agentTool 1.8.3 User's manual
# Table of contents

**BASIC GRAPHIC INSTRUCTIONS**

1. Adding ........................................................................................................ 1
   1.1 Objects............................................................................... 1
   1.2 Connectors............................................................................. 1
   1.3 HotKeys................................................................................. 1

2. Moving & Manipulating .......................................................................... 1
   2.1 Objects............................................................................... 1
   2.2 Connectors............................................................................. 1
   2.3 Snapping to Grid .................................................................... 2
   2.4 Manipulating ........................................................................ 2
   2.5 Multiple Objects ................................................................... 2

3. Deleting .................................................................................................... 2
   3.1 Objects & Connectors ............................................................... 2

4. Printing .................................................................................................... 2

**BUILDING YOUR SYSTEM** .................................................................. 3

1. Goals ....................................................................................................... 3

2. Use Cases .................................................................................................. 4

3. Sequence Diagrams ............................................................................. 2

4. Role Diagrams .......................................................................................... 2

5. Specifying Agents .................................................................................. 2
   5.1 Agent Template Diagram Panel ................................................... 2
   5.2 State Diagrams ........................................................................ 1

6. Deployment Panel .................................................................................. 1

**VERIFYING CONVERSATIONS** .......................................................... 4

1.1 Verifying Conversations ..................................................................... 4

2. Set Up for Verification Purposes .......................................................... 7
   2.1 Within VisualAge ................................................................... 7
   2.2 Outside VisualAge ................................................................... 7

**AGENT TOOL V1.8 ADDITIONS (2-9-01)**........................................... 8
AGENT TOOL V1.8 BUG FIXES .............................................................. 10
BASIC GRAPHIC INSTRUCTIONS

1. Adding

1.1 Objects
To add an object, click on the button on the left hand of the screen that is labeled as the object that you wish to add. (i.e.- the “Add Agent” button adds an agent).

1.2 Connectors
To add a connector, at least one object must already exist on the panel. If it does not, there is nothing to connect and a message in the message box will tell you that you must first add an object. Once you have at least one object on the panel, you may begin to add a connector. There are three steps to adding a connector. They are:
   1. Click the correct button. (i.e.- if you wish to add a conversation, click “Add Conversation”)
   2. Click on the first object that you wish to connect. (Click within the box for that object)
   3. Click on the second object that you wish to connect.
If you wish to create a loop, just click on the same object twice.

1.3 HotKeys
On many of the buttons you will find you can also use hotkeys alt-XX to “press” the button without pressing it. This allows you to quickly add many components without moving the mouse back and forth to the buttons. Also, many of the menu items also have hotkeys associated with them. (XX will be underlined on the button text. i.e. – “Add Agent” button has alt-a for its hotkey.)

2. Moving & Manipulating

2.1 Objects
Moving- To move an object, click and hold the mouse somewhere inside the object you wish to move and then drag your object to wherever you wish on the displayed panel.

NOTE: It is not wise to move the selected object off the displayed panel. You may not be able to recover the object

2.2 Connectors
Moving- Connector endpoints will move with their objects. Internal points may be added to the connectors between the two objects (except for loops). To do this, drag on the connector between the two objects. Try to stay a good distance away from the components when doing this. This will add another point to your connector that can then be moved around just by clicking on it. When Components are moved, the points between the object do not move, so you may have to manually move them to meet your visual desires.
When multiple connectors are connected to the side of an object, the component arranges them in a manner it sees fit. If this order does not mesh with your requests, you can change it. To do so, click on or very near to the point where the connector and object connect. Now, drag the mouse above or below the closest connector. You will see that the two connectors will then swap places.

It is easy to clutter your screen with lines that contain many points, and often you find you have created a point that you do not want anymore. To remove a point from a line, simply drag it on top of another point in the line, or either end of the line. The two points will automatically merge when they are within a short distance of each other. It can sometimes be difficult to see the points in the line, so when you select the line they are now circled in blue.

### 2.2.1 Example

There are three connectors going into the left side of component “A”. Let us call them C1, C2, and C3, from top to bottom.

If we want C1 to be below C2, click on the end point of C1 (outside of the component “A”) and drag the mouse below the end point of C2. They will automatically switch. Now the order of connectors into the left side of component “A” from top to bottom is C2, C1, and C3.

**NOTE**: Do this slowly. Just switch two components at a time

### 2.3 Snapping to Grid

While arranging your numerous objects on the screen you may find it helpful to arrange your items along a common gridline. While there is no visible grid, there is a function to snap objects to a grid point by their top left corner. You can do this by simply holding the shift key down while you move the object around. You’ll notice that it will lock to one position until you move the mouse closer to the next grid point. Java can be a bit touchy, so if the snap is not working try releasing the shift key and then pressing it again.

Possibly more useful than the object grid snapping is the function to snap connector lines. This function also works by holding the shift key while you move a connector point around. However, the line is not snapped to a grid. Instead when your line is close to being vertical, horizontal, or at a 45 degree, the line will snap to these angles. You will find this makes it very easy to arrange many connectors into a clear and organized pattern.

### 2.4 Manipulating

Some of the connectors can be reversed, made to originate from different objects, and made to connect to different objects. If you wish to do this, select the connector with the left mouse button and then click the right mouse button to get the popup menu to appear.
If you choose to reverse the connection, all you have to do is choose “Reverse”. If you wish to change the objects that the connector is attached to, there is a series of three steps to be followed.

1. Decide if it is the initiator or the responder that you wish to change.
2. Make the appropriate selection from the popup menu.
   (i.e.- if you wish to change the initiator choose “Change Initiator”)
3. Click on the object that you wish to connect it to.

2.5 Multiple Objects
To move multiple objects, hold down the shift key while selecting the objects. If you wish to deselect an object, hold down the shift key and click on that object. To move the multiple objects, select one of the selected objects and move it. The rest will follow in formation. (This option does not currently work for all panels. –JAG 1-12-01)

NOTE: It is only necessary to move the objects (not the connectors). Connectors move with their relevant objects. However, the interior points (not the end points) do not move with the objects. Therefore, it is necessary for the user to move these after the objects have been placed appropriately.

3. Deleting
3.1 Objects & Connectors
To delete, right-click over the object or connector you wish to delete and select “Delete” from the popup menu. Please keep in mind that you will have to remove all the connectors from an object before you will be allowed to delete the object. Also, certain objects have references in other panels that will have to be removed before it can be removed. You will be notified of such a case in the output field on the left side of the screen.

New!: You can delete any object by selecting the object and pressing Delete on your keyboard. You will get a popup dialog that asks you to verify your intention. It also gives you the option to never ask you again. If you choose not to ask and then press yes, AgentTool will not ask you again for the duration of this runtime. It will ask again if you restart the program, so don’t worry about messing it up for others.

Finally, in order to remove a point from a line, simply drag it on top of another point in the line, or either end of the line. The two points will automatically merge when they are within a short distance of each other.
4. Printing

By selecting the print options from the File Menu you can print a single panel of your system. Currently there are two options when printing. You can either print the default number of pages determined by panel active width and height, or you can scale that panel down to a single page. Printing is a bit slow at this point in time, so please be patient. Also, be aware that Java printing requires a significant amount of temporary swap space on your C: drive. If your print job does not process within about 1 minute, or you receive full-disk errors, you may need to clear more space. It would be safest to give Java between 75-100 Mb of drive space to operate.
**BUILDING YOUR SYSTEM**

1. **Goals**

If there are no goals currently in your system, then you simply select the “Add Goal” button to create a new goal. This goal becomes your root goal and is positioned at the top center of your screen. Because the goal hierarchy is a tree, subsequent goals need to be placed underneath your root goal. Therefore, when you select “Add Goal” a second time you will be prompted to select the parent of the new goal. Simply select the parent goal with your mouse and the child goal will be added in a position directly beneath its parent.

To change the name of a goal, use the popup menu to pull up the description window and enter the new value. You will not be able to change the numbering of the goals using this method as the numbers are assigned by the system.

Generally speaking, goals must be implemented in a role (see Roles section). Until this has happened, the goal will remain highlighted. In the image above you can see that “1.3 Verify User ID and Password” has been assigned to a role.

There are goals that do not need to be implemented in a role. These goals are completely implemented through their children goals, so they are called “partitioned”. To set a goal to be partitioned, simply select “Set Partitioned” in the popup menu for that goal. The goal will then be displayed gray and will not appear in any role lists. Goal “1.2 Get Password” is one such goal. If a goal is already assigned to a role, it must be unassigned before you can partition it. The program does not check to make sure a goal has children, so it will be up to you to verify the consistency of your system.

Several new features have been implemented on this panel. First, there is a level locking function that restricts movement of goals to the horizontal. This is intended to preserve the visual tree structure of the goal hierarchy by keeping goals of the same depth on the same level. It can be toggled off and on through the panel popup menu. Toggling it back on will lock goals into their current vertical position. The second feature is the Auto-Arrange function in the popup menu. This will process your goal hierarchy and algorithmically arrange the goals into an ordered tree, with parents centered above their children. It is often the case that this will extend the tree beyond the width of the page, in
which case the panel will automatically resize. This function does have an undo in case of an error, or if the result is simply undesired. Undo can be run from the popup menu after an Auto-Arrange has been performed and before any new additions have been made.

More new features have recently been added to deal with reassignment of the goal parents. All of these appear on the popup menu for a goal. To reassign the parent of a goal you would select “Change Parent” from its popup menu and then click on the goal you wish to become the new parent of this goal. You can also assign (and remove) secondary parents. These are handled differently than the primary parent in order to preserve the tree structure of the goal hierarchy. Select “Add Secondary Parent” from your child’s popup menu and then click on the goal you wish to add as a secondary parent. To remove it, select “Remove Secondary Parent” on the child and then click on the goal to remove. If you wish to make a secondary parent the primary, you need only use the “Change Parent” function and the existing primary will be switched to secondary and the selected secondary parent made primary. There are checks in place that will prevent you from assigning a goal’s descendant as its parent. However, it is still possible to create cycles in this system, so it is up to you to make sure that you are being consistent with your assignments.

One final feature that has recently been added is the ability to reorder your goals. In reassigning parents it is very common to wind up with goals in an ordering that is not satisfactory. You will find at the bottom of the popup menus two items labeled “¬” and “→”. These items respectively will decrement and increment the index of the selected goal within its group of siblings. That is, if a goal is labeled 1.2.3 and you decrement, the goal will be relabeled 1.2.2 and the previous 1.2.2 will become 1.2.3.

Please keep in mind that none of the parenting and reordering algorithms can be applied to the root goal. This is because the root goal is a critical node of the tree and we cannot allow it to loop. This means that whatever you do in your system you must first make sure that your root goal is what you really want because it will be difficult to change it halfway through.

2. Use Cases

To add a new use case, select the “Add Use Case” button. The new use case will appear in the list on the left. You can change the name of the use case by selecting the case in the list and selecting “name” from its popup menu. By typing in the description field on the right you can provide a short description for the selected use case.
To add a sequence diagram to a use case, select the use case and then push the “Add” button at the bottom of the screen. A new sequence diagram will then appear in the list of Sequence Diagrams. You can set the name and description of the sequence diagram by selecting the appropriate command in its popup window.

3. Sequence Diagrams

Defining a sequence diagram in a use case is the first step in defining a sequence diagram. Every sequence diagram contains a definition of some subset of roles and the messages that travel between them. You can edit this information by selecting the appropriate sequence diagram on the Use Case Panel and then switching to the Sequence Diagrams Panel next to it. (If you have not selected a sequence diagram before entering this page, you will be gently reminded.) The selected sequence diagram will then be loaded onto that panel for editing. You’ll notice that there is a small plaque in the upper left corner of this panel that tells you what sequence diagram you are working on.

Once you have loaded a sequence diagram you can now add role instances and messages. To add a new role instance, select the “Add Role” button and refer to the dialog that pops up. This dialog allows you to select an existing role (see Role Diagram) or to create new roles. If you choose to create a new role, it will automatically be added to the Role Panel. You can change type of role assigned to a role instance by selecting “Type” from the popup menu for that role. The Role Selection dialog will reopen, allowing the user to change the role assignment at that point.

Messages may be added by pushing “Add Message”. You will be prompted to select the two role instances that the message will connect. Once added, the message will appear between the two line descending from the two selected roles. You can set the message name through the popup commands. A message may be moved by dragging it up and down. Another option is the autospacing command in the role-line popup.

The lines descending from the roles will adjust their length automatically as more messages are added. If it becomes necessary to increase the length manually, this may be done by opening the popup menu over the vertical line and selecting “Increase Line Length”. 

No Sequence Diagram has been selected!
4. Role Diagrams

As you can see from the (simplified) diagram below, role diagrams can become very complicated very quickly. I will endeavor to break down the three primary pieces for you.

The first objects I’ll discuss are the roles. “Airline Manager” and “Scheduler” are examples of roles. These represent fundamental responsibilities that an agent will perform. Roles must be implemented by at least one agent. Until they are, they will remain highlighted. “Scheduler” is a role that has not yet been selected by an agent. The numbers inside the role display represent the goals that the role fulfills.

To assign roles to a goal you must select “Goals” in the role popup menu and refer to the dialog that is spawned. This dialog lists all the unassigned goals and all the goals currently assigned to this role and allows you to add and delete goals from the role.

Next there are tasks. “Develop Plan”, “Get Airplanes”, “Define Airline Problem”, etc. are all examples of tasks. They will appear as green ovals with the task name inside. Every task belongs to a role and is connected to that role by the light green lines you see above. In adding a new task you will be prompted to select the role that task belongs to. You can reassign the parent role by selecting “Change Role” in the task popup menu and then selecting the new parent. The task will then be moved next to its new parent role.
Every task has a state diagram associated with it whose panel (Task Panel) will become available as the task is selected. See the section on State Diagrams for more information.

Tasks are one of the few objects that can be copied. By selecting “copy” in the task popup you create a reference that tells the system which task you wish to copy. You then paste copies by selecting “Paste Task” in the role popup of the new parent role. This will then copy the contents of the task including the state diagrams and create a new task under the new parent role. It is important to realize that this operates by reference. That means if you make changes to the task after selecting it for copy, those changes will carry over into the copy. This can actually be helpful if you want to make lots of copies with progressive changes in each copy.

Finally there are protocols. These are the solid red lines and the dotted blue lines that connect between tasks. The reason there are two types is that protocols have an internal/external setting. You can think of this as defining whether the connection will be a message inside a single role or a message between two different roles. That is, protocols connecting two tasks of the same role are generally internal. You can see this clearly from the protocols into and out of the “Define Airline Problem”. All of these protocols connect tasks of the same role and are internal. However, the protocol “Request Non Local Plan” connects between a task of the “Airline Manager” and one of the “Scheduler” and is therefore external. The reason I am stressing this point is that the assignment is not an automatic one. All protocols are initialized as external. You will be responsible for determining if you want to set them internal. You access this option through the protocol popup menu.

Because the role diagrams get very cluttered very quickly, it is possible to hide the tasks and just see the roles and inter-role-protocols. The popup menu for the panel will have the option “hide Tasks”. By selecting this, the tasks will be hidden and only the roles and protocols will be shown. However, before you can add another protocol the tasks must be made visible again.
5. Specifying Agents

Agents are the core product we wish to produce using AgentTool (who’d have guessed?) These abstracted objects represent a single entity of operation, perhaps a program on a computer. The inter agent relationships we define here are analogous to the network protocols used on the Internet. They define what the agents will say, how they will say it, how they will process that information to accomplish their individual tasks and the overall goal of the system. The next several sections outline the creation of these agent entities and defining their connections and internal operations.

5.1 Agent Template Diagram Panel

The first stop in the process of building our agents is the Agent Template Diagram panel. This panel (shown above) allows us to declare our agents and conversations. You add a new agent by simply pressing the button. Your agent will appear on the screen in an algorithmically determined position. To add conversations you will be prompted to select the two agents being connected. Editing the names and descriptions of these objects is fairly straightforward through the popup menus. One exception is the conversation name. The popup menu instead has a command “Properties” that lets you edit the name and also a field called multiplicity.

An agent will implement some of the roles defined earlier. Different agents can define the same role, as in the above example many of the roles define the Resource role. Abstractly, the roles operate the same way, but each agent implementation may manage a different type of resource. “Truck Manager” manages trucks, the “Airport” might manage its runways and fuel resources, while the “Airline Manager” has a set of
airplanes that it manages. You use the abstract functions the roles define to build a specific function that the agent performs. In order to assign a role to an agent you select “Roles” from the agent popup menu and refer to the dialog that appears. It operates in a very straightforward manner, just as the Goal selection dialog did for the roles.

Agents are another of the few objects that can be copied. By selecting “Copy” from the agent popup menu you create an instant copy that appears at the top left corner of the screen. This new agent contains a complete copy of all the agent’s components, their connections, state diagrams, etc.. It contains everything about the agent except for the inter-agent connections and deployment information.

Lastly, the conversations have a few menu operations that allow you to change the direction of the conversation or to change which agents it is connected to.

5.1.1 The Agent Architecture

Every agent contains an internal definition that defines how the agent will accomplish its declared roles. We call this the agent “architecture” and this is comprised of a set of components and connections between the components. You can edit the agent architecture by selecting the agent in the Agent Diagram Panel and then moving to the Agent panel that is added. In the below example, we had selected the Scheduler agent and the tab “Agent: Scheduler” was added. Moving to this tab we will see the first level architecture for that agent.

![Agent Architecture Diagram]

The architecture pictured above is a fairly simple one. It has two components (large blue squares) and two connections. Creating new instances of these types is straightforward just as every other object and connector. As you can see, there are two types of connectors pictured here (red line and the green dotted line). I will go into more detail about those in the section on connectors.

5.1.1.1 The Agent Components

An agent component is the part of an agent that defines how the agent accomplishes some part of the agent’s roles. You can imagine it like devices in a computer. A sound card processes audio, the modem connects to the network, the video card displays visual information, the CPU processes the computations, etc. There are many levels of components in a system, just as there are in the computer. You could think of the
motherboard as the main component of a computer. The motherboard has many subcomponents that allow it to function: the CPU, memory, hard drive, CD-ROM, video card, sound card, network card, etc. Likewise, each of these has subcomponents in them that allow them to work. This is what you must keep in mind when designing the agent components. Each component you build has an option for a sub-architecture that may or may not need to be defined depending on the scope of your problem.

A component has several parts to it. First of all, as mentioned before you have the option to give a component a sub-architecture. You access this through the “Properties” dialog. (note : the component constraints section of this dialog does not seem to be in use at this time). In order to edit the sub-architecture you must select the component, much as you selected the agent to access its architecture. A new tab will be added that allows you to edit this component’s architecture.

The second part of a component is its attributes. These are sort of like variables in a computer program. You define their name, type, initial value, whether they are user defined or run time defined, and what type of set they represent. Attributes are shown above in the middle section of the component. Specifically, we should look at the “sorry” attribute. The display given here tells you most of the information about the attribute. The + and – in front tell you that (+) it is user defined and (-) that it is run time defined. The name of the attribute is given next. If you have a Set, Sequence, or Bag type set then this will appear next, and then the attribute type (“type” in this case).

The next part of the component is the method definitions. Methods are similar to functions in a computer program. Abstractly, they receive certain parameters under certain setup conditions, and produce a response of a given type. There are several fields we will have to consider. A method has a name, a return type, parameters, a precondition, a postcondition, and visibility. If we look at the “getPlan” method we can see how this will be displayed. The # indicates that a method is not visible. “getPlan” is visible and so does not have a #. Next is the method name. The parameters are listed inside the parentheses of the method. After the colon is the return type. The precondition and post condition are not displayed here.

Each component has a state diagram associated with it. You can edit the state diagram by selecting the component and moving to the State Diagram Panel that is added.
Component connectors represent the message pathways between components. There are two types of these connectors. The first and primary type is the component to component connection. These are represented by red lines. The only information the connectors contain is which components they connect and the direction of message travel in an abstract sense. A connector can be set to allow traffic in only one direction, or to allow two way communications. You access these settings in the connector popup menu.

The second type of connector connects one component to the “framework”. The actual definition of this connection is a bit sketchy. In one line of thought, it can represent an input/output port for the architecture. You add an external connection as if it were a loop, that is, you set the starting and ending component to be your source component. The program automatically interprets that to mean an external connection. Currently all external connectors will link to the same endpoint, the red and green triangle linked to the left side of the panel. In the future, when the external connector concept is further developed this endpoint might be defined as an actual referenced port. For now, treat it however you like because it only means whatever you make of it right now.

5.1.2 Conversations

Agents communicate to one another through conversations. A normal conversation has an initiator and a responder agent. It is very intuitive which is which. You have the option to reverse the conversation, or to change the initiator and responder through the popup menu. Other options you have are to edit the name and multiplicity values through the “Properties” command, and to edit the description.
There are special conversations whose initiator and responder are the same agent. Sometimes this is desired in designing the agents. These special conversations are shown as loops in the example above. As you can see, my schizophrenic agent likes talking to itself a lot. Some of the loops must overlap other lines. When there are loops and lines on the same side of an object, they are handled separately and will not interfere with each other’s position. However, when there are multiple loops on a single side then we have to arrange them in some fashion. I have chosen an algorithm that places the last added loop on the outside and the others on the inside in the order they are added. In order to grab a loop you need to select somewhere within the rectangle defined by its top and bottom edges and its right and leftmost edges. If there are multiple loops, the smallest loop selected by your mouse is the one selected. Therefore it is recommended that you attempt to grasp the loop near its peak where there is the least overlap of area.

Every conversation has two diagrams that describe it: the Initiator State Diagram and the Responder State Diagram. Tabs for these will appear when the conversation is selected.

5.2 State Diagrams

The state diagram allows you to define a standard state table. In practice, our object would begin in the start state and based upon the values of some input data it switches states appropriately. This table is an abstracted way of setting these functions up.

5.2.1 States

Two states always exist in the state diagram. The green circle is the start state and the red circle is the end state. These are special states that can never be deleted or changed. The real power of the state table comes from the normal states. These are the rounded rectangles you see above. Each state has a set of actions it would perform.
To define the actions of a state, select “Define Actions” from the popup menu. To add a new action type its description in the textfield at the bottom of the dialog and press add. This action is added to the list and can be moved higher or lower priority as seen fit. To edit an action, select the action in your list and click the “Edit” button. The action will then appear in the Action text box. Change the action and then click the “Add” button. The new version of the action will then replace the old in the list.

The two remaining options on the states popup menu are “Delete Error” and “Delete”. The second is used for deleting the state. Before the state can be deleted, the transitions that are connected to it must first be deleted. The “Delete Error” is used when verifying a conversation. If there is an error in that state, then the state will be highlighted. Once the error has been corrected, remove the error with the “Delete Error” option and then test the conversation again.

5.2.2 State Transitions
To specify the properties of the transition, select the properties option from the popup menu. A new window will appear. Fill in the appropriate text fields. When finished, click “Apply”.

To define the actions of the transition, select the “Define Actions” option from the popup menu. Then define the actions in the same manner as for a state.

A Transition has markers that label the start and end parts of the transition. These options are selectable in the popup menu.

6. Deployment Panel
The deployment panel is the first step in developing actual working systems of agents. Until now you have been specifying abstract functionality you want your system to have. Now that these things are set up you are ready to define a working system. In this section you will create instances of your agent templates and conversations, and then connect them in systems. Let’s refer back to the package system we looked at earlier. There are agents defined for managing an airport, a post office, a truck shipping hub, etc.. In real life we want to have a separate agent for every airport, every post office, each trucking hub. Each will operate independently while communicating its needs to the rest of the world. This is where deployment comes in. We define, say, three airports, two post offices, and a truck hub. We then set up the way we want their network of conversations to run. Then we can identify certain portions as belonging to a single group, or system.
To add an agent to the Deployment Diagram, at least one agent must already exist in the Agent Diagram. If this is true, push the “Add Agent” button and select the correct agent. To add a conversation in the Deployment Diagram, a corresponding conversation in the Agent Diagram must exist. When you add a conversation a dialog will pop up that will ask you to specify the type of conversation you want to add. The conversations in the list include all the conversation that exist in the Agent Template Diagram. (*NOTE – for old MAML files this information was not given. The system will load them as null conversations and function properly within that context. However, it is suggested that you go back and assign a value to those conversations).

To add a System to the Deployment Panel, click the “Add System” button. Then click and drag the mouse in the diagram to place the system appropriately. The systems can be resized just as a window on your OS. Just “grab” one of the edges or a corner with your mouse and drag it to the correct size.

The systems in the Deployment Panel act as solid units. If you move a system, any agents contained within it will be moved likewise. If you want to move an agent outside the system, just grab it by itself and drag it. You can select multiple objects to move by holding down the shift button as you select items. If you have compound systems (ie – a system completely contained within another system) it will be handled in the same way an agent would be within the larger system. You can picture it like a pyramid. The largest system is the base. By grabbing the base you move everything on top of it. Smaller systems sit on top and can be moved around individually. Agents sit on top of the systems. Conversations float between agents. Any objects that are not completely contained within a system will not be selected unless you select them individually.
In the example image above, you can see compound systems. System 3 is contained within System 1. System 2, however, is not contained by System 1. All three agents are contained within System 1. If we grabbed System 1, all visible objects would be moved except for System 2. If we grabbed System 3, then we would be able to move System 3 and Write Agent #1.
1.1 Verifying Conversations

To verify a conversation, the conversation must first be created. Input the transitions’ properties in both the Initiator State Table and the Responder State Table. Once this has been done, select “Verify Conversations” from agentTool’s Commands Menu.

A window will appear called Verification Messages. When the verification is complete, the line

```
****************** Testing Completed ******************
```

will appear. At this time, you can scroll back up through the Verification Messages window and see if the conversation(s) were verified or if they have errors. If errors exist, the transitions or states with errors will be shown on the screen and will be highlighted in agentTool. A simple example is below.

The initiator of the conversation sends the responder message \( a \) and receives message \( b \).

The responder of the conversation receives message \( a \) and receives message \( b \).

Now, verify the conversation. The following window will pop up.
The verification window shows that the conversation is deadlocked. Now, look at the agentTool panels after verification.

The Initiator Panel has the transition where b is being received highlighted as well as the end state. The transition b is highlighted because b is never received. The end state is highlighted because since b is never received, the end state is never entered.

The Responder Panel also has a transition and the end state highlighted. Once again, b is never received and the end state is never entered.

So, let us change the Responder State Table so that b is sent in the second transition.
Now, by re-verifying, the changes can be checked to see if the error was corrected.

The verification window shows that the conversation is not deadlocked, they do not have any unused states, and so they are correct. The state diagrams have also been changed:

The errors have been erased.
2. Set up for Verification Purposes

2.1 Within VisualAge

Begin by ensuring the resource files are loaded into VisualAge. When importing the resource files from a jar file, the files are placed in a project_resources directory. For example, if VisualAge is installed in the c:\IBM\Java2 directory, and your project name in VisualAge is agentTool, then the resource files would be placed in the c:\IBM\Java2\ide\project_resources\agentTool directory. All of these files and directories must be copied into the c:\IBM\Java2\ide\program directory for the program to work correctly.

When executing the verification routines from within VisualAge, the user directory has to be specified by setting the appropriate system properties. This is done by right clicking on the ATEditor class and selecting the properties menu option. In the system properties box, enter the path to the program directory where the resource files were copied. For example, using the directory above, the command would read user.dir=c:\IBM\Java2\ide\program.

2.2 Outside VisualAge

The verification routines can be tested without invoking agentTool by running the VCRun class in the Verify package. Before VCRun can be invoked, the user.dir command must be setup in the system properties for this class as previously mentioned.
As of this point (June 15, 2001) I have made more changes and updates and little fixes and big fixes that I cannot count or even remember them all. As I have approximately 35 minutes left in my project term I will not be documenting all of this information here. Most of the functionality items have been addressed in the preceding document. If you want to know about the details of bug fixes and how the program operates internally, you are welcome to open ‘er up and take a look at the source code, or you can contact Major Scott DeLoach (sdeloach@computer.org). Your best bet is to forget about it unless you are a developer, in which case you are stuck with trying to understand how I program anyway.

This is a work in progress. Please send any bug reports you have to Major Scott DeLoach (sdeloach@computer.org) or Lt.Col. Jacobs (Timothy.Jacobs@afit.af.mil).

-Jason Gunsch
June 15, 2001

AGENT TOOL V1.8 ADDITIONS (2-9-01)

- Deployment Panel System enhancements – Reworked the deployment panel’s selection and movement functionality. Implemented new standard layout involving a principle of system layers and containment. Agents that exist within systems are recognized as part of a system and dragged with the system box automatically. Nested systems are recognized as subsets of a larger system and are moved with the larger system box automatically. We retain the functionality of moving singular items separately regardless of position, with agents being primary select items, nested systems next, then base systems. We also retain the ability for multiple item selection. (2-12-01)

- Printing – The panel printing functionality has been implemented now. The default function to print all pages has been implemented, as well as a function to scale the panel to one page. Further enhancement to the page layout is in development. (2-9-01)

- Unique Keys now assigned to all objects. This allows all objects to be loaded and saved as unique objects, regardless of name. This eliminates problems with identically named objects being mislinked in our Vectors. It also eliminates the need for unique naming conventions that we had implemented prior to now. Not all these conventions have been abandoned, but they are no longer critical to the operation. (2-5-01)

- Goal Parenting – Goals now can have multiple parents (with one primary parent). All the functions to add, remove, and otherwise manipulate them have been added. One can also reassign the primary parent at will, allowing quick restructuring of the goal hierarchy. Finally, we have implemented functions to increment and decrement the goal numberings to compensate for the jumbling involved with our reassignment processes. (1-29-01)

- Role reassignment in Sequence Diagram – Now our sequence diagram roles can have their role type changed by the user on the fly. (1-19-01)

- Message Line Extension – The message lines in the Sequence Diagram can be extended downward at will, and will also extend themselves when appropriate numbers of messages have been added to necessitate the extension. (1-12-01)
• Goal level lock – by default goals can now be moved only horizontally. This is intended to keep vertical cluttering down and maintain the visual appearance of the tree structure when adding new goals. This can be toggled off in the popup menu if necessary, since older MAML’s were created without level locking and are generally not organized into a tree. Goals can be resituated manually by turning off the level lock, or one can use the new auto-arrange function.

• Goal Depth Labeling – goals of version 1.8 or greater are now automatically labeled vertically instead of horizontally. Prior to this, a goal description had at most two lines of text which spanned horizontally as far as necessary. Now these labels are set to a limited width with an expandable height. This adaptation was done to eliminate extreme horizontal spanning in the tree structure. Goals prior to v1.8 are labeled as before for consistency with their existing relative layout.

• Auto Arranging for the Goal Hierarchy – added popup menu and functions to handle auto-arranging goals. It rebuilds the goal placement into an orderly and visually uncluttered structure using the internal tree layout of the goals. It recursively arranges a goal’s children and then places the goal in the middle above its children. This function also will expand the display as necessary in order to compensate for width and height settings. It will also update all the goal versions to 1.8 to utilize the new depth labeling feature.

• Undo – added an undo function for the auto-arrange operation. This is currently the only automatic main operation that cannot be undone by simply deleting an item or renaming or resetting a property. It provides the basis for a MAML-based function that can be extended to undo nearly any operation.

• Panel Size setting on startup – As a side effect of the Auto Arrange function, it became common for Goals to extend beyond the default window range. Now there is a function in place that will set the size of the display after reading the MAML. The function tests all visible elements and finds the minimum width and height the screen must be to accommodate the display.

• Graphical enhancements to segmented lines - added blue circles at each vertex when selected to aid in grabbing and shaping lines.

• Transform Menu – added a transform menu with menu items for Add Agent Components, Annotate Component State Diagrams, and Create Conversations. These functions will be scripted later to suit the needs of 1Lt. Clint Sparkman in his research.

• Internal vs. External Protocols – added menu popup, internal data members, graphical changes, and MAML I/O to keep track of the locality of Protocols.

• Goal Partitioning – added menu popup, internal data members, graphical changes, and MAML I/O to keep track of the partitioning of goals. We also updated the assignment of goals to roles so that a partitioned goal is not assigned to a role, and an assigned goal cannot be partitioned until it is unassigned.

• Start State / End State Transition – added menu popup, internal data members, graphical changes, and MAML I/O to keep track of the new start / end status markers in Transitions.
AGENT TOOL V1.8 BUG FIXES

- Move operations changed from delta functions to actual position assignments. That is, instead of moving the object however far the mouse moved, we put the object where the mouse currently is, or wherever is closest within the object’s boundaries. This eliminates the problems encountered with getting the mouse shifted off the object. (1-01)

- Memory consumption has been significantly reduced by removing the panel images and rerouting all paint and update functions properly. The update functions now are unused. When repaint is called, each panel is updated properly now using global graphics objects and images. (1-12-01)

- The popup menu malfunction has been patched up. This occurred mainly on the Sequence Diagram Panel. The error would be triggered when no objects were selected because it would attempt to show an empty popup menu. This would cause the menu to be disabled and would cease to activate. This was fixed by checking to make sure that an object was selected before trying to show() the popup. (1-5-01)

- The algorithms for attaching lines to squares and circles have been fixed to more properly arrange them with respect to relative screen position. (12-20-00)

- The algorithm that handles tab addition/deletion has been reworked. (12-4-00)

- Fixed the Menu Clipping error. When a menu was opened in front of the panel display of the Agent Tool Editor, it would be clipped off by the white box in the display. This was fixed by moving the call to superclass update function to the very end of the main update. (11-29-00).

- Dragging the mouse while adding objects in the Role Panel caused many errors in insertion and deletion, including reassigning the last task to a different role. This has been fixed, and the other panels updated to prevent possible bugs in them. (11-29-00).

- Components now have required unique names so that they are tracked properly. (11-29-00).

- Adding lines to a circle (esp. in state diagram) pointed line to center of circle instead of rim. This is now fixed. (11-28-00)

- Components are now counted properly, accounting for sub-architectures as well. Components added are initially numbered according to the current count, including all sub-components as well. (11-28-00)

- File operation cancel – you can now successfully cancel a save or load file operation, and also fixed the subdirectory issues I had with JFileChooser (11-14-00)

- Triple Quote MAML errors – Many MAMLs were being corrupted when saving a null String. The way the read MAML function worked was to read a token in a given location and set the String value to it. However, an empty String would cause the given location to hold a quote instead of a null value, and so the String would be set to a quote. On resaving, the string would save a triple quote. Once that had been done the file could no longer be parsed correctly. To fix this we set a check in the reload that will load a quote mark as a space instead. Second, we changed the initial string from an empty string to a space mark. (11-14-00)

- Goal Happy – in adding goals to a role, only the goals not currently assigned are displayed in the selection list. Upon adding a goal, that goal is removed from the selection list. Upon
deleting a selected goal, it is added back into the selection list. (11-7-00) Also, for agents adding roles, roles are not marked like goals, so all roles are loaded, but only those not already selected are displayed. This makes it possible to assign a single role to multiple agents, which is desired. (11-28-00)

- Component Attribute Editor – now fully tracks the data entered, stores it back into the component attributes, and loads it back into the editor on reload. (11-3-00)

- Eliminated several never-ending operations. Once started, these operations would never end until completed. This often interfered with the logical flow of the interface. We set conditions which would cancel each of these operations: (11-2-00)
  - State Diagrams -> Add Message
  - State Diagram -> Add Transition
  - Role Diagram -> Add Task
  - Goal Hierarchy -> Add Goal
  - Agent : AgentX -> Add Component Connection

- Segmented lines – fixed some errors with selection boundaries, vertex creation, segment processing. (10-31-00)

- Role numbering – roles loaded from a MAML are now counted properly and additions begin from the correct index. (10-31-00)

- Element spacing – added a basic spacing algorithm to Roles, Agents, Components, etc. so that they are no longer generated on top of each other. They may still overlap other elements by circumstance if you have moved things around, but at least if you create multiple elements at one time, they are separated and clearly visible. (10-31-00)

- Protocol Description popup – it has now been scripted to actually do something. (10-31-00)

- Protocol vs Role-Task Connectors – these two items now have different colors in order to distinguish between them visually. (10-31-00)

- Removed redundant commands in Component Connector popup (10-31-00)

- Open Space Selection – in the Goal Hierarchy panel, selecting open space now unselects all goals. (10-31-00)

- “Delete use case” button – now checks for selected use cases (10-27-00)

- Start State / End State class cast exception – you can no longer define Actions for a start state or end state in the state diagrams. They were never supposed to allow that as an option. (10-25-00)

- Asynchronous input with synchronous update – Changes to elements were not being graphically updated because the input dialog boxes were in asynchronous mode. We changed them to synchronous so now the update occurs after the dialog is closed. (10-25-00)

- Arrow Heads – The arrowheads previously did not initialize properly, causing them to draw all over the screen. They work now.

- Component Diagram Arrow Issues – Arrow information was being lost in the AgentX Component Diagram causing all but one arrow to be drawn and none ever deleted. All arrows behave properly now.
- Component Diagram External Arrows - The external conversation arrow (green dotted arrow) used to draw from the left side of a component at a fixed length, this sometimes caused its head to go beyond the window border causing it to be forever lost, until deleted. They now initialize to the top left corner guaranteeing that that will not happen anymore.

- Deployment Systems – When reloading MAML files with systems in the deployment panel the systems would tend to drift due to a mispointing index in the loadMAML function. They initialize properly now.

- Extra Conversations – There were issues on various panels where the internal counters would count extra conversations coming off of a box, thus causing you to be unable to delete said boxes even when all visible conversations were gone. These ‘extra’ conversations were tracked down and dealt with harshly.