# A CLIPS-BASED TOOL FOR AIRCRAFT PILOT-VEHICLE INTERFACE DESIGN *

N92-16605

**Thomas D. Fowler**
California State Polytechnic University, San Luis Obispo

**Steven P. Rogers, Ph.D.**
Anacapa Sciences, Inc., Santa Barbara, California

**Abstract**

The Pilot-Vehicle Interface of modern aircraft is the cognitive, sensory, and psychomotor link between the pilot, the avionics modules, and all other systems on board the aircraft. To assist Pilot-Vehicle Interface designers a CLIPS-Based tool has been developed that allows design information to be stored in a table that can be modified by rules representing design knowledge. Developed for the Apple Macintosh® , the tool allows users without any CLIPS programming experience to form simple rules using a point-and-click interface.

## INTRODUCTION

Development of the Pilot-Vehicle Interface (PVI) is one of the most challenging aspects of advanced aircraft programs. The PVI is the cognitive, sensory, and psychomotor link between the pilot, the avionics modules, and all other systems on board the aircraft. To reduce the pilot's workload while maximizing his situation awareness, the PVI must optimize the flow of information between the pilot and the hardware/software systems, respond smoothly to his immediate needs, and present the required information elements using the most readily comprehensible methods.

It is tempting to believe that advancing technology must have made the PVI of modern aircraft better than ever before. In fact, according to many pilots, the opposite is true. Although modern military aircraft have better weapons, better sensors, better computers, and better displays than in years past, the barrage of information from all the cockpit systems is nearly overwhelming, the data from various systems are not integrated to support the pilot's tasks, and the display types and formats are often ill-chosen. To acquire the information necessary for a given task, the pilot may have to extract data from several different systems and mentally translate or manipulate the data, while simultaneously dealing with the unrelenting workload imposed by aviation, navigation, and communication tasks.

The general requirements for an intelligent PVI for an advanced aircraft are that it be capable of filtering out information that is not currently useful, prioritizing the most useful information, prompting the pilot to examine the most important information, and integrating the functionally related data into an easily comprehended presentation. In order to determine the best presentation, the PVI must rapidly perform a layered series of

information presentation choices to select the best display modality, location, and format for the pilot's current activity.

In order to meet these general requirements for the PVI, it is necessary to base the system design on the pilot's functions and information requirements, rather than focusing on the technology of the aircraft systems. Taking such an information requirements approach, however, requires that thorough, systematic analyses be performed to identify the specific information elements necessary to perform each pilot function and to determine the various attributes of these elements. Next a set of rules must be developed to select the best display modalities, locations, and formats for integrating and presenting the information given the specific information elements and their attributes.

The necessity of a database of information elements, and rules linking the information elements to optimal presentation methods, strongly suggested that an expert system be developed to manage and guide the display method selection process. Such an expert system must be designed to organize the extensive collections of analytical information, document the controlled growth of the rules for information display, and manage the application of the rules to the database for selection of display methods in a methodical, consistent, and easily understood manner.

To assist in the development of such an expert system, a CLIPS-Based tool has been developed that allows the information elements and their attributes to be stored in a table that can be modified by rules representing design knowledge. The tool allows the user to create rules incrementally using a simple point-and-click interface. The following sections describe the tool from the user's perspective and discuss the use of CLIPS (NASA 1991) on the Macintosh for its implementation.

## USER LEVEL VIEW

The tool presents the user with a table contained in a window that has horizontal and vertical scrolling capabilities. Rows and columns can be manipulated using cut, copy, and paste commands available from the menu bar. Entries in the table can be selected by clicking on them with the mouse or by using the arrow keys on the keyboard.

| Info_Type | Info_Element | X | Y | Modality | Location | Format |
|---|---|---|---|---|---|---|
| Flight_Plan | Time_Estimates | Low | Red | | | |
| Flight_Plan | Altitudes | High | Green | | | |
| Weather | Wind_Speed | Medium | Blue | | | |
| Weather | Visibility | High | Blue | | | |
| Aircraft | Fuel_Remaining | High | Red | | | |
| Aircraft | Oil_Pressure | Low | Red | | | |

**Figure 1.** Information Element Table

The first row in the table is used to identify the type, name, and attributes of each information element. Figure 1 shows a table consisting of six information elements each having the attributes X, Y, Modality, Location, and Format.

Columns entries that are set by the user are tagged as "input" columns, while those set as the result of rule execution are designated "output" columns. In figure 1, columns X and Y are input columns with entries that the user is responsible for setting. The Modality, Location, and Format columns are output columns. Associated with each column is a list of value options from which each entry is selected. Figure 2 shows the input of a list of value options for column X and its designation as an input column.
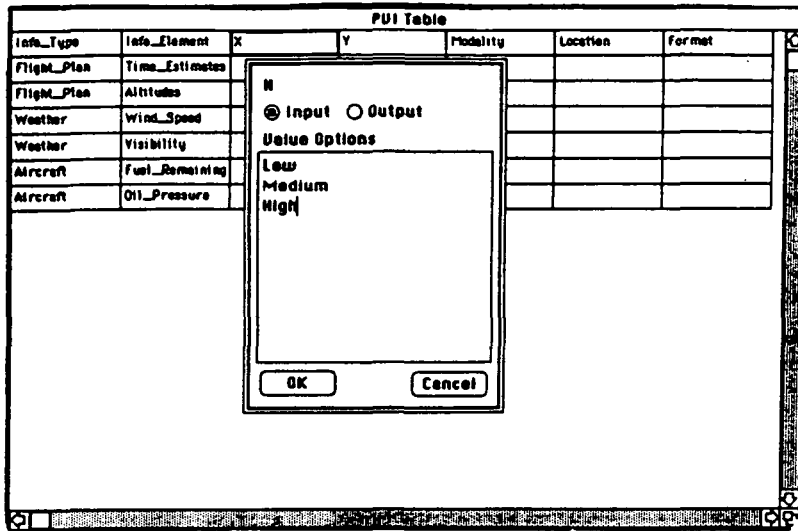


**Figure 2.** Column settings dialog

Once the list of value options has been set for a column, the dialog shown in Figure 3 is used to set the values of its entries. In this case, the value of the attribute X for the Wind_Speed information element is being set to Medium. The information type and attribute for the entry is indicated at the top of the dialog. If an entry already contains a value, then it is displayed in the box below the type and attribute indicators.
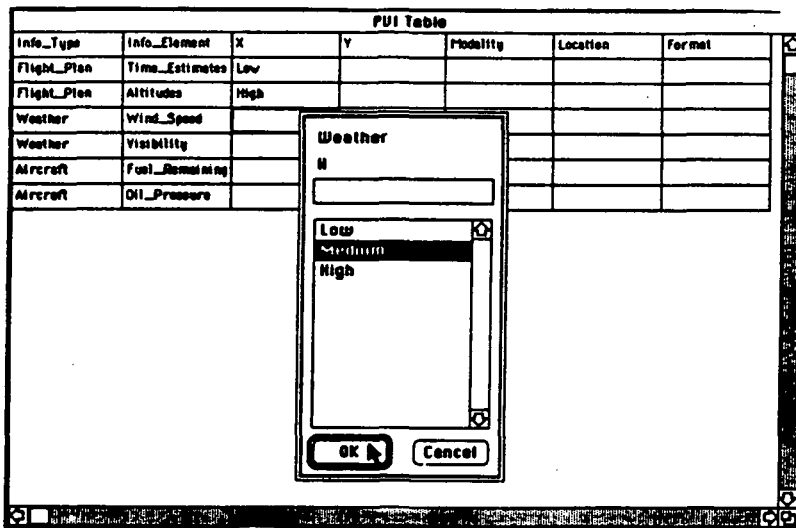


**Figure 3.** Attribute entry dialog

A list of value options must also specified for the Modality, Location, and Format columns for use by the rules which set them. Restricting the values of entries in this manner guarantees correctness of input and insures the consistent usage of symbols by user-created rules.

409

## Adding Rules To The Table

Rules can be formed that have left-hand side (LHS) patterns that match values in input and output columns, and right-hand side (RHS) clauses that set values in output columns. Rule creation is facilitated by the dialog shown in Figure 4(a). Here, a rule with name "Rule1" has been formed that will apply to information elements that have attribute X set to Low and attribute Y set to Red. The consequence of Rule1's execution is to set the attributes Modality, Location, and Format to Visual, Display1, and Iconic, respectively. Figures 4(b) and 4(c) show the formation of additional rules.



**Figure 4(a).** Rule1
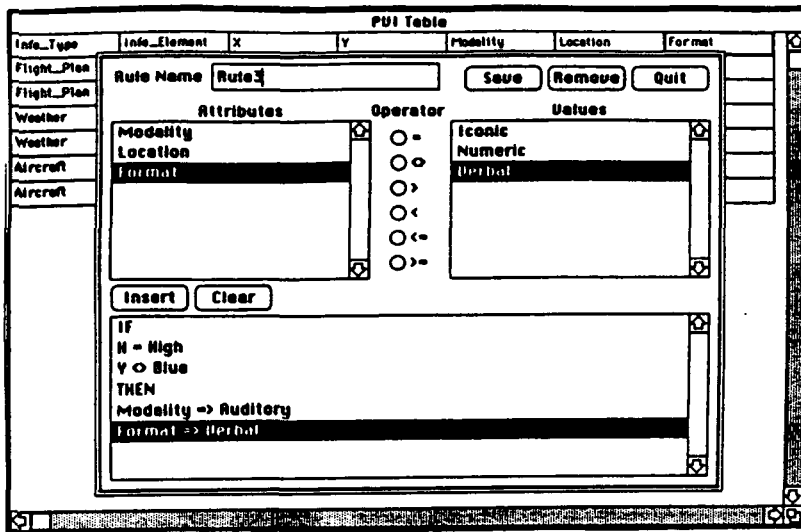


**Figure 4(b).** Rule2

410

**Figure 4(c).** Rule3

**Figures 4(a)-(c).** Rule input dialog

LHS rule patterns are composed of an attribute, an operator, and a value. The list of attributes, displayed in the Attributes list-box, consists of the names of all input and output columns from the first row of the table. Displayed in the Values list-box is the list of value options that correspond to the selected attribute. LHS side patterns are formed by selecting an attribute, an operator (via the Operator radio buttons), and a value. The resulting pattern is displayed at the selected line in the rule display.

RHS rule clauses are composed of an attribute, the assertion operator (=>) and a value. When a RHS clause is being formed, only the names of output columns appear in the Attributes list-box and the Operator radio buttons are not applicable.

New LHS patterns and RHS clauses can be inserted into the rule by pressing the Insert button and deleted by pressing the Clear button. The Save button allows newly created rules to be saved while the Remove button deletes pre-existing ones.

**Rule Execution**

A single rule can fire multiple times to set attribute values for different information elements. However, rules are applied to the table in a row-wise manner. That is, the LHS and RHS parts of rule can only apply to one row at a time. Figure 5(a) shows the state of table after Rule1 [Figure 4(a)] has executed. Rule1's LHS patterns are satisfied by the values of attributes X and Y for information elements Time_Estimates and Oil_Pressure. Rule1 will fire once for each of these information elements, resulting in their Modality, Location, and Format attributes being set to Visual, Display1, and Iconic, respectively. Rule2 [Figure 4(b)] includes a LHS pattern that matches if "X >= Medium." This is permissible because an order relationship is established on the list of value options. Thus, the relationship (Low < Medium < High) holds for the value options for the attribute X.

Figure 5(b) shows the state of the table after the Rule2 and Rule3 have executed. Notice that because Rule3's RHS does not set the Location attribute, it has not been set for the information elements Altitudes and Fuel_Remaining.

PUI Table

| Info_Type | Info_Element | X | Y | Modality | Location | Format |
|---|---|---|---|---|---|---|
| Flight_Plan | Time_Estimates | Low | Red | Visual | Display1 | Iconic |
| Flight_Plan | Altitudes | High | Green | | | |
| Weather | Wind_Speed | Medium | Blue | | | |
| Weather | Visibility | High | Blue | | | |
| Aircraft | Fuel_Remaining | High | Red | | | |
| Aircraft | Oil_Pressure | Low | Red | Visual | Display1 | Iconic |

**Figure 5(a).** Table after Rule1 executes

PUI Table

| Info_Type | Info_Element | X | Y | Modality | Location | Format |
|---|---|---|---|---|---|---|
| Flight_Plan | Time_Estimates | Low | Red | Visual | Display1 | Iconic |
| Flight_Plan | Altitudes | High | Green | Auditory | | Verbal |
| Weather | Wind_Speed | Medium | Blue | Visual | Display2 | Numeric |
| Weather | Visibility | High | Blue | Visual | Display2 | Numeric |
| Aircraft | Fuel_Remaining | High | Red | Auditory | | Verbal |
| Aircraft | Oil_Pressure | Low | Red | Visual | Display1 | Iconic |

**Figure 5(b).** Table after Rule2 executes

**Figures 5(a) and (b).** Effects of rule execution

Once a rule has been entered it will apply to any new information elements that are added to the table. Figure 6 shows the addition of the information element Planned_Speeds in the third row of the table and the setting of its Y attribute to Red. After the Y attribute has been set to Red, Rule3 [Figure 4(c)] will execute and the Modality, and Format attributes will be set accordingly (Figure 7).

The effects of modifying an existing rule are immediately reflected in the state of the table. Figure 8 shows Rule2 modified so that it sets the attribute Location to Display3 instead of Display2. After the modified Rule2 has executed, the Location attribute for information elements Wind_Speed and Visibility will be set to Display3 (Figure 9).

412

**PUI Table**

| Info_Type | Info_Element | X | Y | Modality | Location | Format |
|---|---|---|---|---|---|---|
| Flight_Plan | Time_Estimates | Low | Red | Visual | Display1 | Iconic |
| Flight_Plan | Planned_Speeds | High | | | | |
| Flight_Plan | Altitudes | High | | | | Verbal |
| Weather | Wind_Speed | Medium | | | Display2 | Numeric |
| Weather | Visibility | High | | | Display2 | Numeric |
| Aircraft | Fuel_Remaining | High | | | | Verbal |
| Aircraft | Oil_Pressure | Low | | | Display1 | Iconic |

flight_Plan
Y

Red
Green
Blue

OK    Cancel

**Figure 6.** Addition of new information element

**PUI Table**

| Info_Type | Info_Element | X | Y | Modality | Location | Format |
|---|---|---|---|---|---|---|
| Flight_Plan | Time_Estimates | Low | Red | Visual | Display1 | Iconic |
| Flight_Plan | Planned_Speeds | High | Red | Auditory | | Verbal |
| Flight_Plan | Altitudes | High | Green | Auditory | | Verbal |
| Weather | Wind_Speed | Medium | Blue | Visual | Display2 | Numeric |
| Weather | Visibility | High | Blue | Visual | Display2 | Numeric |
| Aircraft | Fuel_Remaining | High | Red | Auditory | | Verbal |
| Aircraft | Oil_Pressure | Low | Red | Visual | Display1 | Iconic |

**Figure 7.** Table after addition of a new information element

## Conflicting Rules

A rule conflict is defined as two or more rules attempting to set the same table entry. A conflict situation arises when the PVI designer has entered two or more rules that apply to the same output attribute(s) of an information element. A more subtle conflict situation presents itself when a change in the value of a table entry brings two or more previously non-conflicting rules into conflict.

The tool allows conflicting rules to be entered. However, when multiple rules attempt to set the value of the same table entry, the symbol "CONFLICT" is placed into the entry. The user can then request that a list of all rules that caused the conflict be generated and modify the appropriate rule(s) to resolve the conflict. The resolution of conflicting rules is an important feature of the tool that insures the consistency of the design knowledge.
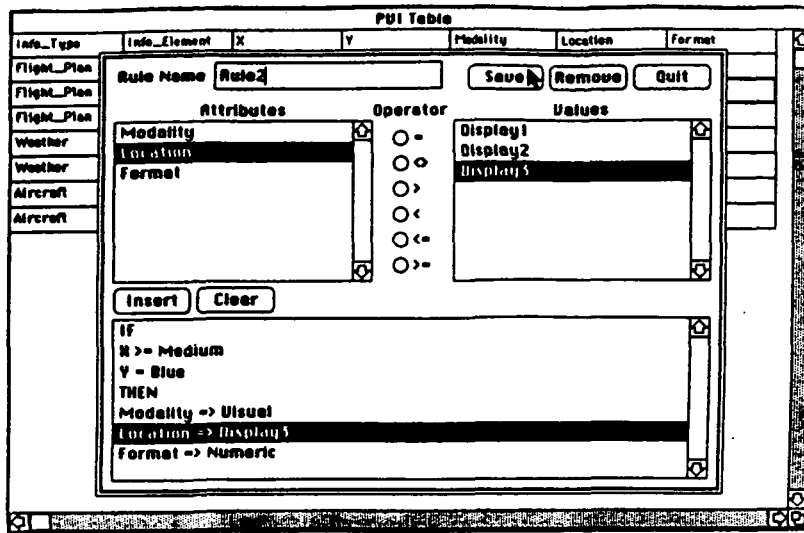
**Figure 8.** Modification of Rule2



**Figure 9.** Table after modified Rule2 has executed

## Column and Value Option Deletion

The deletion of a column or one of its value options is problematic in that rules that reference the column or value will no longer apply. For example, if the column representing attribute X were deleted from the table, then any rules that reference X will no longer fire. Similarly, if a value option for a column is deleted, any rules referencing that value will no longer execute.

There are several alternatives for handling this situation. One is to remove all rules that make reference to deleted columns or values. This has the dangerous consequence that a large number rules could potentially be removed due to a single operation by the user. A second alternative is to modify the rule set by removing all clauses that make reference to the deleted column or value option. While less severe than the first alternative, a large scale modification of the rule set could occur. A third alternative is to simply alert the user that a number of rules have been invalidated. A query can then be performed over the rule set to locate all invalidated rules. The user then has the option of modifying or deleting these rules as appropriate.

Since Invalidated rules are no longer applicable they are "unfired" by clearing all table entries that were set as a consequence of their execution. When a value option is

414

C-3

removed from a column, all entries in the column with that value are cleared. The clearing of table entries in this manner can result in the LHS patterns of previously executed rules to longer be satisfied. These rules must also be "unfired" to maintain the consistency of the information in the table. The resolution of conflicts, as described above, has not yet been fully implemented. Currently, The truth-maintenance facility of CLIPS Version 5.0 is being considered for use.
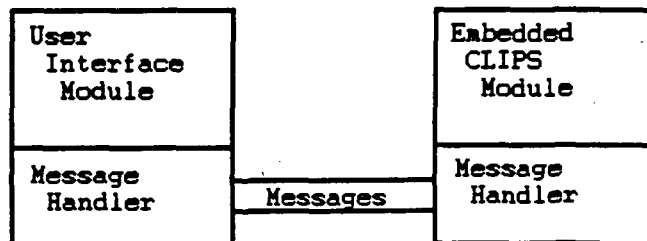
## IMPLEMENTATION



**Figure 10.** Tool Architecture

## Architecture

The tool is comprised of three main components; a user-interface module, a message handling facility and an embedded CLIPS module (Figure 10). The user interface module is responsible for handling all events generated by the user and operating system. The embedded CLIPS component is responsible for maintaining the table and rule-base. The message handling component facilitates communication between the user-interface and CLIPS modules. Whenever the table is modified by the user, a message is sent to CLIPS by the user-interface. CLIPS handles this message by updating its fact-list and executing any rules that have become activated. If rules that result in the modification of the table are executed, then the CLIPS module sends a message to the user-interface module - which responds by updating the display accordingly. When the user creates rules, the user-interface module sends a message to CLIPS, which responds by adding the rule to its environment using the CLIPS Build function.

## Developing With CLIPS On The Macintosh

Programming the Macintosh family of computers requires a substantial investment of time to become familiar with the toolbox routines implemented in ROM. Indispensable to any serious development effort for the Macintosh is the *Inside Macintosh* series (Apple 1985). Knowledge of the basics of Pascal records, procedures and functions is helpful because the toolbox interface is defined in Pascal. Apple's *Programmer's Introduction to the Macintosh Family* (Apple 1988) is a good source of information for those considering a developing for the Macintosh.

Symantec's THINK C compiler was chosen to implement this tool. Since CLIPS for the Macintosh was developed using THINK C, the task of embedding it into an application is made easier by using THINK C. In addition, THINK C comes with a class library that provides an interface to the Macintosh toolbox that can release the programmer from being concerned with many low-level toolbox details. Because Symantec's documentation for the THINK C Class Library (Symantec 1989) is extremely

terse and includes few coding examples, Dave Mark's *Macintosh C Programming Primer* series (Mark and Reed 1989, Mark 1990) is strongly recommended.

## CONCLUSIONS AND FUTURE DIRECTIONS

The problem of transferring human knowledge into an expert system is known as the knowledge acquisition bottleneck (Giarratano 1989). This tool helps reduce this bottleneck by facilitating the incremental acquisition of design knowledge from its users. It accomplishes this by allowing users without formal programming experience to easily create rules while assessing their effects. Using CLIPS to represent these rules provides a measure of power that is not attainable using other tools such as a spreadsheet. CLIPS' use of the Rete Algorithm (Forgy 1979) results in the execution of only those rules affected by incremental changes to the table. The pattern matching capabilities of CLIPS allows rules to be formed that reference table entries via user-defined symbols rather than positional row/column addresses imposed by the implementation of the table. This allows rows and columns to be added or rearranged without having to modify any rules.

Future directions for the tool include allowing the formation of more sophisticated rules, incorporating meta-rules that can be used to evaluate and examine the design rules entered by the user, and providing a rule explanation facility.

## REFERENCES

Apple Computer, Inc. *Inside Macintosh*. 6 vols. Addison-Wesley, 1985 - 1991.

Apple Computer, Inc. *Programmer's Introduction to the Macintosh Family*. Addison-Wesley, 1988.

Forgy, Charles L. *On the Efficient Implementation of Production Systems*. Ph.D. Thesis, Carnegie-Mellon University, 1979.

Giarratano, Joseph, Gary Riley. *Expert Systems Principles and Programming*. Boston: PWS-KENT, 1989.

Mark, Dave, Cartwright Reed. *Macintosh C Programming Primer Volume I*. Addison-Wesley, 1989.

Mark, Dave. *Macintosh C Programming Primer Volume II*. Addison-Wesley, 1990.

NASA. *CLIPS Reference Manual (Version 5.0)*. Software Technology Branch, Lyndon B. Johnson Space Center, 1991.

Symantec. *THINK C User's Manual*. Symantec, 1989.