CaseMIDAS - A REACTIVE PLANNING ARCHITECTURE FOR THE MAN-MACHINE INTEGRATION DESIGN AND ANALYSIS SYSTEM

R. Adam Pease
US Army Aeroflightdynamics Directorate
Mail Stop 262-3
NASA - Ames Research Center
Moffett Field CA 94035
tel. 415 604 0060, fax 415 604 0801
pease@whale.arc.nasa.gov

Abstract

MIDAS is a set of tools which allow a designer to specify the physical and functional characteristics of a complex system such as an aircraft cockpit, and analyze the system with regard to human performance. MIDAS allows for a number of static analyses such as military standard reach and fit analysis, display legibility analysis, and vision polars. It also supports dynamic simulation of mission segments with 3d visualization. MIDAS development has incorporated several models of human planning behavior. The CaseMIDAS effort has been to provide a simplified and unified approach to modeling task selection behavior. Except for highly practiced, routine procedures, a human operator exhibits a cognitive effort while determining what step to take next in the accomplishment of mission tasks. Current versions of MIDAS do not model this effort in a consistent and inclusive manner, CaseMIDAS also attempts to address this issue.

The CaseMIDAS project has yielded an easy to use software module for case creation and execution which is integrated with existing MlDAS simulation components.

Introduction

The joint Army-NASA Computational Human Engineering Re-search Office has been engaged in a effort since 1985 in modeling human performance in complex systems. The major software product of this effort - the Man-machine Integration Design and Analysis System (MIDAS) has begun to reach maturity in its role as a prototype design tool. [1]

Case based reasoning [2] is a significant subfield of AI and as a whole is concerned with many issues that CaseMIDAS does not currently address. Learning is one of these issues, partial case matching and analogy is another. CaseMIDAS embodies one central idea: that planning knowledge is best represented in the form of cases. A case may be viewed as a script or plan of actions, as the aggregation or instantiation of several rules for a given situation, or simply as a frame with a varying number of slots. Case based systems react to situations rather than planning a series of steps as in classical Al planning. Producing a plan consists of matching a current situation to the preconditions of an existing case rather than building a plan by searching through a space of possible component actions.

MIDAS has several components which represent the operator; structures that model the operator's menial state, physical state, representation of the world, and a system to select actions to perform. CaseMIDAS is most concerned with the last component. At this stage in development, it should be noted that CaseMIDAS does not incorporate the full range of functionality found in other versions of MIDAS.

This report discusses the software structure of CaseMIDAS and then presents narratives of program execution. Successive versions of increasing complexity are given. Version I introduces a simple version of case activation and distinguishes between actions which affect the world, and actions which are links to other cases. Version 1.5 adds parallel execution of actions and resource loading of operator actions. Then the current integrated version of CaseMIDAS is discussed along with issues for further development.

Software Structure

The cognitive components of CaseMIDAS have the software structure given in Figure 1. One

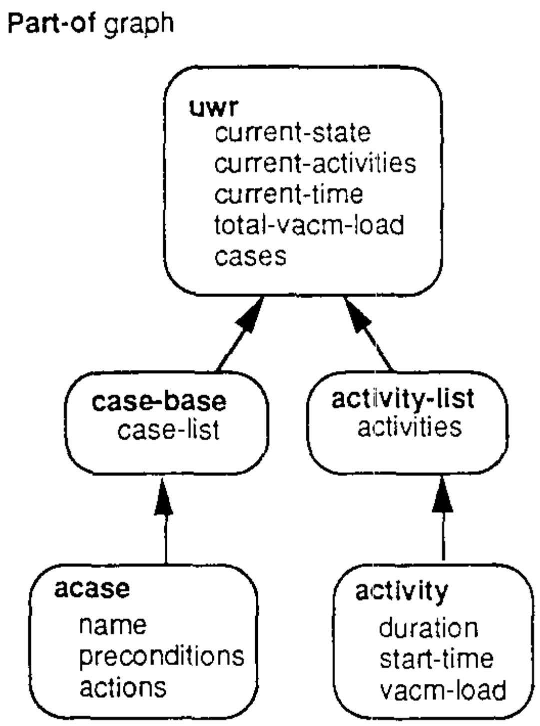

Figure 1.

component is in individual case which has a name, a set of preconditions that must be met before it is invoked, and a series of actions to be performed if the preconditions axe met. Cases are collected together in a *case-base.* This case-base is part of a structure called the updatable world representation (UWR). The UWR is a MIDAS software construct which includes simple representations of the mental processes of the operator, including memory storage and retrieval. These models are by no means definitive but are vital to having a complete (if simplistic) representation of operator cognition. In addition to the case-base, the UWR also contains the current state of the world, the set of low level activities that the operator is performing, the current time, and the total visual, cognitive, auditory, and motor (VACM) loading that the operator is experiencing. The VACM load estimates are from a simplified additive workload model patterned after the work of Aldrich and Szabo [3]. In CaseMIDAS the load estimates have been multiplied by a factor of ten, so the maximum level for each of the four channels is 70 units. In some simulations, the maximum for cognitive load is extended to explore the handling of multiple tasks. The current set of activities is represented in the UWR by the activity-list structure. Each activity contains information on its duration, the time it was activated and the estimated VACM load that the operator will experience due to its activation. At each time step during the simulation, the system looks at the tasks which have yet to be performed, and the current load the operator is experiencing. This load is the sum of the current activities and the cognitive load for planning the next set of actions.

Figure 2 shows the interfaces between the given components of the system. The central method is run-simulation in the case-base. It attempts to find a matching case by calling test-case for each case in the

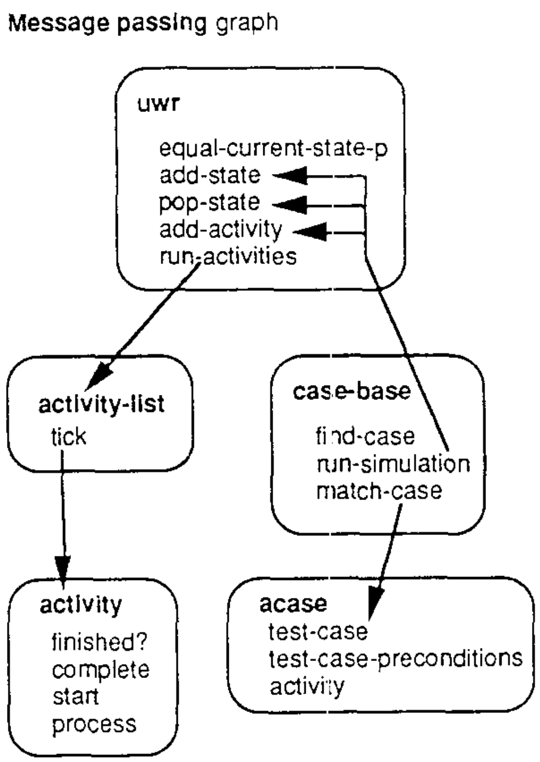

Figure 2.

case-base. If a case matches there are two possible situations. The first is that the actions in the case consist of a list of other case names. Each of the names is added to the current state and run-simulation is called again to repeat the match and run cycle. The second situation is that the action part of the case consists of *leaf activities.* Leaf activities are atomic actions that the simulated human performs such as pressing a button or looking at a display. The activities are added to the current activities list in the UWR and run-activities repeatedly calls the tick method on the activity list to execute the activities. The MIDAS system is a *tick-driven* rather than *event-driven* simulation. This means that a single global clock is used to schedule program actions at discrete time steps as opposed to a system in which world events trigger program actions at a variable time granularity.

<u>CaseMIDAS Version I Narrative</u>

This example is based on a commercial aircraft, pre-taxi checklist scenario previously modeled in MIDAS. Due to space limitations, the complete output is not given. This example shows the current state of the UWR and the processing of activities as they are added to the current-activities list. Note that for this version, all activities have the same duration and are run sequentially. `run-simulation` first tries to find a case in the case-base which has preconditions that are completely true. Each case has a precondition that includes a current state. `match-case` is called to test the preconditions of each case in the case-base until one is found in which all preconditions are true. When a true case is found, each action in the actions portion of the case is performed. If an action is not an executable leaf-activity (e.g. reaching for a button, looking at a display), then that action name is added to the current state, the current state of the system is printed, and the case matching process happens again. The first true case found is

```
(defcase process-clearance
   (preconditions (equal-current-state-p
                  *uwr*
                  '(AIR-MIDAS-ACTIVITY
                      PROCESS-CLEARANCE)))
(actions
  '(receive-clearance
  decide-whether-to-accept-clearance
  accept-clearance)))
```

In this instance, the actions are not leaf activities. The program will add each action in the action list to the current state list and repeat the match cycle. The program prints `(AIR-MIDAS-ACTTVTTY PROCESS-CLEARANCE RECEIVE-CLEARANCE)`
The case that matches that state has actions which are leaf activities. That case is

```
(defcase receive-clearance
   (preconditions
      (equal-current-state-p
         *uwr*
          '(AIR-MIDAS-ACTIVITY
            PROCESS-CLEARANCE
            RECEIVE-CLEARANCE))
        (eq
          (clearance-access-equipment-type-is
          'voice))
   (actions '(listen-to-clearance
               record-clearance)))
```

It also matches a condition that the clearance access equipment type is voice. The case has two actions which will each be executed in turn. So the program prints (AIR-MIDAS-ACTIVITY PROCESS-CLEARANCE RECEIVE-CLEARANCE LISTEN-TO-CLEARANCE) The activity is added to the activity list, the UWR ticks the activity list, and the activity LISTEN-TO-CLEARANCE is executed.

```
“starting activity"
“processing activity"
100
“processing activity”
200
.
.
.
"processing activity"
800
```

The activity is ticked in simulated 100 millisecond increments. Each activity lasts an arbitrary 800 milliseconds. When one activity is completed, that activity name is removed from the current state. The program prints (AIR-MIDAS-ACTIVITY PROCESS-CLEARANCE RECEIVE-CLEARANCE) and then adds the state for the next action (AIR-MIDAS-ACTIVITY PROCESS-CLEARANCE RECEIVE-CLEARANCE RECORD-CLEARANCE)

Once that activity is executed until completion, the program removes it from the current state. That also completes the actions for the receive-clearance case. The state corresponding to the case is also removed. At this point we are back to the first case. The next action is decide-whether-to-accept-clearance. This is added to the current state and the match and execute cycle continues. The program stops when there are no cases with actions left to he executed, and no new cases exist which have preconditions which are all true.

CascMIDAS Version 1.5

This version of CaseMIDAS added parallel activities, operator loading, and cognitive loading for case activation. The architecture is fundamentally the same as in CaseMIDAS version 1. One important method was added to the UWR to test the total operator load. The loaded? method totals the resource load for all the activities in the activity list and the load incurred by case activation, and checks the total load that would result from adding the next possible activity. Note that cognitive load from leaf activities and the cognitive load from case activation are added together. Cognitive load for case activation is added in the following way. When a state is added to the current state list, one unit of cognitive load is added. The choice of one unit is reasonable relative to the load measures assigned to leaf activities but it is arbitrary. When a state is removed, one unit of load is subtracted. When a case is activated, one unit of load is added for each action in the action list. When the case is finished, this load is subtracted.

CaseMIDAS Version 1.5 Narrative

This section shows the output of CaseMIDAS version 1.5. Again, due to space limitations, the complete output is not given. First the system is loaded and simulation on is called to start the program. Just as in the previous version, the program searches for cases with true preconditions.

```
(AIR-MIDAS-ACTIVITY PROCESS-CLEARANCE
RECEIVE-CLEARANCE)
(AIR-MIDAS-ACTIVITY PROCES3-CLEARANCE
RECEIVE-CLEARANCE LISTEN-TO-CLEARANCE)
```

When it encounters a case with leaf activities, it adds as many as it can given the current load and the loads for each activity. Here the program shows the load of the activity LISTEN-TO-CLEARANCE and the proposed total load if that activity is added to the activity list.

```
Activity load : Current loads: V:0 A:49 C:46
M:O
Proposed load : Current loads: V:0 A:49 C:53
M:O
```

Because the activity was successfully added, the program tries to add another activity. Note that 7 units of cognitive load have been added because cases have been activated. This is calculated in the following manner. The scenario starts off with two states in the state list AIR-MIDAS-ACTIVITY and PRCCESS-CLEARANCE. To this, two new states are added which contributes two units of cognitive load, one for each new state. The first case which is executed, process-clearance has three actions which each contribute a unit of load for a new total of 5 units. One of those actions causes the activation of the Case receive-clearance which has two actions. They each contribute one unit for a grand total of 7 units. CaseMIDAS prints the new state and loads.

```
(AIR-MIDAS-ACTIVITY PRCCESS-CLEARANCE
RECEIVE-CLEARANCE RECORD-CLEARANCE)
Activity load : Current loads: V:50 A:O C:46
M:65
Proposed load : Current loads: V:50 A:49 C:99
M:65
```

The proposed load is within the acceptable limits so the activity is added. The load limits were set to 70 except for cognitive load which was set to 100. The program tries to add another activity

```
(AIR-MIDAS-ACTIVITY PRCCESS-CLEARANCE
ACCEPT-CLEARANCE COMMUNICATE-INTENT
COMPOSE-RESPONSE)
Activity load : Current loads: V:50 A:0 C:68
M:70
Proposed load : Current loads: V:100 A:49 C:175
M:135
```

This time the load is too great. The activity is not added and the existing activities are executed in order to free some of the load.

```
startinq activity LISTEN-TO-CLEARANCE
processing activity LISTEN-TO-CLEARANCE
starting activity RECORD-CLEARANCE
processing activity REC0RD-CLEARANCE
Current time: 100
Activity load : Current loads: V:5O A:0 C:68
M:70
Proposed load : Current loads: V:100 A:49 C:175
M:135
```

When an activity completes, a portion the computed VACM load is freed and the system tries to add another activity to the activity list.

Integrated CaseMIDAS

Case.MIDAS has now been ported to C++ and integrated with many of the existing Core MIDAS software components. While the operation of

CaseMIDAS is essentially the same as in version 1.5, it is now possible to visualize the 3d anthropometric model, called Jack®, in the cockpit performing the activities as they are executed.

There exists a version of MIDAS which describes a pilot executing a pre start checklist in a Boeing 747-400 which is called TaxiMIDAS. The TaxiMIDAS scenario has been reimplemented as cases and an outline of the case file is given in following section. The declarative representation of cases has made the reimplementation very straightforward. A first version of the case base was written in a matter of hours. No coding was necessary. This version does not completely mimic the existing TaxiMIDAS scenario. During the scenario, the pilot monitors his equipment for the arrival of a clearance message from air traffic control. TaxiMIDAS models this as an activity which is performed in the background as the pilot goes about other casks. CaseMIDAS does not have a way to model background tasks but instead treats the presence of the clearance message as a case precondition.

Case and Activity Definition

This section shows the implementation of the TaxiMIDAS scenario in case form. The first portion describes the activities to be performed. Each activity has a type and a name. Then duration and VACM load values are listed. Then x, y, and z coordinates are given for the location which is to be looked at or touched. For reach, press, pull and rotate activities the part of the body and side of the body are also given.

Each case is described by a name, state preconditions, information preconditions, and state actions or leaf actions. State preconditions are given by a list of text items. Information preconditions are given as a list of name-value pairs, State actions are lists of text items that are added sequentially to the current state. Leaf actions correspond to the activity names listed in the first part of this File. Note that the majority of the file has been replaced by ellipses.

```
CaseMIDAS_data_file
activities
67
fixate fixateDoorsButton 400 59 0 68 0 -21.7896
-0.768 28.9644

reach reachDoorsButton 200 50 0 10 22
-21.7896 -0.768 28.9644 FINGER RIGHT
.
.
.

casename: taxiMIDAS
statePrecond: 0
infoPrecond: 0
stateActions: 9
checkDoors
checkSeatbelt
makeSeatbeltAuto
checkOxygen
checkFlightInstruments
recheckDoors
checkOil
engineStart
checkEICAS

casename: checkDoors
statePrecond: 1
checkDoors
infoPrecond: 0
leafActions: 5
fixateDoorsButton
reachDoorsButton
pressDoorsButton
fixateDoorsPage
reachDoorsPage

.
.
.
```

Future Work

Some of the work that remains is strictly of a programming nature. For example, a simple user in interface has been created for creating a case base, and monitoring its execution, and more interface work is still required. Some work remains to clean up the software structure and module interfaces. In addition, the current equipment model does not begin to compare to the existing MIDAS model.

CaseMIDAS has also not addressed several issues that Core MIDAS has explored with some success. The first is interruption and resumption of activities. There are several ways in which interruption might be handled within a case based framework. The first method might be called "hard-coded". We determine when an interruption occurs and model it explicitly in the case base. In a case base where there is no variation in the activity durations this is straightforward to do. The cases should simply show the desired sequence of tasks as they should occur if interrupted and resumed. Loads can be modified to appropriately show effort expended in shifting tasks or remembering to resume a previous task.

A second method for implementing interruption could be termed dynamic compilation. During the execution of a case base, if a case needs to be interrupted, the system alters the cases on the fly to conform to the new interrupted sequence of activities. This might be the most cognitively realistic option.

Another method would be most similar to existing activity interruption in Core MIDAS. An interrupted case would be placed in a queue with a pointer to the next activity to be executed when the case is resumed. Multiple cases could be interrupted in this way with no increase in algorithmic complexity. This method would necessitate an ordering of case priority to determine which cases can interrupt which others, and a list of interruptive stimuli.

Another issue which CaseMIDAS needs to address is forgetting, especially in the context of interruption. The CaseMIDAS model should be able to incorporate models of decay or interference based forgetting, similar to Core MIDAS, without conflict with its existing task selection algorithm.

Conclusion

The CaseMIDAS project has explored the use of cases to model human task behavior in a human factors simulation system. This exploration has yielded an easy to use software module for case creation and execution which is integrated with existing MIDAS simulation components.

This effort has yielded a straightforward activity representation for MIDAS. Cases are modular and allow extension of scenarios without the need for global knowledge or manipulation of scenario organization. The representation is simple and does not require programming. Cases are also a natural and intuitive form for describing the groups of procedures or tasks prevalent in cockpit activities.

Clearly much work remains for CaseMIDAS. Several domains need to be implemented in this representation to ensure that it is general enough to have useful coverage. Also valuable would be a head-to-head comparison with the existing activity structure where the ease of use and development time for scenarios in each representation could be compared.

While this effort does not extend the core research in case based reasoning, it does highlight the application of this technique to a novel domain. Most CBR work to date has focused on the goal of better problem solving on a computer inspired by a model of human problem solving. Here the objective is different. This effort examined the application of an Al technique Lo model human performance, not build a better problem solver.

Though much additional work remains to develop a complete and cognitively plausible case based task representation, this effort has prompted a fruitful reexamination of current MIDAS activity representation. In addition, the development of this new task representation has illustrated issues in software complexity and modularity important for the development and continued success of MIDAS.

References

[1] B. Smith and K. Corker (1993), "An Architecture and Model for Cognitive Engineering Simulation Analysis: Application to Advanced Aviation Automation", AIAA-93, San Diego, CA.

[21 C. K. Riesbeck and R. C. Schank, (1989), "Inside Case-Based Reasoning", Lawrence Erlbaum Associates, Hillsdale, NJ.

[31 T. B. Aldrich and S. M. Szabo (1986), "Validation of the LHX One-Crewmember Workload Prediction Model", (Draft Technical Memorandum AS1678-202-86[b]). Fort Rucker, AL: Anacapa. Sciences, Inc.