting SEEV information relevance parameters to drive MIDAS attention model.

5. Set relevance for each Area of Interest (range 0-6, low to high)
6. Set Situational Element to Required or Desired (Figure 241).

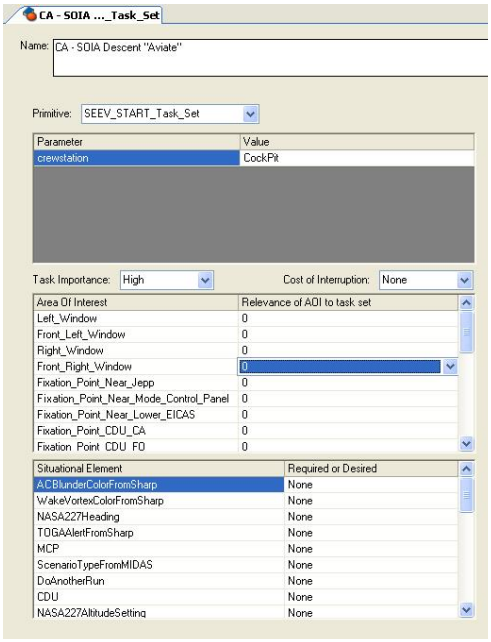


Figure 241. Modifying SEEV AOI and Situation Elements.

Modify Workload User Primitives

1. Open your MIDAS file of interest
2. Select Primitives from the Tree View window (Figure 242)

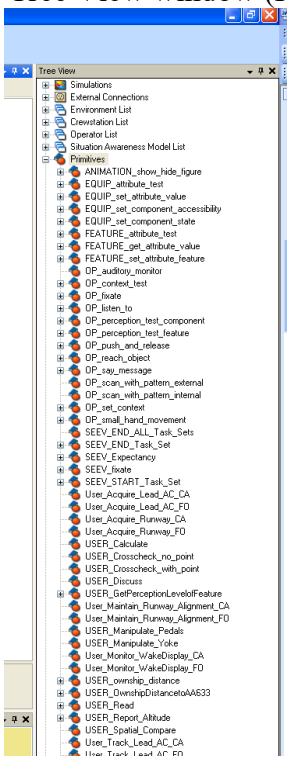


Figure 242. Available MIDAS primitives from the tree view.

3. Double click the primitive you are interested in manipulating and tab to the task load tab (furthest on right) (Figure 243). Note, only user defined primitives can be modified.

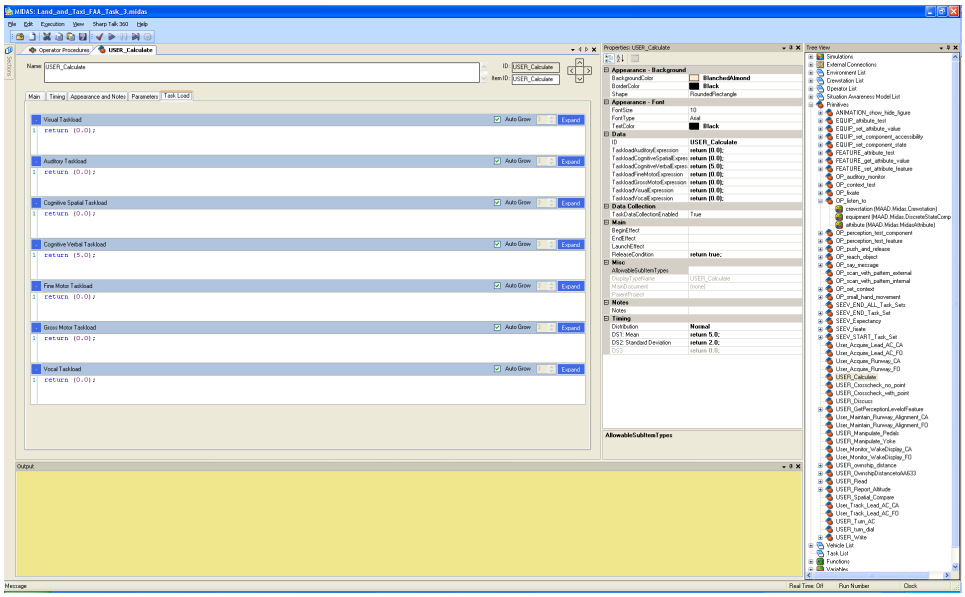


Figure 243. Manipulating a MIDAS primitive's workload.

4. Adjust the taskloads individually in their respective boxes by clicking and changing only the numbers in the brackets (Figure 244). Please be sure to maintain formatting (ie do not delete the bracket or the semi colon.

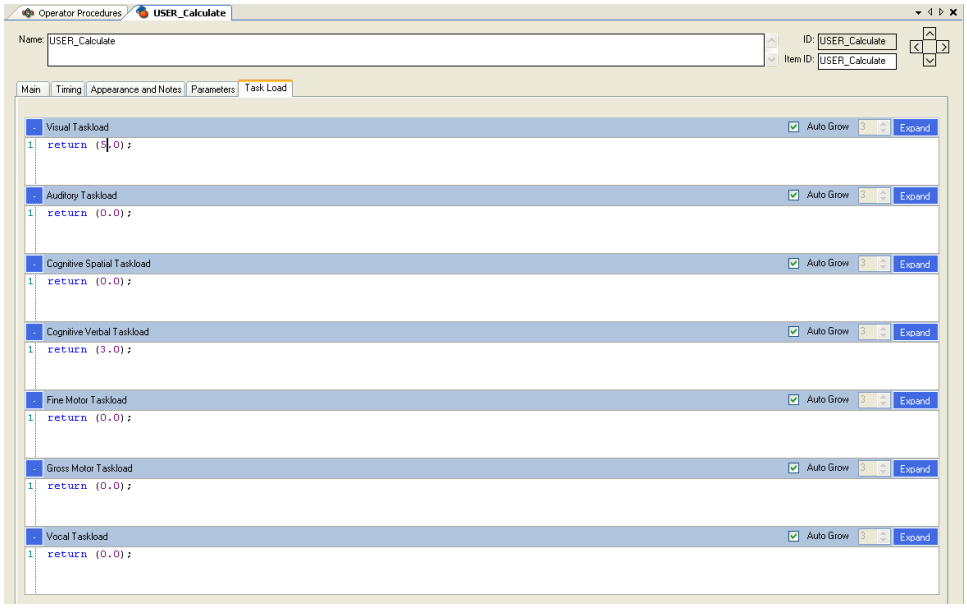


Figure 244. User defined primitive taskload modification.

Manipulating Existing SAINT Scenarios

1. Open Microsaint sharp and double click on subnetwork 305 (Initialize variables) (Figure 245)

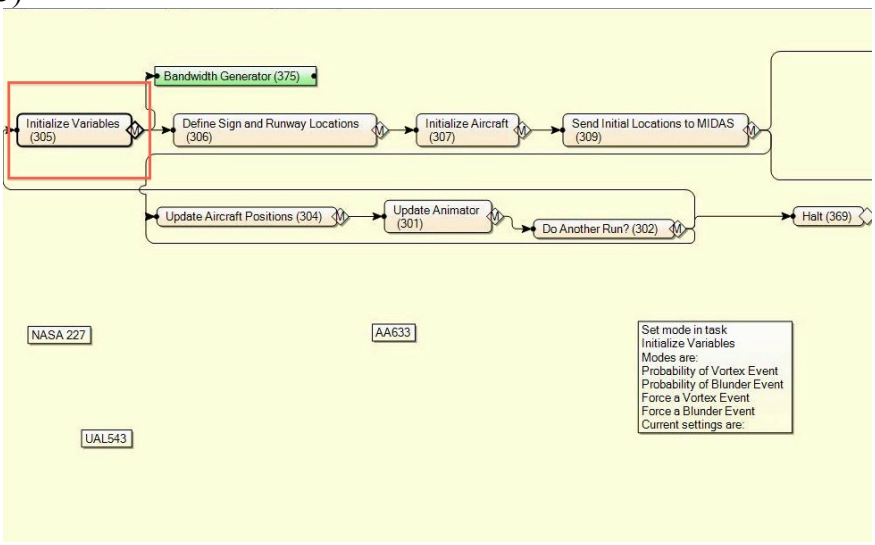


Figure 245. Example of manipulating MicroSaint Sharp models.

2. Adjust code in the ending effect box. Comment it out to turn it off (ie remove the two "/") (Figure 246).

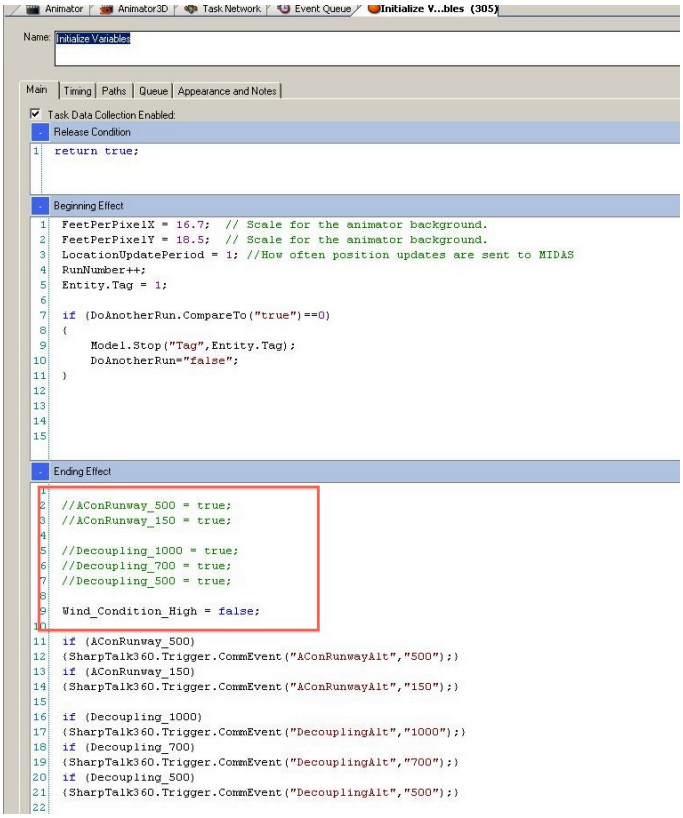


Figure 246. Example of Micro Saint Sharp Code's ending conditions.

Chapter 7: Interpreting and Analyzing MIDAS Data Output

1. MIDAS output is saved into the same directory of the .midas file that was loaded. The output is comprised of the .midas and .saint files that generated the .csv output, and the accuracy, fixation changes, SEEV events, SEEV settings, situation elements, situation awareness, taskbeginend, and workload files (Figure 247).

SIDEBAR: If the model is re-run with this same file, it will update and replace all the .csv files with the new data.

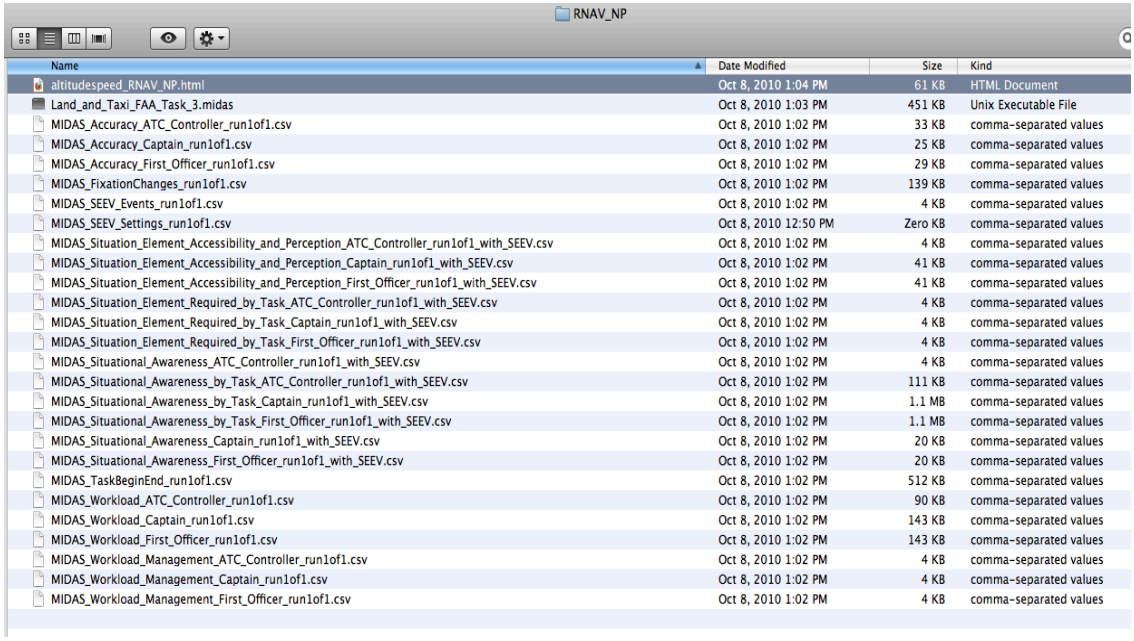


Figure 247. Example of the MIDAS .csv output files.

Most Common Files Used for Data analysis can be found in Table 30.

Table 30. Most common MIDAS output files used during analyses.

File	Use
Land_and_Taxi_FAA_Task_3.midas	Midas project file to open in Midas
MIDAS_FixationChangs_run1of1.csv	Calculate Dwell%
MIDAS_TaskBeginEnd_run1of1.csv	Reverse engineer tasks
MIDAS_Workload_Captain_run1of1.csv	Calculate CA workload
MIDAS_Workload_First_Officer_run1of1.csv	Calculate FO workload

Hint: When you open the CSV files, resave them as .xls or .xlsx so that your formatting changes will be preserved

Analysis: MicroSaint Output Position Data

1. Run a simulation.
2. Before closing MicroSaint Sharp, under the Tree View window, expand Snapshots (Figure 248).

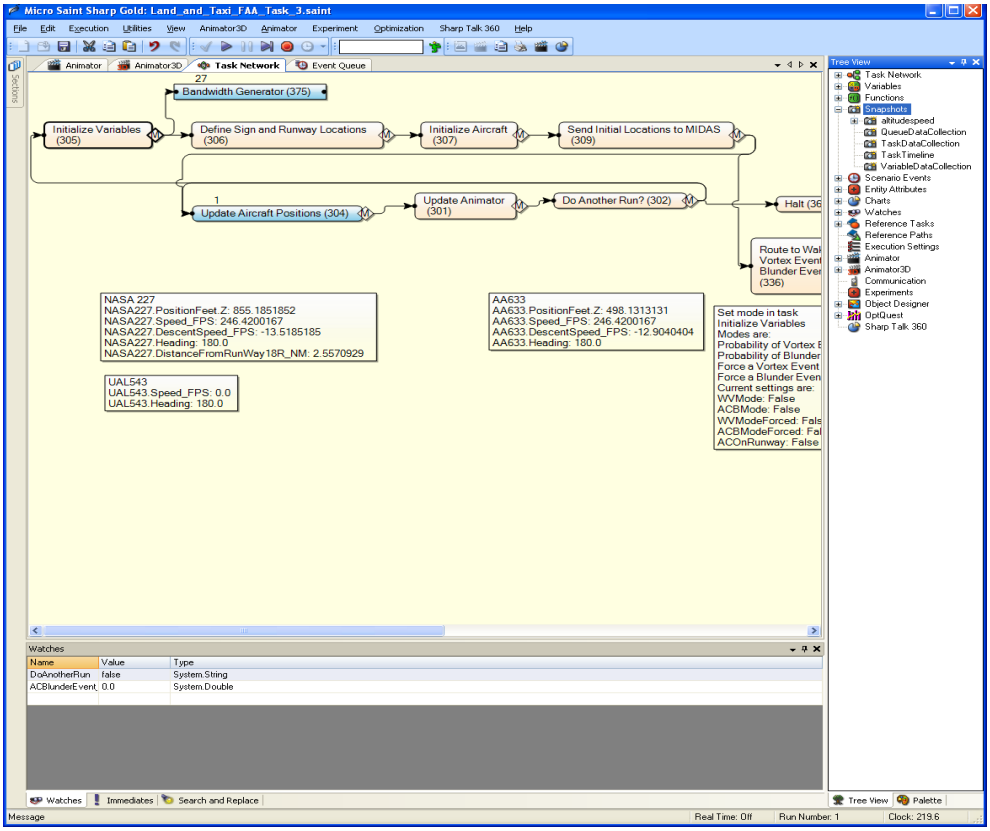


Figure 248. Analyzing output position data from MicroSaint Sharp.

3. Under snapshots, right click altitudespeed and select Open in Spreadsheet (Figure 249)

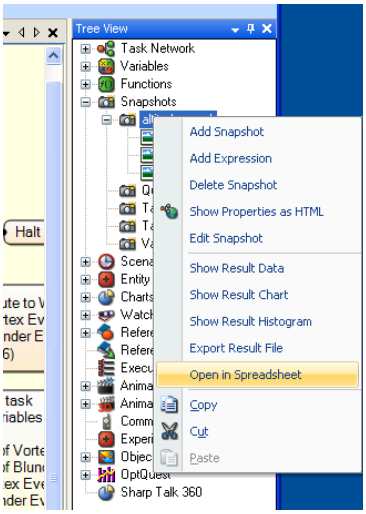
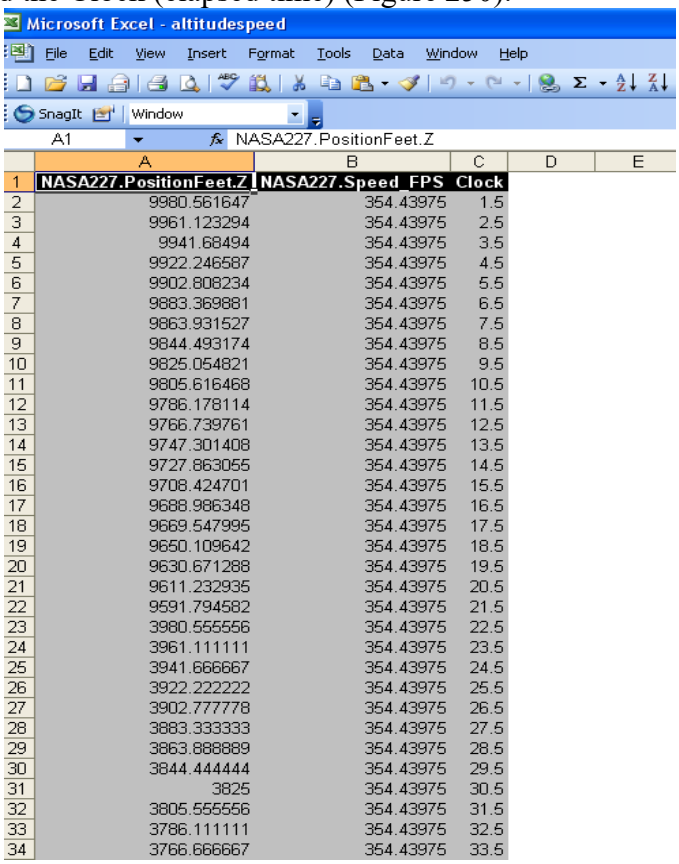


Figure 249. Accessing position data from MicroSaint Sharp.

4. A spreadsheet will open that contains Z position (altitude in feet), Speed (feet per second) and the Clock (elapsed time) (Figure 250).



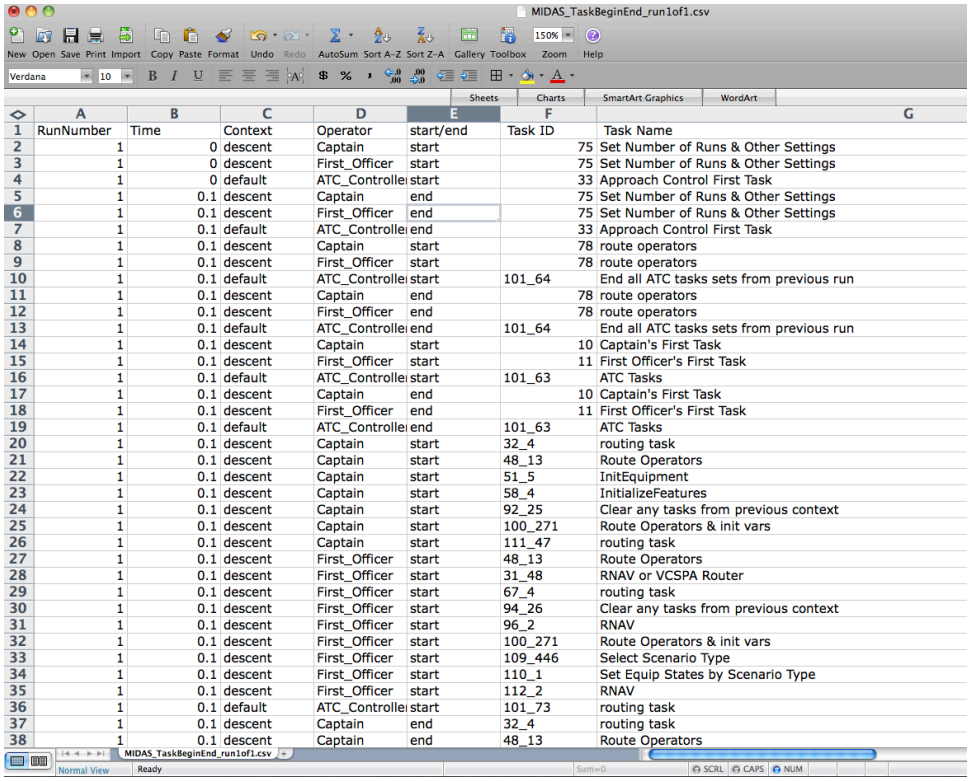
	A	B	C	D	E
1	NASA227.PositionFeet.Z	NASA227.Speed_FPS	Clock		
2	9980.561647	354.43975	1.5		
3	9961.123294	354.43975	2.5		
4	9941.68494	354.43975	3.5		
5	9922.246587	354.43975	4.5		
6	9902.808234	354.43975	5.5		
7	9883.369881	354.43975	6.5		
8	9863.931527	354.43975	7.5		
9	9844.493174	354.43975	8.5		
10	9825.054821	354.43975	9.5		
11	9805.616468	354.43975	10.5		
12	9786.178114	354.43975	11.5		
13	9766.739761	354.43975	12.5		
14	9747.301408	354.43975	13.5		
15	9727.863055	354.43975	14.5		
16	9708.424701	354.43975	15.5		
17	9688.986348	354.43975	16.5		
18	9669.547995	354.43975	17.5		
19	9650.109642	354.43975	18.5		
20	9630.671288	354.43975	19.5		
21	9611.232935	354.43975	20.5		
22	9591.794582	354.43975	21.5		
23	3980.555556	354.43975	22.5		
24	3961.111111	354.43975	23.5		
25	3941.666667	354.43975	24.5		
26	3922.222222	354.43975	25.5		
27	3902.777778	354.43975	26.5		
28	3883.333333	354.43975	27.5		
29	3863.888889	354.43975	28.5		
30	3844.444444	354.43975	29.5		
31	3825	354.43975	30.5		
32	3805.555556	354.43975	31.5		
33	3786.111111	354.43975	32.5		
34	3766.666667	354.43975	33.5		

Figure 250. MicroSaint Sharp positional data output in Excel format.

* Note you can export this report any time with base .saint file. For example, if you want to output position data for a run that was completed six months ago, simply open the original .saint file and repeat steps 2-4 above.

Analysis: Reverse Engineering MIDAS Task Model Output

1. Open MIDAS_TaskBeginEnd_run1of1.csv file in a word processing package like Microsoft Excel (Figure 251).



MIDAS_TaskBeginEnd_run1of1.csv

	A	B	C	D	E	F	G
1	RunNumber	Time	Context	Operator	start/end	Task ID	Task Name
2	1	0	descent	Captain	start	75	Set Number of Runs & Other Settings
3	1	0	descent	First_Officer	start	75	Set Number of Runs & Other Settings
4	1	0	default	ATC_Controlle	start	33	Approach Control First Task
5	1	0.1	descent	Captain	end	75	Set Number of Runs & Other Settings
6	1	0.1	descent	First_Officer	end	75	Set Number of Runs & Other Settings
7	1	0.1	default	ATC_Controlle	end	33	Approach Control First Task
8	1	0.1	descent	Captain	start	78	route operators
9	1	0.1	descent	First_Officer	start	78	route operators
10	1	0.1	default	ATC_Controlle	start	101_64	End all ATC tasks sets from previous run
11	1	0.1	descent	Captain	end	78	route operators
12	1	0.1	descent	First_Officer	end	78	route operators
13	1	0.1	default	ATC_Controlle	end	101_64	End all ATC tasks sets from previous run
14	1	0.1	descent	Captain	start	10	Captain's First Task
15	1	0.1	descent	First_Officer	start	11	First Officer's First Task
16	1	0.1	default	ATC_Controlle	start	101_63	ATC Tasks
17	1	0.1	descent	Captain	end	10	Captain's First Task
18	1	0.1	descent	First_Officer	end	11	First Officer's First Task
19	1	0.1	default	ATC_Controlle	end	101_63	ATC Tasks
20	1	0.1	descent	Captain	start	32_4	routing task
21	1	0.1	descent	Captain	start	48_13	Route Operators
22	1	0.1	descent	Captain	start	51_5	InitEquipment
23	1	0.1	descent	Captain	start	58_4	InitializeFeatures
24	1	0.1	descent	Captain	start	92_25	Clear any tasks from previous context
25	1	0.1	descent	Captain	start	100_271	Route Operators & init vars
26	1	0.1	descent	Captain	start	111_47	routing task
27	1	0.1	descent	First_Officer	start	48_13	Route Operators
28	1	0.1	descent	First_Officer	start	31_48	RNAV or VCSPA Router
29	1	0.1	descent	First_Officer	start	67_4	routing task
30	1	0.1	descent	First_Officer	start	94_26	Clear any tasks from previous context
31	1	0.1	descent	First_Officer	start	96_2	RNAV
32	1	0.1	descent	First_Officer	start	100_271	Route Operators & init vars
33	1	0.1	descent	First_Officer	start	109_446	Select Scenario Type
34	1	0.1	descent	First_Officer	start	110_1	Set Equip States by Scenario Type
35	1	0.1	descent	First_Officer	start	112_2	RNAV
36	1	0.1	default	ATC_Controlle	start	101_73	routing task
37	1	0.1	descent	Captain	end	32_4	routing task
38	1	0.1	descent	Captain	end	48_13	Route Operators

Figure 251. Reverse Engineering Process of MIDAS scenario.

2. Resave as a .xls or .xlsx file
3. Create a new column called ignore and label all non-operator based tasks as ignore
4. Filter the model down to the desired tasks (e.g. reprogram FMS)

Analysis: Calculate Percent Dwell Time

1. Open the MIDAS_FixationChanges_run1of1.csv file (Figure 252)

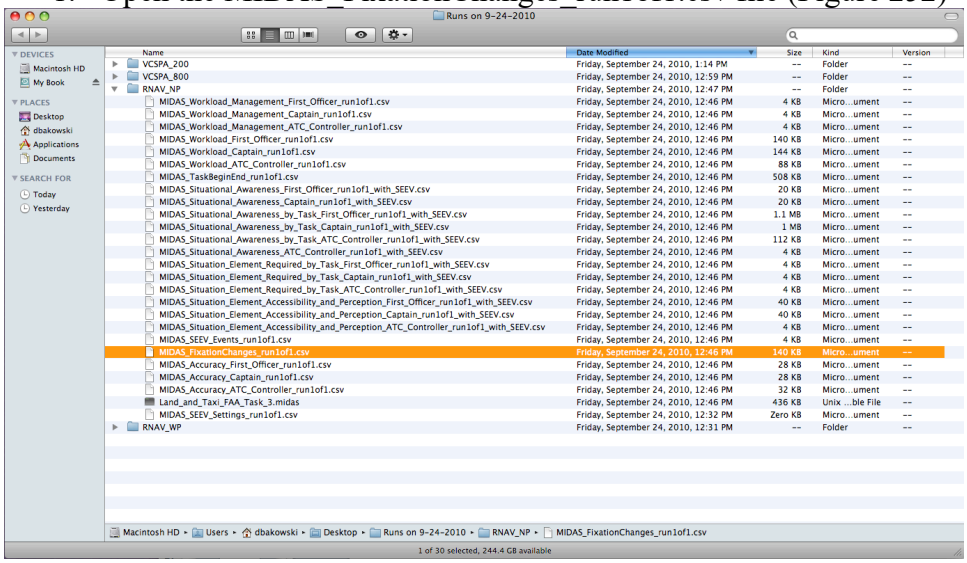


Figure 252. Output file used to create percent dwell time.

2. Resave as a .xls or .xlsx file
3. Create a new column called recoded context and recoded the contexts (Table 31) according to preference (Figure 253)

Table 31. Default and new context settings.

Default Contexts	Recoded Context
Descent, Approach_Initial	Descent and Approach_Initial
Approach_Transitional_1, Approach_Transitional_2, Approach_Final	Approach_Transitional 1,2 and Approach_Final
Land_Initial, Land_Final	Land_Initial and Land_Final

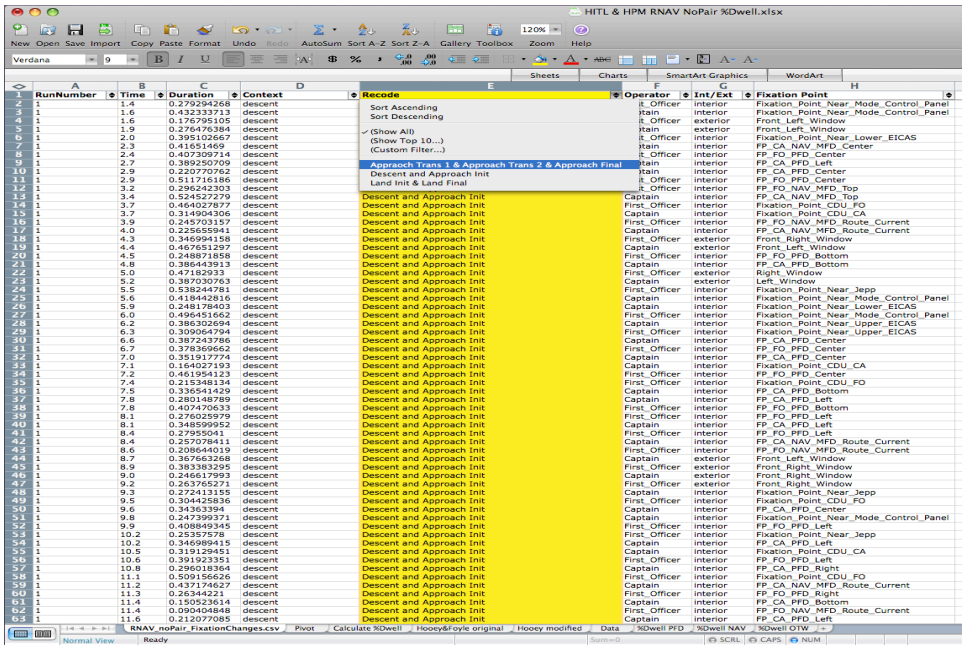
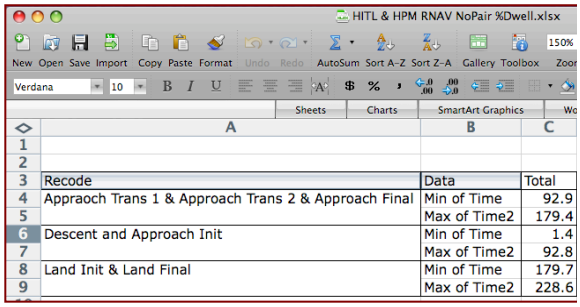


Figure 253. Recoding a variable output from a MIDAS scenario.

4. Create a pivot table to derive total time across each phase (Figure 254)

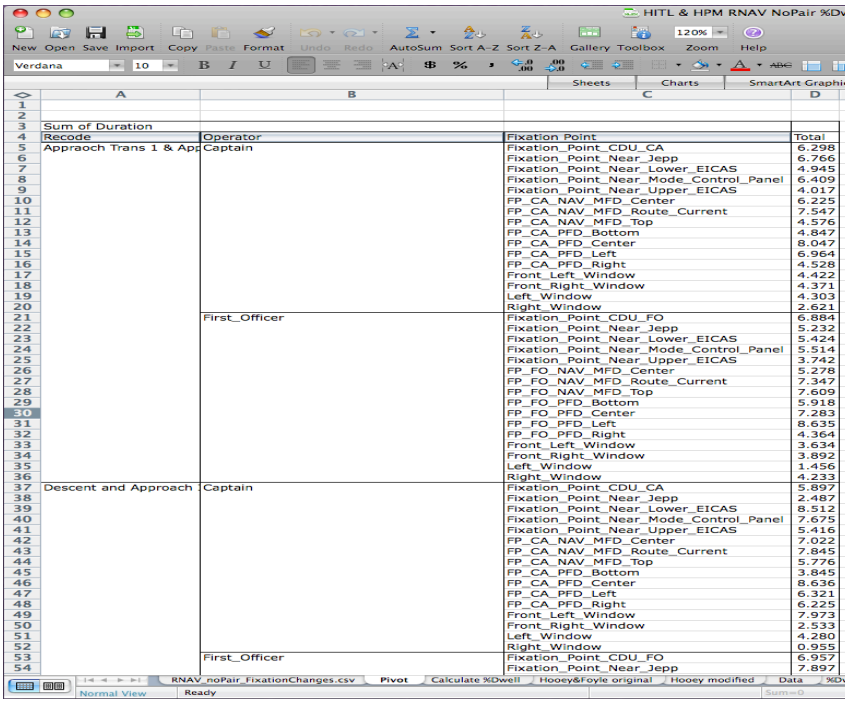


Recode	Data	Total
Approach Trans 1 & Approach Trans 2 & Approach Final	Min of Time	92.9
	Max of Time2	179.4
Descent and Approach Init	Min of Time	1.4
	Max of Time2	92.8
Land Init & Land Final	Min of Time	179.7
	Max of Time2	228.6

	Min Time	Max Time	Total Time
Descent and Approach Init	1.4	92.8	91.4
Approach Trans 1 & Approach Trans 2 & Approach Final	92.9	179.4	86.5
Land Init & Land Final	179.7	228.6	48.9

Figure 254. Create pivot table to derive total time across each phase of flight.

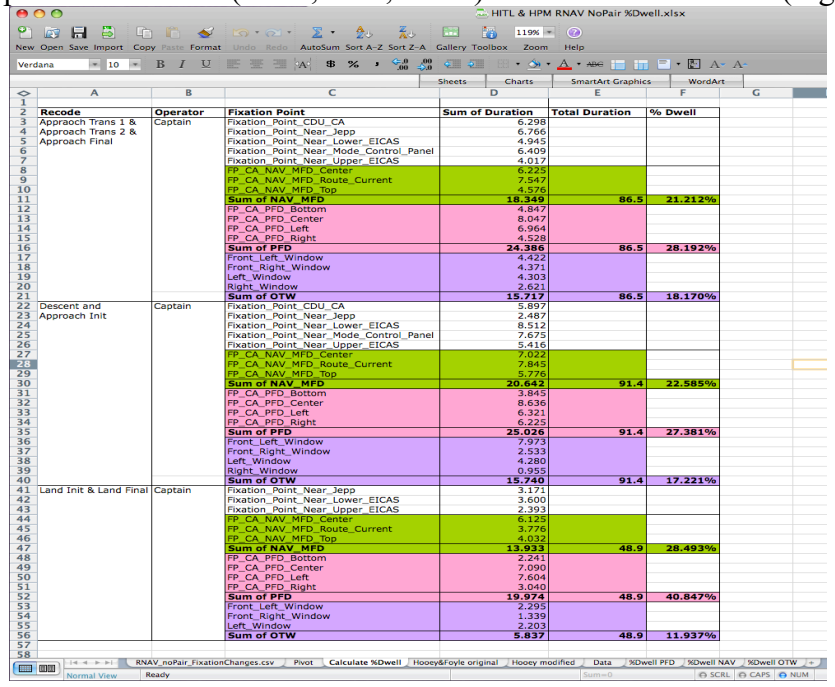
5. Find the Sum of Duration (Figure 255).



	A	B	C	D
1				
2				
3	Sum of Duration			
4	Recode	Operator	Fixation Point	Total
5	Appraoch Trans 1 & App	Captain	Fixation_Point_CDU_CA	6.298
6			Fixation_Point_Near_Jepp	6.766
7			Fixation_Point_Near_Lower_EICAS	4.945
8			Fixation_Point_Near_Mode_Control_Panel	6.409
9			Fixation_Point_Near_Upper_EICAS	4.017
10			FP_CA_NAV_MFD_Center	6.225
11			FP_CA_NAV_MFD_Route_Current	7.547
12			FP_CA_NAV_MFD_Top	4.576
13			FP_CA_PFD_Bottom	4.847
14			FP_CA_PFD_Center	8.047
15			FP_CA_PFD_Left	6.964
16			FP_CA_PFD_Right	4.528
17			Front_Left_Window	4.422
18			Front_Right_Window	4.371
19			Left_Window	4.303
20			Right_Window	2.621
21		First_Officer	Fixation_Point_CDU_FO	6.884
22			Fixation_Point_Near_Jepp	5.232
23			Fixation_Point_Near_Lower_EICAS	5.424
24			Fixation_Point_Near_Mode_Control_Panel	5.514
25			Fixation_Point_Near_Upper_EICAS	3.742
26			FP_FO_NAV_MFD_Center	5.278
27			FP_FO_NAV_MFD_Route_Current	7.347
28			FP_FO_NAV_MFD_Top	7.609
29			FP_FO_PFD_Bottom	5.918
30			FP_FO_PFD_Center	7.283
31			FP_FO_PFD_Left	8.635
32			FP_FO_PFD_Right	4.364
33			Front_Left_Window	3.634
34			Front_Right_Window	3.892
35			Left_Window	1.456
36			Right_Window	4.233
37	Descent and Approach	Captain	Fixation_Point_CDU_CA	5.897
38			Fixation_Point_Near_Jepp	2.487
39			Fixation_Point_Near_Lower_EICAS	8.512
40			Fixation_Point_Near_Mode_Control_Panel	7.675
41			Fixation_Point_Near_Upper_EICAS	5.416
42			FP_CA_NAV_MFD_Center	7.022
43			FP_CA_NAV_MFD_Route_Current	7.845
44			FP_CA_NAV_MFD_Top	5.776
45			FP_CA_PFD_Bottom	3.845
46			FP_CA_PFD_Center	8.636
47			FP_CA_PFD_Left	6.321
48			FP_CA_PFD_Right	6.225
49			Front_Left_Window	7.973
50			Front_Right_Window	2.533
51			Left_Window	4.280
52			Right_Window	0.955
53		First_Officer	Fixation_Point_CDU_FO	6.957
54			Fixation_Point_Near_Jepp	7.897

Figure 255. Find the sum of duration.

6. Group Fixation Points (NAV, PFD, OTW) and calculate %Dwell (Figure 256).

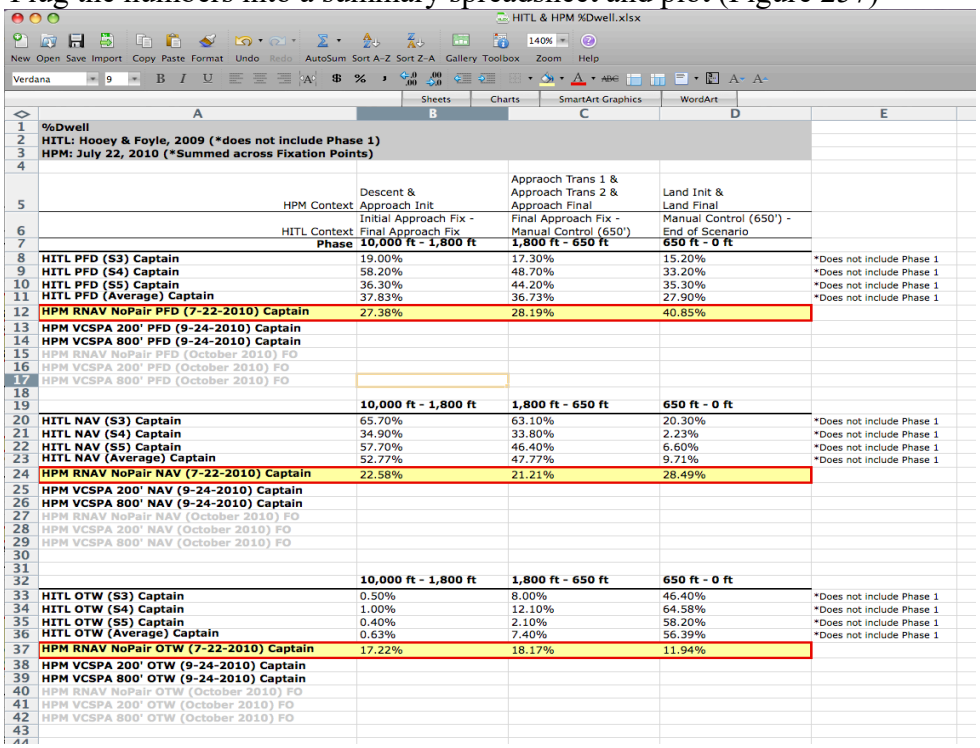


	A	B	C	D	E	F	G
1							
2	**Recode**	**Operator**	**Fixation Point**	**Sum of Duration**	**Total Duration**	**% Dwell**	
3	Appraoch Trans 1 &	Captain	Fixation_Point_CDU_CA	6.298			
4	Approach Trans 2 &		Fixation_Point_Near_Jepp	6.766			
5	Approach Final		Fixation_Point_Near_Lower_EICAS	4.945			
6			Fixation_Point_Near_Mode_Control_Panel	6.409			
7			Fixation_Point_Near_Upper_EICAS	4.017			
8			FP_CA_NAV_MFD_Center	6.225			
9			FP_CA_NAV_MFD_Route_Current	7.547			
10			FP_CA_NAV_MFD_Top	4.576			
11			**Sum of NAV_MFD**	**18.349**	**86.5**	**21.212%**	
12			FP_CA_PFD_Bottom	4.847			
13			FP_CA_PFD_Center	8.047			
14			FP_CA_PFD_Left	6.964			
15			FP_CA_PFD_Right	4.528			
16			**Sum of PFD**	**24.386**	**86.5**	**28.192%**	
17			Front_Left_Window	4.422			
18			Front_Right_Window	4.371			
19			Left_Window	4.303			
20			Right_Window	2.621			
21			**Sum of OTW**	**15.717**	**86.5**	**18.170%**	
22	Descent and	Captain	Fixation_Point_CDU_CA	5.897			
23	Approach Init		Fixation_Point_Near_Jepp	2.487			
24			Fixation_Point_Near_Lower_EICAS	8.512			
25			Fixation_Point_Near_Mode_Control_Panel	7.675			
26			Fixation_Point_Near_Upper_EICAS	5.416			
27			FP_CA_NAV_MFD_Center	7.022			
28			FP_CA_NAV_MFD_Route_Current	7.845			
29			FP_CA_NAV_MFD_Top	5.776			
30			**Sum of NAV_MFD**	**20.642**	**91.4**	**22.585%**	
31			FP_CA_PFD_Bottom	3.845			
32			FP_CA_PFD_Center	8.636			
33			FP_CA_PFD_Left	6.321			
34			FP_CA_PFD_Right	6.225			
35			**Sum of PFD**	**25.026**	**91.4**	**27.381%**	
36			Front_Left_Window	7.973			
37			Front_Right_Window	2.533			
38			Left_Window	4.280			
39			Right_Window	0.955			
40			**Sum of OTW**	**15.740**	**91.4**	**17.221%**	
41	Land Init & Land Final	Captain	Fixation_Point_Near_Jepp	3.171			
42			Fixation_Point_Near_Lower_EICAS	3.600			
43			Fixation_Point_Near_Upper_EICAS	2.393			
44			FP_CA_NAV_MFD_Center	6.125			
45			FP_CA_NAV_MFD_Route_Current	3.776			
46			FP_CA_NAV_MFD_Top	4.032			
47			**Sum of NAV_MFD**	**13.933**	**48.9**	**28.493%**	
48			FP_CA_PFD_Bottom	2.241			
49			FP_CA_PFD_Center	7.090			
50			FP_CA_PFD_Left	7.604			
51			FP_CA_PFD_Right	3.040			
52			**Sum of PFD**	**19.974**	**48.9**	**40.847%**	
53			Front_Left_Window	2.295			
54			Front_Right_Window	1.339			
55			Left_Window	2.203			
56			**Sum of OTW**	**5.837**	**48.9**	**11.937%**	

Figure 256. Illustration of the process to group fixation points to calculated percent dwell time.

*Note: Make sure to confirm which operator (CA or FO) you are calculating in operator column of pivot report as it includes both CA and FO

7. Plug the numbers into a summary spreadsheet and plot (Figure 257)



HITL & HPM %Dwell.xlsx

	A	B	C	D	E
1	%Dwell				
2	HITL: Hooey & Foyle, 2009 (*does not include Phase 1)				
3	HPM: July 22, 2010 (*Summed across Fixation Points)				
4					
5	HPM Context	Descent & Approach Init	Approach Trans 1 & Approach Trans 2 & Approach Final	Land Init & Land Final	
6	HITL Context	Initial Approach Fix - Final Approach Fix	Final Approach Fix - Manual Control (650')	Manual Control (650') - End of Scenario	
7	**Phase**	**10,000 ft - 1,800 ft**	**1,800 ft - 650 ft**	**650 ft - 0 ft**	
8	**HITL PFD (S3) Captain**	19.00%	17.30%	15.20%	*Does not include Phase 1
9	**HITL PFD (S4) Captain**	58.20%	48.70%	33.20%	*Does not include Phase 1
10	**HITL PFD (S5) Captain**	36.30%	44.20%	35.30%	*Does not include Phase 1
11	**HITL PFD (Average) Captain**	37.83%	36.73%	27.90%	*Does not include Phase 1
12	**HPM RNAV NoPair PFD (7-22-2010) Captain**	27.38%	28.19%	40.85%	
13	**HPM VCSPA 200' PFD (9-24-2010) Captain**				
14	**HPM VCSPA 800' PFD (9-24-2010) Captain**				
15	**HPM RNAV NoPair PFD (October 2010) FO**				
16	**HPM VCSPA 200' PFD (October 2010) FO**				
17	**HPM VCSPA 800' PFD (October 2010) FO**				
18					
19		**10,000 ft - 1,800 ft**	**1,800 ft - 650 ft**	**650 ft - 0 ft**	
20	**HITL NAV (S3) Captain**	65.70%	63.10%	20.30%	*Does not include Phase 1
21	**HITL NAV (S4) Captain**	34.90%	33.80%	2.23%	*Does not include Phase 1
22	**HITL NAV (S5) Captain**	57.70%	46.40%	6.60%	*Does not include Phase 1
23	**HITL NAV (Average) Captain**	52.77%	47.77%	9.71%	*Does not include Phase 1
24	**HPM RNAV NoPair NAV (7-22-2010) Captain**	22.58%	21.21%	28.49%	
25	**HPM VCSPA 200' NAV (9-24-2010) Captain**				
26	**HPM VCSPA 800' NAV (9-24-2010) Captain**				
27	**HPM RNAV NoPair NAV (October 2010) FO**				
28	**HPM VCSPA 200' NAV (October 2010) FO**				
29	**HPM VCSPA 800' NAV (October 2010) FO**				
30					
31					
32		**10,000 ft - 1,800 ft**	**1,800 ft - 650 ft**	**650 ft - 0 ft**	
33	**HITL OTW (S3) Captain**	0.50%	8.00%	46.40%	*Does not include Phase 1
34	**HITL OTW (S4) Captain**	1.00%	12.10%	64.58%	*Does not include Phase 1
35	**HITL OTW (S5) Captain**	0.40%	2.10%	58.20%	*Does not include Phase 1
36	**HITL OTW (Average) Captain**	0.63%	7.40%	56.39%	*Does not include Phase 1
37	**HPM RNAV NoPair OTW (7-22-2010) Captain**	17.22%	18.17%	11.94%	
38	**HPM VCSPA 200' OTW (9-24-2010) Captain**				
39	**HPM VCSPA 800' OTW (9-24-2010) Captain**				
40	**HPM RNAV NoPair OTW (October 2010) FO**				
41	**HPM VCSPA 200' OTW (October 2010) FO**				
42	**HPM VCSPA 800' OTW (October 2010) FO**				
43					

Figure 257. Create summary spreadsheet to facilitate generation of PDT.

Analysis: Workload

1. Open MIDAS_Workload_Captain or First Officer_run1of1.csv file (Figure 258).

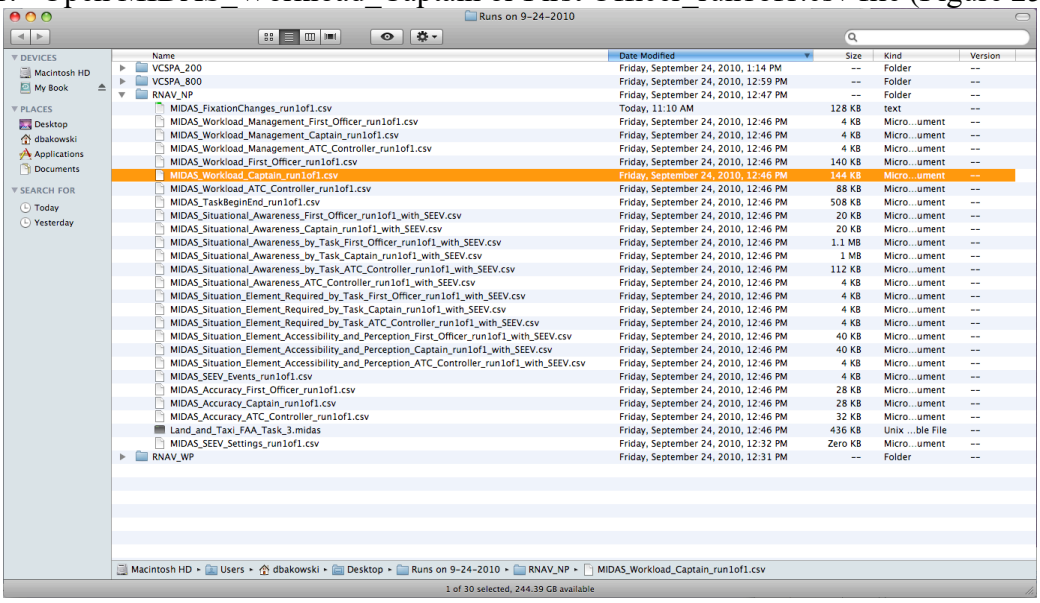


Figure 258. Processing workload output from MIDAS.

2. Resave as .xls or .xlsx
3. Create a new column called recoded context (Table 32) and recode the contexts according to preference (Figure 259)

Table 32. To recode context for workload analysis.

Default Contexts	Recoded Context
Descent, Approach_Initial	Descent and Approach_Initial
Approach_Transitional_1, Approach_Transitional_2, Approach_Final	Approach_Transitional 1,2 and Approach_Final
Land_Initial, Land_Final	Land_Initial and Land_Final

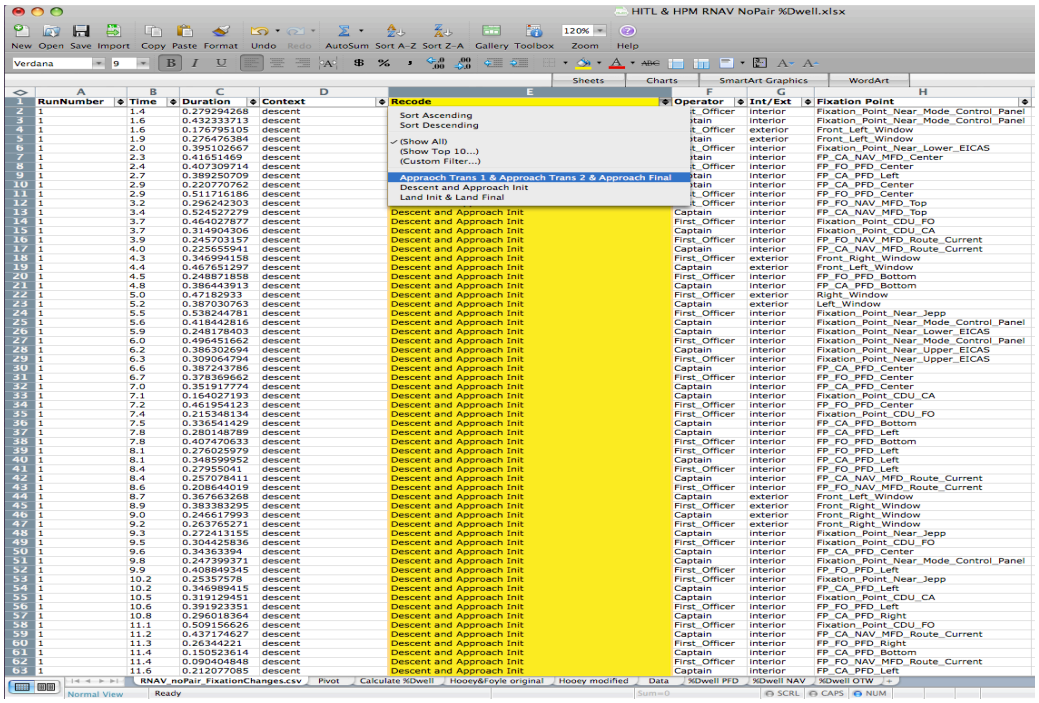


Figure 259. Example of recoding workload file when combining phases of flight.

4. Sum each row and take the average of each workload channel (columns) (Figure 260)

RunNumber	Time	Context	Recoded Context	Visual	Auditory	Cog Spatial	Cog Verbal	Fine Motor	Gross Motor	Voice	Sum
1	281	land_final	land_initial and land_final	24.94	7.64	17.71	9.9	20.42	12.04	0	92.65
1	281.1	land_final	land_initial and land_final	24.94	7.64	17.71	9.9	20.42	12.04	0	92.65
1	281.2	land_final	land_initial and land_final	24.94	7.64	17.71	9.9	20.42	12.04	0	92.65
1	281.3	land_final	land_initial and land_final	19.44	6.53	15.25	9.3	15.36	9.24	0	75.12
1	281.4	land_final	land_initial and land_final	19.44	6.53	15.25	9.3	15.36	9.24	0	75.12
1	281.5	land_final	land_initial and land_final	19.44	6.53	15.25	9.3	15.36	9.24	0	75.12
1	281.6	land_final	land_initial and land_final	19.44	6.53	15.25	9.3	15.36	9.24	0	75.12
1	281.7	land_final	land_initial and land_final	19.44	6.53	15.25	9.3	15.36	9.24	0	75.12
1	281.8	land_final	land_initial and land_final	19.44	12.13	18.98	16.88	15.36	9.24	0	92.03
1	281.9	land_final	land_initial and land_final	16.95	11.77	18.98	16.88	0	0	0	64.58
1	282	land_final	land_initial and land_final	16.95	11.77	18.98	16.88	0	7.34	0	71.92
1	282.1	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	282.2	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	282.3	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	282.4	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	282.5	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	282.6	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	282.7	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	282.8	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	282.9	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	283	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	283.1	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	283.2	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	283.3	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	283.4	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	283.5	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	283.6	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	283.7	land_final	land_initial and land_final	15.4	5.81	13.37	8.8	0	7.34	0	50.72
1	283.8	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	283.9	land_final	land_initial and land_final	16.87	11.77	16.02	15.08	0	7.34	0	67.08
1	284	land_final	land_initial and land_final	23.97	13.24	18.6	15.68	17.29	9.39	0	98.17
				16.806	6.402	14.376	9.503	4.929	6.768	0.000	58.784
											8.398

Figure 260. Analyzing MIDAS workload .csv files.

Calculate the average overall workload (average of columns E-K in this example) which is displayed in Column L (last row) (Figure 260).

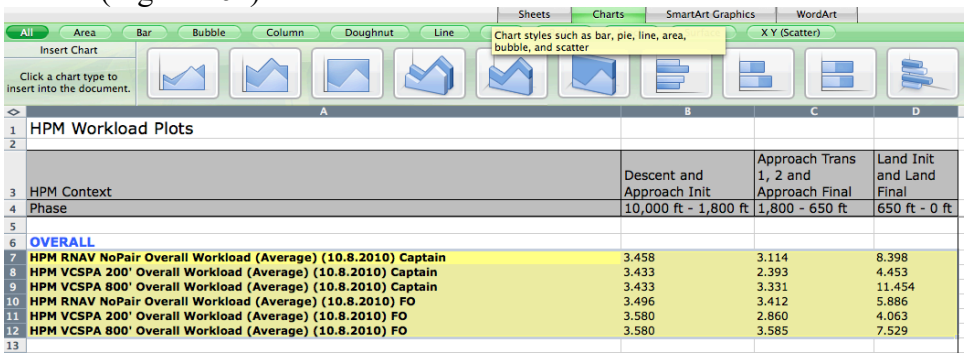
Analysis: How to Plot the MIDAS Output in Excel

1. Open your data summary sheet (Figure 261)

	A	B	C	D
1	HPM Workload Plots			
2				
3	HPM Context	Descent and Approach Init	Approach Trans 1, 2 and Approach Final	Land Init and Land Final
4	Phase	10,000 ft - 1,800 ft	1,800 - 650 ft	650 ft - 0 ft
5				
6	OVERALL			
7	HPM RNAV NoPair Overall Workload (Average) (10.8.2010) Captain	3.458	3.114	8.398
8	HPM VCSPA 200' Overall Workload (Average) (10.8.2010) Captain	3.433	2.393	4.453
9	HPM VCSPA 800' Overall Workload (Average) (10.8.2010) Captain	3.433	3.331	11.454
10	HPM RNAV NoPair Overall Workload (Average) (10.8.2010) FO	3.496	3.412	5.886
11	HPM VCSPA 200' Overall Workload (Average) (10.8.2010) FO	3.580	2.860	4.063
12	HPM VCSPA 800' Overall Workload (Average) (10.8.2010) FO	3.580	3.585	7.529
13				
14	VISUAL			
15	HPM RNAV NoPair Visual Workload (10.8.2010) Captain	7.634	7.793	16.806
16	HPM VCSPA 200' Visual Workload (10.8.2010) Captain	7.570	6.395	10.991
17	HPM VCSPA 800' Visual Workload (10.8.2010) Captain	7.570	8.396	22.835
18	HPM RNAV NoPair Visual Workload (10.8.2010) FO	7.566	7.709	14.626
19	HPM VCSPA 200' Visual Workload (10.8.2010) FO	7.590	6.661	10.293
20	HPM VCSPA 800' Visual Workload (10.8.2010) FO	7.590	8.315	19.192
21				
22	AUDITORY			
23	HPM RNAV NoPair Auditory Workload (10.8.2010) Captain	5.308	4.618	6.402
24	HPM VCSPA 200' Auditory Workload (10.8.2010) Captain	5.272	4.053	5.230
25	HPM VCSPA 800' Auditory Workload (10.8.2010) Captain	5.272	4.649	7.669
26	HPM RNAV NoPair Auditory Workload (10.8.2010) FO	5.196	4.721	5.992
27	HPM VCSPA 200' Auditory Workload (10.8.2010) FO	5.277	4.290	4.998
28	HPM VCSPA 800' Auditory Workload (10.8.2010) FO	5.277	4.808	7.058
29				

Figure 261. Example of summary sheet that was generated for the workload variable.

2. Select rows that you would like to plot and click on charts drop down menu (Figure 262).



	A	B	C	D
1	HPM Workload Plots			
2				
3	HPM Context	Descent and Approach Init	Approach Trans 1, 2 and Approach Final	Land Init and Land Final
4	Phase	10,000 ft - 1,800 ft	1,800 - 650 ft	650 ft - 0 ft
5				
6	OVERALL			
7	HPM RNAV NoPair Overall Workload (Average) (10.8.2010) Captain	3.458	3.114	8.398
8	HPM VCSPA 200' Overall Workload (Average) (10.8.2010) Captain	3.433	2.393	4.453
9	HPM VCSPA 800' Overall Workload (Average) (10.8.2010) Captain	3.433	3.331	11.454
10	HPM RNAV NoPair Overall Workload (Average) (10.8.2010) FO	3.496	3.412	5.886
11	HPM VCSPA 200' Overall Workload (Average) (10.8.2010) FO	3.580	2.860	4.063
12	HPM VCSPA 800' Overall Workload (Average) (10.8.2010) FO	3.580	3.585	7.529
13				

Figure 262. Selecting data for data presentation in Excel.

3. Once you select your desired chart type, the chart will appear on the current spreadsheet (Figure 263).

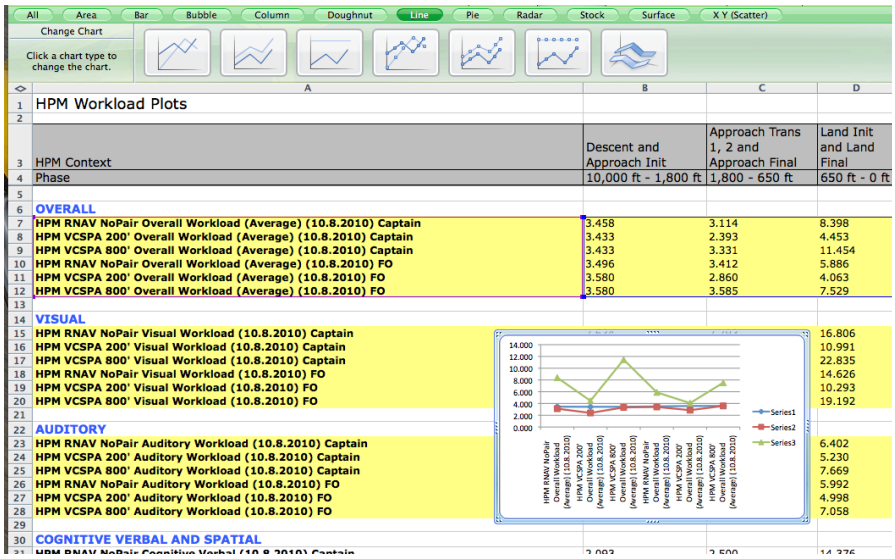


Figure 263. Create initial data plot for examination.

4. Make edits to the chart via the chart drop down menu or the toolbox (Figure 264).

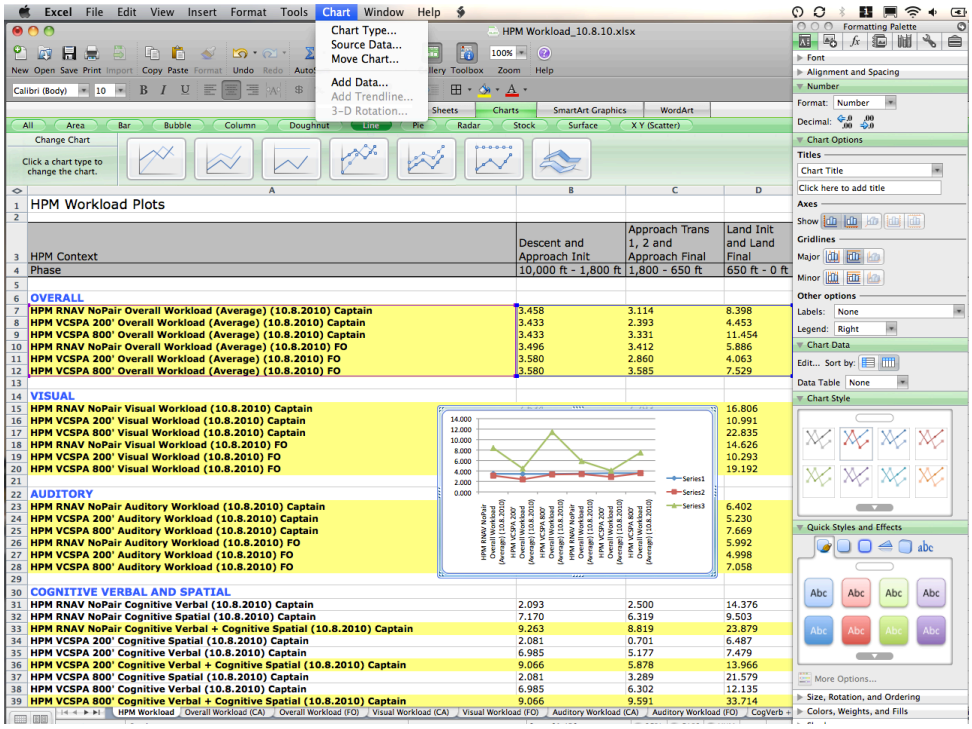


Figure 264. Editing the excel charts.

Analysis: One Sample T-Test with SPSS

1. Import the dataset into SPSS

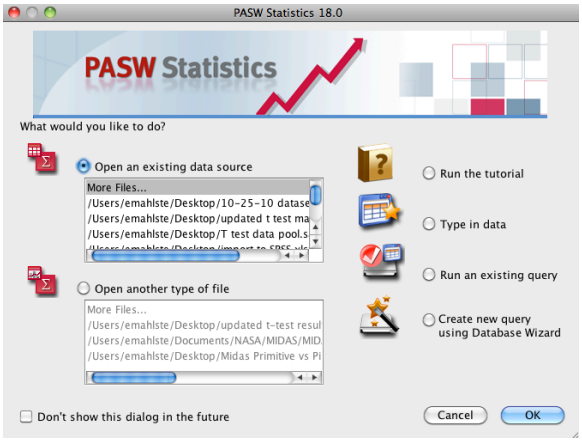


Figure 265. Import data into SPSS.

2. From toolbar select Analyze > Compare Means > One-Sample T Test… (see Figure 266).

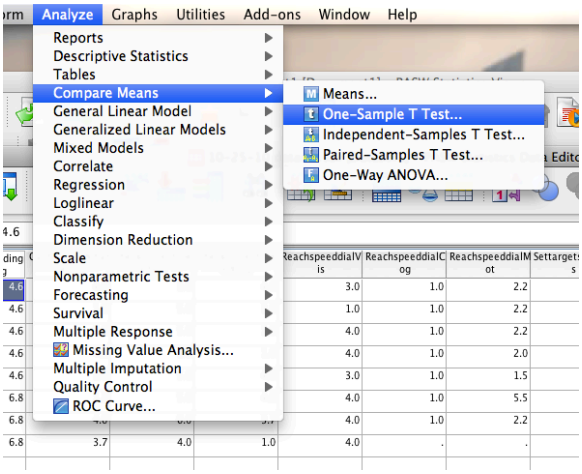


Figure 266. Comparing means using SPSS One-Sample T-Test.

3. Highlight the variable to run test on (Figure 267)

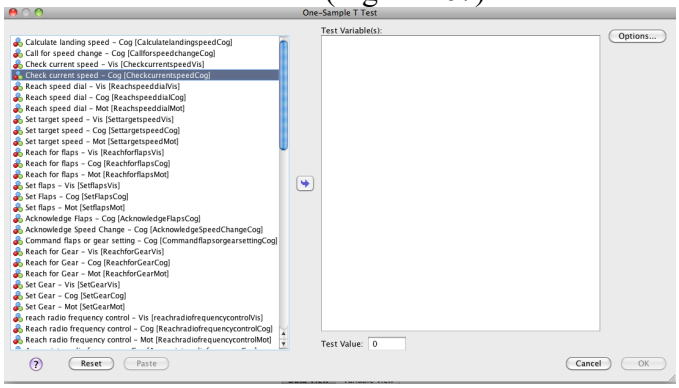


Figure 267. Select the variable upon which the test will be executed.

4. Click on the arrow in the middle of the screen to select it as a test variable (Figure 268).

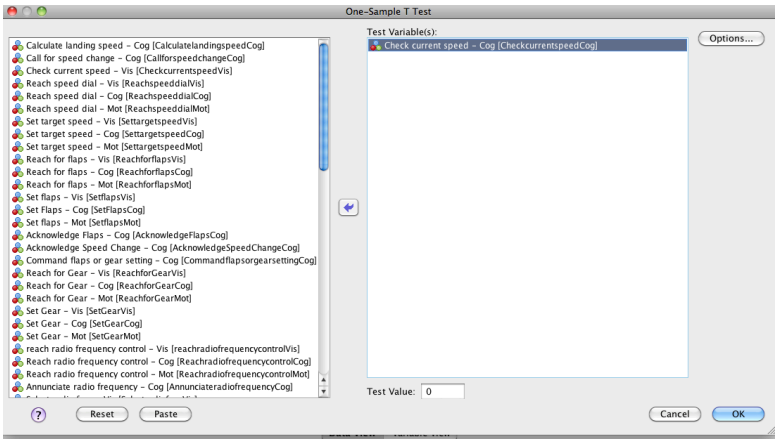


Figure 268. Select the test variable.

5. Enter Test value (Figure 269).

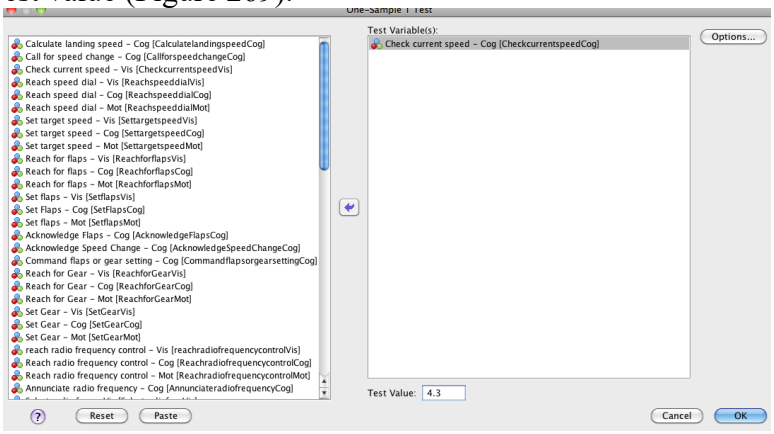


Figure 269. Enter the critical value for T.

6. Press Ok to run test and view output (Figure 270).

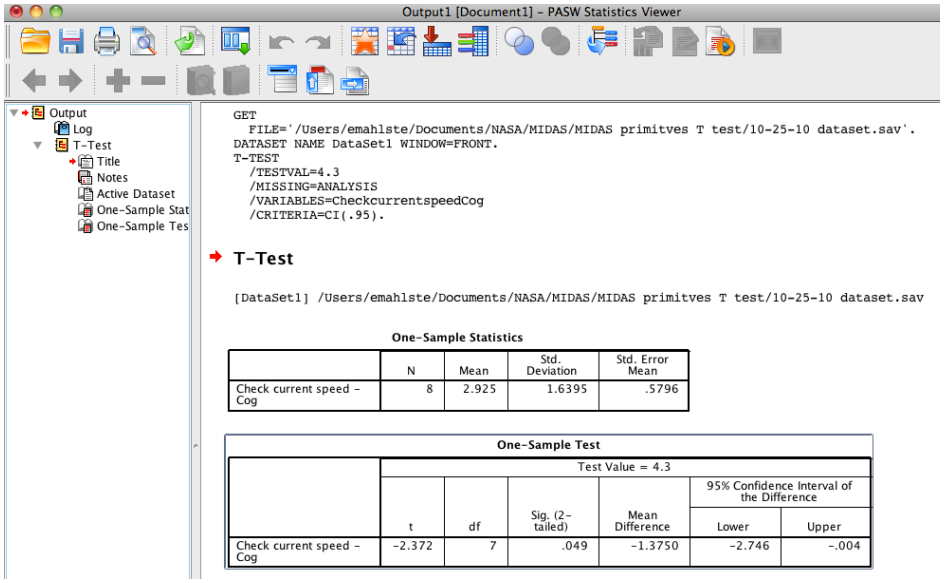


Figure 270. SPSS t-test output.

Analysis: Paired Samples T-test

1. Import or type your dataset into SPSS (Figure 271).

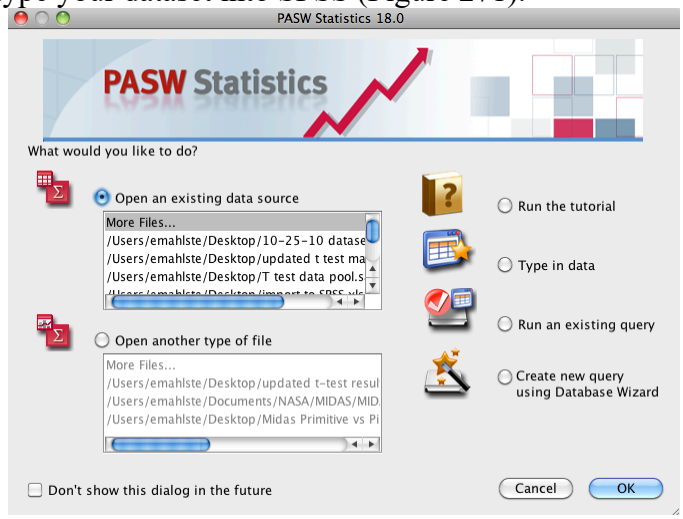


Figure 271. Import data into SPSS.

2. From the analyze drop down menu, select compare means > paired samples T Test (Figure 272)

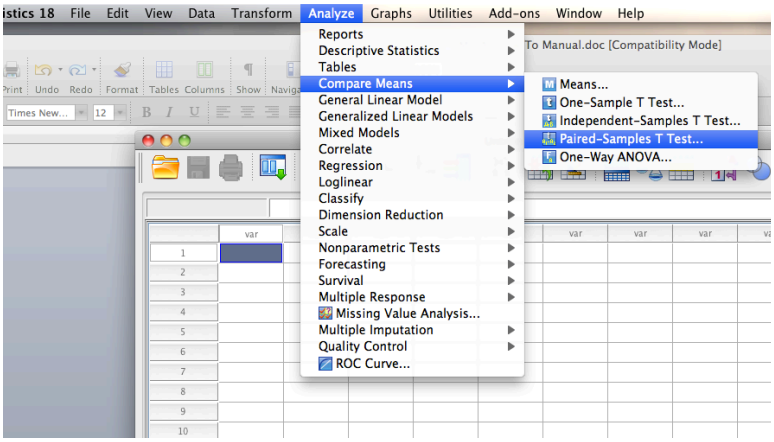


Figure 272. Comparing means using SPSS Paired-Samples T-Test.

3. Select a pair of variables from the left column by clicking on them and then press the right arrow to queue them for the test (Figure 273)**.**

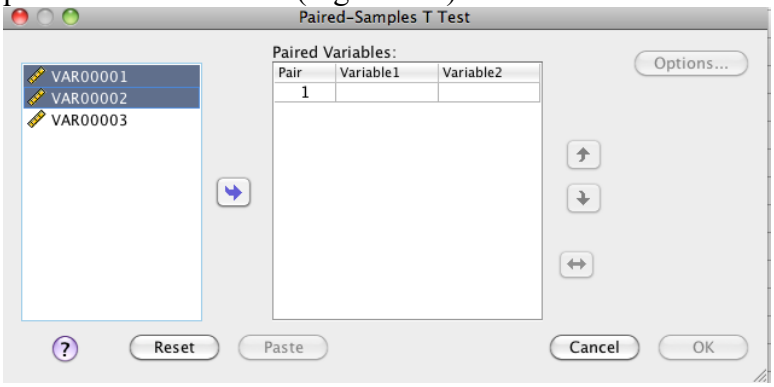


Figure 273. Selecting the pairs for the paired samples t-test.

4. If doing all Pairwise comparisons, include them all in the paired variables sub-window and press ok. The window can be made larger in order to see the names of the variables more clearly (Figure 274).

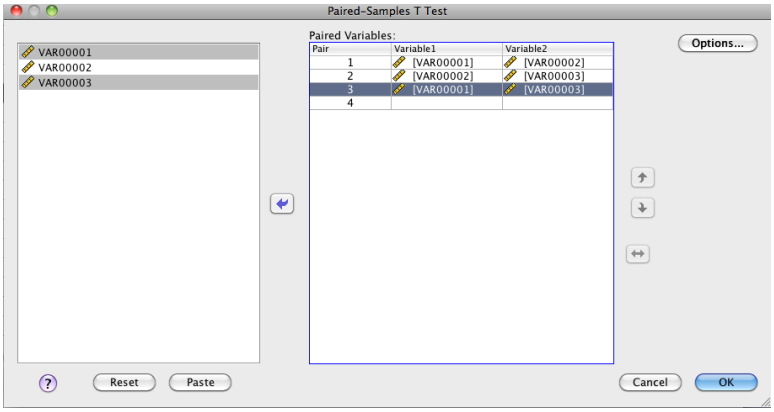


Figure 274. Increase window size to make variable names more visible.

5. An example of the paired sample t-test can be found in Figure 275.

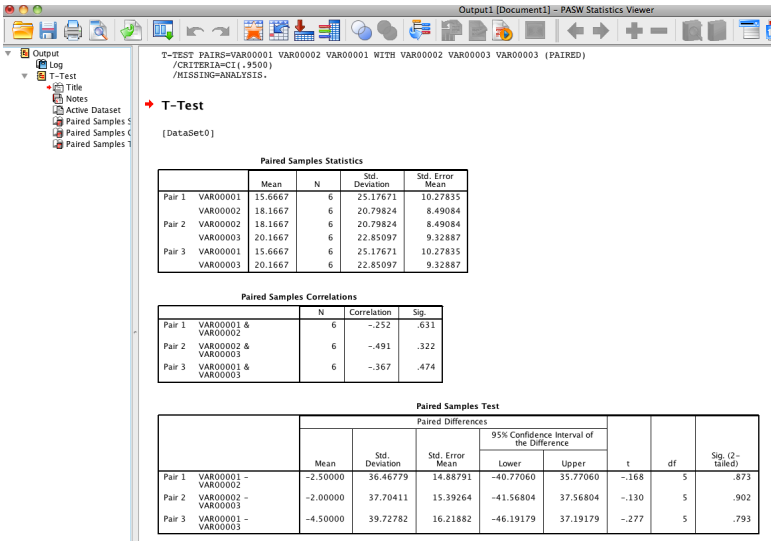


Figure 275. Example of paired sample SPSS t-test output.

Analysis: Repeated Measures ANOVA

1. Import your dataset into SPSS (Figure 276)

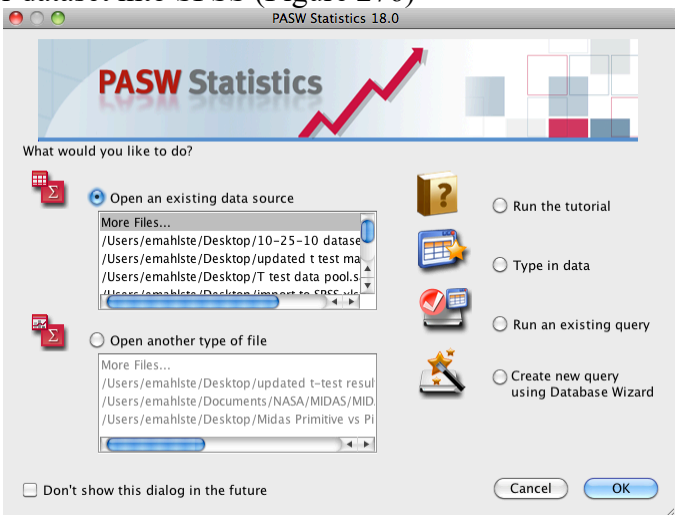


Figure 276. Import data into SPSS.

2. From the Analyze drop down menu, select General Linear Model > Repeated Measures (Figure 277).

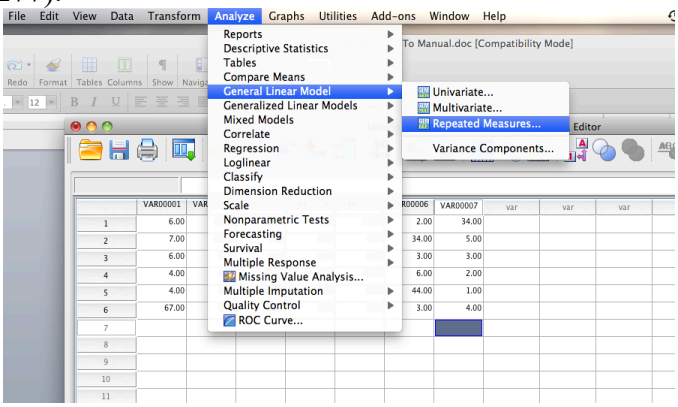


Figure 277. View of the steps to run a generalized linear model with repeated measures.

3. Type in the name of the first within subjects factor, enter the number of levels and add the factor (Figure 278)

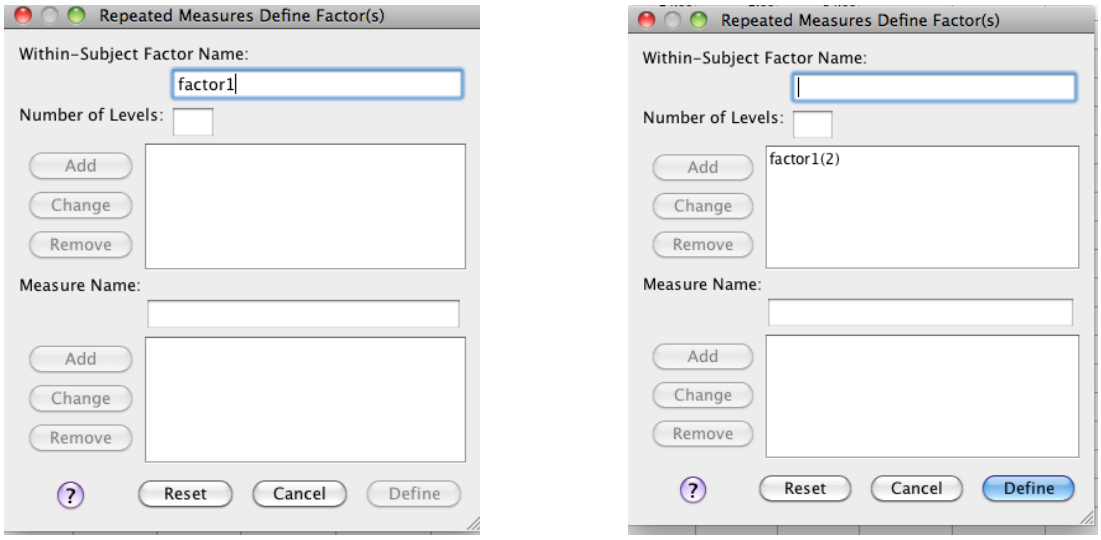


Figure 278. Define the repeated measures anova within-subjects factors.

5. Click define, select the variables to be included in the ANOVA, click on the right arrow and then the blue ok button at the bottom left of the window (Figure 279)

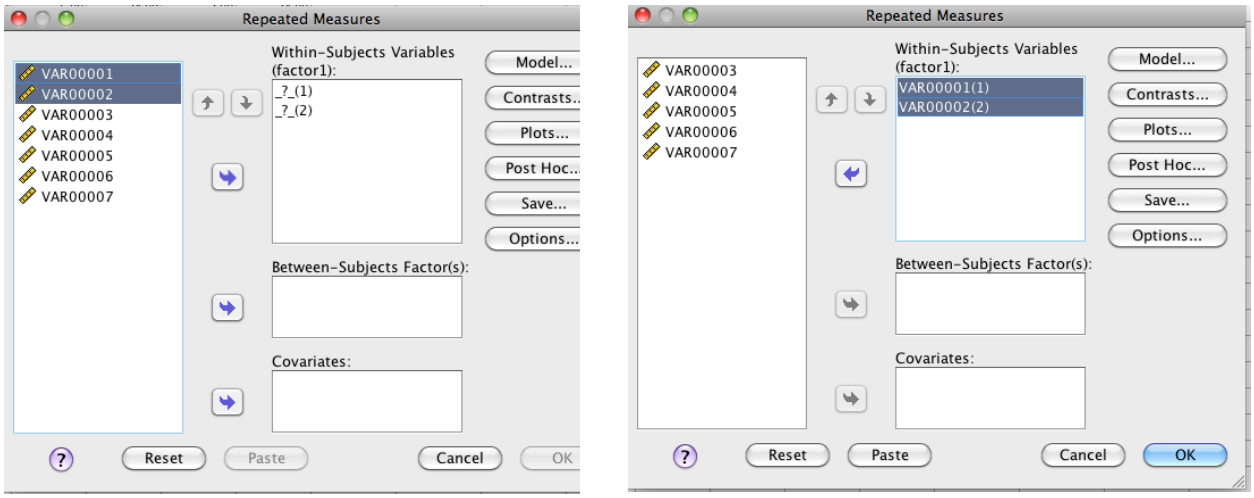


Figure 279. Defining the repeated measures anova.

6. The results will appear as illustrated in Figure 280.

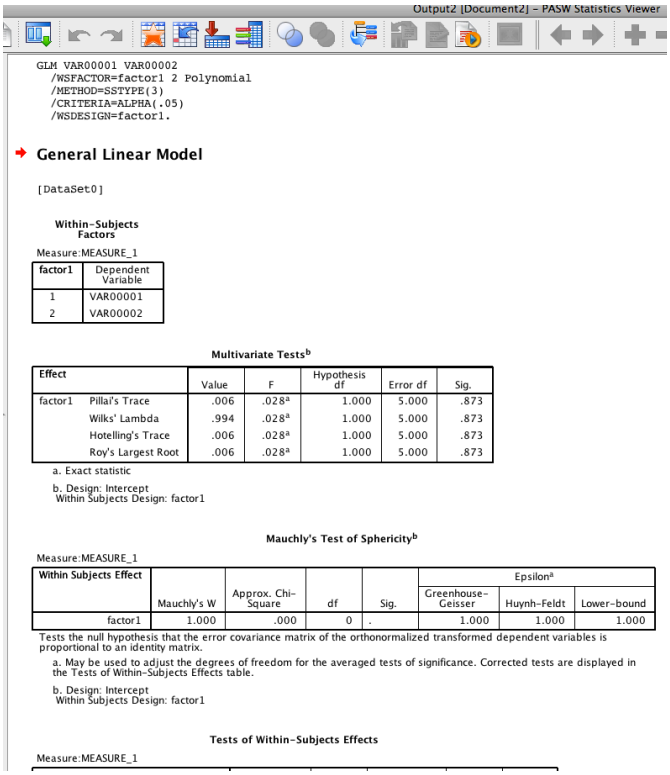


Figure 280. SPSS Output of repeated measures anova.

Trouble Shooting Your Model

- Nothing happens in when I press the start chevron
    - Once you press the start chevron in MIDAS, a communication port is opened at which point microsaint sharp launches a window entitled sharp talk host. This process takes about 10-15s. Once sharp talk host launches, you will need to connect both the environment (microsaint sharp) and MIDAS together so that the pieces of software can communicate with each other. To do this, click on the sharp talk pull down menu in micro saint sharp (the environment) and select connect. Do the same on the MIDAS side. You will see that the connection is made successfully in the sharp talk window.
    - The MIDAS start is accessed through the sharp talk window
- None of the operator activities other than auditory monitor begin.
    - Check to see if you gave the operator a beginning procedure.
- Jack™ does not start
    - Start the Jack™ License server
    - Double click the lmtools icon
    - Click on the Start/Stop/Reread tab
    - Click the Start Server button
- Nothing at all happens in Jack™
    - Make sure the animation flag in the simulation settings is set to true.
    - Make sure the Cport in Jack has been activated.

Customer Support

Contact Brian Gore at NASA Ames Research Center.
Brian Gore
(650) 604-2542
Brian dot F dot Gore at nasa.gov