

AD-A231 347

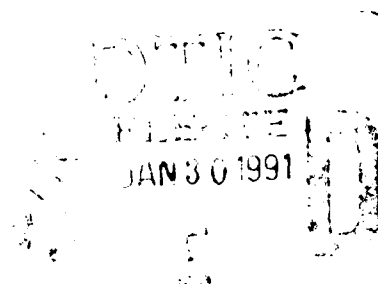
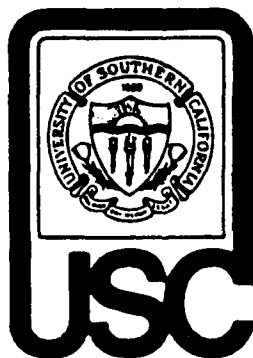
*ONR Final Report
Cognitive Congruence
in Intelligent Simulation Training*

Douglas M. Towne
Allen Munro

Behavioral Technology Laboratories
University of Southern California

Sponsored by
Office of Naval Research
Cognitive Science Program
and
Navy Personnel Research and Development Center

Under ONR Contract No. N00014-87-C-0489



Approved for Public Release: Distribution Unlimited
Reproduction in Whole or in Part is permitted for any purpose of the United States Government

Final Report
Contract N00014-87-C-0489

*Cognitive Congruence
in Intelligent Simulation Training*

Douglas M. Towne
Allen Munro

Technical Report No. 114

Behavioral Technology Laboratories
University of Southern California
250 No. Harbor Drive, Suite 309
Redondo Beach, CA 90277

(213) 379-0844

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 1990	3. REPORT TYPE AND DATES COVERED FINAL TECHNICAL 06/01/87 to 08/31/90	
4. TITLE AND SUBTITLE COGNITIVE CONGRUENCE IN INTELLIGENT SIMULATION TRAINING			5. FUNDING NUMBERS Contract No. N00014-87-C-0489-	
6. AUTHOR(S) DOUGLAS M. TOWNE ALLEN MUNRO				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Southern California Behavioral Technology Laboratories			8. PERFORMING ORGANIZATION REPORT NUMBER Technical Report No. 114	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Cognitive Science Program and Navy Personnel Research and Development Center: San Diego CA 92152 AFHRL			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release: Distribution Unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The feasibility of generating cause and effect explanations in text and graphics is explored for simulation training systems based on device models. Practical and fundamental problems in generating such explanations are explored. Approaches to such explanations based on modeled effects are described. Explanation capabilities include demonstrating the consequences of actions, graphically portraying object architecture and attribute architecture, presenting the initiating condition for an effect, and presenting a theory of operation. Techniques for automatically representing a system at a level of complexity appropriate for an individual learner are described.				
14. SUBJECT TERMS SIMULATION TRAINING EXPLANATION			15. NUMBER OF PAGES 44	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

ABSTRACT

The feasibility of generating cause and effect explanations in text and graphics is explored for simulation training systems based on device models. Practical and fundamental problems in generating such explanations are explored. Approaches to such explanations based on modeled effects are described. Explanation capabilities include demonstrating the consequences of actions, graphically portraying object architecture and attribute architecture, presenting the initiating condition for an effect, and presenting a theory of operation.

Techniques for automatically representing a system at a level of complexity appropriate for an individual learner are described.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

ACKNOWLEDGEMENTS

The work described here was supported by the Office of Naval Research, the Navy Personnel Research and Development Center, and the Air Force Human Resources Laboratory, under ONR Contract No. N00014-87-C-0489.

TABLE OF CONTENTS

Part I. Generating Cause and Effect Explanations in Text and Graphics: Feasibility 1

The Utilities of Explanations	1
State of the Art	2
The Process of Explanation	3
Problems	4
Practical Problems	4
Basic Problems in Explaining Causality	7
Basic Issues in Generating Explanations	7
Understanding the Question — Causal Level	8
Understanding the Question — Causal Remoteness	8
Completeness in Explanation	8
A Principle of Parsimony	9
The Elements of Explanation	10
An Explanation Hypothesis	10
Combining Words with Pictures	10
Feasibility of Generating Cause-Effect Explanations	12
General Purpose Expert System Shells	12
Device Models	13
Types of Technical Explanations	15
Effects of Actions and Conditions	15
Alternatives for Observing Effects	17
Explanations About Device Architecture	17
Depicting the Attribute Architecture	18
Depicting the Object Architecture	18
Explaining Causation of Effects	21
Time Issues	22
Explaining Theory of Operation	23
Summary and Conclusions, Part I	23

Part II. Generating Device Representations of Varying Complexity 25

Background	25
Scope	25
Generating Models of Varying Complexity	26
Selection According to Specific Object Specifications	27
Selection According to Generic Object Specification	28
Selection by Analysis of Device Behavior Rules	29
Reconstituting the Graphical Representation	31
Text-based Explanations of Functionality	31

References 34

List of Figures

Part I. Generating Cause and Effect Explanations in Text and Graphics: Feasibility

- Figure 1. *Profile* Rationalization of a Recommended Test 2
- Figure 2. Assessment of Student's Test 3
- Figure 3. *Profile's* Recommended Strategy 3
- Figure 4. An Authored Explanation 11
- Figure 5. RAPIDS Display of Effects 14
- Figure 6. A More Complex Diagram of Effects 15
- Figure 7. A Block Diagram of a Simple System 19
- Figure 8. A Simple system with *Potential* Causal Relationships 20
- Figure 9. A Simple Discrete-State System 22
- Figure 10. A Digital Decoder of an Analog Value 23

Part II. Generating Device Representations of Varying Complexity

- Figure 1. Simplified Bladefold 26
- Figure 2. Fluid Flow control System 30
- Figure 3. Representation Formed Based on Proximity of Connectivity 30
- Figure 4. Representation Formed Based Upon Functionality 31
- Figure 5. Generic Slider Control 32
- Figure 6. Flow Control System 32

Final Report

Cognitive Congruence in Intelligent Simulation Training

Part I

Generating Cause and Effect Explanations in Text and Graphics: Feasibility

Douglas M. Towne
Allen Munro

This is an exploratory paper that considers the issues associated with developing automated processes for producing technical explanations. It outlines

- the kinds of questions that arise during learning of technical matter, particularly as related to learning device operation;
- the alternative ways in which technical questions might be interpreted;
- the manner in which human instructors explain technical material; and
- the feasibility of responding to questions in an instructionally equivalent manner via generic automated processes.

Automated systems that support human beings in learning or performing complex tasks would be most useful if they could *generate* explanations to accompany the individualized instruction that they provide. This is especially true in simulation-based environments, where the learner has the opportunity of attempting to perform complex tasks, as opposed to interacting with the instructional system about the subject in a less direct manner. The experience of operating upon a device simulation may evoke questions about unexpected effects, about the requirements to achieve desired effects, or about the functions or architecture of the device.

Because the variations in performance among individual learners operating complex systems are immense, it would be impractical to require that instructional authors anticipate all the device conditions produced by the learners or all the questions about those conditions that will arise. Thus it is imperative that tutoring systems possess generic intelligence capable of formulating explanations in an ad hoc manner.

In the past, explanations provided by automated instructional systems have been largely restricted to verbal forms, either spoken or written. Now, with the availability of device simulations in the form of high-resolution computer graphics, the term 'explanation' can be broadened to include any mixture of verbal and pictorial (and sometimes audio) forms, including dynamic images.

The Utilities of Explanations

The primary intent of providing explanations is to enhance understanding so that the learner can accomplish job-related goals more effectively. Some of the purposes that explanations serve include these:

- an explanation may be crucial to understanding and performing the recommended actions;

- an explanation may lead to greater learning of the associated procedure or situation-response knowledge;
- an explanation can be of use to the technician if the recommended activity is not successful - it may allow the technician to make minor modifications to the recommended procedures, or to generate other good courses of action;
- the user is more likely to value and follow the recommendations of the support system if the system also supplies rational explanations (Swartout, 1983).

The challenge facing developers of intelligent tutoring systems is to generate meaningful explanations that are highly context dependent, i.e., the explanations relate to the specific device conditions produced by the individual learner.

State of the Art

While progress has been made in recent years in producing experimental systems that exhibit a modest ability to explain (Brown, Burton, & deKleer, 1982; Clancey, 1979; Feurzeig, Ash, & Ricard, 1984; Bonar & Cunningham, 1988; Lesgold, Lajoie, Bunzo, & Eggan, 1988), no existing automated system extracts functional phenomena in a general way appropriate for a wide range of devices, learners, and types of question. The experimental systems are typically termed "intelligent", because they embrace some ability to interact gracefully and appropriately with an individual learner, yet the explanation *generation* capabilities they possess is generally quite limited.

One system, IMTS (Towne, Munro, Pizzini, & Surmon, 1987), does have the ability to generate situationally sensitive explanations in the domain of fault diagnosis. Here a generic rationalization capability is possible because the domain is restricted to a relatively well-defined task. While the range of device functionalities and architectures may vary greatly, the underlying structure of diagnostic reasoning can be applied to the spectrum of specific devices, and can be explained in specific applications. Thus, IMTS can 1) recommend a good test to perform next, following any sequence of learner-selected tests, 2) explain how to perform the recommended test, in terms of the device conditions established by the learner, and 3) explain why the recommended test is most effective, in terms of knowledge gained by past actions, and the future inferences that could be made from each possible outcome of the recommended test. Figure 1 shows an example.

```

File Edit Window

The best test to do now is to look at
SafetyValveLight
To perform this test, you must:
Set the following switches:
  MainPowerSwitch
  SafetyValveSwitch

A value of On would be normal for SafetyValveLight

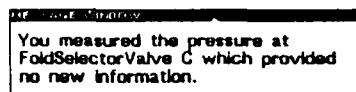
If Off (abnormal), the failure is one of:
  SafetyValveSwitch
  CB80

If normal, then the failure can't be any of the above; we still
suspect:
  AccessoryDriveSwitch
  MasterSwitch
  K107D-SafetyValveInterlock
  CB87
  K107-Coll
  K112-Coll
  K108D
  K112C-BladeFoldInterlock

```

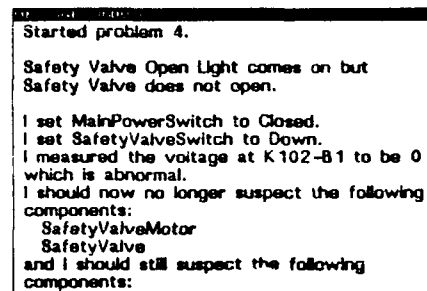
Figure 1. *Profile Rationalization of a Recommended Test*

Using similar logic, IMTS can assess a learner's diagnostic approach (Figure 2), and it can generate a complete testing sequence that exhibits expert diagnostic reasoning (Figure 3).



You measured the pressure at FoldSelectorValve C which provided no new information.

Figure 2. Assessment of Student's Test



Started problem 4.
 Safety Valve Open Light comes on but Safety Valve does not open.
 I set MainPowerSwitch to Closed.
 I set SafetyValveSwitch to Down.
 I measured the voltage at K102-B1 to be 0 which is abnormal.
 I should now no longer suspect the following components:
 SafetyValveMotor
 SafetyValve
 and I should still suspect the following components:

Figure 3. Profile's Recommended Strategy

Of course, a learner may want to respond to an explanation by asking "Why?" again. For example, if IMTS explains that a test is effective because it discriminates among two likely hypotheses, and it explains what symptoms would be observed under each condition, a learner might wonder why the cited symptoms would occur. At this point the question concerns how the device operates, and how various failures would affect that functionality, rather than how one implements a diagnostic reasoning process.

At present, IMTS has no capability to continue the dialogue along these lines. Similarly it cannot generate assistance that will overcome a learner's particular problems in achieving some desired device state, and it cannot explain why some unexpected results occurred. There appears to be, however, considerable potential in the underlying IMTS device model for doing just these things.

The Process of Explanation

The word *explanation* implies the existence of prior partial knowledge on the part of the learner, concerning, for example, some observed effect or happening that is not fully understood. This is opposed to a *description* which supplies a body of information that is not necessarily associated tightly with pre-existing knowledge. Thus we can *describe* Custer's last stand, as it may have looked or sounded, or we can *explain* various aspects of the event to someone who does not understand its ramifications.

An important aspect of the information conveyed in an *explanation*, as distinct from a description or other forms of pedagogical discourse, is that it deals with *causality*. Explanations convey the causes of things. Different types of causes may be found in different explanation, including proximal physical causes, temporally or physically remote enabling causes, and, in some cases involving volitional agents, rationales or motivational causes. (See Shank, 1972, 1975, for a discussion of the semantics of causality in natural language.)

Effective instructors typically employ a diverse combination of media and actions to convey explanations of technical matters. The instructional media may include the spoken or written word, the actual device, two-dimensional diagrams of the device, three-

dimensional models of the device, and sometimes motion pictures or video images, possibly accompanied with sound. The instructor may employ these resources according to an explicit course plan, or in an *ad hoc* manner to address an immediate problem or question.

While the word "explanation" typically suggests a highly verbal form in our culture, other cultures may minimize or virtually forego the verbal component of instruction and rely almost exclusively upon demonstration and apprenticeship. Also, with our current technology, some portions of verbal components of explanations may be replaced with non-verbal forms.

In addition to acquiring and presenting the instructional media, the good instructor typically performs actions that bring these media to the learner in the most effective manner. These actions may include 1) positioning and configuring the media and the learner to maximize experiencing the crucial events, 2) focussing the learners' attention upon the time or locale of significant events, and 3) demonstrating system behaviors using the real equipment or a model thereof, or acting out aspects of the device operation.

The positioning and configuring actions taken by the instructor are generally basic and obvious. These actions allow the learners to see, hear, or feel the device or its representation as critical events occur.

The focussing actions attempt to direct the learner's concentration to the time or place where crucial events occur. Location is often indicated by pointing to crucial parts of the device or the representation. The time of important events are often signified by voice, as, for example, saying "See the gear engage the latch . . . now."

When possible, the device behaviors of interest are demonstrated. When that is not practical, the instructor may resort to acting actions in an attempt to emulate aspects of the device with movement of the body, arms, and hands. This body language is very common in technical instruction, and it is usually accompanied by a verbal account, such as "Next the locking pins engage (the instructor engages his fingers), the circuit interlock closes (the instructor makes a closing gesture), and the motor begins to turn (the instructor makes a turning motion). These hand and body gestures are deeply integrated into common instruction and may provide stimuli that strengthen learning and recall in a significant manner.

Finally, an effective technical explanation often includes a discussion of those unseen events that occurred or phenomena that were involved in the process but were not observed. As before, the instructor may employ diverse media and actions to convey these unseen effects.

Problems

Two kinds of problems confront the learner of difficult technical matter, 1) limitations in his or her access to the expertise and the device to be learned, and 2) difficulties in communicating questions and in interpreting answers. The former of these is a practical issue, the latter is a difficult basic issue.

Practical Problems

One of the most common deficiencies in technical training is limited access to instructional resources that can effectively explain complex technical effects. The deficit includes a lack of appropriate graphical representations that show the interactions of device parts over time. At best, standard technical documentation, including that used for instruction, provides a static view of the system in one or several important states. Rarely

does the instructor have access to all the necessary figures to represent all the states of interest of a system as it responds to inputs over time.

If the real device is available and amenable to direct observation, especially at controlled speeds that allow inspection, then it may serve to illustrate interesting intermediate states. More typically the instructor must fabricate diagrams (often at a chalk board) that convey the changes over time. Such activities may consume considerable amounts of student and instructor time as they are repeated class after class.

Another approach to providing explanatory presentations is to create films or videos that demonstrate the effects of interest. These animated pictorials could offer presentation modes that would not be possible with real equipment, such as displays of cut-away modules carrying out their functions normally, slow motion presentations, and so on. There are two problems with this class of pictorial solutions to the explanation problem. The first is that the expense of developing film or video presentations is so high. Such presentations cost too much to be justified except for use in high-volume training contexts or where the economic or social costs of inadequate training clearly outweigh the expense of such training. The second problem with film/ video explanations in interactive training is that they are typically not sensitive to the pedagogical context. That is, such explanations, which are developed in advance of need, must be somewhat generic in nature. They are not customized for the requirements of the individual student who requires a particular explanation at a particular point during his or her training.

A related practical limitation on explanations in training has to do with the availability of representations that are of just the right level of complexity for a student's ever-changing capabilities. While most instructors have access to some simplified drawings and some complete drawings, there are rarely sufficient intermediate representations appropriate for the student who is familiar with the basic configuration of a system, but has not yet fully grasped the working of the many parts.

One of the most fundamental practical limitations to explanation-based training is sufficient instructor time and device access to provide individualized attention to each learner. Opportunities for individual students to practice, to encounter areas of confusion or surprise, and to obtain answers to their exact questions are often limited by the paucity of explanation resources.

Computer technology, with associated high-resolution graphic displays, offers great promise in addressing these difficult instructional requirements. Certainly today's graphic displays offer adequate size, color, and resolution to convey complex and dynamic phenomena. The most serious limitations are not in the quality of display or the price of the hardware. Instead, they result from difficulties in
generating the wealth of presentation graphics required,
storing these presentations,
retrieving appropriate presentations in the pedagogical context, and
providing associated technical explanations that are appropriate to the needs of the
individual learner.

At the heart of the problem is the immense number of system states and object interactions that could occur, depending upon the initial state of the system and the actions that are performed upon it. For most real devices or systems of even moderate complexity, it is completely impractical to prepare hardcopy diagrams in anticipation of the explanation requirements that could arise.

The same problem confronts simulation-based training systems. Classical frame-based instructional systems avoid the problem by limiting the student's course of action to a fixed sequence — or to selections from a number of simple 'branches, each of which is

itself a fixed sequence. In these systems the learner is not free to perform in a realistic fashion.

Some training systems (e.g., Fault) avoid the problem of dealing with this complex world by simply not displaying the inner workings of the device. Instead the device is represented in block diagram form or in a form that does not respond to the learner's actions. The student selects desired tests and modes from written lists, and observes the test result either via text or some graphic. While some aspects of device knowledge can be conveyed in such an instructional environment (specifically, relationships between modes and test outcomes), clearly the student cannot gain a deep understanding about the device from such presentations.

An alternative to avoiding the representation problem is to provide graphics for some subset of the possible device states, as was done in GMTS (Towne & Munro, 1981). While this approach was relatively successful in portraying a large number of the meaningful front-panel states of complex devices, it could not effectively address representing the inner structure and behavior of such devices. The graphic display in GMTS presented complete photographic images (accessed either by a random-access microfiche projector or by a videodisc player). This meant that the elements in an image — the individual controls and indicators, for example — could not be manipulated independently. It was possible to display a large number of symptom patterns in GMTS only because the strategy adopted was to display only a few elements at any one time. Restricting the number of variable elements that could appear in any one image helped to prevent the combinatorial explosion of images. This approach, the only one that was feasible at the time of development, is inconvenient when the learner must understand the interactions of many parts operating in concert.

Significant progress has been made in representing any of a huge number of possible device configurations, first with the development of ESAS (Towne and Munro, 1984) and later IMTS/RAPIDS. The ESAS system employed an underlying data base virtually identical to that of GMTS, but it used object-oriented computer graphics rather than fixed pictorial media (e.g., videodisc) to represent the device. As a result, a front panel could be presented in exactly its proper condition, depending upon the settings of the various switches and any malfunction condition being simulated, and it could be shown in its entirety rather than in restricted views of individual display elements.

With the development of IMTS, the graphical objects acquired a significant measure of intelligence. Each object in a specific IMTS model computes its graphical appearance and functional behavior by employing rules appropriate for the object class to which it belongs. (Graphics and rules are stored with each generic object prototype). When stimulated with inputs from other such objects in a specific modelled device, the IMTS object executes its generic behavior rules to determine how it should appear and what outputs it will send to neighboring objects. Because IMTS objects can represent virtually any physical or functional entity, some representations can be constructed to reflect the physical attributes of the device while other representations portray the inner workings in schematic form.

The most recent advance in this technology has been to completely recast the deep device-modelling environment of IMTS to allow for continuous, real-time changes in objects, rather than instantaneous changes from one static state to another. This simulation environment, supported by a special-purpose expert system shell and associated inference engine called RAPIDS-II, also offers the simulation developer virtually complete freedom in choosing the level to which any part of a device model will be specified. Thus some portions of a model may be driven by highly detailed component models interacting in a manner that can be observed by the student, while other subsystems can be modelled at a very high level so that they support the total simulation, but require minimal development effort. RAPIDS-II makes it possible for simulation effects to propagate naturally from one level of representation to another.

Basic Problems in Explaining Causality

Systems like IMTS and RAPIDS-II have the potential to resolve many of the practical issues of limited access to real equipment and to expert instructors. We turn now to more basic problems related to explicating cause and effect.

Volumes have been written about the nature of and intriguing mysteries concerning physical causality. In modern quantum mechanics the obvious truths that seem to hold in the scale of time and space relevant to human activity have been called into question. We will not deal with these puzzles. Two important points about causality are made here, however. First, natural language is extremely imprecise with respect to issues of causality, and this results in a high rate of error in human reasoning about causality. Therefore, great care must be taken to specify what we mean by "A causes B". Second, the treatment of causality with respect to explanation in technical training cannot be naively derived from characteristics of the propositional calculus (which employs formulas such as if p then q , or $p \rightarrow q$). The propositional calculus is *extensional*, i.e., "there is not asserted any causal relation between the sentences connected by the conditional operator" (Dougherty and Giardina, 1988).

What do we mean if we say that event A causes effect B? Immediately the ambiguity of our language causes difficulties, for a number of differing interpretations could be offered. Of course there is no ultimate correct interpretation; the statement means whatever we want it to mean. We will begin by exploring some of the subtle issues that are unstated.

Exclusivity of Cause and of Effect. The English statement "event A causes effect B" is ambiguous partly because it doesn't state if event A is the only one that can cause effect B to occur, and we don't know, from the statement, whether there might be other effects of A. Legal documents attempt to resolve such ambiguity by saying "event A, and only event A, will result in effect B and only in effect B".

Directness. A second common ambiguity clouding cause and effect discussions has to do with whether an effect is a direct result of an event or a subsequent, indirect, result. As before, verbal responses about effect can be ambiguous if nothing is stated about the directness of the result. Diagrammatic forms of cause-effect expositions would tend to avoid confusion of this type, whereas approaches involving graphical highlighting of the device model could lead to confusion concerning directness.

Enablers versus Processors. Some parts of devices act as facilitators; they enable or disable other parts. Sometimes the enabling part plays no role in the output produced by the enabled part; it simply controls whether the part will function or not. In other cases the enabling part is partially or completely responsible for the quality or quantity of the output. Thus, there are multiple ways in which one part can be said to "cause" an output, and these differences must be recognized to avoid confusion.

Basic Issues in Generating Explanations

When expert instructors produce explanations they take into account a host of factors related to the nature of the question, the characteristics of the individual learner, and the situation that elicited the explanation. Two expert instructors would be expected to produce somewhat different explanations to the same question in the same situation, owing to their different impressions of the problem. The effects of these differences may be largely mitigated in a normal instructional context by the fact that the communication is usually two-way. In a tutorial context, for example, the learner ordinarily has the opportunity and willingness to participate in the discussion by indicating if the instruction is addressing his or her problems. Additionally, different explanations from different instructors may

address the issues quite satisfactorily. For example, there may be several equivalent approaches to an explanation, all of which provide the necessary information.

This section considers some of the factors that affect the nature of the explanation provided. We have identified the following *issues* for consideration:

- The causal *level* that is being queried
- The *remoteness* of the cause that is being queried
- Economy of explanation
- Functional completeness

We then sketch a three strategies that might be incorporated in an explanation delivery system:

- The primitive elements of an explanation
- A top-down explanation strategy
- Combining words with pictures

Understanding the Question — Causal Level

If we ask "Why did the CRT go blank?" we might be seeking a response in terms of electrons and phosphorescence, or we might be looking for an explanation at a much higher level. We may be wondering about the system architecture that caused the observed effect, or we may be asking what we did to cause it.

Here are three possible interpretations of "Why did the screen go blank?"

1. What did I do that made the screen go blank?
2. After I pressed the CLEAR key, what internal functions came into play?
3. How do the screen clearing functions work?

Usually a human instructor understands what is being asked, using his or her knowledge of the individual, the course prerequisites, and so on. We might be surprised, however, to find the frequency with which the instructor misunderstands the question. Clearly, we cannot currently contemplate an explanation facility that responds to free form questions. Instead some repertoire of explanation types must be developed and selected by the learner.

Understanding the Question — Causal Remoteness

Sometimes it is difficult to tell how proximal a cause is being queried. In the section immediately above, we considered the issue of determining how deep a causal relationship should be described in an explanation. A separate issue is how remote a cause should be described.

If a number of mechanical rods are joined, so that when someone pushes on the first one, the last one pushes a ball, the proximal cause of the last rod's pushing the ball is that the rod immediately before pushed it. In many cases, however, this would *not* be a satisfactory answer to the question 'Why did rod *n* push the ball off the edge?'

Completeness in Explanation

Occasionally when we ask for an explanation, we want to know all the facts. A judge might instruct a witness to tell absolutely everything seen or done in the minutes preceding a crime that was observed. But in the technical world we rarely want or need a complete explanation.

Generally, a complete explanation, covering all levels of phenomena is wasteful of resources. A complete technical explanation would continue to decompose the components

of the answer down until we reach some limit of understanding, such as quantum mechanics. We all know some individuals who provide more answer than we typically want, and occasionally any instructor will respond at a level other than what may be needed. Thus, it is sometimes the learner's task to interrupt the instructor, if socially acceptable, and clarify the question. The instructor may ignore the student's clarification, realizing that the answer must be at a different level than requested, or the instructor may attempt to reform the explanation to meet the student's expressed needs. In any case, explanations are almost always geared to address an appropriate level, rather than being complete. Again, we must look to the student to request the explanation type that is appropriate to his or her needs. The explanation resource must, however, be designed to facilitate this process.

Now, what is a complete explanation of what an element does? We suggest that a functionally complete explanation is one which allows the learner to predict the output of the element, given any input in the future. Clearly this explanation, which we will call *f-complete*, does not tell everything there is to tell about the element. It says nothing about *how* the element does its job, but it does say everything about what the element does. As elaborated below, we can explain how the element functions by supplying a functionally complete explanation of all the parts.

Examples:

Suppose we wish to fully explain an element that has the function of accepting six bits and shifting them left. We need to explain that -

1. all digits are moved left one position
2. a 0 always goes into the least significant bit (LSB) at the right
3. digits shifted beyond the sixth place are lost.

Now we can predict the output of all inputs, such as these:

000000 gives 000000
111111 gives 111110
101010 gives 010100
010101 gives 101010
111000 gives 110000

Suppose we have a device that outputs 10 if its input is above 2, else it outputs 0. We now have an *f-complete* explanation, for we can generate the output for any input. Note that, for this device, we cannot guess the input that produced the output. This is a characteristic of the device.

Having defined functional completeness, we must ask if providing a functionally complete explanation of all elements in a device provides a good explanation of the device. We hypothesize that if the element is well-characterized, that is, the elements used to represent it are appropriately chosen, then a functionally complete explanation will also be an effective explanation.

A Principle of Parsimony

One reasonably effective strategy in conventional instruction is to provide the least detailed response that seems to answer the question, then respond to any further follow-up questions as necessary. The advantage of this strategy is that minimum time is invested in delivering the initial response, one that may be found to be at an inappropriate level. This can be an irritating approach, however, if taken to an extreme. If the learner asks how the screen clearing functions work, and we respond by saying that they send a screen clearing pulse to the screen, very little new information has been supplied. It is almost certain that the learner wanted to know more than this. There is probably some sense of the minimum amount of new information in a response that is understood between the parties in a learning environment.

The Elements of Explanation

Some journalists rely on a simple checklist (what, who, when, where, why, and how) to determine if an account is complete. For a technical explanation of a past event, the relevant questions are:

- *why* did the event occur? (what was its initial cause, or trigger?)
- *what* happened as a result of the initial trigger?
- *how* did the effects of the trigger propagate to produce the final effect?
- *who* produced the triggering event? (not always relevant in technical discourse)
- *when* did the initial cause, intervening effects, and final effect occur?
- *where* did the triggering and subsequent events occur?

A technical explanation facility should be able to address all of these question types, as well as one additional question type that is often significant: "Why *didn't* some event occur as a result of some performed actions?"

An Explanation Hypothesis — Top-Down Exposition of *What*

Suppose we wish to explain how a system, composed of *n* submodules, works, at its highest level. That is, we wish to offer an explanation of some effect that can be observed at this top level. It appears sufficient to meet this goal by explaining *what* each of the *n* submodules does, but not *how*. Of course, we may explain how each of the *n* submodules operates, but then we are providing a second level explanation. To explain how one of the *n* submodules works, we again explain *what* each of the components in the submodule does. This process can be continued to any level necessary and possible. This approach seems to meet the parsimony principle without producing responses that are so stripped of content that they will be insufficient much of the time.

Suppose the learner knows what each of the sub-elements in a system does, *generically*. Then, we can explain the system by explaining what each element does *in this system*. If the learner does not know what each sub-element does generically, then we could first supply that information. However, now the learner must hold in memory the newly-learned information while attempting to understand the behavior of the device.

Clearly, it is not a sufficient explanation, at a level, to tell what each part does out of context. The explanation must make clear what the elements do in the particular device. Of course, one who knows what each element in a device is capable of doing could possibly figure out what the device does, but this is not certain nor is it ordinarily easy.

Combining Words with Pictures

While RAPIDS simulations represent powerful tools for conveying crucial technical information, they cannot be expected to be employed to their fullest potential when used solely as substitutes for the real device, i.e., when used without any accompanying instructional elaboration. While some highly motivated and self-directed individuals might require no other support than a fully functional model of the device to answer certain questions or to resolve some types of confusion, it is clear that 1) some students do not function effectively in this self-directed learning style (Chi, Bassok, Lewis, Reimann, & Glaser, 1987), and 2) even the most self-directed of students may require external direction and support to acquire certain aspects of technical knowledge.

This is not to downplay the potential of discovery worlds that can be explored by the adventurous and partially knowledgeable learner. Rather it is a recognition that discovery learning cannot support all instructional requirements.

To take the fullest possible advantage of the RAPIDS simulation, Behavioral Technology Laboratories has developed tools that allow the technical expert to convey his or her knowledge in an explicit, pre-planned, manner, using the simulated device as the primary vehicle for presentation (graphics from other sources can also be employed). These tools, also part of the RAPIDS system, allow the technical expert to demonstrate procedures, to quickly produce and explain important device behaviors (Figure 4), and to create a wide range of instructional scenarios that expand the learner's skill and knowledge.

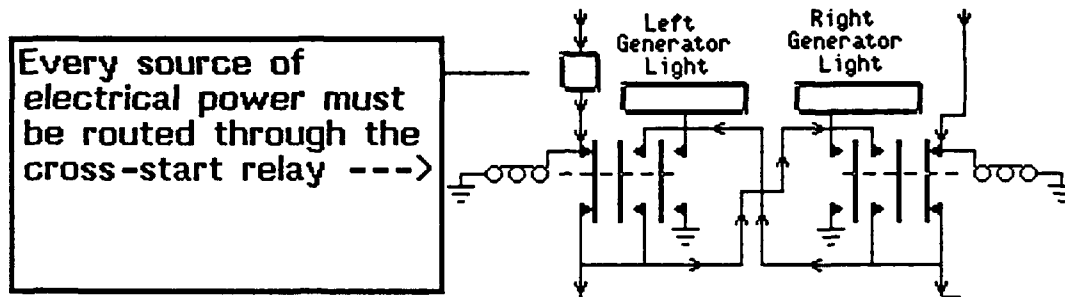


Figure 4. An Authored Explanation

The explanations provided can be both verbal and graphical. The graphical support includes bringing the appropriate simulation scenes to the screen, updating the simulation to reflect the results of various actions and conditions, highlighting pertinent areas to focus the learner's attention, and positioning verbal explanations in close proximity to the relevant portions of the simulation.

The important distinction between authoring RAPIDS instruction and authoring conventional frame-based instruction is that with RAPIDS the technical expert devotes a very high percentage of his or her time conveying the technical knowledge about the device as opposed to producing the graphic displays that accompany each situation, managing the delivery of the technical material, handling student-computer interactions, recording student performance, and managing resources such as time, all of which are handled automatically by RAPIDS.

With the advent of dynamic and interactive simulations of devices, there is no longer a clear distinction between a device representation and an explanation. Illustrations are provided in technical manuals to support the textual accounts. Together they constitute explanations, and considerable research has been devoted to the manner in which these static figures are used to better understand the text (Hegarty & Just, 1987). The question now concerns how responsive, dynamic, and soon full-color simulations are best supported with additional technical content.

Dynamic simulation with accompanying instructional material as provided by RAPIDS address a significant portion of the requirements for producing and delivering technical explanations: 1) they satisfy the need to orient the learner in relation to the device representation (by displaying the appropriate sections of the device for easy viewing), 2) they focus the learner's attention on the critical sections of the simulation, via graphical highlighting, and 3) they provide detailed simulations of device responses and functionality.

Even with RAPIDS the learner is not currently able to inquire into the causes and effects of system behaviors that were not sufficiently explained by the authored instruction. Thus we need to extend the power of the underlying RAPIDS model and simulation tools to generate both verbal and graphical explanations in response to individual requests.

Feasibility of Generating Individualized Cause-Effect Explanations

Two types of system have potential for producing intelligent explanations: 1) classical (general-purpose) expert system shells, and 2) tutoring functions that operate upon device models.

General-purpose Expert System Shells

At considerable expense, a general-purpose expert system shell can be developed with a large inventory of expert rules, thereby causing it to appear intelligent about certain issues. Typically such expert systems express recommended actions under various conditions. One problem with such formulations is in verifying that all conditions that will be encountered have been covered within the rule base. As a practical matter, the labor involved in producing such expert systems is typically so substantial, that few of them even attempt to handle all conditions that might arise.

In practice, conventional expert systems run into serious limitations and difficulties when extended to include rationalizing the rules or their implications. Typically, the reasoning by the expert that led to a particular rule is based upon some deeper mental conception of the device, or at least expert knowledge acquired by, and related to, experience. *This much deeper device model is typically not expressed within the expert system, and is thus inaccessible.*

The additional problem with expert systems is that the natural-language rules are non-computable, i.e., they cannot be explored for further implications, applications, or qualifications. At the heart of the problem is the lack of precision of the words used in the rule base and the inability to apply a range of computations to the rule base to derive conclusions not explicitly stated. As discovered by Clancey (1983), ambiguities in a rule base of production rules work to thwart deriving explanations therefrom. The following gives some flavor of the problems encountered (Phillips, Eike & Flegler, 1986):

Careful examination of the rules exposed a variety of subtleties in terms of the way the clauses were ordered. The ordering of clauses within a production rule will influence conclusions drawn by the inference structure. This obscuring characteristic, compounded by the unavoidable presentation of rules used to control the troubleshooting path prevents the true explanation from being explicit.

When people converse they bring much more to the table than the words actually uttered - they express and interpret meaning in terms of a broad and deep body of world knowledge about the subject and about the other parties to the communication. Lacking this wealth of mediating information even the largest of rule bases is typically fragile and shallow. Its fragility is made evident when it fails to respond appropriately to questions dealing with combinations of events that were not adequately covered by the individual rules. The shallowness is evident from the limitations placed upon the types and content of questions that can be asked.

Even when the expert system can make a satisfactory reply, any justification that it might give can only be in terms of citing the rules that yielded the result. While this might be necessary and appropriate in some cases, it is hardly sufficient if the elucidation is a mechanistic parroting of rules. A fair analogy might be drawn from the computer programming world, in which a learner wishes to understand how a particular computer algorithm operates. We could simply hand the learner the program code, and say "See, that's how it works." Generally, however, the learner is seeking an explanation that goes beyond a mere listing of tautologies. The good instructor, it seems, is skilled at providing

some new information to augment what is already known, or is able to facilitate the application or reorganization of what is already known.

The imprecision of our natural language can be overcome by greatly restricting the language allowed in the expert system. For example a rule base concerning only causality could be formulated as follows:

- A causes B
- B causes C

This rule base could be probed to generate responses to the following questions:

- Does B cause A?
- Does A cause C?
- What are all the possible causes of C?

Unfortunately, the benefits from developing restricted expert systems rarely justify the labor required. A possible exception to this is a restricted expert system that supports both device simulation and explanation, as will now be considered.

Device Models

One form of restricted expert system is a rule base that expresses the functionality of a device, as was done in the IMTS/RAPIDS system. Rather than expressing ways to operate or maintain that device, this more fundamental rule base prescribes how the individual components of a device behave and how those components interact. From this can be inferred, automatically, the behaviors of the device as a whole under a wide range of conditions, and, hence, the ability to simulate the device as a user operates upon it.

In addition to supporting device simulation, these explicit device models have the potential of supporting the generation of recommendations to achieve various goals and of providing the reasoning that led to those recommendations. Furthermore, this more robust device-oriented knowledge base has the potential of addressing many of the types of questions that would be asked by technical users.

The language into which the device model is cast must be sufficiently restricted and precise that no ambiguity exists concerning the implications of the individual rules or any combination thereof. This is done in RAPIDS II by predefining a language of operators and modifiers sufficient to prescribe the functioning of virtually any component type, and by allowing virtually any number of individual components to be combined in a simulation.

A deep model of a particular device is made up of expressions, produced by technical experts, describing the workings of the parts of the device and the way in which the parts are organized. If the model has been prepared as a unique computer program (as in STEAMER, Hollin, Hutchins, & Weitzman, 1984), then the program expresses the developers' conception of the device. In this case, however, the computer program is not merely a collection of rules of device behavior, but also a structure of algorithms necessary for executing the device rules to produce device behaviors.

If the device model is a collection of predefined objects, and possibly rules about their interactions, as in the case of IMTS, then the collection of rules stands apart from the inference engine (simulation algorithms) necessary to produce device behavior, and this rule set constitutes a type of expert system.

The substantial differences between RAPIDS II and a conventional, general-purpose, expert system, however, are these:

THIS
PAGE
IS
MISSING
IN
ORIGINAL
DOCUMENT

Fig 5

display is even more useful when the flow of effects is more complex, as in Figure 6, below.

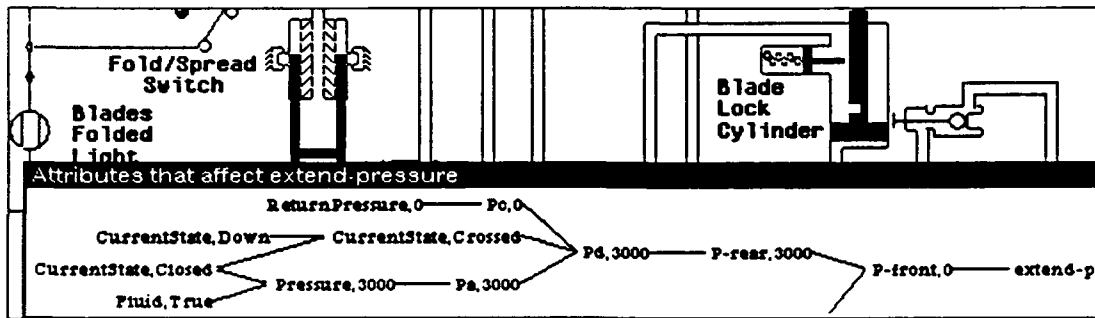


Figure 6. A More Complex Diagram of Effects

These RAPIDS II effects diagrams show not only the attributes that affect an attribute of interest, but also the current values of those attributes.

We now consider the range of technical issues that could be supported with the RAPIDS device model structure.

Types of Technical Explanations

While a large number of domain-specific cause-effect questions might be posed for a moderately complex device, the variety of question types is relatively limited. Thus the capability to address such issues rests primarily upon an ability to generate inferences about the device behavior rather than an ability to deal with natural language processing.

We classify the kinds of explanations that might be generated about device behavior into four types:

- 1) explanations about *effects* of various actions and conditions upon the device,
- 2) explanations about the device architecture as it relates to object interactions and underlying attributes,
- 3) explanations about *causes* of device behaviors
- 4) explanations of theory of operation

Effects of Actions and Conditions

One significant difference between a student that has only received 'text-book' instruction from one who has gained pertinent knowledge on the job is that the latter has actually experienced operating and observing the device. Simulation-based instruction can provide similar experiences in the classroom. Often, the simulation can provide experiences that cannot be replicated in the field, by allowing the learner to experiment in ways that could not be tolerated with real equipment. Such experiments are of maximal instructional value if accompanied by a patient and knowledgeable tutor who can explain the observed effects. The remainder of this paper explores the feasibility of developing such a tutor within RAPIDS.

Two types of questions about effects arise often during learning: 1) how would the device respond to certain *actions* performed upon it, and 2) how would the device respond if certain *failures* occurred. To an extent, these two types of question are already addressed in RAPIDS, as the learner can perform actions of interest and observe the responses of the simulated device if he or she can find them. Currently missing in RAPIDS, however, are features 1) to identify those system responses that result specifically from particular actions or new conditions, and 2) to focus the learner's attention upon the changes in the device.

To provide such support, RAPIDS would save all object states and attribute values at the instant the learner indicated a desire to explore the effects of some forthcoming actions or malfunctions. The learner would then proceed to perform the actions of interest. These might include actions that an operator or maintainer might normally perform, such as setting switches and controls. They might also include changing conditions not normally under the control of the operator, such as introducing one or more failures, altering inputs coming from external sources, or possibly even changing the characteristics of some of the objects in the device.

As the learner introduces change, the RAPIDS simulation would respond, as it does now, by displaying the device in its new condition. With the additional support function implemented, however, the student would have the important option of asking (via a mouse action on a menu item) "What parts were affected by my actions?" Of course a diligent learner could answer this question by making a full accounting of all the device parts before and after the actions, but this places a heavy clerical load on the learner. We would prefer to support the learner by automatically identifying all significant changes.

This support function would direct the student's attention to those portions of the simulation that were affected by his or her actions (thus performing the *orienting* actions of a human instructor). It would highlight all the parts that had changed in some significant manner, with pedagogical consequences analogous to those due to the focussing actions of the human instructor. These orienting and focussing actions might simply draw attention to those parts of the currently displayed world that should be noticed or they might guide the learner to view sections of the simulated device not currently displayed. These functions could be of great value in learning about complex devices, for the number of changes could be considerable, and many changes might occur on sections of the device not currently displayed.

One issue to be addressed relates to deciding which kinds of changes to include in the presentation. One alternative is to highlight all the parts that changed *appearance* in the graphical simulation as a result of the user's actions. Such a function would simplify the learner's task in accounting for the effects of his or her performance. This function would direct the attention of the learner to objects whose discrete states changed, such as relays, fuses, and indicator lights, and to continuous objects whose changes may have been obvious or not. One additional function would allow the learner to alternatively view a selected object as it was before the performance and after, for comparison.

Alternatively, the learner may want to see all parts whose *attribute values* changed in any way, even if the attributes were not visible. More precisely stated, the student may wish to know which objects processed signals whose attributes changed as a result of the actions or the malfunction that was introduced. For example, a simple amplifier might be outputting 100 times its input of 3.5, prior to the learner's actions, then be outputting 100 times an input of 5.4, after the actions. In one sense the amplifier has not changed at all; it continues to amplify its input by 100. But it may be meaningful to the student that the output of the amplifier changed as a result of the actions performed, even though there were no visible affects for this part. In addition to highlighting such parts, we may wish to also list the attributes that changed.

It is important to appreciate that RAPIDS II, the latest version of simulation offering real-time responses, operates upon attributes, such as voltage, current, pressure, or temperature, whose values change. Typically many objects will be affected by a single attribute. Attributes are bound to objects for convenience in authoring and for naturalness in generating verbal interactions about simulation events.

Both appearance change highlighting and attribute change highlighting appear to be of instructional value, and could be chosen by the learner. After studying simulation effects in

one or both of these modes, the learner could either continue on from the current device condition or could select a menu item that automatically restores the device to its condition prior to the changes introduced for study.

Alternatives for Observing Effects

The approach described above provides a type of before-and-after view of effects. The learner is shown the parts of the device that changed in response to the designated actions. In a complex device the learner would usually be guided to several different scenes, each of which exhibits one or more changes. There will also be situations in which the learner wishes to see the effects simulated multiple times. Suppose, for example, that RAPIDS has shown the learner that a particular part of the device changed in response to some actions of interest. The learner now might wish to concentrate on that portion of the device while the effects are simulated again (without having to repeat the actions a second time). This might reveal interesting real-time effects that were not originally noticed, and it might provide deeper insight to the student as to how the events took place.

Students should have the option of changing the simulation clock speed between repetitions of the observed simulation effects. This feature could help to clarify rapid effects by imposing slow motion; it could also be used to speed up very slow effects for more convenient observation.

The learner might also wish to understand better how two or more objects in the device interact (or alternatively, an instructor might wish to demonstrate this matter). To accomplish this, the user would ask RAPIDS to repeat the simulated response to the actions, ensuring that the objects of interest are continually and simultaneously shown (perhaps by putting copies of the objects in *active snap windows*). Maximum instructional flexibility will be achieved if the learner can repeat the simulation update at will without having to go through the motions each time and can observe possibly disparate objects in the device simultaneously.

This capability also resolves a potential problem, in which an object changes in response to the designated actions, but ends up in exactly the same state as that in which it started. If the student can only see the before and after states, and happened to be concentrating on a different portion of the device when the simulation update occurred, then some confusion could arise. With the ability to repeat the simulation update, the student will see the object pass through at least two stages and ultimately return to its original condition.

Implementation Note. Providing the capabilities to monitor transient changes, of course, has implications in the manner in which this instructional feature is implemented, in that it will not be sufficient to compare attribute values after the actions to their values before the actions. Instead, the monitoring functions must note any changes that occur, even if the final status, after the student's actions, happens to be identical to the original status.

Explanations About Device Architecture

RAPIDS represents devices in terms of objects and collections of objects. Whether viewing the most detailed view of a device or a high-level representation yoked to the detailed model for fidelity of behavior, the learner sees the device in terms of objects. Some of these objects, such as indicators, may display values of certain attributes, such as pressure or voltage. Other objects take on static state appearances that are determined by the values of their attributes. Some attributes have no direct appearance effects, but rather affect the values of other attributes (which may, in turn, have appearance effects). Similarly, the user of a real device may infer many attribute values, but rarely experiences any of them directly.

For instructional purposes, a capability to make explicit the relationship between certain important attributes in a device and the parts of the device that affect those attributes may be very important. Likewise, an ability to depict what objects in a device affect others can benefit both the student operator and the student maintainer.

Depicting the Attribute Architecture

Both operators and maintainers may wish to know what parts of the SH-3H Blade-folding subsystem 1) *depend upon* 28VDC-Bus, and 2) *affect* 28VDC-Bus. Stated another way, what objects *cause* this attribute to have particular values and what objects are *affected* by this attribute. The proposed user function would allow the user to select an attribute from the list of attributes used in the model, and to specify whether the interest is in those objects *depending upon* the attribute, objects *affecting* the attribute, or both.

An alternative point of view might also be instructionally useful, one in which the learner wishes to know all attributes affected by a particular *object* or all attributes that affect the *object*.

Developing the function to analyze the model rules to meet these objectives will be a significant undertaking. Clearly the analysis procedure is not as simple as just finding those objects that explicitly refer to the attribute of interest. First, the way in which an object relates to the attribute must be considered, in order to distinguish objects that depend upon an attribute from those that affect the attribute. In addition, the function must find all consequential effects, those objects whose behaviors are affected by objects that are affected by the attribute of interest, and so on to all levels of indirectness.

Furthermore, as one object operates upon a particular attribute, it may produce an attribute of a new name. All objects affected by this new attribute reference must also be identified. Thus, for example, a hydraulic actuator object in Blade-fold may have no explicit association with 28VDC, yet its position may be determined by an object that is ultimately affected by 28VDC.

For the student to be able to selectively produce and study graphic displays depicting these functional/architectural relationships could be a major instructional asset. Without doubt such power could also lead to great confusion and loss of instructional productivity if not employed carefully.

Depicting the Object Architecture

Closely related to the *attribute* architecture of a device is its *object* architecture, i.e., which objects in the device affect others. The questions of interest include the following:

- What parts are affected by <some identified part>?
- What parts affect <some identified part>?
- What parts in the device function in loops?

These three questions are essentially equivalent, the primary difference being the directionality of effect. In the first case we wish to trace 'forward' to identify all those parts whose behavior is affected by the identified object. One alternative here is to show all those parts affected *directly* by the identified object. This would simply be all parts handling attributes that come directly from the identified part. The second alternative of this first question, showing all indirect affects as well, simply perpetuates the search until all affects are identified.

In the second case we trace backward to find all the parts affecting a given part (again, either identifying direct affects only or both direct and subsequent affects).

If the selected object is in a loop, then all objects in the loop would be identified (highlighted) whether the question concerned objects affected or affecting. Since the user might not discern that a loop exists, some indication should be provided that the highlighted elements are in a loop. If the learner explicitly asks what objects are in loops, the instructional system should identify each loop, possibly with a different color.

The technique for analyzing the RAPIDS rule base may be similar or identical to that used to generate inferences about attribute architecture. There are some rather subtle questions of interpretation related to these questions, however. The difficulties seem to spring primarily from lack of precision in our language and in the two-dimensional diagrams we commonly use to represent more complex entities. In the nearly trivial diagram of Figure 7, for example, it appears that block A is affecting block B directly and block D indirectly. We could conclude that A is not affecting C because switch S1 is open. Thus it appears that we can determine effectivity by determining if attributes are being passed from one object to another without considering the values of those attributes.

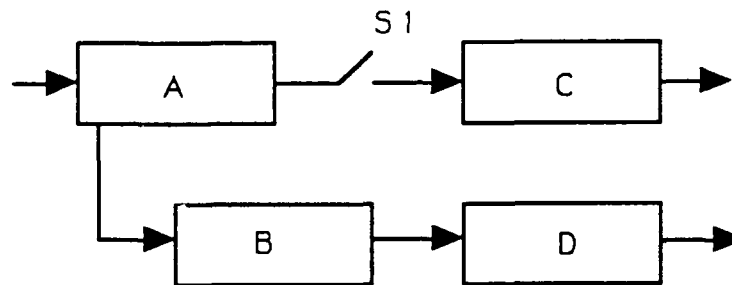


Figure 7. A Block Diagram of a Simple System

But what if S1 were closed, and block A, because of the particular inputs it is receiving at this time, is outputting a value of zero to block B, more precisely it is outputting nothing to block B? We might conclude that in this case A is not affecting B. But what if this nil value is precisely the value that causes B to do something important, like outputting a high value to D? Then it appears that A is affecting B, by outputting nothing to it. Yet this is also the value that block C is receiving from A. So is C being affected by A, or just by S1?

In a properly constructed simulation, there may be a straightforward solution to this potential problem. Consider the behavior rule for S1. Suppose its form is

```
if CurrentState is 'Closed
    then Assign MyOutput MyInput
    else Assign MyOutput 0
```

Here the currently active assignment determines the flow of affects from A to C. If the state of S1 is 'Closed, then a chain of values can be traced back to A from C (because, of course, the MyInput attribute of S1 is assigned the value of an attribute of A). On the other hand, if the state of S1 is 'Open, then there is nothing to trace back to from the MyOutput attribute, because it is simply assigned the value 0.

For purposes of tracing causal effects backwards, references to attributes count in both the condition and in the effect parts of attributes. Since the CurrentState attribute of the switch S1 in the above figure will not be influenced by any other objects in the simulation, tracing causal effects will not propagate backward from the CurrentState attribute in this particular example.

Now consider the second case mentioned above, in which the value 0 is propagated from A to B and has an important effect on B. In this case, some attribute of B will have a rule that refers to an attribute of A explicitly. For example:

```

if PowerOutput of A = 0
  then Assign MyBusOutput EmergencyOutput
  else Assign MyBusOutput PowerOutput of A

```

Here the *effect* clause of the rule has no path back to object A when A's output is 0. But the *condition* clause of the rule does lead back to A. A is therefore understood to affect the output of B.

Now consider the diagram shown in Figure 8. Here, blocks A, B, and C are connected to block D which performs the function of selecting the greatest input, and outputting that value. Suppose that block A is supplying 1, block B is supplying 10, and block C is supplying 3. Then the output of D is 10, as supplied by block B. In a functional sense, the output of D is related only to B. Yet, blocks A and C, and their predecessors, are involved in the output of D in the sense that D's output could change depending upon their behavior. Thus, there is a *potential* relationship between A and C to D. The correct diagram to show, therefore, seems to depend upon the point of view of the questioner. If we wonder where the value of D's output, 10, is produced, then we want to see block B and all its predecessors. Certainly we do not wish to explore the functional chains leading to A or C. In fact, these blocks can be turned off or even removed from the system, and we still see an output of 10 from D.

On the other hand a diagnostician might wonder what failure could have occurred that would cause D to output 10, when it should be 100 in the current configuration. To answer this question we must include A and C and their predecessors in the set of possibilities, for it might be that A is outputting 1 instead of 100, due to a failure in it or one of its predecessors.

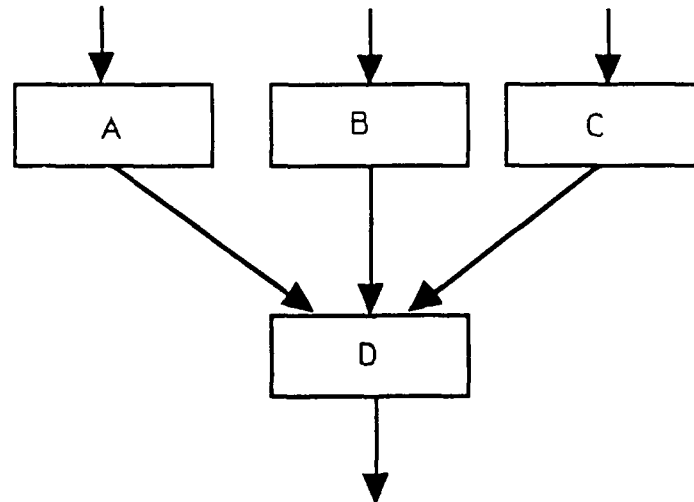


Figure 8. A Simple System with *Potential* Causal Relationships

It appears that the diagram, developed for the purpose of representing the real device, or at least some aspect of the real device, is a poor vehicle to use to judge effectivity. The precision we require is gained only by turning to the rules that specify the behavior of the device and by being very precise concerning the exploration to be conducted.

It becomes increasingly apparent that the diagrams and explanations of system behavior serve as starting points for the technician, at least for diagnosis.

The function that traces the connectivity of attributes must evaluate all conditional expressions to determine if one object is currently affecting another. For example, consider the following simulation rule:

IF TEMPERATURE > 215 THEN ATTRIBUTE2 = f (ATTRIBUTE1)
where f is any function, i.e., ATTRIBUTE2 is a function of ATTRIBUTE1. Here, the object 'owning' ATTRIBUTE1 affects that owning ATTRIBUTE2 if and only if TEMPERATURE exceeds 215, thus the traces of effectivity are clearly dependent upon the current mode of the device.

At times the learner will be interested in placing the device into a particular configuration of interest and then inquiring about the ways the objects affect one another. In other cases the learner will be more interested in how these interactions change as the configuration (mode) is changed. Thus, in the latter case, the learner may wish to select a single object, indicate if the interest is on affected or affecting objects, then step the device through a number of operating modes. Here is a case where a 'modal' user interface design would work to the benefit of the user (the IMTS/RAPIDS design avoids modal interactions for the most part). By placing the training system into a 'display effects' mode either from or to the selected object, the learner could watch how the effects change as various switches are thrown.

It may also be useful to offer a mode in which students can ask about potential effects. This would highlight all the objects that can possibly be affected by a selected object in any non-malfunctioning configuration of the device.

Explaining Causation of Effects

Questions about causation in a device have to do with how and why certain events did or did not occur, in response to the actions or conditions applied to the device. These questions generally arise out of the learner's surprise or confusion, i.e., something happened that was not expected, or the expected did not happen. Now the learner wants to know why. Some examples of this question type are

Why did the Power Alert Light come on?
Why didn't the interlock latch disengage?

The issue of causation is a somewhat more specific case of effectivity in general. To determine why the Power Alert Light came on, a learner could request and examine a display of all the objects and attributes affecting the light. A more direct approach, however, would be to simply ask why the light came on, and rely on automated processes to derive the answer from the rule base.

There are several difficulties associated with responding to this type of query. First, the query is made *after* the causing actions are performed. Since no forewarning was given that the learner would be asking the questions, no particular data were recorded to track the events. A second problem involves the best way to convey the information requested. We can be succinct, by only relating the most recent action that triggered the event in question, running the risk that the learner will over-generalize from the short answer. Alternatively, we can provide a complete enumeration of all the events that worked to cause the event in question, running the risk that the learner will become confused or exasperated with our level of detail. The following example illustrates the issues.

Imagine that a human instructor is observing a student learn to start a car. The student enters the car, moves the transmission lever to Park, turns the ignition key, and hears the engine start. The student asks "Why did the Engine Start? One possible way to respond is "Because you turned the ignition key to the Start position." While this is certainly not a false statement, it also does not tell the complete story. The learner might generalize from the response to believe that the car can always be started by simply turning the ignition key to Start, not realizing the additional requirement that the transmission lever be in Park. The

answer "Because you set the transmission lever to Park and turned the ignition key to Start" sounds good when there are just these two actions to explain. But what if there were 20 necessary conditions. Do we enumerate them all?

In the general case there are multiple necessary actions required to achieve a particular result, and that result is observed when the last necessary step is performed. This generality is true whether the actions must be performed in a fixed sequence or a variable sequence.

Time Issues

Some systems can be rather fully explained with a static graphic representation. Generally, such systems reach a stable condition in a relatively short period of time relative to their operating cycle, and thus they can be viewed in a relatively steady-state status. Furthermore, the details about how the steady-state was reached is unimportant. Thus we can observe the system, and the values at its test points, and know what is going on. Figure 9 illustrates a simple (and imaginary) system of this type.

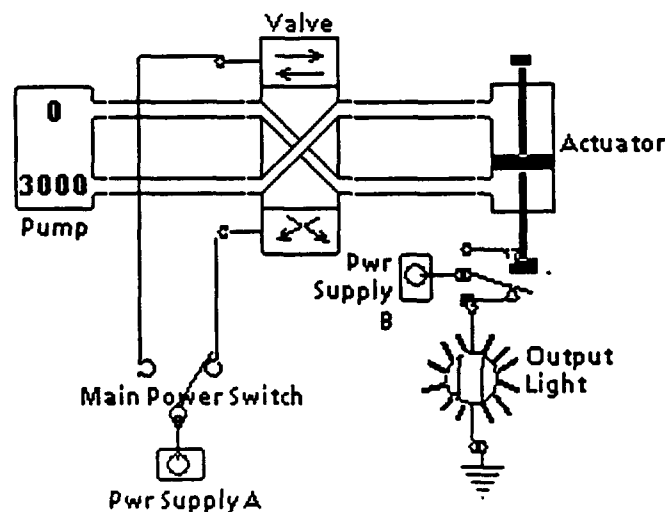


Figure 9. A Simple Discrete-State System

Other systems can only be understood by observing them over time. Some of these change gradually, others change in discrete steps. One very simple system, a crude voltage-to-digital converter is shown below Goode & Machol, 1957). This system accepts a voltage as input and outputs the value of this voltage as a binary number, after some operation time. The system operates as a simple counter, adding one to the value in the counter whenever its current (estimated) value is low, and subtracting one whenever the current value is high.

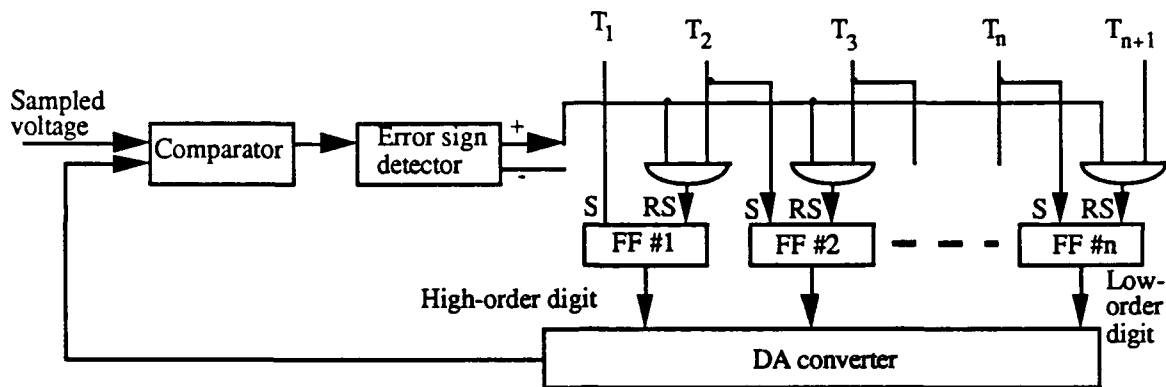


Figure 10. A digital decoder of an analog value.

Explaining Theory of Operation

A long term objective is to generate expositions (combinations of text and graphics) that explain the theory of operation of a device. Such an exposition might give an account of what the major sub-elements of the element in question do and why. The checklist (what, when, where, why, who, how) outlined above would seem to provide a good measure of completeness of such explanations. It is not clear, however, that the rule base that supports simulation, and probably explanations of type outlined above, can also support generation of these more natural, and less restricted, forms.

One critical issue involves recognizing important functions from ancillary ones, and discriminating the role of each element. A start toward making such a recognition would involve identifying the general nature of elements of the device. Four basic element types are 1) sources, 2) enablers/disablers, 3) routers, and 4) processors.

Source. A source is any element that is the originator of an attribute, such as 12 VDC, and requires only external inputs for its operation. Other sources might be identified which produce original attributes, but do require inputs from other elements in the device (as distinguished, say, from a power supply). To recognize such functions might require that the attribute tagging involve suffixes, such as POWER.MODULATED, to recognize that one attribute is a modified version of another.

Enablers/Disablers. An enabler/disabler does not enter into the transformation of an attribute, but instead determines whether an element operates or not. A rule of the form
 IF $E > 5$ then $X = 2 * Y$ else $X = 0$
 reflects an enabling function, as the output (X) is not a function of the input (E).

Routers. Routers, such as pipes and wires, simply convey signals or other attributes from one device element to another, without performing any transformation upon them.

Processors. Processing of a signal is indicated by an output being a function of an input. For example, IF $Z = RED$ then $X = 2 * Y$. Here, the object that applies this operation is processing the input Y and outputting X.

Summary & Conclusions, Part I

The central components of a technical explanation are 1) the representations of the device or phenomena being instructed, either functionally or physically, 2) the verbal amplifications, and 3) actions that focus the learner's attention on the salient aspects of the topic.

The variability of performance among different learners virtually precludes use of any technique that involves preparing canned verbal or graphical material in expectation of the needs. Likewise, classical expert systems in which a large rule base is built up by interrogating technical experts, do not exhibit the power required to generate specific inferences about device behaviors.

Instead we must find ways to generate explanations and responses to technical questions by computing upon the RAPIDS-style rule base, a more restricted knowledge representation form that is designed specifically for representation of device behaviors. The responses must be technically correct, and they must be presented in a manner that is effective and natural. Ideally, the explanations should respond appropriately to a perceived knowledge deficit on the part of the student. While maximizing the instructional effectiveness of such responses is a legitimate goal, we must begin by developing the ability to generate and present technically sound explanations first. Once this facility has been demonstrated, it would then be appropriate to proceed to questions of optimality. Answering such questions is likely to require additional research on the effectiveness of presentation media and styles and on the relationships between the effectiveness of different presentation approaches and the knowledge deficits that are to be remedied

Appendix C, below, provides an example.

Part II

Generating Device Representations of Varying Complexity

Douglas M. Towne

This paper is concerned with techniques for automatically representing systems to differing levels of complexity. The primary emphasis is on feasible approaches for producing alternate views of complex systems, with some additional consideration of manners of automatically producing textual accounts of the roles of objects in devices.

Background

It is a common practice for equipment technicians to learn about new devices using the technical documentation prepared for reference purposes. In some cases, where the device is not overly complex and the documentation provides adequate supporting text and figures, this situation is tolerable, if not ideal. For devices of even moderate complexity, however, technical reference material generally falls short for training purposes.

Even when materials are specifically prepared to address training requirements, their hard-copy form is increasingly inadequate for explicating systems whose operation involves substantial reconfigurations and dynamic effects. Only a dynamic medium, such as computer graphics, can hope to portray the inner workings of complicated devices, under a multitude of conditions, and in response to specific learner needs.

Systems such as STEAMER, ESAS, IMTS, SHERLOCK, and RAPIDS have demonstrated some of the potential of computer graphics for portraying the functionality of devices. The object-oriented approach taken in STEAMER, IMTS, and RAPIDS is particularly well positioned to exploit computer graphics and intelligent computer processes.

This orientation creates an opportunity for capturing human technical expertise in an efficient and powerful manner. IMTS and RAPIDS, specifically, demonstrate the high productivity achieved when technical experts produce intelligent graphic objects that can then be employed by them and others in producing future device models.

Scope

This section is concerned with exploring the next logical step - investing human expertise into the specification of objects and device representations specifically for the purpose of allowing those elements to explain and/or represent themselves in instructive ways. Of course IMTS and RAPIDS already offer considerable instructive power, by allowing the learner to manipulate the simulated device and to observe its responses. The concern now is with extending the object definition to include instructive intelligence as well as functional intelligence.

The prior section outlined the possibilities for generating domain-specific explanations of device operation, using text and graphics. The graphic forms considered are of two types 1) highlighting objects in the RAPIDS device model to indicate various cause-effect relationships (such as What objects are affected by this object?), and 2) representations of causality in the form of directed graphs (object names linked via directed lines).

This section considers ways in which the RAPIDS graphical device representation can be processed to enhance understanding, particularly for the purpose of generating representations of appropriate complexity for individual learners. Of course one critical mechanism for reflecting device functionality is already operational within RAPIDS - the ability of the objects in a device model to graphically reflect their current status, whether affected by the user or by other objects.

It is rather clear, however, that this graphical animation is not necessarily sufficient for understanding effects. Thus, a novice can watch the detailed Blade-fold simulation respond as he or she sets the controls that position and fold the blades, and can be impressed by the extent of changes required to effect the blade-folding process, yet be overwhelmed by the diverse and seemingly simultaneous events and processes that are displayed.

A simplified (and real-time) model of the Blade-folding system (Figure 1) has been developed to more closely meet the needs of the novice. Here, the learner can see all the critical functions at one time. It is possible that an even simpler representation could be produced that would retain the critical functionality of the system, yet be comprehensible by the novice. In any case, there is a huge leap from the one-screen simplified representation to the complete 14-screen model. There needs to be some way to progress, in a principled fashion, from the most simplified form to the most complete form.

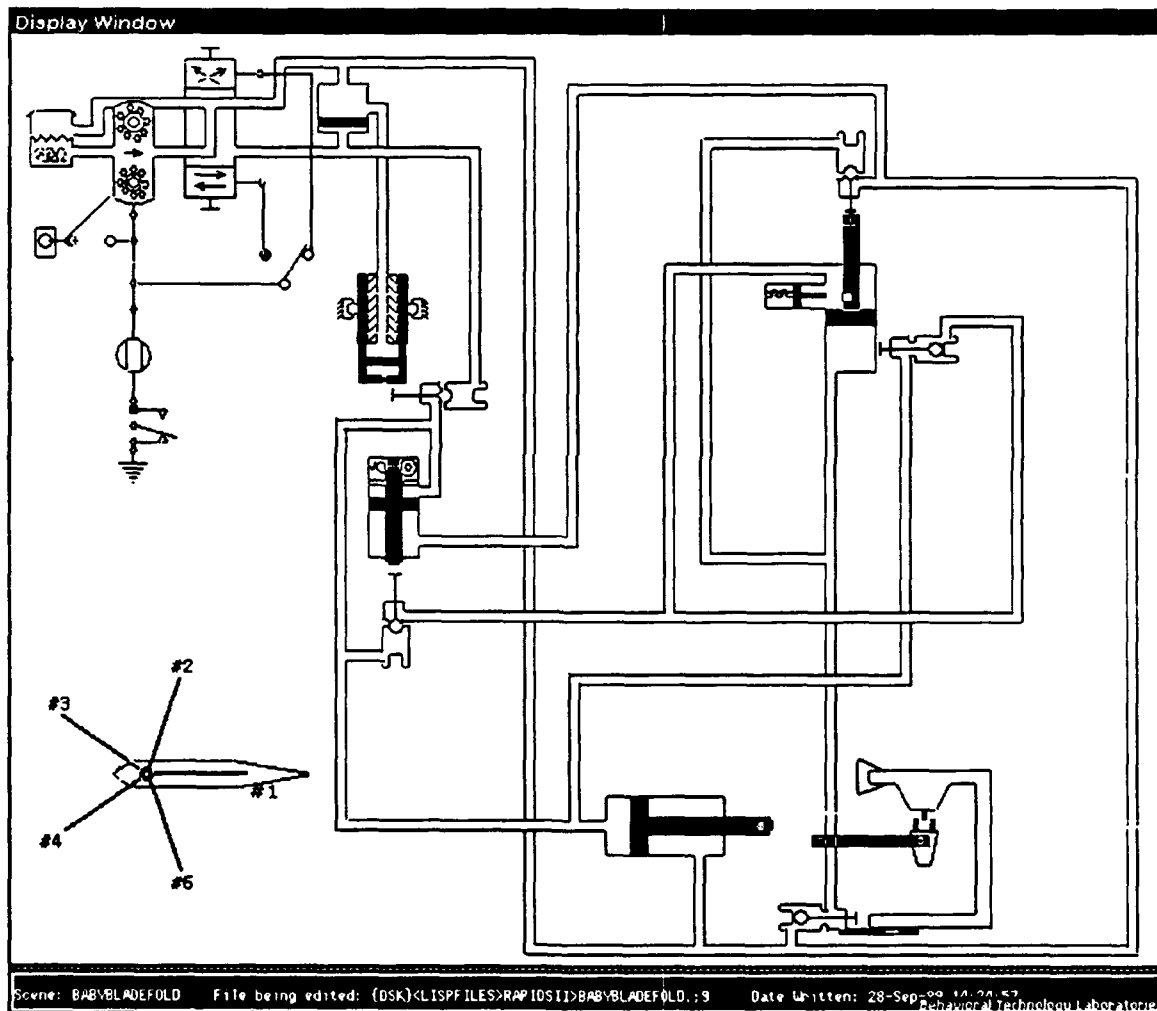


Figure 1. Simplified Blade-fold

Generating Models of Varying Complexity

The following describes three possible techniques for automatically generating device models of varying degrees of complexity. The most significant difference among the three approaches is the level of human intelligence required versus the degree of automaticity involved in the

production of the graphical forms. Instruction based upon these alternative representations would usually follow a progression from simple to complex, however in intelligent tutoring environments there will be occasions for reverting to simpler models.

Unfortunately, there is little instructional theory to guide the formation of an appropriate device model to correspond with an individual's level of understanding. Another problem is that the graphical form that a model should take is probably affected by the purpose we seek. If we are introducing a new device to someone, we may well wish to employ a simple block diagram that reflects the major sections and functions. But if we want the novice to comprehend the key functions that a device performs, we probably need a highly simplified, yet functionally complete (seemingly operational) model. How can we automatically produce a partial model of a device that is cohesive and instructionally meaningful?

Three possible approaches for selecting objects for inclusion in a model of some specified level of complexity will be considered here:

1. select according to specifications attached to the *specific* objects in the device model;
2. select according to specifications attached to the *generic* objects from which the specific device model was constructed; and
3. select according to an analysis of the underlying behavior rules of the device model.

Selection According to Specific Object Specifications

Perhaps the most direct approach is to have the subject matter expert assign to each part in the complete device model a number that indicates its importance in understanding the behaviors of the device as a whole. Suppose, for example, that the most significant parts are assigned 10, the next most significant parts are assigned 9, and so on, with the least meaningful parts being assigned 1.

With an appropriately designed selection algorithm, it would not be necessary for all numbers between 1 and 10 to be assigned. Additionally, the technical expert need not use a 10-point system; he or she might choose to use a three-point system, thereby producing three levels of representation rather than ten.

To produce the simplest model, those objects having the highest ratings are selected and displayed. Interestingly, the simplest model conveys the most critical functions. To produce a slightly more complex version involves adding in those objects having the next smaller rating, and so on. When the most complex representation is formed, the least important parts join the others.

Obviously this approach will work to guide the formation of progressively more complex representations. Three problems associated specifically with it are:

1. the human effort involved in rating the objects could be inhibiting;
2. special representations could not be produced in an ad hoc manner to address the needs of an individual learner; and
3. the assembly of graphics from individually rated objects could produce device representations that lack technical cohesiveness.

As a practical matter the first two problems do not appear to be particularly distressing. While some hours of labor might be required to rate all the objects in a complex model, this effort could be small compared to the potential gain from providing this instructional power.

The inability to adaptively form representations is of unknown seriousness. If the manually assigned object specifications yield an adequate repertoire of representations, then an automated instructional system has the opportunity to select the most appropriate one for an individual learner, and to remain with that representation until that individual is ready to move on. To what extent additional, but unavailable, representations would be required will depend very much upon the domain and the quality of representations made available. Thus, the inability to

dynamically generate representations does not seem to be a major limitation, especially since additional figures could be produced as outlined in the earlier paper to address a learner's questions concerning causality.

The most serious potential problem related to this approach concerns the technical cohesiveness of particular object combinations that might result. For example, a representation could result that omits some objects that were properly omitted in simpler versions, but should appear in the current level of representation, since other objects related to the functions just entering the current level have been added. This problem would be overcome if the rating of individual components were done with a view toward producing specific, collective representations. This appears to add to the human intelligence and effort required, but it does not affect the potential mechanism proposed.

Selection According to Generic Object Specification

The human effort involved in attaching specifications to objects would be greatly reduced if the assignments could be to generic objects rather than to specific ones. The specifications could then be brought into a specific device model automatically, from the generic object library, thereby eliminating the human effort required during development of specific models.

There are probably some few objects in the world that are important in virtually all devices in which they are employed. A nuclear detonator might be an example (after all, in how many different devices can a nuclear detonator be involved?). And there may be some objects that are unimportant (again, from a learning viewpoint) in most devices. Unfortunately, the importance of most objects cannot be predetermined with high reliability; the purpose and architecture of the device partially dictates the significance and roles of its component parts.

Thus it appears difficult to categorize generic parts according to their relative importance in applications. There is, however, a way of describing generic objects that has some potential of yielding successively more complex models when employed in specific devices. In this approach, each generic object is assigned values that describe its purpose, technology, and function. The rating scheme employed, consistent across all objects in the device, could involve whatever properties are deemed important by the subject matter expert.

When a particular device model has been constructed in RAPIDS, it (the device as a whole) is rated on exactly the same properties as were the generic parts. To produce a model of fewest parts, hence maximum importance, the specifications of the specific parts in the model (carried over from their generic prototypes) are compared to the specification of the device, and those most closely fitting the device specification are selected. Successively more complex models are formed by adding in objects whose properties correspond less well with those of the device. The algorithm for doing this must also decide how extensive each successive model is to be, i.e., exactly how many more parts should be added in to the previous model?

As a simple and qualitative example, the SH-3H helicopter blade-folding functions are primarily mechanical, operated by changes in hydraulic pressure in the lines controlled by electrical signals. Thus the primary function of blade-folding is understood by viewing a model of mechanical objects, driven by hydraulic elements. The less important control functions of the electrical elements can be understood at a later stage.

In some cases this role correspondence rule appears weak. Why should an object be considered unimportant just because its properties differ greatly from those of the device of which it is a part? In other cases the role correspondence rule seems to produce a reasonable progression of representations. For example, the properties of an electrical relay are quite different than those of the SH-3H helicopter, thus the relays would not be shown to explain the helicopter blade-folding function until the primary hydraulic operation had been covered.

The problem of cohesiveness is even more pronounced with this method, however, as the individual components are added to the emerging progression of representations without any concept of the collective forms that will emerge.

In general, it appears that objects that affect the connectivity and topology of the device are less important from a learning standpoint than the objects that actually process signals and effect conversions. Such differences can be discerned by analyzing the underlying rules of the device model, in RAPIDS, as explained below.

Selection by Analysis of Device Behavior Rules

A much more ambitious, but possibly more powerful, approach is to analyze the behavior rules of the device itself to infer the objects that are playing key roles in its behavior. Such an approach would require only that the simulation author identify those few objects whose operation reflects the purpose of the device itself. For the SH-3H blade-folding system, the helicopter blades would be identified as the key objects. Those objects having the most direct affect upon the blades would be regarded as central to understanding the functionality of the system. In a subsequent version, the parts supporting those appearing in the first version would be added, and so on.

It is important, in selecting objects for a particular representation to distinguish between objects that are merely close to the key objects, in terms of intermediate *connections*, from those that have meaningful impact upon the *behaviors* and *functions* of the key objects. If we wish to produce a representation of important objects, we might be tempted to select only those objects that are 'close' to the key objects, in terms of directness. Thus, we would select all those objects that either input to the key objects directly, or send signals to the key objects with just a few intervening objects.

Figure 2 illustrates an example device, a pumping and storage system with temperature and pressure controls. The purpose of the system is to maintain the level of fluid in Tank E at a proper level and temperature (thus the tank is identified as the central object in the device). Various controls and sensors route the fluid back through a temperature adjustment circuit as necessary. Some additional hardware adjusts the pressure in the line, as required, and additional manually controlled valves, N and P, are included for maintenance and emergency override purposes.

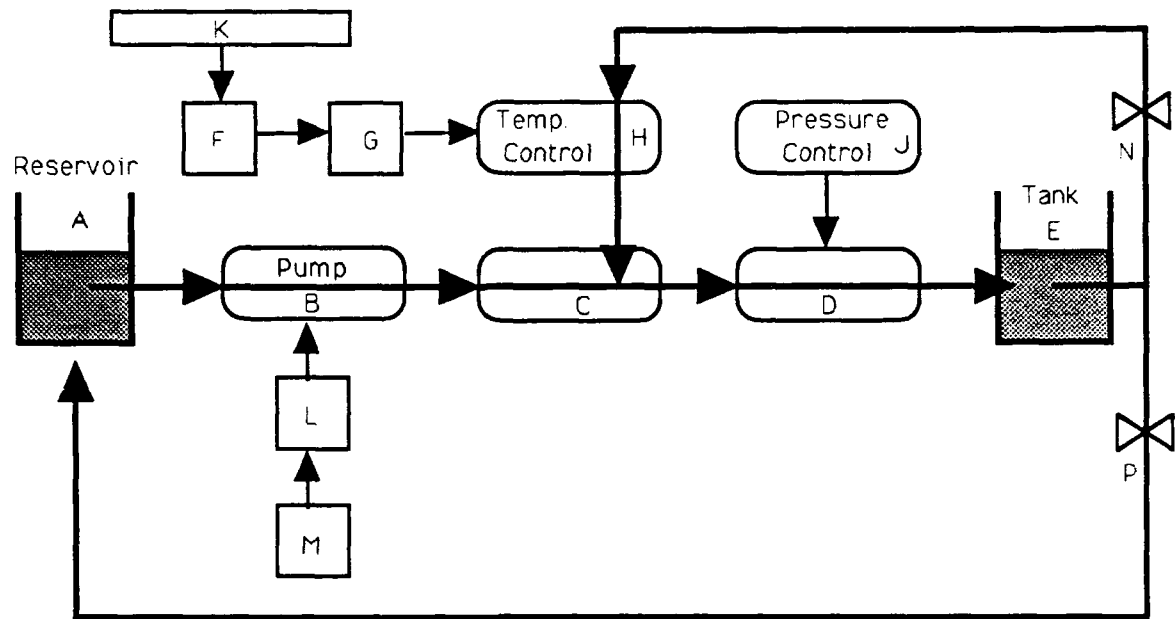


Figure 2. Fluid Flow Control System

Here, we would identify the tank E as being the object representing the key function of the device. If we select objects that are functionally close to the tank for a simple representation, we would end up showing just the components surrounding the tank, as shown in figure 3. Such a representation hardly reflects the essence of the device. Instead it reflects all those components that happen to be near the end of the chain of effects. This representation fails to show that the fluid is recirculated to adjust the level of the tank and the temperature of the fluid.

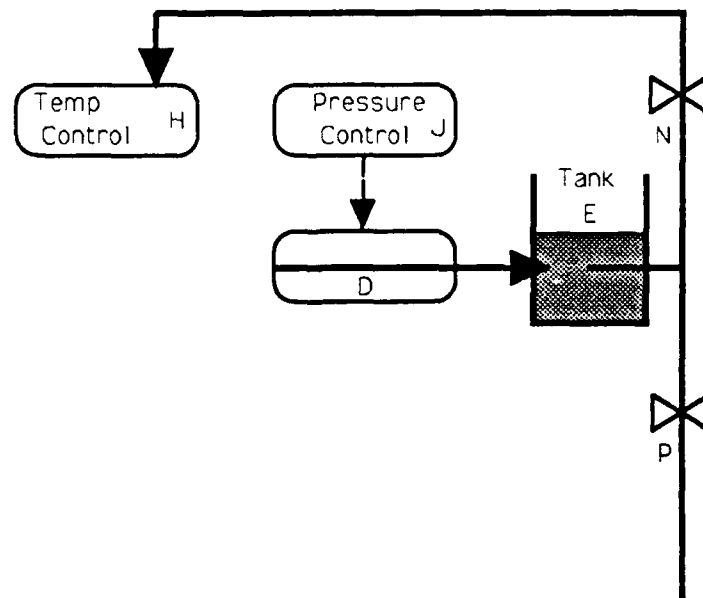


Figure 3. Representation Formed Based Upon Proximity of Connectivity

How, then, do we recognize an object that may be far removed from the key objects, but crucial to understanding the device behavior? A deeper analysis of the behavior rules of the device would show that the attributes of the tank (level and temperature of fluid) are affected

primarily by objects A, B, C, D, H, and J. For example, block H is concerned with adjusting the temperature of the fluid.

Several associated components are required and crucial for H to perform its function, and must ultimately be understood. But for a simple view of critical functions, those ancillary objects should be omitted, as shown in Figure 4. In a similar fashion the objects supporting objects B, H, and F are of lesser significance, as they act as facilitators to these objects.

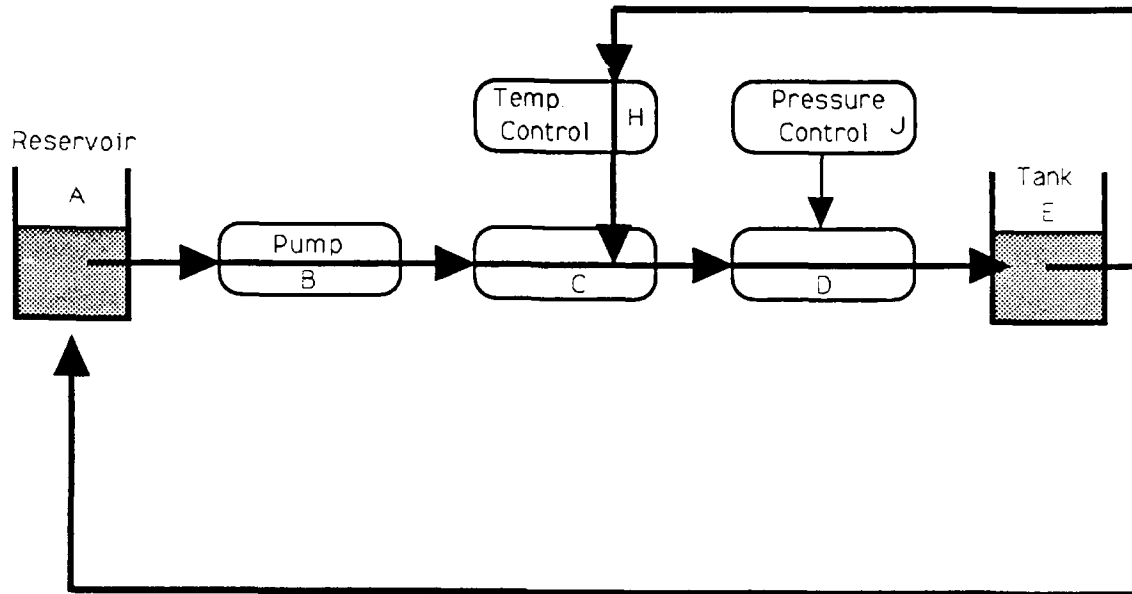


Figure 4. Representation Formed Based Upon Functionality

Fortunately, the active processing of a key attribute, such as fluid temperature, can be recognized by analyzing the object rules, and can be distinguished from support functions. For example, the output of block H is fluid of a particular temperature and rate of flow. While the output depends partially upon the functions controlling block H (from F and G), the primary attributes being processed by H are the fluid temperature and rate, as received from the tank. Thus blocks F and G, and their supporting units, can be determined to be performing secondary functions. Possibly blocks F and G will be added to the model at the second stage, however, their supporting elements would not come in until the third representation, at the soonest.

Reconstituting the Graphical Representation

Whatever the approach used to identify those objects that should be included in a particular representation, a problem remains in reconstituting a cohesive graphical representation. To simply remove the unnecessary objects from the 14-scene Bladefold simulation would result in a highly disbursed representation that is mostly blank screen. This problem is primarily one of spatial layout. We note the necessity to develop techniques for consolidating a simplified representation, but offer no guidelines here.

The two examples in the Appendices carefully avoid this problem. The most complex versions are single-screen figures, while the simplified versions simply omit some of the objects. This approach would be acceptable for small device models but not for complex ones.

Text-based Explanations of Functionality

Another long term goal is to generate textual explanations of how a particular device operates, from its underlying simulation data, without requiring human effort at the domain-specific level.

One significant step toward that end would be to endow each generic object the ability to explain what it does in a *particular* device model.

This could be done by authoring a functional explanation for each generic object, written in terms of its local input/output ports. When the object is used in a particular device, the surrounding objects supplying inputs and using outputs, and their attribute names, would be automatically substituted into the generic explanations, thereby forming a domain-specific explanation.

For example, consider the slider control shown below.



Figure 5. Generic Slider Control

A general statement of the function of this object is as follows:

The <object_name> is used to control <using_attribute> between <source_object> to <using_object>, between the limits of <minimum_value> and <maximum_value> <units>. Each gradation represents $(\text{maximum_value} - \text{minimum_value}) / 8$ <units>.

Suppose this generic object is employed in the flow control system shown in Figure 6.

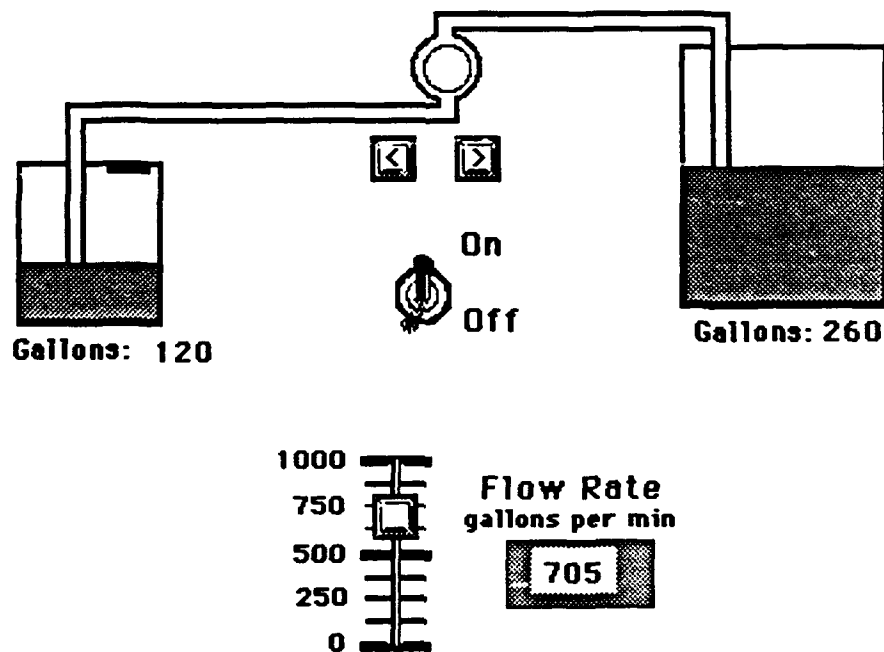
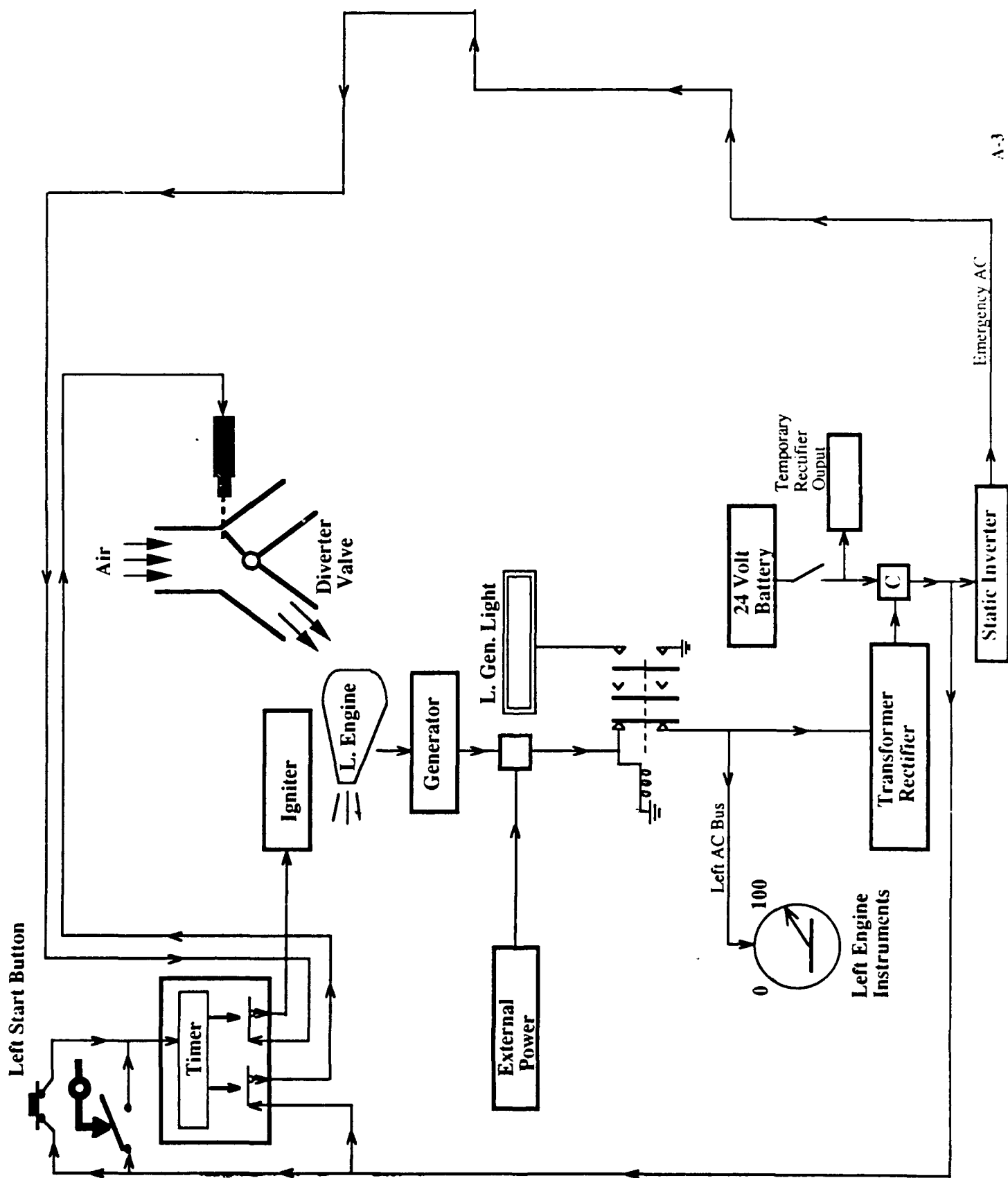
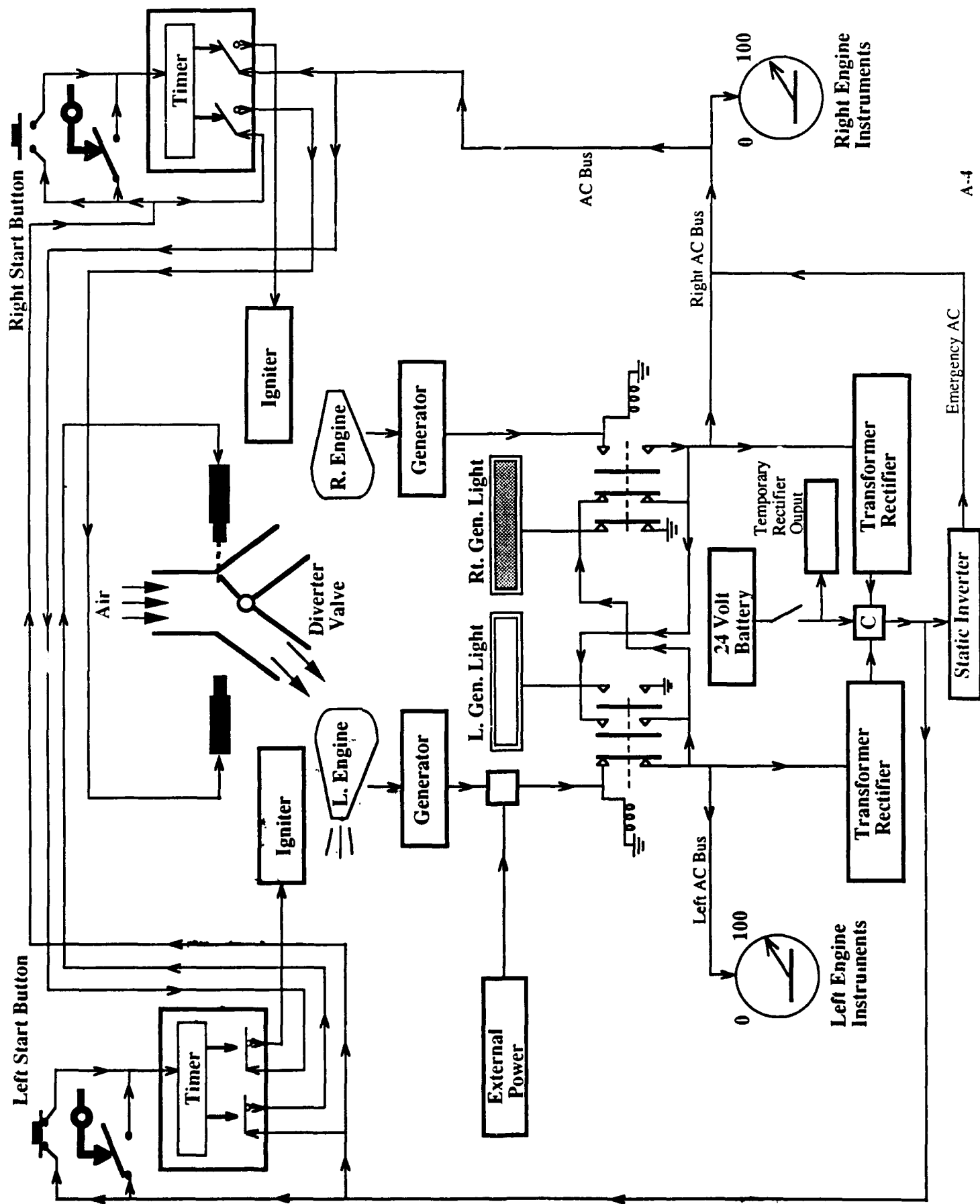
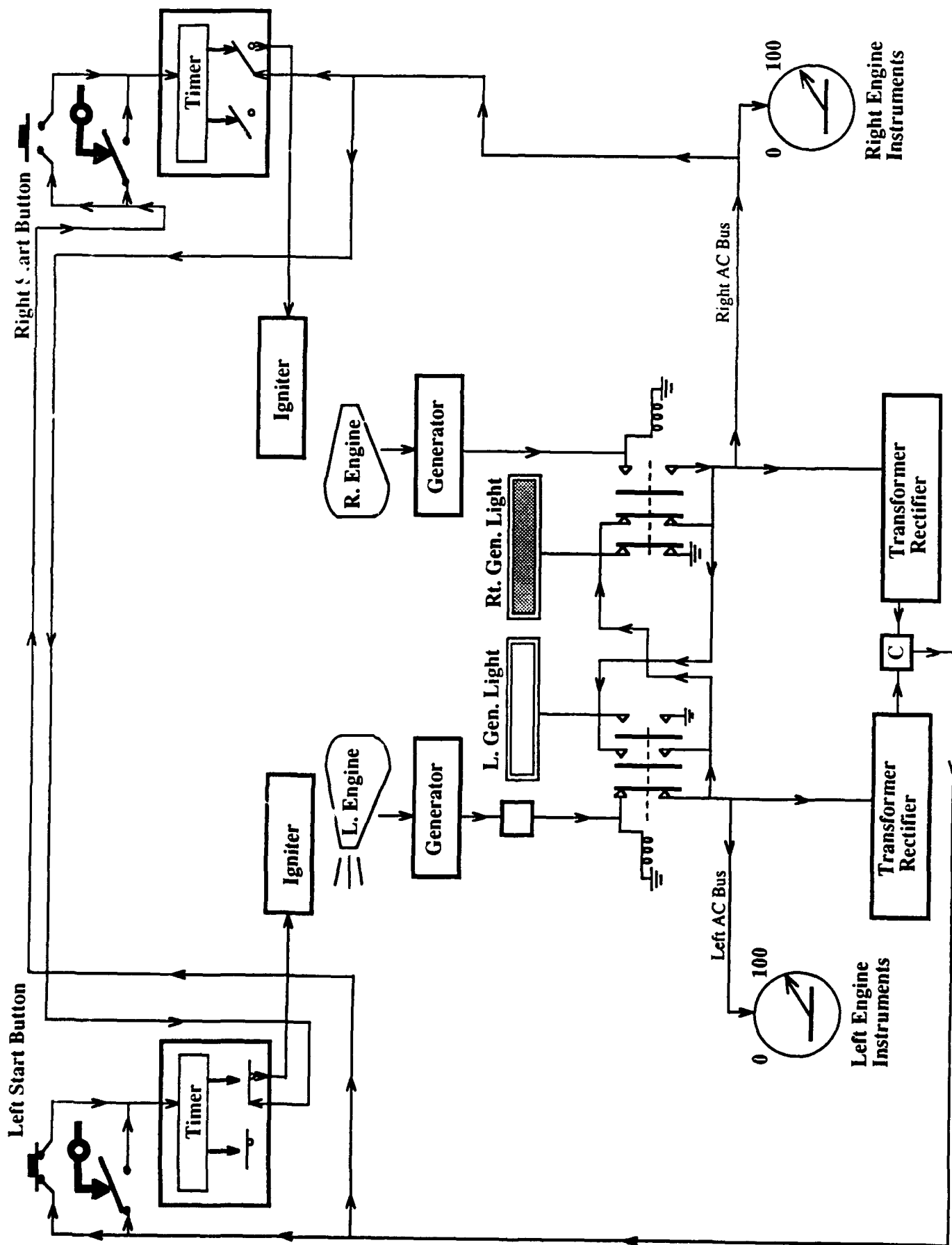
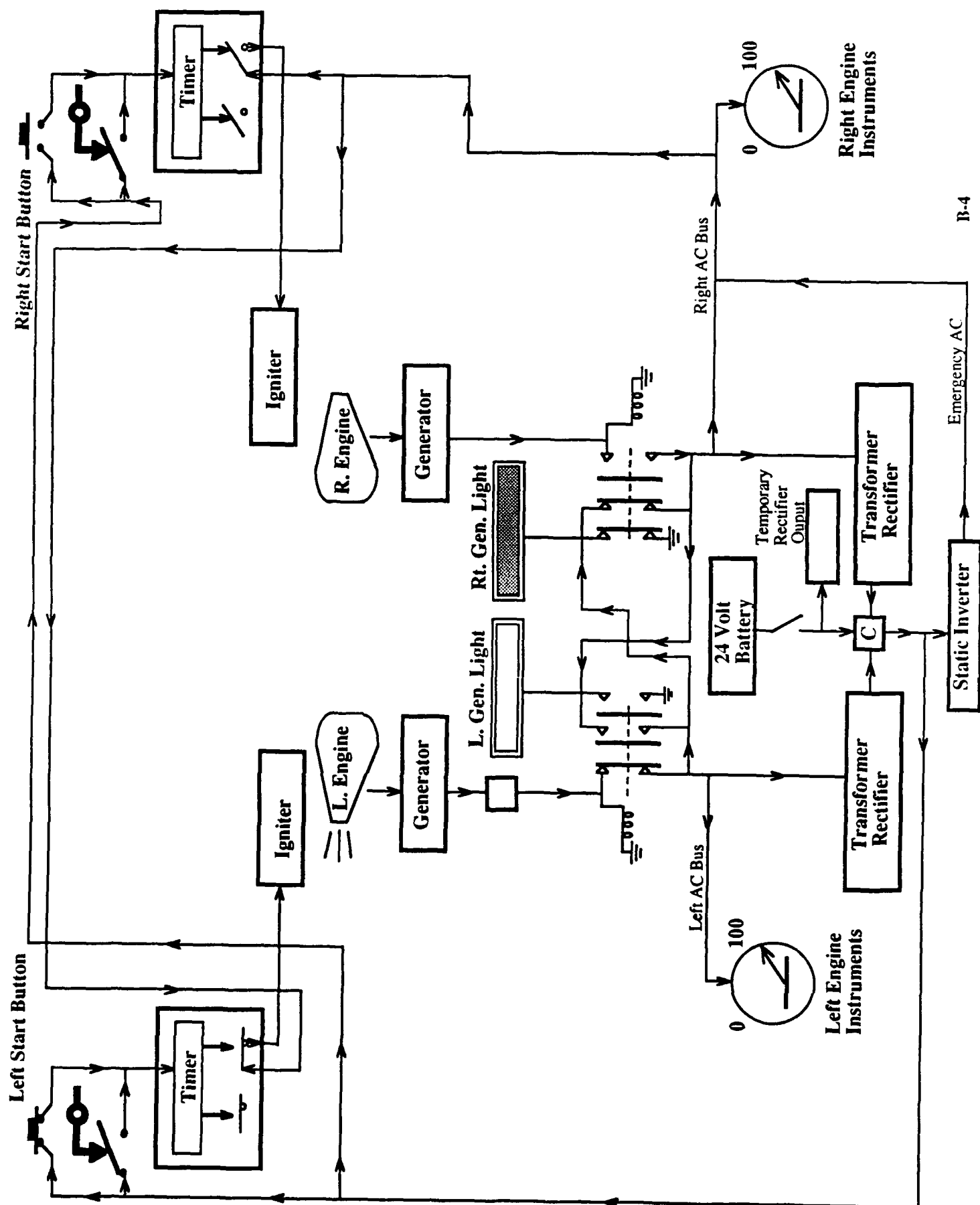


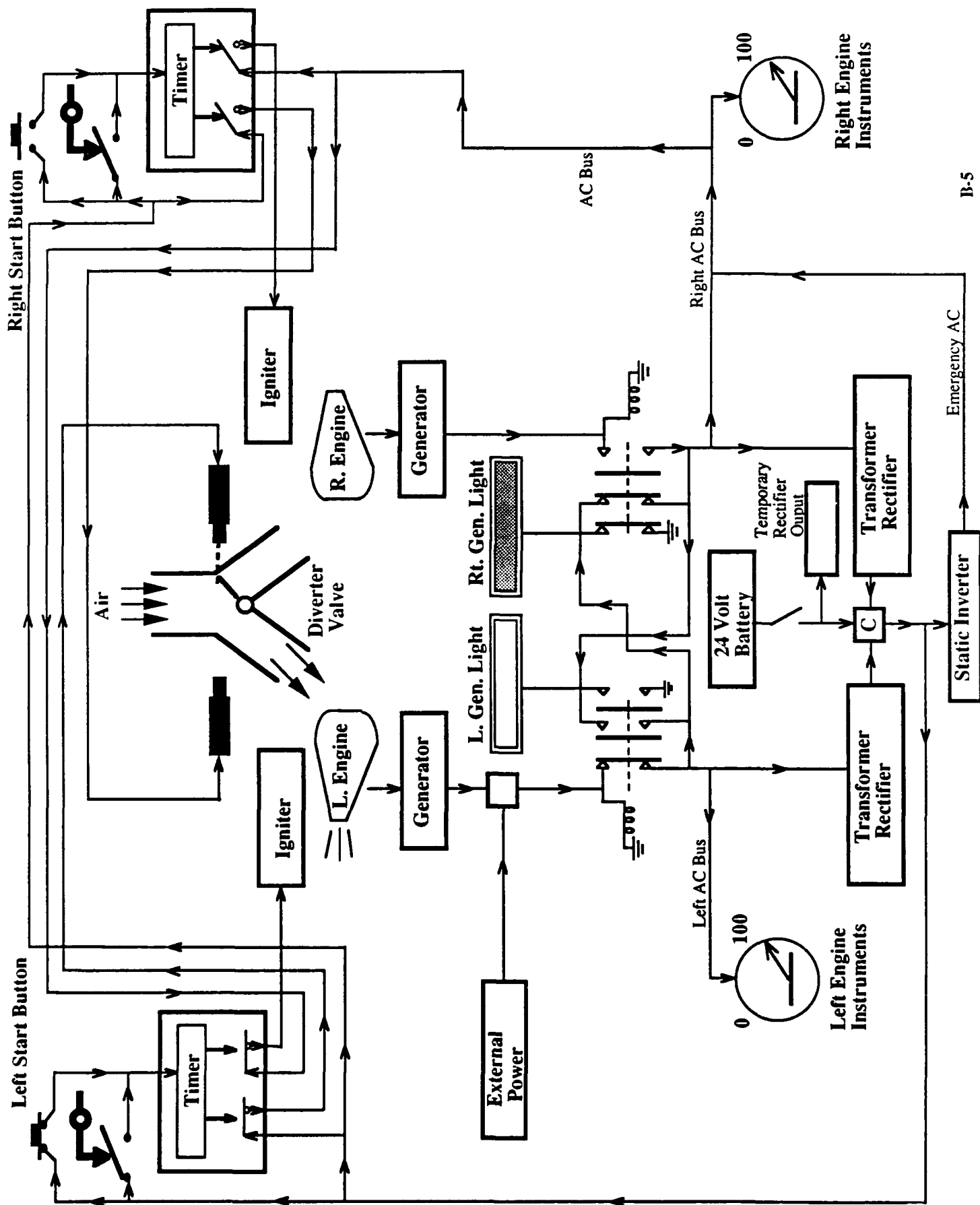
Figure 6. Flow Control System





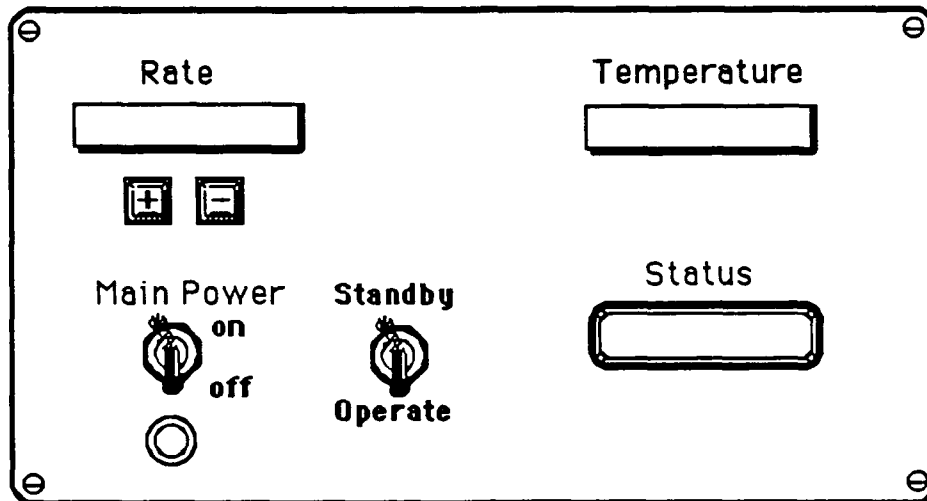






Appendix C

Consider the device whose front panel is shown below.



The graphical representation portrays a system with two digital readouts, for **Rate** (of change) and **Temperature**, two push buttons labelled + and -, a two-position toggle switch, labelled **Main Power**, a similar toggle switch labelled **Standby/Operate**, a power-on/off light, and an indicator labelled **Status**.

An explanation of the general purpose of the device is as follows:

This device is used to control the temperature of cooling water in a nuclear reactor.

The following is intended to be a complete explanation of the *behavior* of the device, in response to operator actions (this explanation purposely avoids any explanation about *what* is going on behind the front panel - it is intended to specify device behaviors only):

Initially, with the power off, all indicators are blank. When the Main Power switch is set to **on**, the power light comes on, the Rate indicator reads the initial value of 0, and the Temperature indicator reads the initial value of 0. The Status indicator remains blank.

The Rate indicator displays the rate at which the water temperature will change, in degrees per minute. Initially, Rate is 0. Each time the + button is pressed, with the power on, the rate increases by 5; each time the - button is pressed, with the power on, the rate decreases by 5. The water temperature does not actually change, however, until the standby/operate switch is set to **operate**.

When the Standby/Operate switch is set to **Operate** (with power on), the temperature changes at the rate indicated. While the rate can be negative, temperature can only go as low as 40.

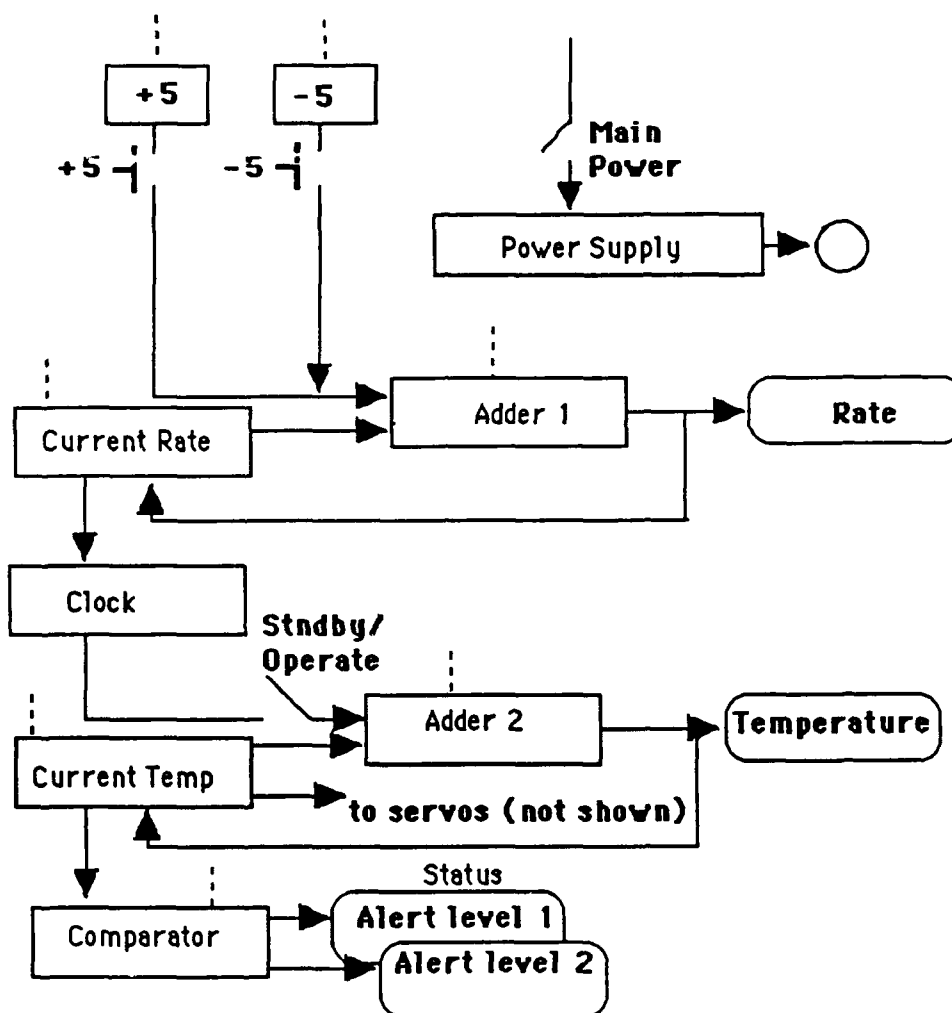
If temperature hits 100, the Status light reads "Alert Level 1"; at 125 it reads "Alert Level 2"; at 160 it reads "Melt Down" and the system ceases to change. If temperature drops below 160, the alert changes back to "Alert Level 1"; if it drops below 125 the indicator returns to a blank condition.

When the standby/operate switch is set back to **standby**, temperature drops at the rate of 1 degree per second until it reaches 40. Whenever the device is put back into operate mode temperature begins to change at the current rate, as shown in the rate indicator.

When power is set to off, the device returns to its status prior to being energized, i.e., all displays are blank. The water temperature continues to drop at 1 degree per second, however. If the device is reenergized the temperature display would reflect the current water temperature. The rate, however, always reads 0 until it is changed, in operate mode.

From this explanation we can infer a few things about how the device works. For one, Rate must somehow be stored in volatile memory, so that it is reset to 0 when the power is turned off. The functions that store and change Rate seem to work as long as power is turned on, but the water is not actually affected until we go into operate mode.

There are many ways that the functionality explained above could be implemented. The figure below illustrates one such way.



The following is intended to be a complete explanation of this implementation of the device (at a high level):

The power supply provides power to all the working modules, as shown with dashed lines. When either rate button, a or b, is depressed a +5 or -5 is added to the current rate by Adder1. As long as power is supplied, the current rate is stored in the "current rate" register and is displayed.

When the standby/operate switch is set to **operate**, the clock begins to count. Each second, the Current Rate (+ or -) is added to the current water temperature by Adder2. The Current Temperature register is also used to control servos (not shown) that actually open and close valves in the reactor.

The Current Temperature register is also accessed by the Comparator, which compares the value to 100 and to 200. If the value is between 100 to 199, the Comparator activates the Alert Level 1 light. If the value is over 200 it activates the Alert

Level 2 light (these two lights are overlaid so that just one shows in the indicator area).

The rule base that implements this device follows:

```
/* Water Cooling System: Temp starts out at 0, with rate of change =0. */
init=1
if changed(Init) clearscren()

meltdown=0
power=0
temp = 0 rate=0
alertlev=0

if (sv_clock) do
  (gotoxy(0,3) print ("Time is: ", sv_clock)
   gotoxy(0,4) print ("Temperature is: ", temp, " ")
   gotoxy(0,5) print ("Current rate: ", rate, " ")
   gotoxy(0,6) print ("Alert level: ", alertlev)
   gotoxy(0,7) print ("Power is ", power))

if changed(sv_keychar) do
  {if sv_keychar=="c" do
    {stopprocess(temp) /* if any */
     rate=0
     if power==0 do {power=1 startprocess(temp,400,rate,meltdown)}
     else do {power=0 startprocess(temp,0,-1)} }

    if sv_keychar=="a" do rate=(rate + 5) * power /* 0 if power is off */
    if sv_keychar=="b" do rate=(rate - 5) * power}

if (temp <= 0) stopprocess(temp)

/* turn on alert messages when temp exceeds safety levels */
if firsttime(temp >100) do
  {gotoxy(0,9) print ("stage 1 alert.") beep() alertlev=1}
if firsttime(temp >200) do
  {gotoxy(0,9) print ("stage 2 alert.") beep() beep() alertlev=2}

/* turn off alert messages when temperature returns to safer regions */
if (firsttime(temp <100) & (alertlev==1)) do
  {gotoxy(0,9) print (" ") alertlev=0}
if (firsttime(temp <200) & (alertlev==2)) do
  {gotoxy(0,9) print ("stage 1 alert.") alertlev=1}

if meltdown do
  {beep() beep() beep() gotoxy(0,10) abort("We just melted down.")}
```

This information would allow one to predict the effect of a number of malfunctions:

An open in the Power Switch would cause power to not come on, as reflected by all blank indicators. A short in the switch would cause power to remain on, even with the switch set to Off. Opens in either rate switch would cause rate to be unchanged when the open switch is depressed. A short would cause rate to increase or decrease without any action by the operator. Complete failure of the +5 or -5 registers would be indistinguishable from opens in their corresponding switches. Alternatively, failures in the +5 or -5 registers would cause the rate to change at some value other than 5 each time the button is pressed. If

the clock failed to operate the temperature would not be affected over time. If the clock operates at the wrong speed, the change in temperature would be incorrect, but would increase and decrease as the rate is positive or negative. If any of the indicators fails completely it would fail to display any value. If Adder1 fails, the rate would not change when either +5 or -5 is pressed. If Adder2 fails, the temperature would not change over time (a symptom indistinguishable from a failed clock). If the Comparator fails the alert lights would either come on at the wrong temperature, or would never come on.

In addition to these failures a host of other possible malfunctions could occur whose effects could not be ascertained with certainty from this technical representation. For example, what would happen if the Current Rate register lost one bit of memory? We could not predict the actual rate of change without knowing more about the way in which the register stores its value. Likewise the clock could fail in ways that would cause seemingly random or intermittent errors, even though it is following deterministic rules of behavior. And, finally, there could be short circuits that produce a topology quite different than that shown in the nominal view, with correspondingly aberrant behaviors.

1990/December

Behavioral Technology Laboratories/Towne, Munro

Dr. Susan E. Chipman
Cognitive and Decision Science
Office of Naval Research
800 North Quincy Street
Arlington, VA 22717-5000
Code 1142CS

Robert Bachman
Office of Naval Research
University of California, San Diego (A-034)
8603 La Jolla Shores Drive
San Diego, CA 92093-0234

Director Naval Research Laboratory
Attn: Code 2627
Washington, DC 30275

Defense Technical Information Center
Bldg. 5, Cameron Station
Alexandria, VA 22314

Mr. Vernon Malec
Navy Personnel Research & Development Center
San Diego, CA 92152

Dr. Wesley Regian
Air Force Human Research Laboratory/IDI
Brooks AFB, TX 78235-5601

University of Southern California
Department of Contracts and Grants
University Park
Los Angeles, CA 9089-0591
ATTN: Ofelia Galvan MC1147