NASA Contractor Report 4336

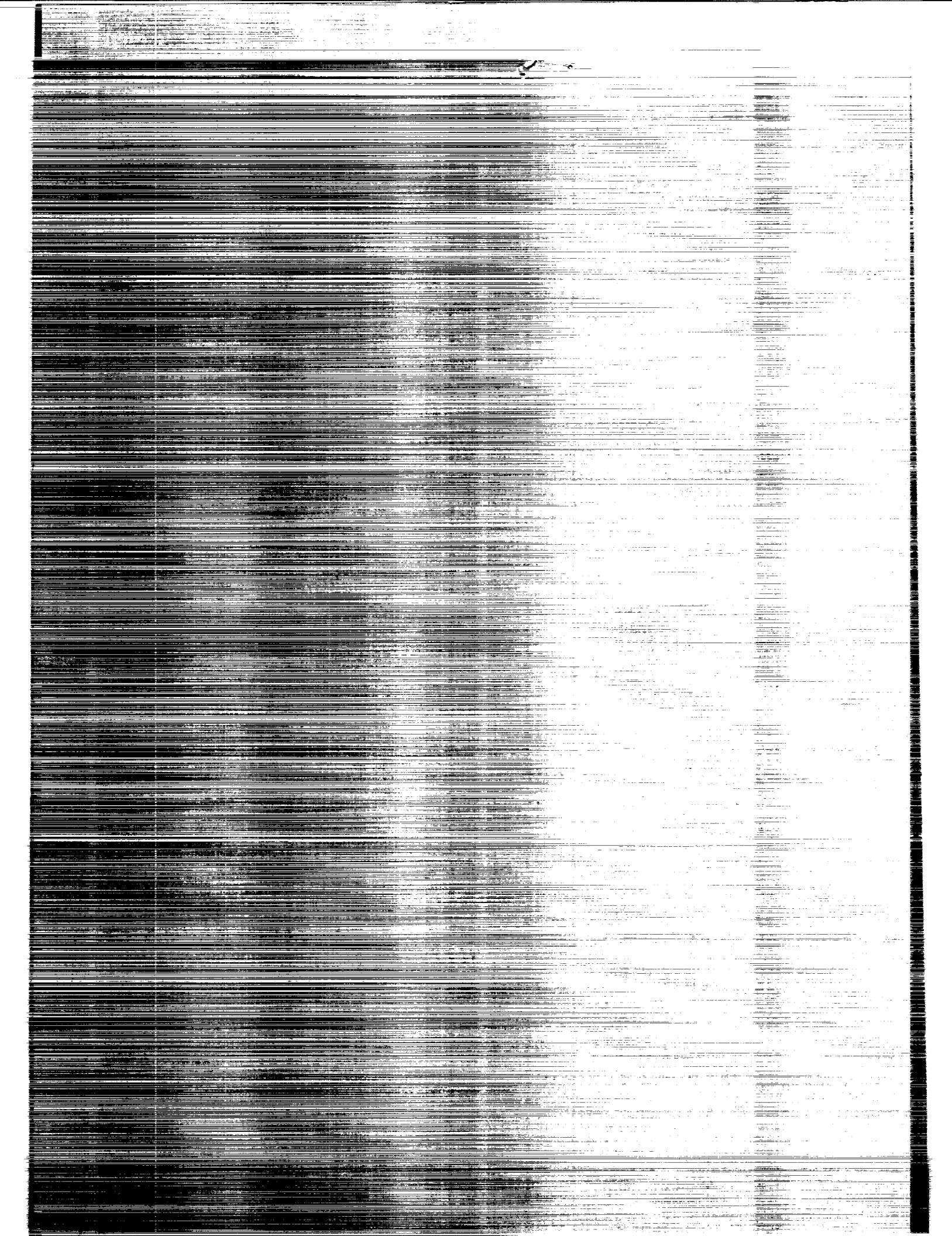
# Assessing the Impact of Modeling Limits on Intelligent Systems

William B. Rouse and John M. Hammer

CONTRACT NAS1-19021
NOVEMBER 1990



NASA



NASA Contractor Report 4336

# Assessing the Impact of Modeling Limits on Intelligent Systems

William B. Rouse and John M. Hammer
*Search Technology, Inc.*
*Norcross, Georgia*

Prepared for
Langley Research Center
under Contract NAS1-19021

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division

**1990**

# SUMMARY

This report is concerned with validating the knowledge bases underlying intelligent systems. In particular, the concern is with identifying and remediating modeling limits that are likely to lead to unacceptable consequences in terms of system performance or safety. The first part of this report provides a general conceptual framework for considering the roles in intelligent systems of models of physical, behavioral, and operational phenomena. Ten types of modeling limit are identified and their likely consequences discussed. The second part of this report describes a methodology for identifying limits in particular intelligent systems, and illustrates the use of the methodology via an experimental evaluation of the pilot-vehicle interface within the Pilot's Associate. The essence of this methodology involves interactive probing and tracing of concept graph representations of knowledge bases. The insights provided by the results of this experiment demonstrate the utility of this methodology. The third and final part of this report outlines the requirements and functionality for a computer-based knowledge engineering environment which would embody the approach advocated and illustrated in earlier discussions. Issues considered include the specific benefits of this functionality, the potential breadth of applicability, and technical feasibility.

## INTRODUCTION

Models of physical, behavioral, and operational processes often form the basis for developing intelligent systems. These models are embedded in software systems in terms of rules, networks, equations, and algorithms. The goal is to use these models as a means to automate, offer expert advice, and perhaps provide explanations. In addition, this functionality must be provided without compromising software safety.

There are a variety of methods for developing models. They may be derived from first principles, identified from data, or compiled via knowledge engineering. Regardless of the method employed, the resulting models inevitably cannot capture all of the phenomena of interest. This is true for all behavioral, physical, and operational phenomena -- the limits of modeling are fundamental (Casti, 1989; Glymour, et al., 1987; Rouse, 1980; Rouse & Hunt, 1982; Rouse & Morris, 1986; Rouse, et al., 1989). In this report, the nature and implications of these limits are considered for these three classes of phenomena.

There are several types of limit. The data sample, upon which a model is based, is always finite. The variables chosen to capture a phenomenon may be incomplete or incorrect. Structural assumptions may be inadequate or inappropriate. Parameter estimates may be non-unique. These limits hold for both signal processing and symbol processing models (Rouse, et al., 1990).

These types of limit have important implications. The most obvious, of course, is the possibility of not fully modeling the phenomena of interest. Other implications include a poor fit relative to the possibilities with other models, or multiple equally good fits with two or more models. As a result, it is quite feasible for system control to be based on non-unique state estimates and/or system reasoning to be based on non-unique explanations.

Non-uniqueness can occur either because a chosen model can produce multiple state estimates or multiple explanations that fit the available data, or because the model chosen was only one of multiple models that fit the data. In either case, non-uniqueness can lead to, for example, the conclusion that an aircraft's landing gear are not locked in place when, in

fact, a sensor failure in the gear light circuit could also have produced the same symptoms. Model-based reasoning that does not include a representation of the gear light circuit will "confidently" reach the conclusion that the gear are not locked in place.

The consequences of the above limits and implications are shown in Table 1. The consequences differ depending on whether the intelligent system is for the purpose of supporting humans via computer aiding, or replacing humans via automation. These consequences, however, are very undesirable in both cases.

What is needed are methods for detecting these types of consequence, diagnosing their causes, and correcting or compensating accordingly. This report presents an overall formulation of this problem and discusses an initial approach to developing the required methods. In addition, concluding discussions in this report consider the implications of the proposed approach for addressing software safety issues (Parnas, et al., 1990; Rushby, 1988).

## PROBLEM FORMULATION

Figure 1 depicts the overall problem being addressed. An intelligent system interacts with the world by sensing inputs $\underline{x}$ and producing outputs $\underline{y}$. The intelligent system includes models $M_i$, i = 1,2, . . .. These models provide the basis for decision making, as well as functions that make changes to the world. The relationship between decisions and functions is such that a decision is a choice, by the intelligent system, to use a function or to use one of multiple available functions.

Model $M_i$ receives inputs $\underline{u}_i$ and produces outputs $\underline{v}_i$. The input vector $\underline{u}_i$ is composed of elements of $\underline{x}$ and $\underline{v}_j \; \forall \; j \neq i$. Similarly, the output vector $\underline{v}_i$ is composed of elements $\underline{y}$ and $\underline{u}_j \; \forall \; j \neq i$.

Two classes of limitation may have undesirable effects on the relationship between the intelligent system and the world. One class concerns the nature of the models. Four types of limitation are of interest in this class:

| Implications of Limits | Consequences | |
|---|---|---|
| | **Aiding** | **Automation** |
| Inappropriate Model | Advice Wrong | Control Wrong |
| Inadequate Model | Advice Incomplete | Control Incomplete |
| Non-Unique Model | Misplaced Compliance | Misplaced Confidence |

Table 1. Consequences of Modeling Limits and Their Implications

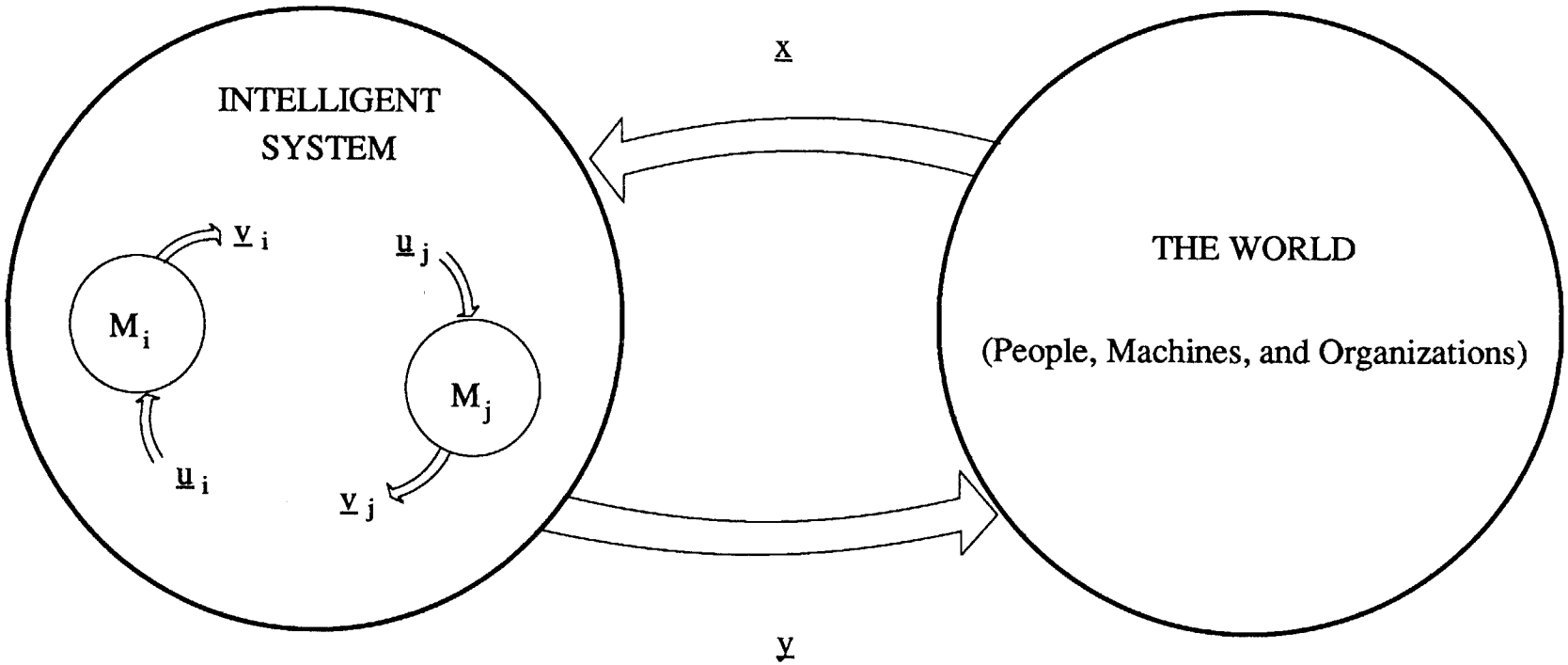



Figure 1. The Overall Problem

- $M_i$ may be incomplete, e.g., one or more relationships between $\underline{u}_i$ and $\underline{y}_i$ is missing,
- $M_i$ may be inaccurate, e.g., representation of phenomena is a poor fit relative to actual phenomena ($\underline{x}$, $\underline{y}$) in the world,
- $M_i$ may be incompatible, e.g., representation is complete and correct, but level of representation is inconsistent with available $\underline{x}$, necessary $\underline{y}$, and/or $M_j$ $\forall$ $j \neq i$, and
- $M_i$ may be incorrect, e.g., representation of phenomena is a very bad fit relative to actual phenomena in the world.

An example of an incomplete model is the aforementioned lack of inclusion of a representation of the effects of the gear light circuit. An illustration of an inaccurate model is a first-order approximation of the response dynamics of the landing gear that, by definition, ignores the inertia of the gear but, nevertheless, may provide "ballpark" predictions. An incompatible model might accurately represent extension and retraction of the gear in terms of exchanges of potential and kinetic energy but not provide an appropriate basis for control. Finally, an incorrect model might represent gear extension and retraction as a simple pendulum which neglects many important aspects of the dynamics of landing gear.

Considering remediation of these types of problem, relationships would be added to incomplete models. Relationships would be modified, perhaps parametrically, for inaccurate models. Incompatible models would be transformed structurally. Finally, incorrect models would be replaced with a different representation or structure.

The second class of limitations relates to the ways in which $\underline{x}$ and $\underline{y}$ are represented. Six types of limitation are of interest, three for x and three for y:

- $\underline{x}$ may take on unanticipated values, e.g., $\{x_i\}$ take on values outside the ranges specified in $\underline{u}$,
- $\underline{y}$ may require unanticipated values, e.g., $\{y_i\}$ require values outside the range supplied by $\underline{v}$,
- $\underline{x}$ may have missing elements, e.g., $x_i$ is relevant but not included in $\underline{u}$,
- $\underline{y}$ may have missing elements, e.g., $y_i$ is needed but not included in $\underline{v}$,
- $\underline{x}$ may have extra elements, e.g., $x_i$ is included in $\underline{u}$ but not relevant, and
- $\underline{y}$ may have extra elements, e.g., $y_i$ is included in $\underline{v}$ but not needed.

These limitations relate to underspecification or overspecification of the inputs and outputs between the intelligent system and the world. Underspecification implies that relevant phenomena in the world are not modeled; overspecification means that phenomena that are modeled are not relevant.

Examples of unanticipated values might be state variables such as velocities or pressures taking on values outside the ranges expected by the software, and control signals (e.g., position commands) from the software being such that the world inherently cannot satisfy them. Illustrations of missing elements include missing state variables and controls -- not infrequently this problem may occur in conjunction with an incomplete model where the missing relationship(s) involves the state variables and controls identified as missing. Examples of extra elements include measurement of variables that do not play a role in the phenomena of interest and production of control signals that do not affect these phenomena. This type of lack of understanding of the phenomena being modeled may evidence itself in conjunction with the development of a model that is incomplete and/or has missing elements.

The ten types of limitation outlined in this section could lead to several types of consequence. If a decision, which is based on one or more models, is wrong, functions will be activated inappropriately. A function can be thought of as a controller, and a decision as turning the controller on or changing the controller's setpoint (i.e., its goal). A controller turned on or off inappropriately represents an error of commission or omission, which usually leads to degraded performance. If a decision results in an incorrect setpoint (or goal), the controller will inexorably pursue this goal and possibly take the system into an undesirable state such as oscillation or instability. Consequences can also include inaccurate, inadequate, or incorrect performance of a function -- of course, these types of problem are not unique to intelligent systems.

## SOLUTION REQUIREMENTS

Remediating modeling problems within intelligent systems involves accomplishing three things:

- Determining what happened -- detecting that unacceptable consequences have occurred.
- Determining why it happened -- diagnosing the causes of unacceptable consequences, i.e., problems with M, x, or y.
- Determining how to remediate -- compensating for consequences and/or causes.

Each of the three elements of the solution can be approached in three ways, as shown in Table 2. In addition, each of these three approaches can be pursued during three different time periods or phases: design and development, test and evaluation, and operational use. A balance among these three approaches and phases can provide a comprehensive framework for detecting, diagnosing, and compensating.

Analytical/simulation methods enable one to predict consequences of initial design choices and, hence, redesign before the fact. This is the predominant approach in most design endeavors. The disadvantage of this approach is that one inevitably predicts, and designs solutions for, many possible consequences that will not actually occur.

Controlling via intelligent systems/automation has very desirable characteristics. By responding during the fact, consequences do not propagate, at least not as far. Further, resources are only devoted to problems that have occurred. Online detection, diagnosis, and compensation as operational problems arise would represent the type of self-monitoring skills that is one of the hallmarks of true expertise (Chi, Glaser, & Farr, 1988). While the state of the art is such that the concept of an intelligent system monitoring itself is not yet feasible, the approach outlined later in this report may enable taking a first step in that direction.

The least preferred of the three approaches in Table 2 is observational methods in that the remediation of problems depends on limitations manifesting themselves in terms of

| | DETECTION (What) | DIAGNOSIS (Why) | COMPENSATION (How) |
|---|---|---|---|
| PREDICTING (Before the fact) | ← Analytical/Simulation Methods → | | |
| CONTROLLING (During the fact) | ← Intelligent Systems/Automation → | | |
| OBSERVING (After the fact) | ← Experimental Methods → | | |

Table 2. Alternative Approaches

unacceptable consequences. The result is redesign after the fact. When this approach is necessary, it hopefully can be employed during design and development, or test and evaluation, rather than during operational use. Of course, as Parnas and his colleagues emphasize, problems inevitably emerge with software as use proceeds (Parnas, et al., 1990).

## PRIMARY ISSUES

Several issues must be resolved if solutions are to be produced that satisfy the above requirements. An obvious and predominant issue is the tradeoff between desirability and feasibility. While self-monitoring systems (i.e., controlling in Table 2) are highly desirable, they are unlikely to be feasible without first pursuing approaches based on observation, and subsequently developing methods of prediction. In other words, observation and prediction will provide the basis for online control.

A particularly important issue concerns internal versus external evaluation. Internal consistency and completeness is an important concern. Nguyen and his colleagues (Nguyen, et al., 1987) discuss methods for identifying syntactic problems such as:

- Redundant, conflicting, subsumed, and circular rules,
- Unnecessary if conditions,
- Unreferenced and illegal attribute values,
- Unreachable conclusions, and
- Dead-end if conditions and dead-end goals.

These are clearly types of problem that one would rather not have in an intelligent system. However, as Rushby (1988) emphasizes in his review of efforts similar to that of Nguyen, assuring that these problems do not exist, i.e., assuring consistency and completeness, is not sufficient for success.

Beyond the necessary condition of successful internal evaluation, one must also satisfy the sufficient condition of successful external evaluation. This is the primary focus of

the ten types of limitation discussed earlier in the context of Figure 1. The external reference for these limitations is the world -- people, machines, and organizations.

A central issue concerns how external reality is conceptualized and how it is accessed. This is a subtle issue because one is concerned with assessing the extent to which the representations embedded within the design of the intelligent system are incompatible with the actual phenomena in the real world. The subtlety is due to the need to represent the real world in a manner broader than implied by the design basis of the system.

This issue is of most concern at the "knowledge level," which lies above the symbol level where representations are encoded (Newell, 1981). At the knowledge level, distinctions among behavioral, physical, and operational phenomena are not usually crucial. The focus is on comparing knowledge within the intelligent system to knowledge about the real world, the source of which is inevitably either human operators, maintainers, managers, or designers. In other words, in contrast to physical measurements, knowledge about the real world is not accessible other than through experiences and interpretations of humans.

One alternative approach to representing the real world is to evaluate the intelligent system in the real world, perhaps via a flight test or equivalent. While this type of test is essential to operational credibility, it can be a dangerous way to find the types of problem discussed earlier. Further, the expense of this type of evaluation precludes consideration of the wide range of conditions that the real world will inevitably produce after the software is put into operational use (Parnas, et al., 1990).

Most fundamentally, such an evaluation may have limited validity for addressing the types of issue emphasized in this report. Since tests of this type are usually "designed," it is unlikely that one would incorporate in the test plan conditions and phenomena of which one is unaware, e.g., situations not described in the design requirements documents. Put simply, it is difficult to know what you do not know. This difficulty can be lessened by using independent testing organizations (Parnas, et al., 1990) -- however, this difficulty is unlikely to be eliminated with this approach.

What is needed is a means of interacting with the real world in a manner that does not inherently constrain the world to operate in conformance with the design basis of the intelligent system. One obvious approach is to simply field the system. However, this presents even more problems of cost and danger than flight test or its equivalent. Thus, this approach is only feasible to the extent that appropriate means of control (i.e., Table 2) can be established.

This problem can be resolved, at least in part, by recognizing the fact that evaluation seldom requires all of reality. One need only have that portion of the real world for which the knowledge embedded in the intelligent system needs evaluation. Thus, one can often capture primary effects and low-order interactions by considering only a portion of the real world within which the intelligent system is intended to operate. Nevertheless, this does not mean that the problem of validating intelligent systems is now easy -- simply that it may be tractable.

This idea is far from novel. It is quite common to initially evaluate displays, controls, procedures, etc. using real pilots in a high-fidelity simulator. In a similar manner, engines are initially evaluated using real operating conditions on a test stand, not an airplane.

While the basic approach is not new, the ways in which this approach has to be used to satisfy the objectives of this project are rather different. In particular, the level at which the intelligent system and the real world interact must be such that the phenomena represented in the intelligent system and the phenomena accessed in the real world are comparable. Further, the interaction must be such that the real world is not inherently constrained to reflect the intelligent system. This latter issue can be particularly problematic when the portion of the real world of interest is highly adaptable -- as is the case with humans.

Of course, adaptability is also the source of many individual differences in humans. If these differences reflect varying preferences, then modeling problems are likely to be primarily incompatibilities. On the other hand, if individual differences are due to

incomplete or incorrect views of the world, then either training or aiding (or both) is needed to remediate these differences. In general, it is quite common to find differences in preferences, but not common to find that trained and experienced personnel have incorrect models of the world they operate within (Rouse, 1990).

## AN EXAMPLE

To further refine the issues outlined thus far, as well as illustrate how these issues can be addressed and resolved, it is useful to continue the discussion in the context of an example. The display selection functions within the Pilot-Vehicle Interface (PVI) of the Pilot's Associate provide a rich illustration of how the impact of modeling limits can be pursued*.

Display selection is performed by two modules: event-based display selection (EBDS) and plan-based display selection (PBDS). EBDS results in displays of bottom-up data that are determined to be significant by assessors (i.e., situation assessment or system status functions) or the PVI.

PBDS results in displays of the information needed to execute active plans. This is considered to be a top-down process because the information displayed is necessary for the active plans. Active plans and goals can originate from interpretations of pilot actions or from plans created by planners (i.e., mission planning, tactics planning, or system status functions) that were proposed to and accepted by the pilot.

### Structure and Validity

To discuss validity in the context of this example requires some understanding of the structure of PVI knowledge bases for EBDS, PBDS, and plan and goal interpretation.

---

* See Rouse, Geddes, & Curry (1988) and Rouse, Geddes, & Hammer (1990) for comprehensive discussions of the PVI.

The EBDS knowledge base structure is organized around the concept of a message expert. This expert may correspond either to a type of assessment (e.g., afterburner blowout or compressor stall), or it may correspond to a controller of a particular display attribute (e.g., the range setting of a map display). Each message expert examines the global state of the aircraft, pilot, and external environment before making a display choice.

Because of its small size and simple format, correcting a message expert is relatively straightforward. A message expert is rarely over fifty lines of code in length and appears to be a table (although it is in fact invoking numerous "macros"). Its form is a simple expression of the display options and external state considered, and thus its behavior is apparent. Consequently, changing a message expert is relatively easy.

In contrast, PBDS validity depends on several PVI modules and is, therefore, substantially more complicated than EBDS. PBDS chooses displays to satisfy the information requirements of active plans. Validity concerns may be raised in the interpretation that leads to active plans or in the mapping of plans to information requirements. For example, the interpretation process could produce an active plan that should not be active, or an active plan could be incorrectly connected to an information requirement.

The concern in this example is with the rules that relate actions to plans, and plans to goals, as well as information requirements associated with plans. The objective is to assess the external validity of these rules relative to the real world, which in this case means relative to pilots' perceptions of goals, plans, and information requirements. To accomplish this objective, pilots must be probed at the goals, plans, and requirements levels as they perform real piloting tasks.

Knowledge Engineering

Knowledge acquisition for the PVI was done in several stages. The initial knowledge was acquired using pencil and paper forms. Initial design of the knowledge representation

was completed before this initial acquisition. Domain experts were required to express the knowledge within this design structure. After initial acquisition, considerable modification and restructuring were required to make the knowledge-based modules perform adequately. Only rarely did the knowledge engineers consult domain experts about the suitability of this modification and restructuring.

Later in the process, domain expert comments often were not even recorded on paper. Instead, the knowledge engineer interacted with the domain expert to determine where the problem was in the PVI structure. The correction to the structure was the only permanent result of the domain expert's original comment.

Some people hold a naive view that domain experts' original comments are useful in and of themselves. The only use for such information would be a historical analysis of the evolution of knowledge bases. However, the original information is of insignificant value compared to that represented in the operational knowledge base.

The relationship between modeling limits and the knowledge base entities -- plans, goals, information requirements, and rules -- is that knowledge bases can be considered to be models. The goals, plans, and interpretation rules can be considered to be a model of the pilot's domain -- what the pilot does and why it is done. Probing pilots for goals and plans will allow evaluation of this domain model in terms of the modeling limits previously described.

Individual Differences

Individual differences among fighter pilots are often seen as a severe problem in knowledge acquisition. Our experience is that this problem is rather minor for the following reasons. First, two pilots may have different preferences but be unable to state any reasons for their preferences. An example is the amount of clutter on a head up display (HUD). Some pilots prefer much symbology on the HUD; others want only the minimum necessary.

Neither can explain why, and of course, this very fact prevents it from being entered into a knowledge base.

Second, pilots may differ strongly over relatively minor and insignificant points that are of no consequence. For example, a message about a threat that was out of range could contain either the literal range or a simple clause "out of range." The relative merit of one or the other is of very little importance relative to the significant problems in constructing and validating knowledge bases.

Third, improper framing of a question can result in the question being answered at the wrong level of abstraction. For example, one could ask pilots for the range when the PVI should report the radar resolution of a blip into two closely separated blips. However, to ask this question in terms of range is likely to result in problematic answers because the correct answer relates to whether or not the pilot is attacking these threat(s) or vice versa. We have observed that domain experts do not usually restructure questions on their own -- they simply answer whatever question is asked.

Concept Graphs

Within the Pilot's Associate are knowledge bases that can be viewed as concept graphs. These concept graphs contain concepts $\underline{c}$ and links $\underline{l}$. A concept $c_i$ describes a state or assertion about the world. For example, a concept may be a normal-landing plan or an information requirement for the range to a threat aircraft. While a concept is similar to a state variable in a conventional system, it is richer in several ways.

First, a concept has attributes, each of which may take on values. For instance, the normal-landing plan has an attribute naming the destination airport. An information requirement has an attribute giving the precision with which the range should be known.

Second, a link connects two concepts. Usually, a link is implemented by a production system rule. This rule may modify attributes of existing concepts or instantiate new copies of known concepts. The modifications caused by the rules are intended to model

the dynamics and semantics of the domain. For example, three kinds of links -- interpretation, exclusion, and information need -- are used in the Pilot's Associate knowledge bases.

An interpretation link describes (or models) when a pilot's action, plan, or goal, may be explained by a higher level plan or goal. For example, the action of raising the landing gear has four explanations as shown in Figure 2: *gear cycle plan, climb, gear up landing*, or *go around.* If the pilot raises the gear, one of these four plans is an explanation. The plan that is the most appropriate explanation is determined by the constraints in the rule that implements (at a symbol level) the link. For example, gear up landing is an appropriate interpretation (for most aircraft) if only at most one of three landing gear operates correctly, even after several attempts (i.e., use of the *gear cycle plan*).

An exclusion link prevents two plans (or goals) from being active simultaneously when such would make no sense. For example, the plans *normal-landing* and *low-thrust-landing* exclude each other. The exclusion links are not shown in Figure 2, although a graph could be drawn with both exclusion and interpretation links.

An information need link is between plans and information requirements. In the case of the normal landing plan, the information required could include a map display of the airport, radio and navigation frequencies for the particular airport, etc.

The benefits of concept graphs are in the structuring of the knowledge base design and the designer's ability to perceive relationships in the knowledge base's design. The design practices in knowledge engineering for the PVI illustrate an attempt to design before substantial knowledge acquisition. Preliminary knowledge acquisition is done only to structure the concept graph. After the initial structure is determined, a drawing of the concept graph can be inspected for design errors. This visualization of the knowledge base allows the knowledge engineer to view the knowledge level rather than the symbol level. Similar visual aids such as control and data flow diagrams aid the software engineer.

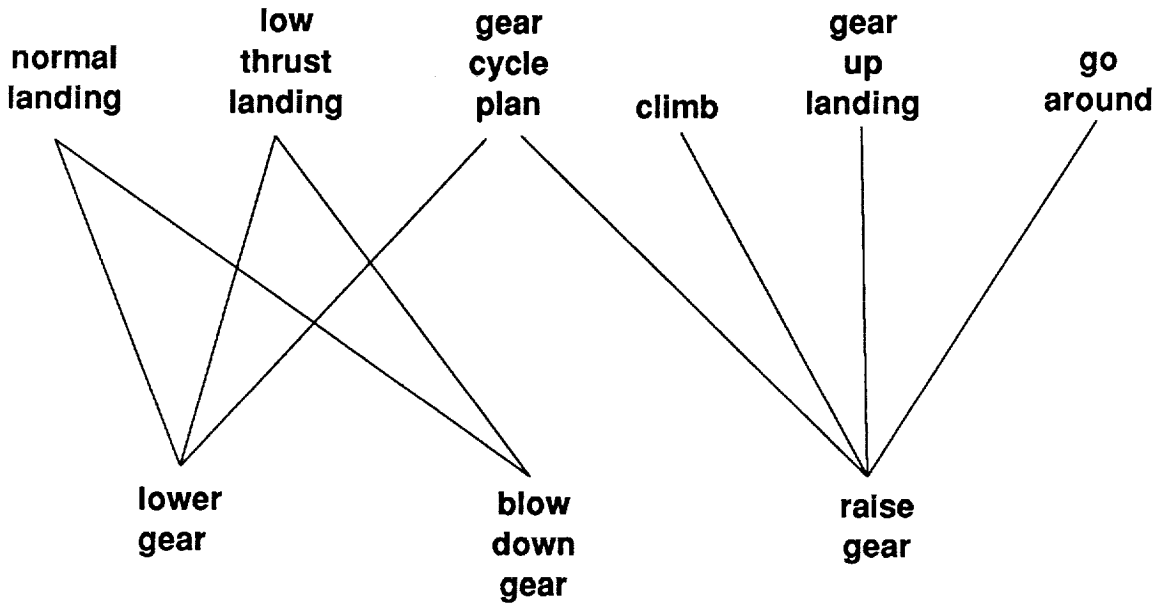



Figure 2. Partial Plan Goal Graph

Identifying Limits

The overall approach for evaluating the PVI is to probe pilots for plans and goals related to their reactions to displayed information in the course of a flight scenario. To assess validity, these probes should be of two types. First, pilots should be asked to react to the goals, plans, and requirements inferred by applying the rule base to interpret their actions. A negative response indicates a problem. However, a positive response only indicates the absence of evidence of a problem. This is a very weak conclusion.

To strengthen the test, queries can be used whose negation represents support for the knowledge base being tested. These queries have to be carefully designed to provide this type of evidence. However, use of these types of query enable the kinds of results depicted in Figure 3. Thus, for example, a negative response to a negative query (e.g., rejecting the intent-based presentation of an FMS display) can be interpreted as strong support to the extent that this query crisply counter-indicates the information requirements in the plan goal graph.

The testing process is a backwards exploration through the concept graph(s) in search of the error that caused the output to take an incorrect value, as judged by the pilot. The test is initiated by focusing on a decision, an observable output y produced by the intelligent system. Differences between the pilot and the intelligent system are traced backwards, looking for limits of incompleteness, inaccuracy, incompatibility, and incorrectness as described earlier.

The simplest description of the nature of a test is that it pursues in depth any differences in judgment between the pilot and the knowledge-based system. When a pilot action occurs, or at points in time where no action has occurred for some time, the pilot is asked if he differs with the intelligent system's choice of displayed information. For example, the pilot might want the tactical map display in the horizontal view (latitude and longitude but no altitude) while the system is displaying a vertical view (azimuth and

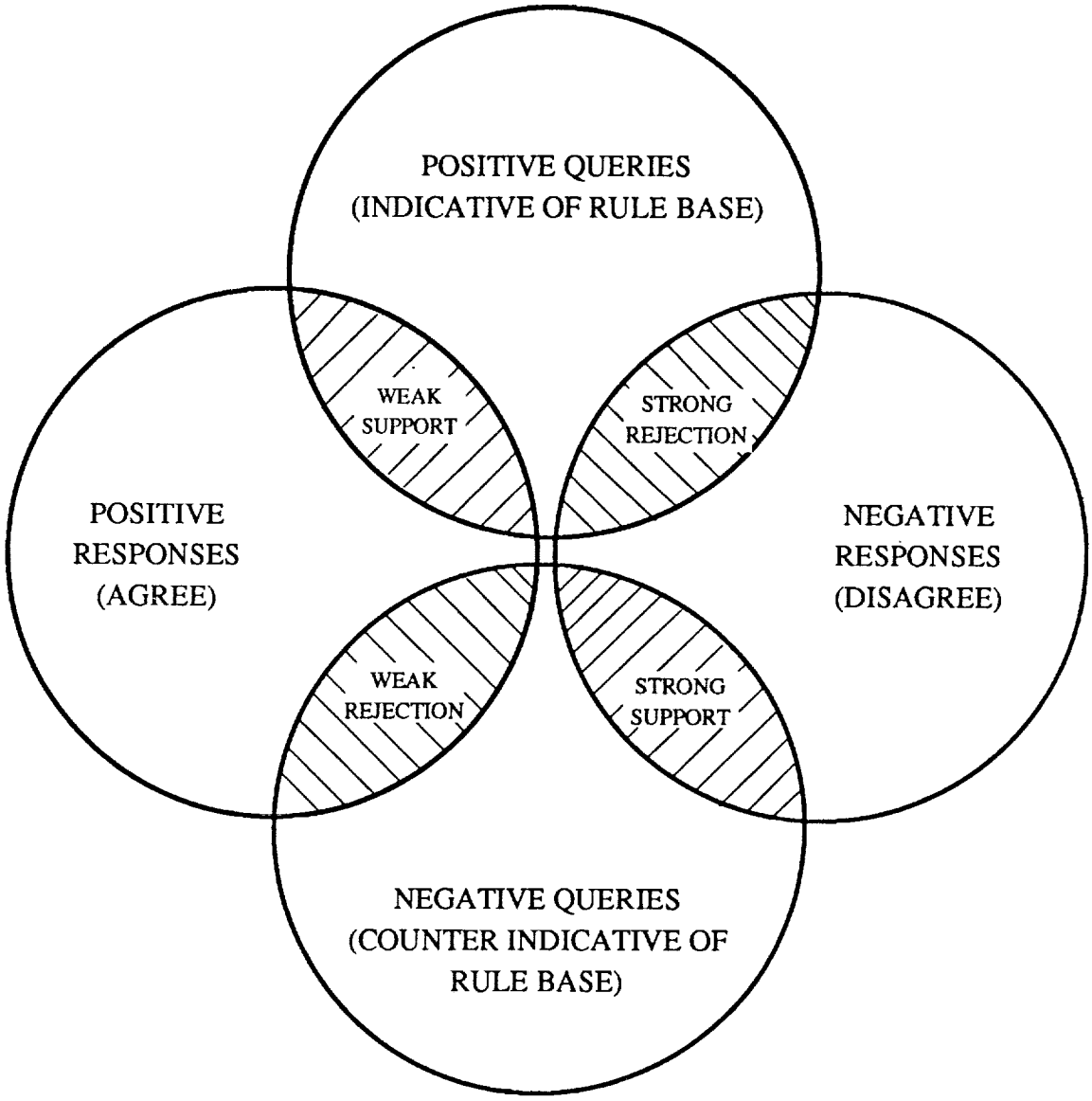



Figure 3. Possible Test Results

elevation but no range). The pilot's judgment is interpreted as correct, and the intelligent system is therefore considered to be incorrect.

Each difference in judgement is explored individually as follows (see also Figure 4).

1. Find the information requirement (IR) that generated the unacceptable display.

2. Determine what caused the IR to be displayed. The error could be in the concepts, the links between concepts, or both. An example of a link error would be a plan that is correctly activated; however, its link to the information requirement is incorrect. By the link being incorrect we mean that the pilot does not want the information when the plan is activated. By the plan being correctly activated we mean that the pilot agrees that the plan should currently be activated.

3. If the error is in the link, then a link from the plan to the IR should be deleted and tracing ends for this difference.

4. If a plan activation error occurs, then the process continues as below. A plan activation error is either a plan that the system has activated that should not be or a plan that the pilot says should have activated but was not.

5. At this point, there is a difference between pilot and intelligent system over the activation of plans or goals. This activation occurs by a bottom-up, rule-based search in the plan goal graph. Beginning at the pilot action at the bottom of the graph, the search traverses links (fires rules) to reach a plan that is currently active or could be active. From the plan, the search continues upward to a goal, then to a plan, then to a goal until reaching either a plan or goal that is already active. The question then is whether the error is in the rule (link) that connects a child to the parent, or is the child incorrectly activated?

6. If the rule is erroneous, then further analyze the rule as described in more detail in the later section and end the tracing of this error.

7. If the child is inappropriately activated, then continue tracing recursively as described in step 5.

Interacting With the Pilot

When actual system behavior differs from what the pilot desires, one first should ask if the aircraft system can express the desired behavior (e.g., is there a way to show such behavior on the displays). If this cannot be done because the displays or aircraft simulation do not support it, one should note the discrepancy and continue. One should then find the concept that seems to express the desired behavior if possible.

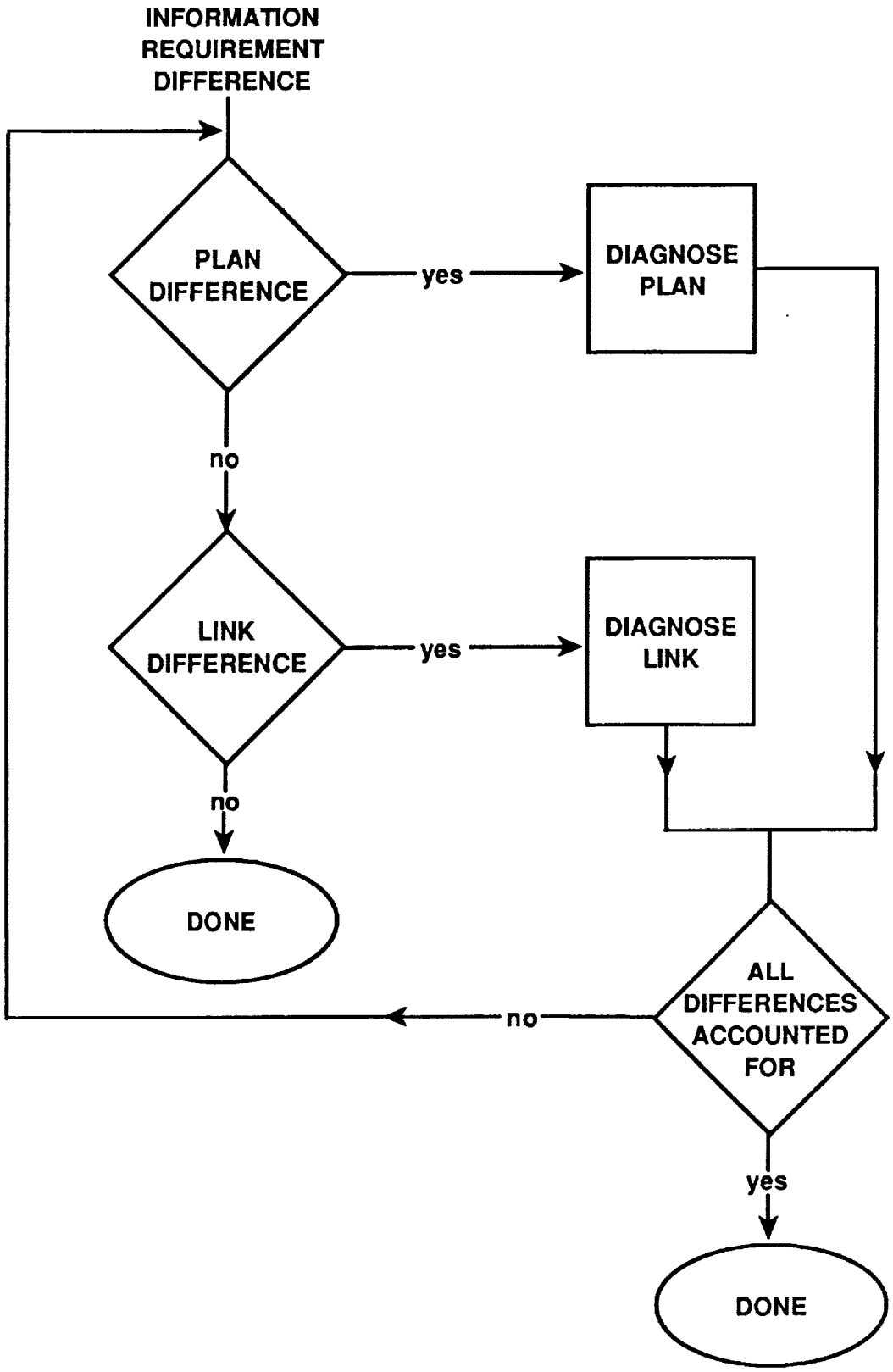



Figure 4. Identifying Limits

The next step is to verify that the internal definition of the concept is the same as the pilot's. Some exploration with the pilot is often necessary because a word may mean different things to the pilot than it does within the concept graph. The knowledge received from the pilot must be reconceptualized in terms of existing (or new) structures in the concept graph.

If one cannot understand the pilot, then one asks the pilot what the concept looks like concretely on the display, how the display is driven by sensed signals from the environment, and how the display behavior changes with changes in the sensed signals. Even if the behavior is of aircraft action systems (e.g., a jammer) rather than a display, the same approach is appropriate.

An Illustration

In this illustration, a simple knowledge base is developed for a sample problem: interpreting the raising and lowering of the landing gear. The knowledge base functionality is limited to determining if the pilot's actions are sensible. Actions that do not make sense, as well as inactions that do not make sense, would be processed elsewhere in pilot-vehicle interface (i.e., the error monitor) and are not described here.

The key landing gear inputs are the commanded and actual gear position and the gear lock indicator. It is assumed that all inputs are sensed, and that sensors may fail. A single hydraulic system supplies the gear hydraulic cylinders and doors with sensors for pressure, fluid quantity, and drive pump status. The backup to the hydraulic system is a pressured bottle of air that is sufficient to lower the gear should the hydraulic system fail. The monitor has access to engine, fuel, etc. status and the position of the aircraft relative to the runway.

While a simple knowledge base might only consider altitude, vertical velocity, and airspeed, the following scenarios illustrate the richness of interpreting landing gear actions:

> Failure to get a locked gear indication may be only a sensor failure. The pilot may have other reasons (a fly by of the tower) to hope or believe the gear are locked. After cycling the gear several times, the pilot may choose to land while minimizing the load on the suspect

gear until that last possible moment. The PVI could issue a message in this situation, but it should not imply that the pilot has committed an error.

- The pilot may be attempting a gear up landing because of hydraulic failure or because two gear will not indicate locked.

- The pilot may intend to delay lowering the gear until the last possible moment because this action will commit him to landing. The aircraft may have damage, particularly engines inoperative, that could prevent a go-around if the gear are down.

- The pilot might lower the gear early to increase drag to decrease speed. Although feasible on commercial aircraft, this is very unlikely on fighter aircraft due to the effectiveness of fighter speedbrakes.

- On takeoff, the pilot may leave the gear down due to hydraulic failure and the intention to land immediately. Should the aircraft go too fast, the error is really exceeding the gear speed limit, not failing to retract the gear.

- The indicators used in the rules - airspeed and altitude (AGL rather than barometric altitude) may be erroneous due to sensor error. An error message about lowering the gear based on these erroneous signals would be confusing to the pilot.

- Under some gear malfunctions, ejection is appropriate in tactical aircraft.

- The landing gear are lowered early because the hydraulic system is starting to fail. If the pilot waited, the hydraulic system may not be able to lower them at all.

- In some areas, landings on highways rather than airstrips may be possible. The PVI may not know the locations of highways, especially those areas designated for landings.

The partial plan goal graph shown in Figure 2 provides an initial knowledge base. The structure is partial because it deals only with actions related to landing gear. The plans described in this knowledge base are defined as follows.

- Normal-landing: The normal landing plan in which all landing gear are extended and properly locked into position.

- Low-thrust-landing: With limited thrust, the approach is flown at higher speed and the gear are lowered late, only after the pilot commits to the landing.

- Gear-cycle-plan: The gear may be retracted and extended several times to get them into the desired state: down and locked. This is usually done if they do not lock in position on the first attempt.

- Climb: Climb is the normal successor to takeoff.
- Gear-up-landing: Because the gear cannot be placed in the desired state, they are all retracted and the aircraft lands while sliding down the runway on its skin.
- Go-around: If a landing does not look good, it will be rejected by accelerating and climbing out over the runway. The aircraft will circle for another try at landing. The gear would ordinarily be raised to decrease drag and thus increase aircraft performance. However, if they were blown down or were hard to lock, they may be left down.

Example Limits

Based on this illustration, examples of incomplete, inaccurate, incompatible, and incorrect limits can be elaborated as follows:

Incomplete. In the example given there is a difficulty in interpreting the *gear cycle plan.* If this plan is appropriate, then both lowering and raising the gear could be connected to it. If extending the gear a second time corrects the problem, then it should be interpreted as part of *normal landing* rather than *gear cycle.* The difficulty is that the correct interpretation will not be known until after the action takes place, for the gear take a few seconds to lower and lock into position.

One solution is to delete the link from *lower gear* to *gear cycle plan.* Then, lowering will always be interpreted as some kind of landing plan. Although the cycling plan would seem to contain both raising and lowering the gear, it should not. In fact, the *gear cycle plan* should probably have been named *retract-and-retry gear plan.* The old name of the plan suggests that both raising and lowering are parts of the plan. In fact, the semantics of interpretation depend on a variety of links around the concept rather than the name of the concept.

Links may be added as well as deleted. It is possible, depending on the hydraulic design, for the gear-blow-down action to raise the landing gear (once) instead of lowering them. This would be an advisable action should a gear up landing be prescribed and the hydraulic system has failed. Whether this link should be added depends on whether the

design would permit such an action to work. The presence or absence of check valves (that permit one-way flow) would determine whether this would work.

Another example of concept modification is the addition of an arrested-landing plan. During this plan, a hook is extended from the back of the aircraft. At touchdown (or the far end of the runway), the hook catches a restraining wire that stops the aircraft in a very short distance. An arrested landing is appropriate when the aircraft is overweight, braking has limited effectiveness, or control over the aircraft would be limited after touchdown.

Inaccurate. A simple error that occurs frequently is the incorrect threshold for a state variable. For example, the interpretation from *lower gear* to *normal landing* must look at state variables to determine if lowering the gear is appropriate. One possible comparison would be that the absolute altitude be below 2,000 feet (a constant) and the vertical velocity must be less than another constant. It is possible for these constants to be erroneous.

Incompatible. There is another interpretation of using the absolute altitude to determine whether the gear were lowered appropriately. Altitude may in fact be a poor state variable to use. For instance, consider landing on a plateau. A better state variable would be to use the distance from the runway, or perhaps position relative to glideslope and localizer (assuming that is known). The more general problem here is that some state variables while feasible are rather noisy for making good decisions. Because the state space is large, choosing the right state variable is difficult. Asking the pilot which variables to use is not necessarily going to work, because the variables mentioned in the explanation are still not necessarily the best to use.

One place to begin is to explain the rule to the pilot and ask what conditions are not accounted for. This may or may not work because the pilot may need to understand the definitions of the concepts used. This is problematic when the concepts are abstract or are purely the invention of the knowledge engineer. Another approach is to ask the pilot what the rule should be and then attempt to translate that into the available concept space.

When modifying the concept graph by changing concepts, some care is required. The structure of the domain, to the extent it has already been understood, should be used wherever possible as a source for the concepts in the knowledge base. In other words, one should try to use what one already understands as a model for the knowledge base structure. In particular, employing precise definitions from a particular discipline or domain is probably a good idea. Practitioners created precise definitions for the same reasons that knowledge bases should employ them.

Choosing concepts at the correct level of abstraction is a significant difficulty. Pilots often describe events rather than goals and plans associated with a situation. They also talk in terms of detailed examples rather than abstract terms. Consequently, some dialogue with a pilot may be necessary to represent knowledge in terms of goals and plans, as well as at the appropriate level of abstraction.

Incorrect. A low thrust landing could be detected by a rule that looked for various malfunction states in the aircraft engine. This would seem quite appropriate in an aircraft equipped with a Pilot's Associate because such assessments about engine health would be easily available to the intent interpretation function. However, the above approach is wrong. Further, this example well illustrates the difference between planning and plan interpretation. A planning approach is to prescribe what the pilot should do. Planning requires information about engine health. Plan interpretation is to describe what the pilot is doing. A plan understanding approach would use a rule that should look at the engine state(s) to determine if only limited thrust is currently being employed. If the pilot has configured the aircraft for a limited thrust landing (perhaps for practice), then that should be the interpretation.

Lowering the landing gear does increase drag. While there are some situations where increased drag is desired to slow down the aircraft, landing gear are not used to do this in tactical aircraft. If the gear are lowered at too high a speed, the doors will be torn from the aircraft. These doors are likely to do further damage as they bounce through the air along the underside of the aircraft.

## AN EXPERIMENT

To test the proposed approach to assessing model validity, data were collected in a flight simulator that is used to demonstrate the Pilot's Associate. The data were pilots' reactions to the choice of displays and display content in the simulator. The data were from observation rather than controlled experimentation. The reasons for this type of data are the level of maturity of the analysis and the similarity of observational data to knowledge engineering.

The remainder of this section describes the aircraft, its displays, the simulated mission, and the observation method. The aircraft simulation was a generic Advanced Tactical Fighter (ATF). The primary displays were thirteen inch (diagonal) color raster CRTs arrayed left to right. All CRTs were touch sensitive, and virtually all controls were menus with touch sensitive buttons. Also part of the simulation, but not part of this study, were sound/voice output, voice input, a touch sensitive plasma panel display/control between the pilot's knees, and a simulated head up display (HUD).

The flight simulator behavior was predetermined and the pilot did not and could not interact with the simulation. This was done to improve the reliability of the demonstration and to demonstrate what the Pilot's Associate would look like in real-time. A live demonstration would run several times slower than real-time. To create this "canned" demonstration, the live message traffic between computers was collected and stored in a disk file. To present the canned demonstration, the disk file was read, and messages were transmitted to the display generator computers at the time given by the message timestamp.

The simulated mission was an escort of eight F-15E fighter-bombers (call sign Hammer 01 through 08) by four ATF fighters (call sign Knight 01 through 04). The F-15E aircraft flew at a low level along a pre-planned path to an enemy airbase. The ATF fighters flew at high altitudes to draw off, chase off, or destroy any threat aircraft that endangered the F-15E aircraft or the ATF fighters. After the F-15E aircraft completed their attack, they were

responsible for their own protection, and the ATF fighters were freed from escort responsibilities.

The single subject in this experiment was a retired USAF pilot with five thousand hours in tactical and strategic military aircraft. He was intimately familiar with the mission, the displays, and the Pilot's Associate. No time for familiarization was needed. This situation is similar to a pilot with considerable experience with a fielded aircraft.

The data were verbal reactions of this pilot to display selection and contents. The pilot was encouraged to identify any discrepancies or errors in the displays. A number of probes were made by the experimenter. A probe was a question about display behavior that appeared questionable to the experimenter.

## RESULTS

All of the data were time stamped with the mission time, which was always displayed in the crewstation. The data and detailed analysis appear in Appendix A. Some of the analysis was made possible by viewing the demonstration a second time and by examining the prerecorded message traffic described earlier.

Table 3 contains the experimental data and a summary of their interpretation. Each table contains the time at which the discrepancy was detected, a description of the discrepancy including a causal analysis, and a classification according to the scheme presented earlier.

## DISCUSSION

Limits that have been classified as incomplete, incompatible, incorrect, or missing input tend to have a single interpretation. Many of the discrepancies are the result of the models that do not exist in the PVI, or of phenomena that need to be incorporated into the PVI. An example of a missing model is deciding whether to show to the pilot adaptive aiding task allocation and execution messages -- the communication task needs to be

| Time | Class of Problem | Explanation |
|---|---|---|
| 8:11:42 | incorrect | New model is needed to decide whether to communicate adaptive aiding actions to the pilot. |
| 8:11:43 | output unanticipated | Proposed plan name was not recognized by pilot. |
| 8:12:00 | inaccurate | To avoid items only partially displayed at edge of screen, display logic must assume display is slightly smaller than its actual size. |
| 8:12:36 | incorrect | Same as 8:11:42. |
| 8:12:55 | bug | PVI knowledge base assumes incorrectly that previous position display feature is only implemented for threat aircraft. |
| 8:13:43 | missing output | On tactical map display, position of symbol for pilot's own aircraft should be varied from center to the bottom of the screen. |
| 8:14:01 | missing input | No representation of information requirements for positions of cooperating aircraft. |
| | incompatible | |
| 8:14:36 | missing input | No representation of information requirements for portions of future route. |
| | incompatible | Same as 8:14:01. |
| 8:17:06 | missing output | SAM sites not labeled when they first appeared. |
| 8:20:37 | missing input | Same as 8:14:01. |
| 8:21:15 | missing input | Same as 8:17:06. |
| 8:21:16 | incorrect | Adaptive aiding should not announce its task allocations to the pilot if their effects can be seen in the displays. |
| 8:22:43 | incomplete | Model is needed to reflect fact that a substantial display change in peripheral vision would demand pilot attention. |

Table 3. Data Analysis and Results

| | | |
|---|---|---|
| 8:32:35 | training | A map display changed to maximum range due to lack of any nearby threats. With experience, pilots would understand this. |
| 8:37:00 | missing input | Two display labels overlaid on each other. Only one was necessary, because threats were part of same group. |
| 8:37:08 | incomplete | Particular offensive tactic requires display of both offensive and defensive situation. Usually, defensive situation (only missiles directed at pilot's own aircraft are shown) determines display range. |
| 8:37:09 | missing input | Perspective view switched to sideview for all missile evasions. Did not consider whether missile was launched from ground or at high altitude. |
| | incompatible | Same as 8:14:01. |
| 8:38:10 | training | Same as 8:32:35. |
| 8:38:30 | inconsistent | Should change map display range enough to show route home, rather than automatically switching to maximum range |

Table 3. Data Analysis and Results (cont'd)

modeled explicitly. Examples of missing phenomena are the missing inputs related to cooperating flights, future routes, and mission routes.

On the other hand, the knowledge base did execute correctly for those problems it was designed to represent. Selection of system displays (engine, fuel, weapons, sensors, and electrical) functioned correctly. While there were some complaints from the domain expert about the selection of these displays, these faulty selections turned out to be the choice of the human pilot who flew the original mission when the data was collected for replay. This "pilot" was not a domain expert and was choosing displays to show the capability of the system. The selections were not intended to be appropriate for the situation. During the replay, distinguishing these actions from those of the PVI and/or PA was difficult unless one stared continuously at the menus.

Only a single parametric limit was found in the data -- this limit related to display size. This is surprising given that the information requirements knowledge base has more than five hundred information requirements in the knowledge base, each with five to ten parameters.

An aspect of the data analysis we expected to use but did not was the process of tracing links in the concept graph. Considerable "link tracing" has been done in the past to diagnose problems in the knowledge bases. The explanation for this is consistent with the previous observation that the limits uncovered were virtually all phenomena outside what was represented in the knowledge base. The first steps in the tracing process can be followed, but the process ends when the information requirement that needs to be expressed in the knowledge base is not expressible in the current knowledge base. For example, there was no way to express the information requirement for the position of certain friendly flights.

Another explanation for the limited use of link tracing is the nature of interactions between the intent interpretation function, which is descriptive, and the planner function, which is prescriptive. Intent interpretation describes what the pilot is doing, and planners prescribe what the pilot should be doing. The combination of planning and plan

understanding tend to mask each other's limitations. The ideal test would be to run the PVI without planner input. Unfortunately, this was not possible with available replay data.

The nature of the above results also caused us to realize what can be termed root causes of the types of problem exhibited. Many large systems are conceived of in terms of **functions**. To do so with the Pilot's Associate is possible but results in missing its essence. It is more appropriate to think of the Pilot's Associate as making decisions between two or more alternatives. The Pilot's Associate makes intelligent decisions between alternative functions or whether to use a function -- it does not provide functions per se. In other words, the Pilot's Associate functions are decision functions, which will be referred to as decisions to emphasize their choice behavior.

From this perspective, interpretation of the root causes of some of the above limitations is possible. Clearly, some functions should have been identified as needing controlling decisions. There are two reasons for not identifying this need. First, the existence of a function is not always noticed in a design. For example, the primary decision of adaptive aiding was whether or not to aid the pilot by allocating a task to automation. The output by adaptive aiding of a message about this decision was not considered to be a decision in itself. This oversight occurred even though this function was identified early in the design of adaptive aiding.

The second reason for functions without associated decisions relates to the growth of functionality of intelligent systems. New functions are often added to a system under development. However, experience with that function is often necessary before the designers come to realize that the function is not always appropriate. Once this is realized, a decision element can be designed for the function -- prior to this realization, such decisions go unmade.

A second root cause is an insufficient number of inputs into a decision. Decisions based on a single input should quite naturally be suspect. The reason is that few real world phenomena depend on a single input. Similarly, a decision made from a large number (e.g.,

ten) inputs is also suspect. Relatively few real world phenomena depend on large numbers of inputs. We did not find decisions made with too many inputs, but we found many made with too few inputs.

By the phrase "number of inputs used in making a decision" we mean the number referred to in making any specific decision, not the total number used in all possible outcomes of a particular decision. Assume the decision can be represented as a decision tree, where each fork in the tree represents a reference to a single variable. The number of inputs used to make a decision is defined as the depth of the tree, i.e., the number of decision points in a path, rather than the size of the set of inputs (the union of all variables across all forks).

This observation can be generalized to an entire knowledge base. A plot of relative frequency versus number of inputs to a decision might be a useful design aid in two ways. First, relative improvement in a new version of a knowledge base could be measured by comparing its plot to its predecessor's plot. Second, such plots could be used as absolute judgements of maturity. Assuming that a causal theory of the domain could give rise to a prescriptive plot, a reference would then be available, relative to which the actual knowledge base plot could be compared.

## CONCLUSIONS

In this section, we both review the results of this effort as well as discuss their implications for computer-based methods and tools for assessing the impact of modeling limits on intelligent systems.

### Review of Overall Problem

First and foremost, it is important to assert again that the types of modeling limit discussed in this report are unavoidable. There is no simple (or complex) way to guarantee that modeling limits will not affect an intelligent system. Consequently, methods and tools are needed for discovering and remediating modeling problems.

As noted in the Introduction, this has important implications for software safety issues. Of most importance, since we cannot avoid modeling problems, it is necessary to have a process that is likely to uncover these problems, both during design and ongoing use. In this way, software can be certified as having been subject to a particular design and evaluation process, even though it can not be guaranteed to be problem free.

It is essential that this process consider more than only internal consistency and completeness. External validity must also be a central issue. Further, validity must be assessed in a context broader than the design basis of an intelligent system. It should be possible to discover that requirements were inadequate.

What is needed to support such a validation process is methods and tools for detecting, diagnosing, and compensating for modeling problems. In the near-term, we need computer-based methods and tools for doing this efficiently during design and evaluation. Eventually, we would like methods and tools that could be built into intelligent systems to provide detection, diagnosis, and compensation during operational use of the system.

General Approach

The approach developed in this project is, in essence, a process of interactively probing a knowledge base and comparing entities to the real world. Since we are concerned with the "knowledge level," the real world inevitably is one or more humans with knowledge about the domains of interest. Thus, whether we are concerned with physical, behavioral, or operational phenomena, the real world is typically accessed via human experts.

The interactive probing follows the seven-step tracing process illustrated in Figure 4. This process involves tracing through concept graph representations, while attempting to locate sources of unacceptable presentations as judged by domain experts. Concept graphs are a very general form of representation to which many types of intelligent system can be transformed.

Ideally, the probing process should include both positive and negative probes. Positive probes yield strong conclusions when they are rejected by experts, while negative probes yield strong results when they are supported. In practice, as discussed below, meaningful negative probes can be difficult to generate.

Example Application

The general approach outlined above was applied for evaluating the display selection knowledge base within the pilot-vehicle interface of the Pilot's Associate. This knowledge base is represented as a concept graph involving the goals, plans, and information requirements of the pilot's domain. Probing was initiated by the pilot rejecting all or a portion of a display. Tracing then involved considering first information requirements, then plans, and finally goals.

Experimental Results

Summarizing the results must begin with a caveat -- the experiment only involved one scenario and one pilot. Clearly, more data and experience with the proposed approach is needed before we can reach definitive conclusions. This, of course, begs the question of how many scenarios and how many pilots? Our conjecture is that a diminishing returns phenomenon will determine the answer to this question. In other words, evaluation should continue as long as new types of problem are being encountered. This implies that some level of evaluation is likely to be needed throughout the life cycle of an intelligent system.

Turning to the specific results of the experiment, 22 problems were identified in a 37 minute scenario. Many of these problems involved missing models (or components of models) or missing inputs. Only one parametric problem was identified, which speaks well of the models that were reflected in the knowledge base -- most of what was planned to work in a particular way did perform acceptably.

Two general classes of root causes for these problems emerged. First, missing models often involved missing decisions to employ functions, rather than missing

functionality. Second, missing inputs appeared to be related to a general problem of insufficient inputs to capture the portion of the real world being modeled.

The experience afforded by this experiment clearly provided important insights into display selection in the pilot-vehicle interface. However, the primary goal was to demonstrate the utility of the proposed approach to validation. Overall, we found that manual execution of the method made it difficult to perform all of the tracing of interest. Similarly, it was difficult to generate negative probes "on the fly." To fully employ the proposed approach, a computer-based method or tool is needed to support the evaluator.

Supporting Knowledge Engineering

Building a knowledge engineering environment for evaluating intelligent systems using the approaches proposed in this report appears to be a tractable problem. However, would this environment or any results derived from it be useful in general? For problems represented as concept graphs or semantic networks, the concepts of the environment would be directly applicable. For example, the error monitoring function in the PVI uses a concept graph to interpret the severity of pilot actions that may be errors. In another effort, we have demonstrated a situation assessment function that interprets threat behavior using concept graph representations.

In qualitative simulations of physical processes, at least some of the reasoning can be modeled using concept graphs. Nodes could represent components and links represent their interactions. Diagnosis of faults, as opposed to prediction of behavior, may be better represented in other ways. Certain aspects of planning, such as hierarchical decomposition of plans and description of situations, are suited to concept graphs. Conflict resolution may not be well suited to concept graphs.

The knowledge engineering environment envisioned would be less well suited to an expert system consisting of arbitrary rules. At least, early in development, access to internal states (called test points on circuit cards) is essential. An expert system with no hierarchical

domain decomposition (one benefit of a concept graph) or, at least, no test points is, in our opinion, as bad as FORTRAN programs with lots of interwoven GOTO statements. The chances of adequately testing such FORTRAN programs are low.

Elements of a Knowledge Engineering Environment

This section describes the problems faced by the knowledge engineer, why the problems are difficult, and what the requirements are for better supporting the knowledge engineer. These problems are based on our observations in this research and in our building PVI knowledge bases in the Pilot's Associate program.

Given a discrepancy between a human expert and a knowledge base system, a knowledge engineer may choose one of the following ways to remedy the problem.

Add a New Decision or Function. In relatively few cases a completely new capability must be added. This activity requires designer creativity and insight and is difficult to support. One of the most difficult aspects of design is predicting the consequences of a design choice. The tool environment could make predictions only in well structured situations and where it has access to considerable information about the design. Currently, programming languages and CASE tools still lack the structure and the representations necessary to support this activity.

Modify Structure. Knowledge base structure could be modified by adding new inputs or new relationships (relatively easy) or by transforming or replacing all or part of the structure (relatively hard). The interpretation of the data suggests that this activity needs to occur frequently. There are a variety of ways in which this may be supported: editing, design checking, visualization and execution. Each of these forms of support is described below.

The simplest support is that of structured editing with a graphical display of the knowledge bases. Structured editing, in which the editor has some representation of the objects being modified, is better than text editing because object changes can be checked.

For example, if a new concept such as a noise abatement climb plan is added to the plan-goal graph, its parameters may be checked for definition.

Checking can be extended into design rules, which are more global in purview than the local editing checks. For example, after a *noise abatement climb* plan was added to the graph, a design rule could detect that the *raise gear* action could not distinguish a *normal climb* from a *noise abatement climb*. (As far as we know, there is no way to distinguish them when the gear is raised.) The design rule would suggest combining the two plans into a single plan *normal climb* with an additional parameter to represent whether noise abatement procedures were being used. Design rules can aid the designer but cannot fully automate the design problem. Full automation of design requires a complete model of the domain, an undertaking at least as difficult as building an intelligent interface.

Visualization of a knowledge base would allow the designer to find limitations that could not or have not been formulated as design rules. The designer needs to be able to focus on particular aspects of knowledge in the knowledge base. For example, the knowledge engineer must at some time look at the landing gear interpretation plan goal graph from the perspective of mutual exclusion rather than interpretation. For example, in Figure 5 the mutual exclusion links are shown for various landing plans. The designer should be able to see and to understand that the four plans are mutually exclusive. A great deal of knowledge would be required for the environment to be able to infer this independently. For this reason, the designer should solve the hard problems and the environment the easy and mundane problems.

Modify Mapping of Inputs to Output. This is a more fine-grained activity than structure modification. Mapping modification is concerned with the semantic aspects of the predicates in production rules, the instantiation and matching of patterns, etc. Modification occurs frequently and can be supported in a variety of ways.

When modifying the mapping, editing support is valuable. The variables referenced in rules and their values can be checked against a data dictionary for sensibility. Design

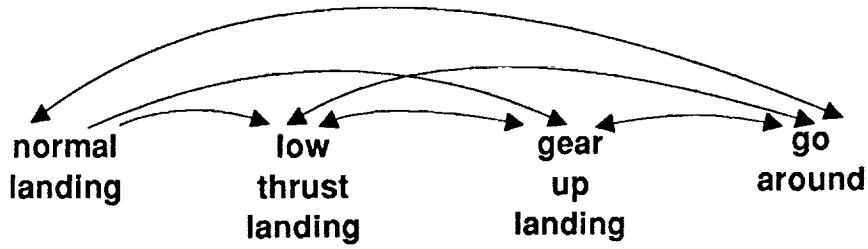



Figure 5. Mutual Exclusion Links

rules can extend checking to more global considerations. For example, symbol manipulation can show that goal X can never be reached through plan Y.

Other kinds of checking support -- overlapping rules, subsuming rules, etc. -- can also be incorporated here. Visualization of the input-output mapping is also valuable for finding errors. The most basic level of display is simply the inputs to a decision. In the context of the intent interpretation problem discussed earlier, a domain expert could be asked directly if the inputs are sufficient to make the decision. A second level of display is a decision table, a more readable way to look at the rule base.

Display of the knowledge base while it executes is another form of support. For the plan goal graph, visualizing the activation and deactivation of plans and goals during action interpretation would be a valuable form of support. Execution could be driven by test cases or data from a canned or live simulation run. By treating knowledge base execution as a process to be debugged, the types of control and inspection typically exercised by a debugger would be valuable.

Modify Inference Mechanism. Occasionally, the knowledge engineer needs to examine or even modify the code of the inference mechanism. For example, the intent interpretation knowledge base needs to be able to detect the termination of a plan or goal that is no longer being pursued by the human. One mechanism, not entirely sufficient by itself, by which this happens is the activation of new plans and goals that are mutually exclusive with (and thus delete) the ignored plans and goals. Another mechanism is required, however, because plans and goals may be abandoned simply by not advancing them rather than activating any new plans and goals to replace them.

If the new mechanism does not fit into any existing intent inference mechanism, it has to be added by modifying the code of the inference mechanism. Such changes are relatively infrequent and are as difficult to support as adding a new decision or function.

Knowledge Engineering Test Stand. Even if the knowledge engineering environment does not support a specific application, some of the principles espoused here can be applied

to all knowledge-based systems. As electronics designers learned some time ago, engineered systems should be designed to be tested. Test points are part of every circuit, and large integrated circuits incorporate extra logic to make internal states accessible. While internal state access is often for circuit repair and manufacturing quality assurance, the concept is applicable to all software.

The knowledge engineering environment, whatever it is, fills the role of a test stand or test jig for a traditional engineered system. Even if access to internal state is not permitted, the test stand should support creating inputs for the engineered object. Of course, no test stand can fully exercise the system. The idea is to exercise it as fully as possible as a unit before moving to the next, more highly integrated test.

The knowledge engineering environment also supports the concept of concurrent engineering. That is, design can be closely followed by test. The earlier problems are fixed, the less their cost to repair. If testing is easier, then more of it can be done early. A knowledge engineering environment that simply supported more rapid feedback to the knowledge engineer would result in a better end product.

Although not addressed directly in this report, the notations used for describing knowledge bases are inadequate or nonexistent. Traditional software notations can be used to describe inference engines, conflict resolution, etc. These notations do not apply well to the specific knowledge in knowledge bases. It is likely that new notations need to be developed to describe clearly the structure of specific knowledge.

Integration of Design and Evaluation

Clearly, the proposed approach to validation has important implications for design. The need to design knowledge bases for testability is an obvious implication. Equally important, however, is the desirability of integrating design and evaluation -- in the current parlance, to perform concurrent engineering of intelligent systems. The approach espoused and illustrated in this report can provide the basis for this concurrent engineering, as well as

the continuous improvement necessary to assuring software safety and avoiding unacceptable consequences of unavoidable modeling limits.

ACKNOWLEDGEMENT

This research was sponsored by the NASA Langley Research Center under Contract No. NAS1-19021. Wendell R. Ricks of NASA Langley is the technical monitor of this project. The authors gratefully acknowledge Mr. Ricks' helpful comments and suggestions as this work progressed, as well as the useful inputs of Dr. Kathy Abbott.

REFERENCES

Casti, J.L. (1989). *Alternate realities: Mathematical models of nature and man.* New York: Wiley.

Chi, M.T.H., Glaser, R., & Farr, M.J. (1988). *The nature of expertise.* Hillsdale, NJ: Erlbaum.

Glymour, C., Scheines, R., Spirtes, P., & Kelly, K. (1987). *Discovering causal structures: Artificial intelligence, philosophy of science, and statistical modeling.* Orlando, FL: Academic Press.

Newell, A. (1981). The knowledge level. *AI Magazine, 2,* 1-20.

Nguyen, T.A., Perkins, W.A., Laffey, T.J., & Pecora, D. (1987). Knowledge base verification. *AI Magazine, 8,* 69-75.

Parnas, D.L., van Schouwen, A.J., & Kwan, S.P. (1990). Evaluation of safety-critical software. *Communications of the ACM, 33,* 636-648.

Rouse, W.B. (1980). *Systems engineering models of human-machine interaction.* New York: North Holland.

Rouse, W.B. (1990). *Design for success: A human-centered approach to designing successful products and systems.* New York: Wiley.

Rouse, W.B., Geddes, N.D., & Curry, R.E. (1988). An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems. *Human-Computer Interaction, 3,* 87-122.

Rouse, W.B., Geddes, N.D., & Hammer, J.M. (1990). Computer-aided fighter pilots. *IEEE Spectrum, 27,* 38-41.

Rouse, W.B., Hammer, J.M., & Lewis, C.M. (1989). On capturing human skills and knowledge: Algorithmic approaches to model identification. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-19*, 558-573.

Rouse, W.B., & Hunt, R.M. (1982). Issues in the identification of models of human behavior. *Proceedings of the 6th IFAC Symposium on Identification and System Parameter Estimation.*

Rouse, W.B., & Morris, N.M. (1986). On looking into the black box: Prospects and limits in the search for mental models. *Psychological Bulletin, 100*, 349-363.

Rushby, J. (1988). *Quality measures and assurance for AI software* (NASA CR-4187). Hampton, VA: NASA Langley Research Center.

## APPENDIX A: DATA TRANSCRIPTS

8:11:42 Adaptive aiding issues messages saying "turned on X" and "tuned radio channel X to frequency Y." These messages should not be shown. If the action is pre-briefed to occur at a certain time or under certain conditions and it does so occur, then it should not be mentioned. If a frequency change were made, then the message would be "going AWACS" rather than frequency. The cause of this problem was that the PVI did not have a satisfactory mechanism for determining whether a change should be pointed out to the pilot. This problem had been identified before, but there was insufficient time to solve the problem.

The cause was that the decision whether to display such information was not identified early enough for it to be solved before it was implemented. Not displaying pre-briefed plans is only a part of the solution. More generally, any plan that would be strongly expected to be executed automatically would not need to be displayed. Pre-briefed plans are a special case of this.

8:11:43 Low observable (LO) cell formation was proposed -- the internal name is radar-cell-formation. The pilot did not know the meaning of "LO-cell-formation." There are two problems here. First, LO-cell-formation probably means "close formation," not radar cell formation, even though they sound the same*. Second, the intent inference function failed to determine what formation the pilot and wingman were already in.

Only a domain expert could have caught this problem. The problem could be caught during testing if the test stand could show this message appearing in a crewstation mocked up on a CRT display.

---

* LO-cell-formation would mean that the two aircraft would fly in such a way that their combined signature would appear to be a single blip. This practice prevents threat sensors from determining how many aircraft are present in the blip. A radar-cell-formation is used for heavy aircraft to fly through bad weather. (Fighter aircraft call the same formation radar trail.) The second aircraft is several miles behind the lead aircraft. The second aircraft uses radar to keep track of the lead and remain in position.

8:12:00 The center threat situation display does not completely show a threat that is right on the edge of the screen. The software needs to consider the screen to be slightly smaller than its measurements when setting the magnification/range of this display.

A good human factors critique of the crewstation might have caught this problem if the conditions were right to repeat the input conditions exactly.

8:12:36 Radar antenna center positions for pilot's aircraft and wingman were proposed as plans. See comment at 8:11:42.

8:12:55 Trail dots were on for the pilot's aircraft and wingman, which was unnecessary and extra clutter. Trail dots are displayed at the previous positions of an aircraft to give the pilot an idea of its previous movement. The mechanism that caused the error was that trail dots were controlled only for threat aircraft. The dots were turned on for the pilot's aircraft at initialization and never considered again by the software.

8:13:43 The position of the pilot's aircraft on the horizontal view needs to be controlled more fully. If there are aircraft behind the pilot's aircraft (or expectation of the same), the symbol for the pilot's aircraft needs to be moved toward the center of the screen in order to see behind (assuming these data are available). Outbound on a mission would lead to the expectation of aircraft behind the pilot's aircraft. A mission deep into enemy territory would force a selection of 150 nm range all the time. This position of the pilot's aircraft was very low on the screen during the replay. At times, the pilot could not see all of the strike or escort aircraft. Magnification (range) could be modified to handle this, but so could the relative position of the pilot's aircraft.

8:14:01 On the center threat situation display, the pilot could we can just barely see Knight 03 and 04. The perspective view does not show all of the F-15E aircraft.

8:14:36 Perspective view should show more of the future path rather than just what is directly ahead.

The cause of the previous two problems was that the model that controlled the perspective viewpoint did not receive the correct form of input from the plans. In other

words, the representation of the input to the model needs to change. The model for controlling the perspective eyepoint received eyepoint offsets as information requirements from various plans. This is the wrong space or dimension for this kind of information requirements. Instead, the information requirements should describe geometric quantities--such as range, bearing, and altitude differences of particular threats and friendlies--that are needed for the active plans. The model would then compute a perspective viewpoint that best showed these quantities. In the case of the second problem, the information requirement should be to show a certain amount of the future flight path on the display.

8:17:06 Some SAM sites have just appeared on the tactical display. The type of SAM is not displayed but should be. The route of the pilot's aircraft was to pass close to these sites. As soon as a threat shows up, it should be identified for the pilot if possible, assuming the pilot is not busy doing something else.

This information is needed because of a display-oriented event -- the appearance of SAM sites on the edge of the screen. This required output was not planned for in the design. The PVI would display the type of threat but only if the SAM site did something, such as emit, or if the pilot's aircraft passed fairly close to the SAM site.

8:20:37 Perspective view does not show future route or any cooperating flights. See 8:14:01.

8:21:15 Numerous SAM sites appear just over the forward edge of the battle area. They should be identified with tags but are not. See 8:17:06.

8:21:16 Radar volume change is announced but it should not be because it is already visible on the screen. See 8:11:42. Despite the pilot's comments, mere visibility on the displays is not reason enough to announce.

8:22:43 Interchange of flight management and sensors displays on the left hand CRT. The left CRT may display two system displays side by side and one menu beneath. The software will sometimes interchange the displays. The perceptual phenomena (i.e., flashing) causing pilots to object to this change is not part of the model that makes this decision.

The cause is that moving displays between the two areas is done by the program, but during design there was no thought that there must be a conscious decision to move the displays.

8:32:35 Center display switches to 200 nm range. This behavior did not make sense to the pilot. The reason it was done was that there are no nearby threats. Consequently, the range goes to maximum. One might interpret showing this much area on the screen as an error if there was no sensor coverage of the area. For example, when in friendly territory, the AWACS sensor data is provided by datalink to the pilot's aircraft. When in enemy territory, the AWACS coverage is minimal or non-existent. To show 200 mile range when there is no coverage beyond 100 miles could be considered misleading.

8:37:00 Two labels are displayed over slightly offset threats, making reading impossible. The PVI was informed that both threats were in the same group. The PVI should have put up only a single label but it did not consider threat grouping when asking for labels. Another possible interpretation is that the positions of the threats on the screen should be considered when labels are requested. If the labels overlap, they are at best difficult to read.

8:37:08 Center CRT display switches to 50 nm range to view anti-radiation missile (ARM) engagement. This caused the pilot's aircraft to lose view of the attack and the status of its own AMRAAM missile, which is extremely important. This situation is not dominated by defensive concerns alone. Basically, the pilot's aircraft has to wait until AMRAAMs switch to active self-guidance before radar can be turned off (which will defeat the ARM). The model assumes no interaction between offensive and defensive concerns.

8:37:09 Right hand CRT goes to side view to better show missile engagement. However, since the anti-radiation missile is an air-to-air, the side view does not show very much. The geometric plane of the attack is just a line when viewed from the side. A horizontal view would be best, provided the attack was pretty much planar.

The cause is similar to that of 8:14:01 and 8:14:36. Event-based display selection needs to reason with geometric quantities. Existing reasoning only worked for surface-to-air missiles, not air to air missiles.

8:38:10 Center CRT on 200 nm range. This has the same interpretation as 8:32:35.

8:38:30 Right CRT on 400 nm range and horizontal view to show new mission route to get pilot's aircraft home. Unfortunately, 400 nm is not really right. The range that is right needs to be computed from the data in the route itself. Also, the route only needs to be seen to the forward edge of the battle, not all the way home. One might want to show the whole route if the pilot's aircraft was to land at a base other than the one initially planned. The cause of this problem is that the model needs to consider geometry, including numerical values, not just symbols.

NASA
National Aeronautics and Space Administration

# Report Documentation Page

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| NASA CR-4336 | | |

**4. Title and Subtitle**

Assessing the Impact of Modeling Limits on Intelligent Systems

**5. Report Date**

November 1990

**6. Performing Organization Code**

**7. Author(s)**

William B. Rouse and John M. Hammer

**8. Performing Organization Report No.**

**10. Work Unit No.**

324-01-00

**9. Performing Organization Name and Address**

Search Technology, Inc.
4725 Peachtree Corners Circle
Norcross, Georgia 30092

**11. Contract or Grant No.**

NAS1-19021

**13. Type of Report and Period Covered**

Contractor Report

**12. Sponsoring Agency Name and Address**

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

**14. Sponsoring Agency Code**

**15. Supplementary Notes**

Langley Technical Monitor: Wendell R. Ricks
Final Report - SBIR Phase I

**16. Abstract**

This report is concerned with validating the knowledge bases underlying intelligent systems. A general conceptual framework is provided for considering the roles in intelligent systems of models of physical, behavioral, and operational phenomena. A methodology is described for identifying limits in particular intelligent systems, and the use of the methodology is illustrated via an experimental evaluation of the pilot-vehicle interface within the Pilot's Associate. The requirements and functionality are outlined for a computer-based knowledge engineering environment which would embody the approach advocated and illustrated in earlier discussions. Issues considered include the specific benefits of this functionality, the potential breadth of applicability, and technical feasibility.

| 17. Key Words (Suggested by Author(s)) | 18. Distribution Statement |
|---|---|
| Artificial intelligence<br>Functional validation<br>Concept graphs<br>Pilot-vehicle interface | Unclassified - Unlimited<br>Subject Category - 59 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 52 | A04 |



For sale by the National Technical Information Service, Springfield, Virginia 22161-2171