

Improving the Aircraft Cockpit User-interface: Using Rule-based Expert System Models

By Lance Sherry and Mike Feary

In a real-time embedded control system, such as an aircraft flight-control, an operating room device, or a power plant command and control center, the ease-of-use of the user-interface becomes critical to determining their commercial success. A well-designed user-interface accepts operator instructions without requiring excessive manipulation of the user-interface. A well-designed user-interface also presents the operator with information to aid decision-making.

This article demonstrates how rule-based expert system models of control system behavior aid the designers in building superior user-interfaces.

Complex User-interfaces Increase Operator's Cognitive Workload

Complex control automation pervades our lives with appliances ranging from the personal (e.g. digital watch) to the most complex (e.g. aircraft flight control system). These systems, constructed using digital microprocessors, perform increasingly more complicated tasks. To match this growth in complexity, the functionality of their user-interfaces must increase. Figure 1 illustrates the primary user-interface for pilot interaction with the flight control system on the Boeing 747 aircraft. There are four knobs for setting targets (shown in the windows) and 13 buttons. Some buttons affect the aircraft's ability to meet the objective (white connecting lines) while others do not.

Although the introduction of complex control automation has generally decreased the operator's *physical motions*, studies show



Figure 1: User-interface for Boeing 747 flight control system.

that the operator's *cognitive workload* often increases. Instead of controlling the vehicle or plant directly through physical action, operators of these control systems supervise sophisticated suites of automation, performing memorized action sequence to "program" the automation in advance and then monitor its behavior.

Researchers, studying operators of these systems have found that during high tempo (e.g. abnormal and emergency) situations, in which the operator's cognition is already taxed to its maximum levels, the additional

burden of programming a complex user-interface may exceed the limits of human performance (Figure 2).

Don't Make Me Think About the User-Interface

Designers recognize that the traditional approach of designing control systems to automate the tasks formerly performed by the operator turns operators into system monitors. Monitoring, over an extended period, is something at which humans inherently perform poorly.

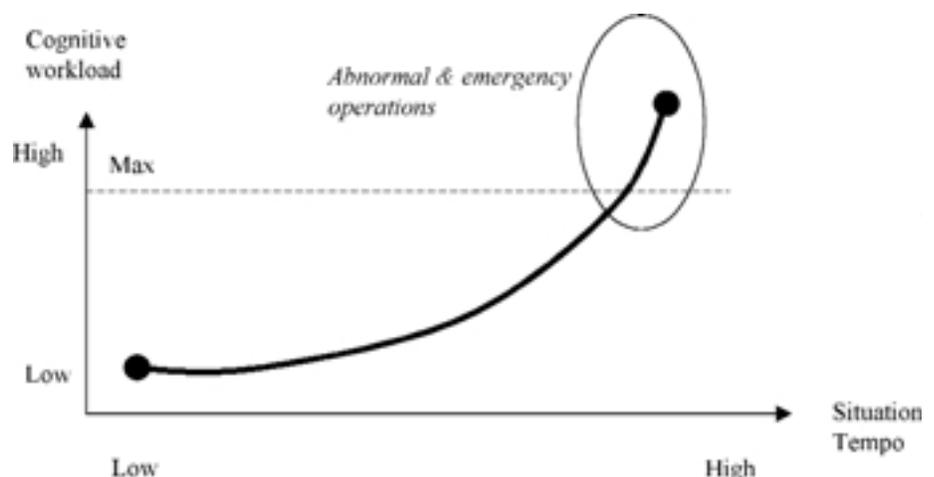


Figure 2: Cognitive workload increases above the limits of human performance while operating complex user-interfaces during abnormal or emergency situations.

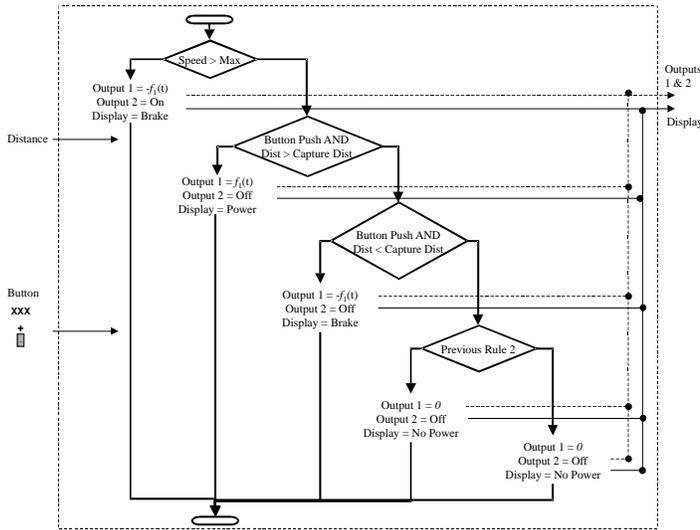


Figure 3: Determine example automation Behavior by the functions generating values on the outputs. Determine these functions by the combination of input conditions (including input device selection by the operator).

Instead of automating tasks normally performed by the operator, designers are now building automation that aids the operator in performing their task by providing information on the future trajectory of the vehicle or plant parameters, and by offering mode options and optimum settings for control. By design, these systems *make the operator smarter*; instead of simply building smarter and more complex automation.

There are two fundamental design principles for building automation that makes the operator smarter²:

Provide input devices for each of the operator's task goals: The user-interface allows the operator to issue commands to the automation that directly reflect the operator's task goals. It does not force the operator to manipulate input devices in long sequences of steps to achieve a task goal.

Provide feedback display devices that acknowledge operator's inputs and visualize the effects of these inputs: The user-interface acknowledges receipt of the operator's inputs and presents pertinent information enabling visualization of the instructions short-term and long-term effects. In many cases, the user-interface also presents the operator with additional information such as safe operating regions and optimal settings.

Indications that these characteristics have been satisfied include: (1) a very small user manual, and (2) operators consistently using the automation in abnormal or emergency situations. The user-interface allows the operator to concentrate on controlling the vehicle or plant (and not manipulating the user-interface).

Designing User-interfaces Using Rule-based Expert System Models of the Automation

Rule-based expert system models, as described in this article, have emerged as a powerful tool for designing easy-to-use user-interfaces for complex control systems. These models supply an analytic technique to ensure that input

devices meet operator's goals and that feedback displays announce the automated behavior.

Example Automation

Figure 3 illustrates an example automation device with one input device invoking the behaviors commanded by the automation. The system also exhibits a feedback display that announces or illustrates the behavior commanded by the automation. The system has two inputs (speed, distance) and one actuator command (output). The flowchart represents the real-time software algorithm.

A rule-based model of the automation is included in Table 1. The left hand-side of the table defines the combinations of the input conditions, including input device selection by the operator. The table's right hand-side defines the combinations of functions used to generate values on the output commands. The right hand-side also includes the content of the feedback display.

Input Conditions				Rule Number	Output Command	Output Command	Display
Speed	Distance	Button	Past Rule	(Behavior Description)	Motor	Brake	
Unsafe region				1 (Brake Protect Vehicle*)	$-f_i(t)$	On	Brake
Safe region	>Capture Region	Push		2 (Hold Speed to Target)	$f_i(t)$	Off	Power
Safe region	\leq Capture Region	Push		3 (Brake to Capture Target)	$-f_i(t)$	Off	Brake
Safe region	\leq Capture Region		2	4 (Decel to Capture Target)	0	Off	No Power
Safe region	No action			5 (Default on power-up)	0	Off	No Power

* Abnormal or emergency situation

Table 1: Rule-based model of the automation behavior. The rows on the left hand-side define the possible combinations of the input conditions. The rows on the right hand-side define the combinations of output commands and displays for each combination of input conditions.

Each row defines a unique combination of input conditions and associated output commands and display. The middle column numbers each rule and describes the automation behavior.

Rule 1 — an input (speed) exceeds a safe threshold. The automation protects the vehicle-under-control by commanding a return to a safe operating region.

Rules 2 and 3 — the pilot selects the input device to instruct the automation to perform a specific maneuver. Rule 2 handles pushing the button in normal condition. Rule 3 handles pushing it in an abnormal condition.

Rule 4 — an automatic maneuver performed by the automation as an intended consequence of the user selecting the button. This condition occurs automatically following Rule 2 activation.

Rule 5 — the default idle condition (e.g. on power-up).

User-interface Design Using the Rule-based Model

A significant amount of information on the utility of the automation user-interface is available from the rule-based expert system model used in the design of practical user-interfaces.

Design Principle 1: *Automation should support all the operator's task goals*

The automation should be designed to perform each of the goals required by the operator's task. The rule-based model defines a complete list of behaviors performed by the automation (middle column) and can be used to ensure that all the operator's task goals are supported. As simple as this design rules appears, one does not need to look far for automation examples requiring significant manipulation of the user-interface to achieve operator's goals. For example, the cordless telephone that requires the removal and re-insertion of the power chord after a power interruption

does not provide the operator the means to “reset” the automation (as do PC’s with Alt-Cntl-Delete).

Design Principle 2: *Automation should respond to all possible combinations of input conditions.*

Notice that the rule-based model defines rules for all possible combinations of input conditions. Combinations not in the model identify situations in which the automation will not respond. Missing situations are common in very large embedded control systems with coupled components developed by different design teams.

Design Principle 3: *Automation should respond in one way for each combination of input conditions.*

In a similar vein, combinations of input conditions that appear more than once in the model identify more than one behavior assigned in response to a situation. Duplicate input combinations, common in very large embedded control systems with coupled components developed by different design teams, must be resolved.

Design Principle 4: *Design automation to present the operator with clearly labeled input devices for each task goal.*

The rule-based model is also critical to defining the input devices for each operator task goal. The rule-based model, documents the agreement between the operator and designer of how the operator will invoke specific commanded behavior.

Mislabeled or poorly labeled input devices are an obvious source of confusion to the operator. For example, the cell phone that is turned on by hitting the button labeled “NO.”

A well design user-interface allows the operator to translate their task goals directly into actions on the input devices. Failure to label the input devices with operationally meaningful labels that reflect the operator’s goals is a significant contributor to the need for training.

The behavior description (middle column) in the rule-based model serves as an excellent source of operationally meaningful labels. For example, an aircraft pilot’s task goal of *hold an altitude* could be reflected on the user-interface by an input device labeled hold (see Figure 1)

Design Principle 5: *Automation input devices should result in one, and only one behavior.*

The astute reader studying the example rule-based model will notice two combinations of input conditions that use the same input device (rule 2, and rule 3). The operator’s selection of this input device commands one behavior for rule 2 and a

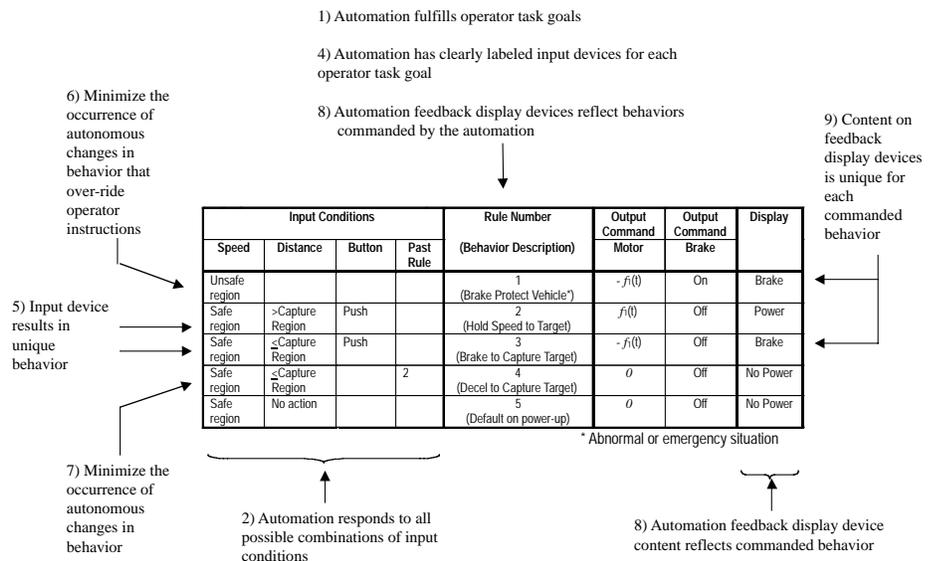


Table 2: Summary of design principles reflected in the rule-based model of the automation.

different behavior for rule 3 (See Table 2).

An input device that commands different behaviors depending on the situation is functionally overloaded. In most cases, the behavior input device either has an operationally meaningful or incorrect label. This characteristic drives up training costs and is a potential weak link in operation of the automation. Despite training, an operator may use the input device inappropriately in the heat-of-the-moment.

Design Principle 6: *Automation should minimize automatic changes in behavior that over-ride the operators instructions.*

Rule 1 in the example automation illustrates a behavior that is not the result of an operator’s manipulation of an input device. This behavior, commanded by the automation, is invoked autonomously by the automation when the input (speed) value leaves the safe operating region.

This type of behavior, known as an *autonomous mode change*, legitimately appear in the rule-based model when the automation is required to override the operator’s instructions to preserve the safety of the device. For example sudden failures in equipment or rapid changes in the environment that operators cannot detect (e.g. windshear for an aircraft) demand immediate response from the automation.

Studies of eye fixation of operators demonstrate that operators frequently do not notice these autonomous changes in commanded behavior⁴. For this reason autonomous-mode changes should be avoided where possible. Alternatively, the designer should consider an additional alerting output device such as an aural warning or a flashing display to capture the operator’s attention. Keep the number of

autonomous changes (and alerts) small since humans are very good at ignoring annoying sounds and flashing displays when they occur too frequently.

Design Principle 7: *Automation should minimize the automatic changes in behavior over time that are implemented in the automation to automate long-sequences of commanded behaviors.*

Another form of autonomous change in commanded behavior occurs in the example automation in Rule 4 (see Table 2). This type of change in commanded behavior automatically executes a sequence of tasks that otherwise would be made by the operator. They are invoked by the operator through the selection of an input device that engages a long sequences of commanded behaviors over an extended period.

These input devices are known as *mega-modes*. There are several problems with mega-modes. First, similar to autonomous mode changes, operators frequently do not notice subtle changes in control strategies. Secondly, mega-modes turn operators into monitors. They become complacent or confused and “get behind” the automation. In addition, as the environment evolves rapidly, it is easy for the automation to rapidly command behaviors that “paint the operator into a corner” that even the operator cannot escape.

Experience has taught us that it is imperative for safety and efficient operations to team the operator and the automation and allow them to operate synchronously to perform the task. When these mega-mode autonomous changes appear in the rule-based model, they should be scrutinized very carefully and possibly eliminated.

Design Principle 8: *Automation should present the operator with clearly labeled content on the display feedback device to reflect the behavior commanded by the automation.*

A critical element of the design is the content of the feedback display devices. The feedback display devices should reflect the

automation's commanded behavior to fulfill the task goal. The behavior description (middle column) in the rule-based model serves as an excellent source of operationally meaningful content for the displays. For example, an operator task goal to *hold an altitude* commanded by an action on the

input device labeled hold (see Figure 1), should have an accompanying feedback display annunciating *hold altitude*.

Despite the obvious nature of this design principle, it is easy to find automation examples where the content of the display feedback has to be interpreted by the operator to infer the behavior commanded by the automation. This type of inference requires that the operator memorize and recall information during operation. This type of memorized knowledge is error-prone and subject to neglect when things get busy.

Design Principle 9: *Content on feedback display devices should reflect one, and only one, behavior commanded by the automation.*

The true behavior, commanded by the automation via the values of the output commands, is not reflected in the display feedback device in Table 1. In fact, the display feedback device only reflects the behavior commanded by Output Command 1 and does not take into account the combinatorics of Output Command 1 and 2.

In this example, rules 1 and 3 share the same display feedback ("Brake"), despite the different output commands. This display is *functionally overloaded*. Operators learning the behavior of the automation by observation are likely to build inaccurate mental models of how the automation works. This has implications on how they use the automation — especially in abnormal situations.

The solution is including unique feedback displays for each combination of output commands. Again, the behavior descriptions (middle column) can be used as the content for these displays. An experiment conducted by NASA showed that pilots flying a modern airliner with unique displays for commanded behavior outperformed their peers with functionally overloaded displays.

Conclusion - Turning Intuitions into Design Habits

A good user-interface design starts and ends with a definition of the operator's task goals. The

Version 3.0 Coming Soon **DISCIPULUS™** **Fast Genetic Programming Based on AIMLearning™ Technology**

Discipulus is *extremely* fast software for modeling and mining engineering and scientific data. It runs 60-200 times faster than competitive technologies; so it is the first software that makes Genetic Programming practical.

On the market since October 1998, Discipulus has been deployed by numerous Fortune 500 companies for a wide range of applications, including:

- Data Mining for scientists and engineers;
- Embedded systems program development;
- Process control; and
- Chip design and testing.

"Discipulus is an outstanding product. It recently solved an extremely difficult process control problem for us that had stumped our best neural network researchers"

Larry Deschaine, PE. Science Applications International Corp.

"We are very pleased with this product. It's fast and the user interface is good. In the past year, Discipulus has been the key to fast deployment of several important, in-house modeling solutions."

Emerging Technologies Director, Fortune 500 Company

"Until you have access to tools this fast, you really cannot appreciate how much of a difference it makes."

EvoNews, Winter 1999.

www.aimlearning.com
720-981-8710
"Speed Matters. . ."™

rule-based expert system model is a powerful way to capture the operator's task goals and ensure the design of a complete and consistent user-interface.

With hindsight, it is hard to imagine how user-interfaces are designed that violate the nine design principles defined above. It has been our experiences that, for the most part, these less than optimum designs are not created intentionally. Instead, they emerge through long development and upgrade cycles with design teams staffed with changing personnel.

To combat this emergent phenomenon we found rule-based expert system models are useful "book-keeping" design tool that keeps us honest. Although building these tables for large complex systems is time-consuming and difficult, our experience designing user-interfaces for aircraft cockpits is that it is well worth the effortⁱⁱⁱ. One interesting phenomenon that we have noticed is that designers working with these rule-based models quickly build intuitions about what will result in good and bad user-interfaces designs without having to construct the rule-based model. In this way constructing the rule-based model is a powerful learning mechanism that converts designers with intuitions about good user-interfaces, to designers with good user-interface design habits.



- 1 Lance Sherry is a researcher at Honeywell Aerospace Electronic Systems in Phoenix, Arizona. He can be reached at lance.sherry@honeywell.com
- 2 Mike Feary is a researcher in Human Factors at NASA Ames Research Center. He can be reached at mfeary@mail.arc.nasa.gov
- 3 Norman, Don (1988) The Design of Everyday Things. Doubleday: NY. NY.
 - i Mumaw, R., N. Sarter, C. Wickens, S. Kimball, M. Nikolic, R. Marsh, W. Xu, and X. Xu. (2000) Analysis of pilots' monitoring and performance on highly automated flight decks. Boeing Commercial Airplane, Seattle, WA.
 - ii Feary, M., D. McCrobie, M. Alkin, L. Sherry, P. Polson, and E. Palmer (1998) Aiding Vertical Guidance Understanding NASA/TM-1998-112217
 - iii Sherry, L., M. Feary, P. Polson, R. Mumaw, and E. Palmer (2001) A cognitive engineering analysis of the flight management system vertical navigation function. Human Factors and Aerospace Safety 1(3), 223-245..

Content Analysis - Information Extraction - Email Response - Categorization
eCRM - Filtering - Knowledge Management - Summarization - Text Mining
e-Intelligence - Natural Language Query - Natural Language Understanding

NEW

VisualText™

Brings deep text analysis capability to these applications, and more...

An innovative Integrated Development Environment for building deep text analyzers that integrate with your broader applications

Features

- Integrated GUI
- Many development accelerator tools
- NLP++™ - a C++ like language especially for deep NLP programming
- Knowledge base management system
- Automatic rule generation
- Runtime API

Benefits

- Deeper text analysis capabilities
- Rapid analyzer development
- Reduced programming resources
- Quicker time to market
- Fast processing with compiled mode
- Extensible, maintainable text analyzers
- Ready for integration with applications

Text Analysis International, Inc.

www.textanalysis.com

1669-2 Hollenbeck Ave., #501, Sunnyvale, CA 94087
Email: info@textanalysis.com

Voice: (650) 417-2059 x2452
FAX: (650) 417-2059 x2452

CIRCLE 21 ON READER SERVICE CARD