AI Magazine Volume 5 Number 2 (1984) (© AAAI)

STEAMER: An Interactive Inspectable Simulation-Based Training System

James D. Hollan
Edwin L. Hutchins
Louis Weitzman

Future Technologies
Naval Personnel Research and Development Center
San Diego, CA 92152

Abstract

The Steamer project is a research effort concerned with exploring the use of AI software and hardware technologies in the implementation of intelligent computer-based training systems While the project addresses a host of research issues ranging from how people understand complex dynamic systems to the use of intelligent graphical interfaces, it is focused around the construction of a system to assist in propulsion engineering instruction The purpose of this article is to discuss the underlying ideas which motivated us to initiate the Steamer effort, describe the current status of the project, provide a glimpse of our planned directions for the future, and discuss the implications of Steamer for AI applications in other instructional domains

---

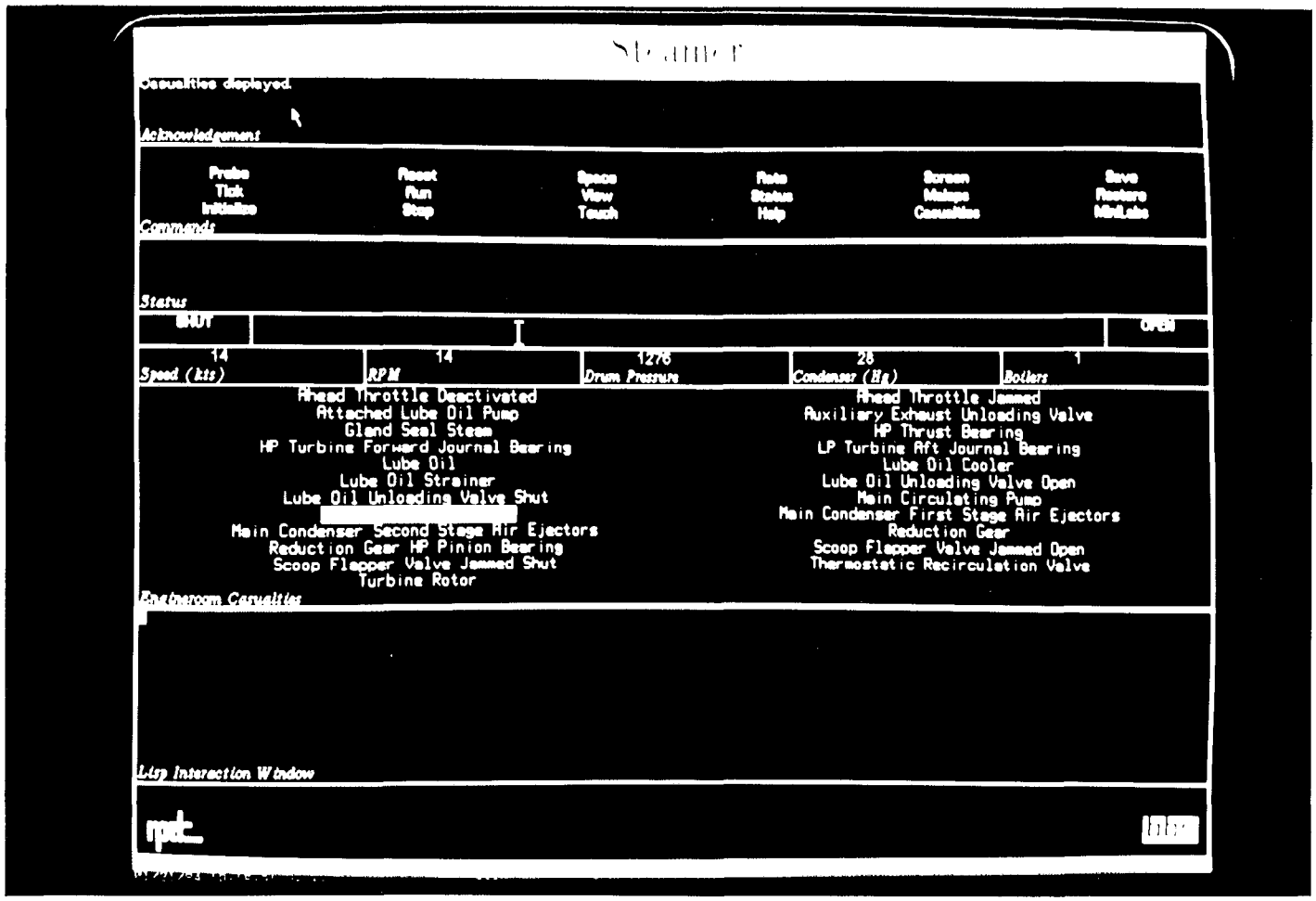
The senior author of this paper, James Hollan, initiated the Steamer effort five years ago in collaboration with Mike Williams, (then at NPRDC and now at IntelliGenetics). Al Stevens of Bolt Beranek and Newman (BBN) joined the collaboration shortly thereafter Since that time, a significant amount of the effort has been conducted under contract to BBN and a number of people have been involved in the project At NPRDC, the Steamer Project currently involves the efforts of James Hollan, Edwin Hutchins, and Louis Weitzman. The BBN side of the effort presently includes Bruce Roberts, who in many ways is the principal software architect of Steamer, Terry Roe (a retired boiler technician chief with 22 years of propulsion operational experience who serves as our subject matter expert and although employed by BBN works with us in San Diego) and the part-time efforts of Albert Boulanger, and Glenn Abrett. Also over the years, we have been most fortunate to have received the capable assistance of Larry Stead (the original implementor of the Steamer Graphics editor; now with Symbolics), Ken Forbus (MIT) and Brian Smith (then at MIT and now at Xerox Parc).

The views expressed here are those of the authors and should not be interpreted as representing the official policies of any government agency

SINCE WE ARE FIRMLY CONVINCED that ideas like people have histories and can only be fully understood in the context of those histories, we will begin by discussing the underlying ideas that motivated us to initiate the Steamer effort. They include the following:

- *Mental Models* — We were and still are caught up in the notion of mental models and of how important it is to understand the models people use to think and reason about complex dynamic physical systems and devices. Without richer and more detailed understandings of the nature of these models, instructional applications will be severely limited.

- *Graphical Interfaces for Interactive Inspectable Simulations* — We believe that graphical interfaces to simulations of physical systems deserve extensive exploration. They make possible new types of instructional interactions by allowing one to control, manipulate, and monitor simulations of dynamic systems at many different hierarchical levels The key idea in Steamer is the conception of an *interactive inspectable simulation.* We have consistently sought to make the system inspectable. This includes not only providing graphical views of the system but also allowing one to inspect various aspects of the procedures for operating the system Interactive inspectable simulations have the potential of being major mechanisms for supporting the development of understandings of *process*



Steamer Interface. This interface is used to select views of the plant to be displayed on the color screen, to control the mathematical model, and to impose casualties. It also provides basic plan status information and the ability to control the throttle. Most control of the plant is accomplished by "touching" icons of the color screen with the mouse.

Figure 1.

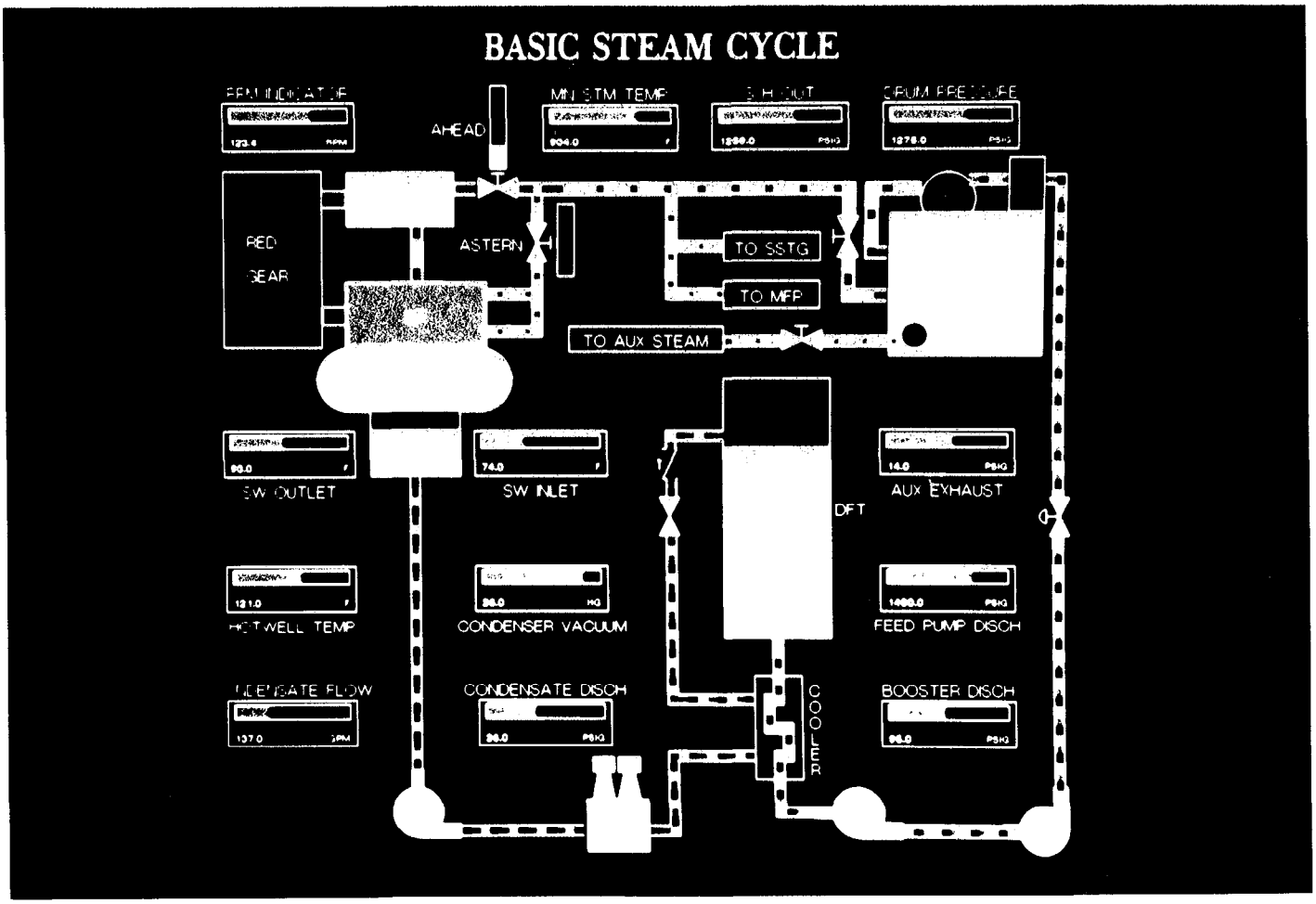
- *Conceptual Fidelity* — We are very much concerned with graphically depicting models that attempt in a fundamental sense to approximate those that experts employ to reason about a physical system. We want to focus on the conceptual rather than physical fidelity of the system to gain a deeper appreciation for how one might support and encourage the development of the mental models people need to understand and reason about dynamic physical systems.
- *Implementation Philosophy* — From the first we wanted to build a non-toy system and to keep the tools we constructed as generic as possible. We felt very strongly that to establish the credibility of these ideas in the training community, we needed a usable system which addressed a real training problem in a complex training domain. It had to be more than a demonstration of the technology's potential, and it had to cover that domain. Also, we have tried to keep the focus beyond just implementing Steamer, but on the more general questions associated with teaching people to understand complex dynamic systems.

We hope these underlying ideas are still evident after the many design, pragmatic, and political decisions that comprise the making of a system like Steamer.

An Overview of Steamer

The Choice of Domain

The fundamental research goal of the Steamer project is to evaluate the potential of new AI hardware and software technology for supporting the construction of computer-based training systems. Just as Papert (1980) holds that one cannot think about thinking without thinking about thinking about something, one cannot evaluate technology in the abstract. We choose to work in the area of propulsion engineering for a number of pragmatic and scientific reasons.



Basic Steam Cycle. This is a high level view of the whole steam plant. One of the important aspects of Steamer is the ability to depict the propulsion system at many different levels.

Figure 2.

1 There is a critical need for improvement in training in this area in the Navy. Thus, it has wide visibility, and we saw the potential for adequate research funding.

2. Alternative forms of training are quite expensive. A high-fidelity simulator costs about 7 million dollars. This has allowed us to explore hardware alternatives which currently are expensive, but which we anticipate will be much less expensive in the near future.

3. We had access to a detailed mathematical simulation model of a common (1200 psi) steam propulsion system This permitted us to focus on the interface, tutorial, and explanation issues which are our major interest

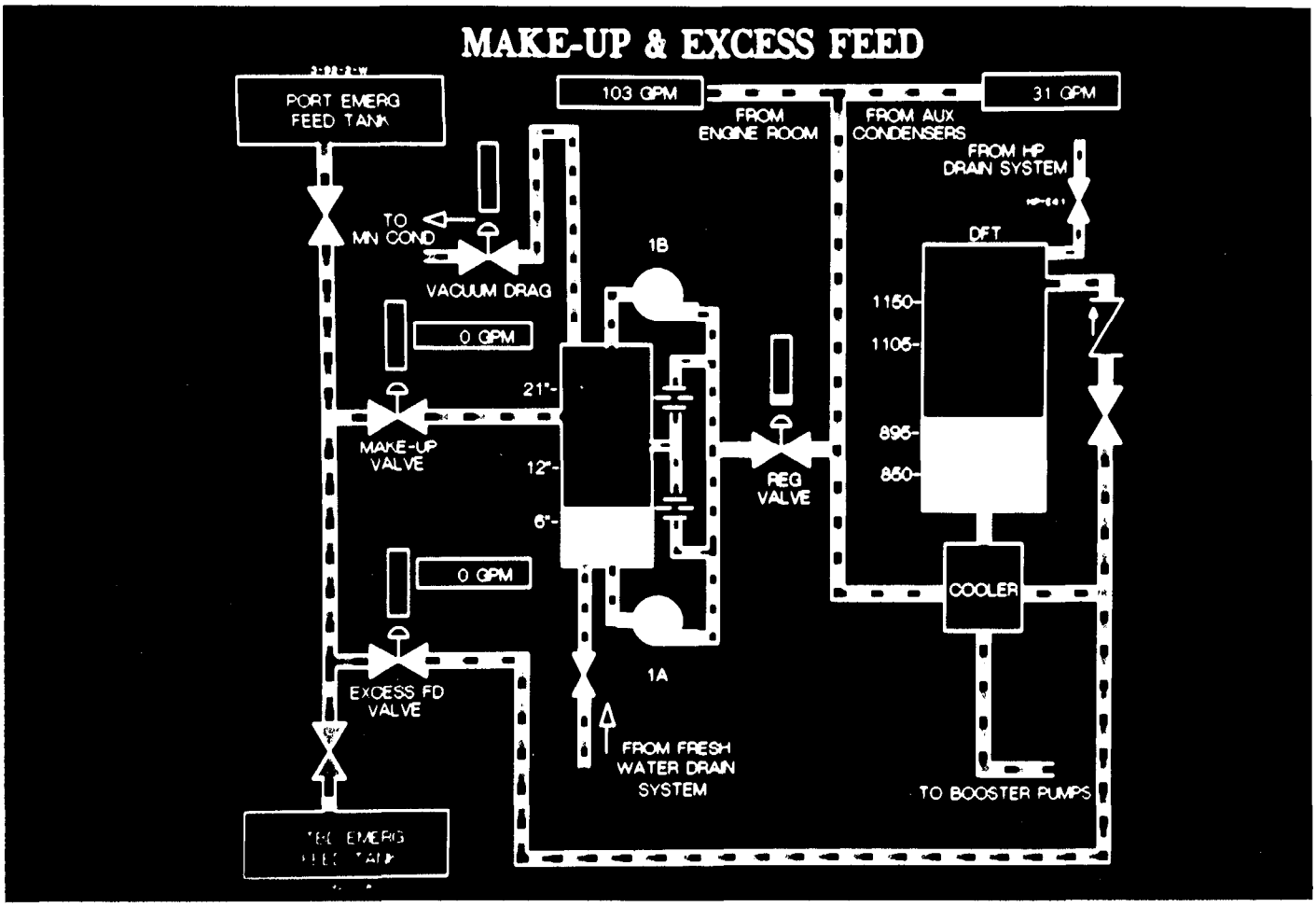
4. We wanted to work in a nontactical area for both personal and pragmatic reasons.

5. We wanted to focus on the use of graphical interfaces to support the development of useful mental models.

6. It seemed engineering domains provided the most instructional leverage from the use of these techniques

Since engineering is an area concerned with designed systems and physical mechanisms, it appeared to be promising for exploring the nature of mental models.

A steam propulsion system is an exceedingly complex physical system. The propulsion spaces account for about one third of the space in most Navy ships. There are thousands of components interconnected by miles of pipes. The operation of the plant is supervised by an engineering officer of the watch and controlled by a team of 16 to 25 individuals who operate in the most trying of circumstances. They often work long hours in a hot, dirty, and quite dangerous environment. Frequently, an individual must cover more than one watch station in a seemingly unending sequence of watches (6 hours on / 6 hours off). The status of the plant is primarily revealed by observing gauges depicting important operational parameters, although operators also make use of other forms of evidence as indicators of plant status, particularly how the plant *sounds* and *feels*. It takes years of instruction and experience to be able to understand and competently operate a propulsion plant. In addition,



Make-Up and Excess Feed. This diagram provides a more detailed view of a subsystem. Adequate coverage of the plant has required approximately 100 diagrams.

Figure 3.

rich robust mental models of the plant are needed to be able to respond to the myriad casualty conditions that can and do arise.

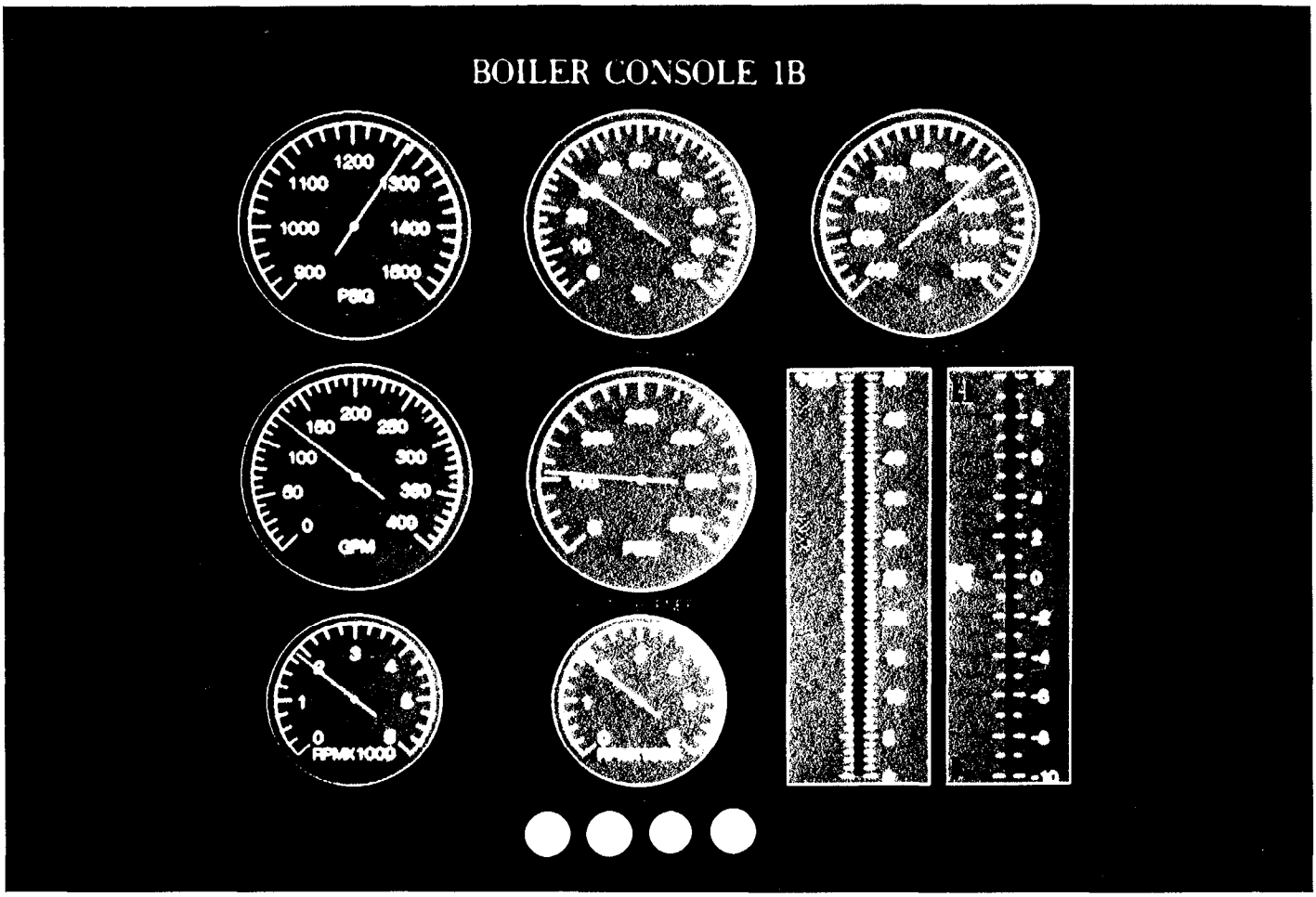
The Steamer Graphical Interface

A principal intuition behind Steamer is that it could be quite valuable to be able to provide a color graphics interface to a simulation of a propulsion plant so one could view and manipulate the plant at a number of different hierarchical levels. Overall control of the system is accomplished by means of the multi-paned window interface depicted, in Figure 1.

This interface provides a view of the overall status of the plant, the ability to make major transitions of plant state, controls for running the mathematical simulation, the ability to impose casualties, and access to a large number of diagrams of the plant.

In the current system we have one hundred color views available and have devised a quite powerful object-based graphics editor for modifying and expanding this set of views. The views range from high-level fairly abstract representations of the plant like the *Basic Steam Cycle* depicted in Figure 2 to views of subsystems such as *Make-Up and Excess Feed* shown in Figure 3. Other views show gauge panels which depict sets of gauges quite like actual gauge panels in a ship, as in Figure 4, and diagrams specifically constructed to reveal aspects of the system not normally available in a real plant but which might be beneficial for understanding some particular aspect of plant operation.

It is important to understand that this graphical interface functions in two ways. First, it reflects the state of components in the simulation. Thus, it reveals whether a particular component is operating or not by means of changes in color or other graphical features of the iconic representation on the color screen. For example, a pump's state is depicted as green if it is operating and red if it is off. The interface also allows a person to view many aspects of the plant that one cannot normally witness in a real plant. One can, for example, see flow rates in pipes. This information provides



Boiler Console 1B Traditional gauge panels like this boiler console panel can also be depicted.

Figure 4

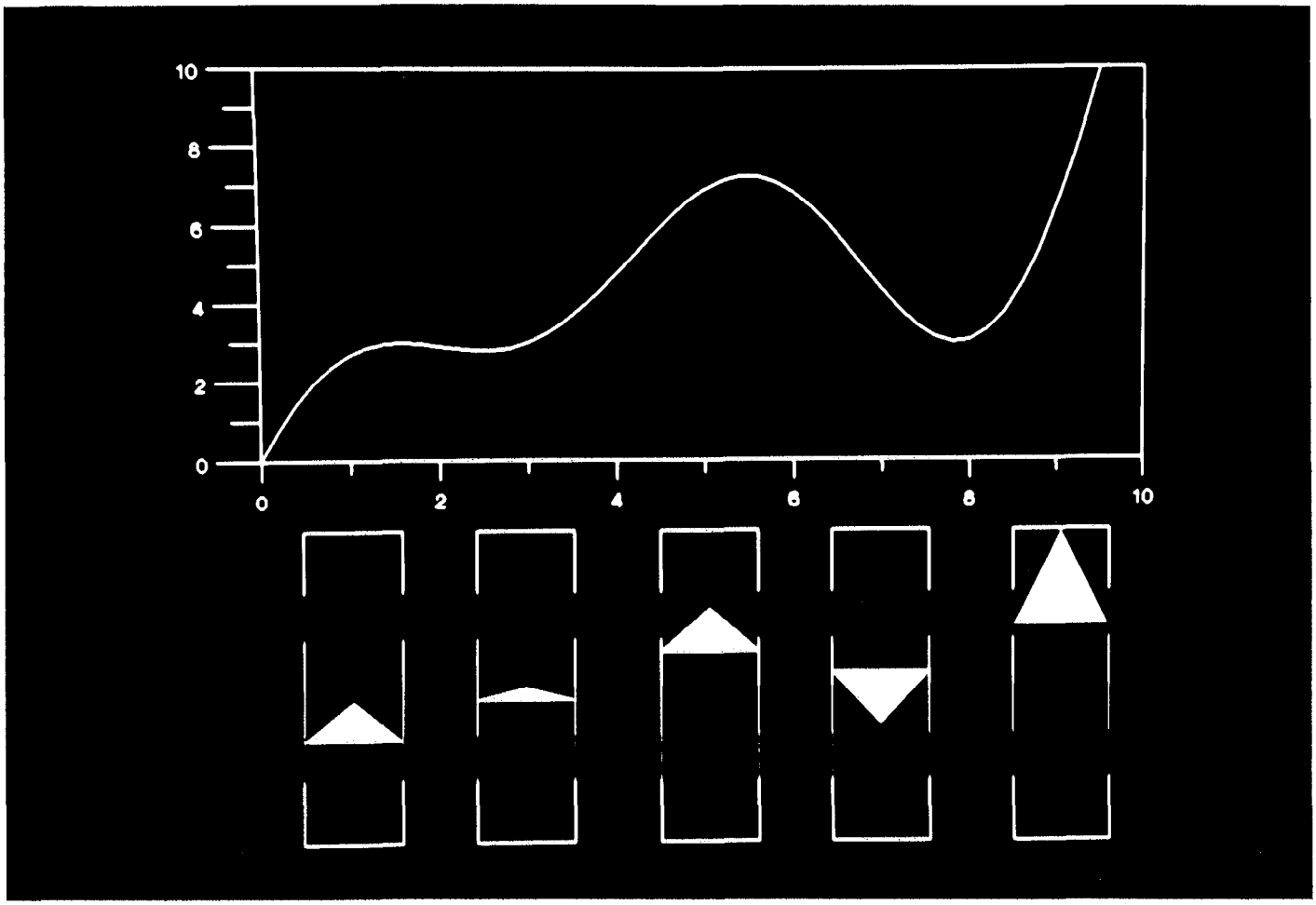
more than just state information. It can make much of the causal topology of the system more directly apparent. We think the ability to depict such characteristics of the plant is quite important for assisting an individual attempting to build a mental model of the operation of the plant. The second function of the graphical interface is to permit control of the components within the simulation. This control is provided by pointing to components with a mouse pointing device and clicking on them.

As an example, consider the Make-Up and Excess Feed Diagram depicted in Figure 3 If one were to increase the level of the deaerating feed tank (DFT[1]) (by pointing to a high position in the tank and clicking), the DFT's level would rise to the position indicated. As a result of this change in tank level, the Excess-feed Valve would go fully open and flows would increase through that portion of the system. Thus, the graphical interface allows both the monitoring of the state of the plant and also its manipulation. It is important to note the potential instructional significance of allowing students not only to interact with things that exist in the real plant (*e.g.*, valves), but also of allowing students to manipulate things which one could not directly manipulate, *e.g.*, DFT levels, which potentially can be of import to supporting the development of an understanding of the operation of a system.

One aspect of the graphical interface arising from our concern with mental models and conceptual fidelity is the ability to provide the user with depictions which approximate the models experts seem to use in reasoning about the system and which have the potential of supporting the development of useful reasoning models. The ability to provide dynamic interactive graphical interfaces is one of the real virtues and powers of the new *display engines* Their high-resolution bit-mapped displays make possible a very different form of explanation which one might term *dynamic graphical explanations.*

These forms of graphical explanation can be of considerable benefit in revealing important aspect of normally opaque systems. For example, one portion of a steam

[1]The deaerating feed tank is a storage tank intended to accommodate fluctuations in demand for water above



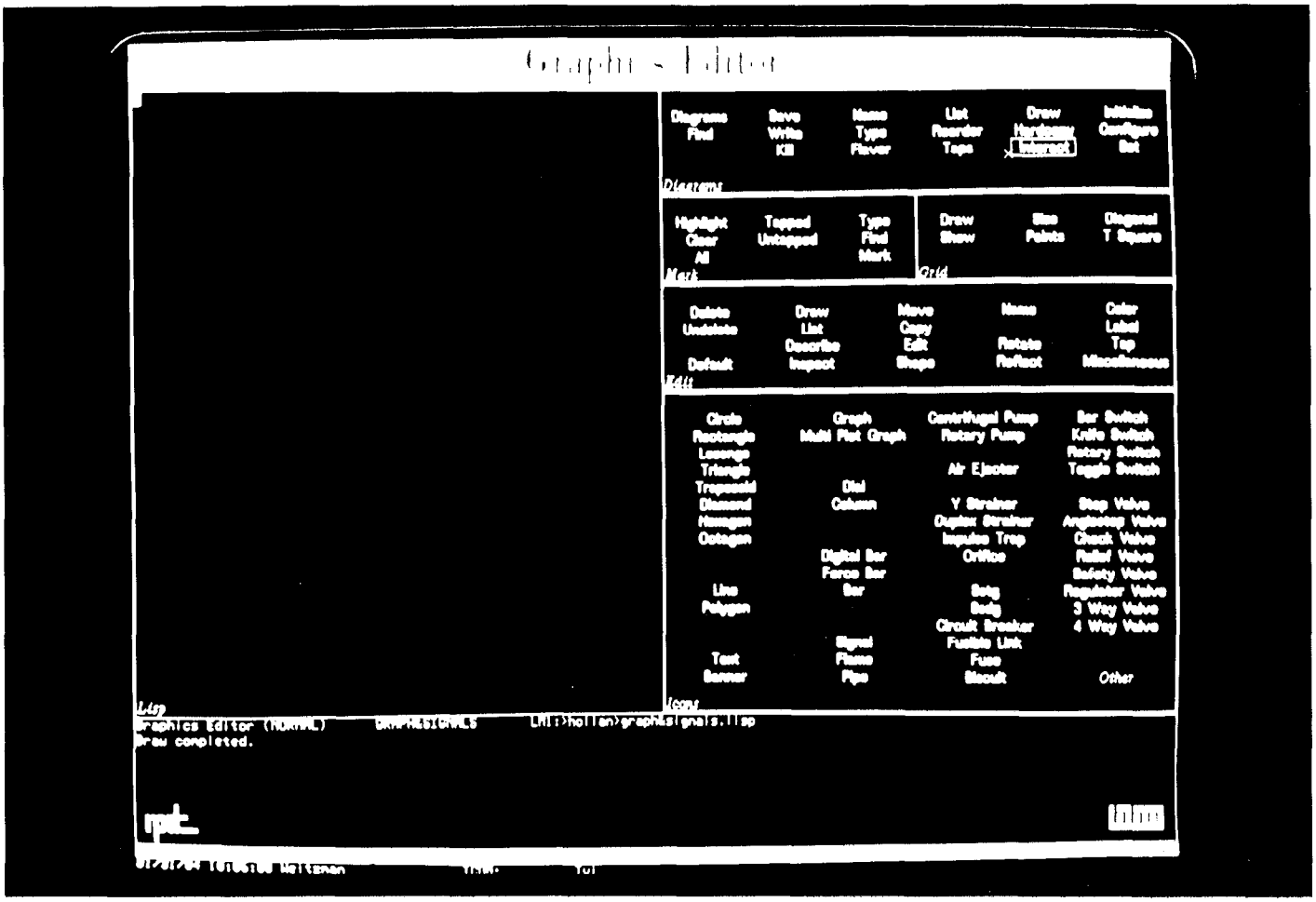
Signal Icon. These signal icons depict rate of change of the variable shown in the graph. We have used them to make visible aspects of automatic control systems which are difficult to see with traditional gauges.

Figure 5.

propulsion system that is quite difficult to understand is the automatic boiler control system. This part of a propulsion plant is a complex system of negative feedback circuits that senses variables such as steam pressure, steam flow, and supply of combustion air and fuel in order to control the rate of firing of a boiler. The internal behavior of the system is characterized by the propagation of pneumatic signals in a world of multiple dynamic equilibria. Normally in a propulsion plant, this system would be viewed by means of a set of gauges like those depicted in Figure 4. The flow of causality and the nature of the response of the system to various perturbations is very difficult to see in the readings of the gauges. Furthermore, in this system the first derivative of the signal is more important than the absolute value of the signal. Thus, what matters is not the actual level of the signal but whether the signal is, in any particular instant, rising, falling or steady. We created a *signal* or *derivative icon* to depict this information explicitly.

Figure 5 shows how this icon would appear at various points in time if it were reflecting the variable whose values are shown on the associated graph. This icon can be used to depict graphically the rate of change of a variable.

We have used the signal icon to create a series of diagrams to assist in explaining the behavior of an automatic boiler control system. Dynamic systems are particularly difficult to explain in language, in part, because of the serial nature of language. However, relationships that are difficult to describe unambiguously in words are often easily depicted graphically. Putting a layer of interface computation between a user and a quantitative model provides a graphical qualitative view of the underlying model. Such a qualitative graphical interface can operate as a *continuous explanation* of the behavior of the system being modeled by allowing a user to more directly apprehend the relationships that are typically described by experts. In a number of views we have instrumented the control air lines with signal icons to reveal the pneumatic signals that are being transmitted. A typical use of these views is to make some throttle change and then single-step the model and watch the transmission of signals. What evolves is a graphical description of the



Graphics Editor Interface. The editor is a powerful facility for creating graphical interfaces to control the underlying simulation and to reflect the state of its components.

Figure 6.

plant's behavior which closely resembles an expert's qualitative explanation of the same perturbation.

It should be clear that these various graphical depictions are appropriate for use by people with very different levels of knowledge about the automatic combustion control system.

The gauge panel is appropriate for an expert who has a rich understanding of the system and needs very little support for his model. A series of signal icons arrayed in an order to fit a causal explanation is more appropriate for someone just developing an understanding of the system. An integrated view with both signal icons and normal gauges provides information to support a bridging from the causal depiction to operational gauge panel displays.

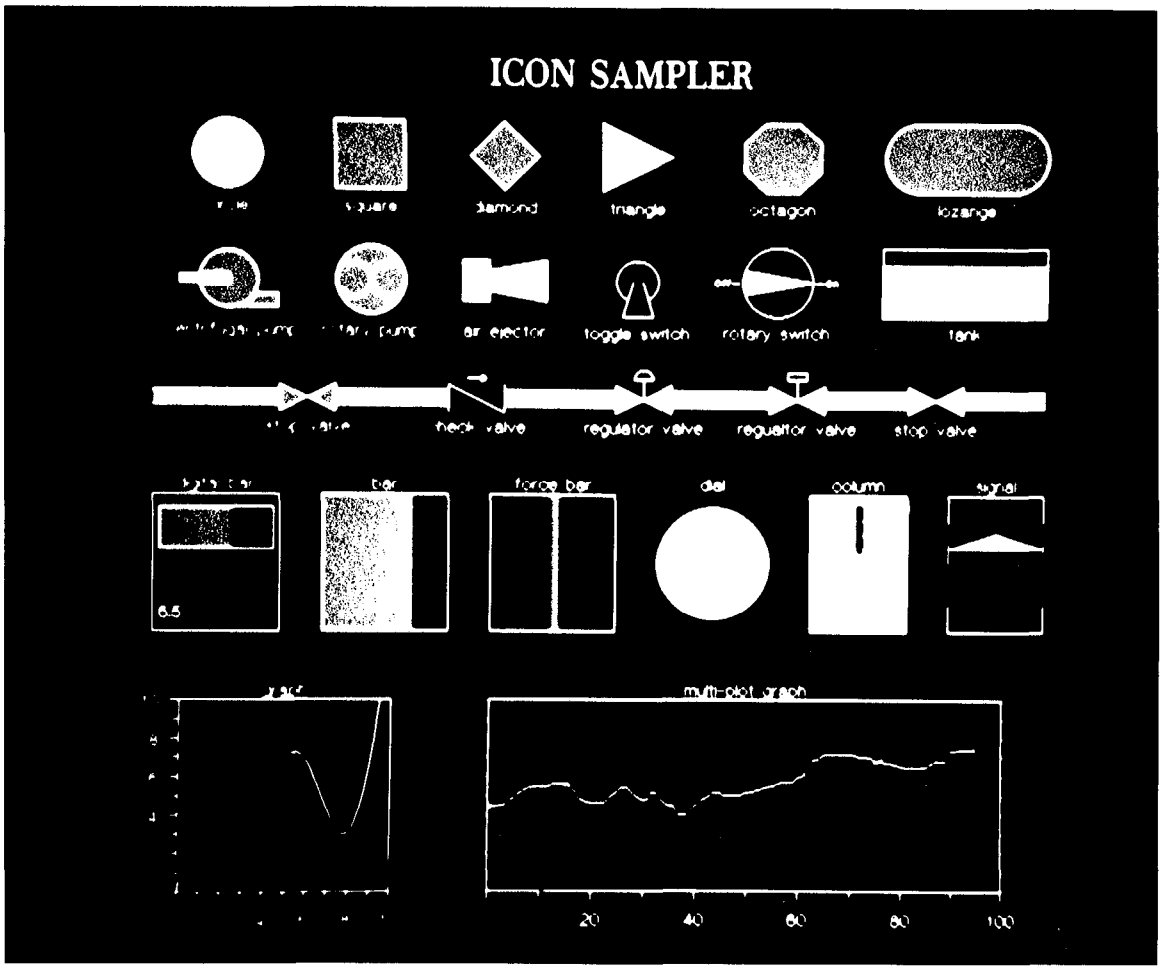
The Steamer Graphics Editor: A flavor is worth a thousand pictures

In order to build and modify the large number of views required to adequately cover the complex propulsion domain, we have implemented an object-based graphics editor.

Figure 6 depicts the user interface to the editor. A user interacts primarily by choosing options from this display and positioning and critiquing graphical icons on the color display.

In order to give you a bit of feel for the graphics editor, we will go through some events in a scenario of constructing a diagram. We choose the icons from a menu of available icons (See Figure 6). The available icons consist of basic graphical primitives (lines, circles, etc.), various indicators (dials, columns, graphs, etc.), and a large set of icons specifically designed to depict objects in the propulsion domain (a variety of pumps, valves, pipes, and electrical components).

A sampler of icons is provided in Figure 7. The user interacts by choosing items from menus and positioning the icons on the color screen. His major actions are *pointing* and *selecting*. When he selects an object and points to a position for it, he immediately gets a specific instantiation of the object with many characteristics defaulted (*e.g.*, the color of a dial, its minimum and maximum values, the number



Sample Icons. Icons are implemented using the object oriented flavors system of Zetalisp. The icons pane in Figure 6 contains a list of all the currently available icons. A subset of these icons is depicted above.

Figure 7.

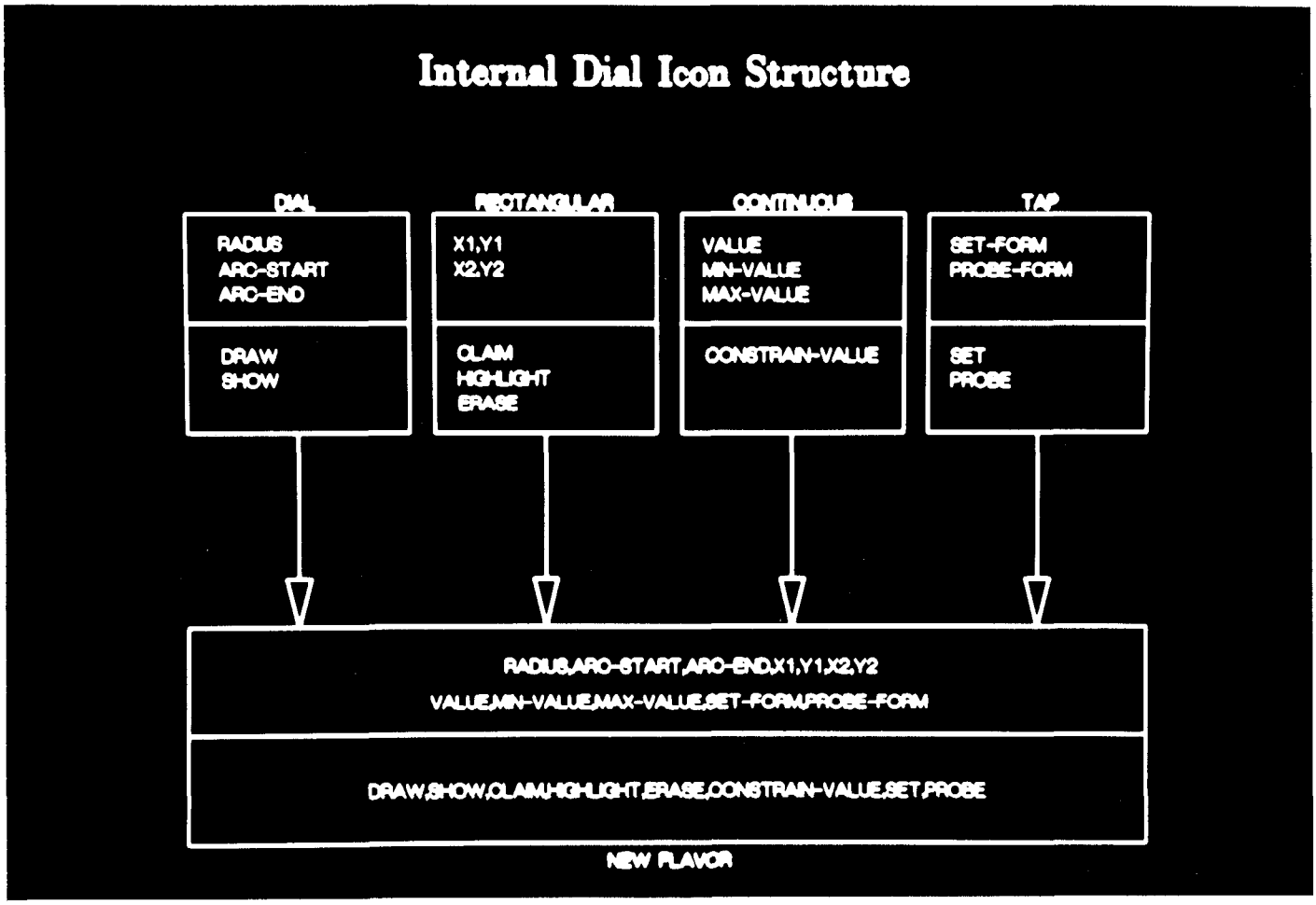
of divisions on its scale, *etc.*). Then, through a process of incremental refinement, he critiques the display, reformulates it and eventually makes it into what he wants. We think this form of interchange is very important. It seems quite natural and allows both the machine and the person to do what each does best. The critique is facilitated by requiring only the choice of different values for parameters which have not defaulted to the required values. What is created as a result of this interaction is not just the color graphic depiction of a diagram, but a program which contains a number of dynamic entities capable of responding to messages and of providing graphical support to an interactive instructional system. The editor then is a facility which makes it possible for a nonprogrammer to create some fairly complex pieces of LISP code.

For purposes of demonstration, consider creating a "dial" for use in a Steamer diagram. You would click on dial on the black and white screen. The cursor would then be taken to the color screen where you would position and size the dial. You immediately would get a default dial on the screen. Then, one would critique that dial by changing its parameters (position, size, scale, font, color, label, etc.) to match those required parameters.

Figure 8 shows some characteristics of a dial which would get created by this simple interaction. Not only are all of the specific details of the dial created, but also, as a particular type of object, it inherits a large collection of messages from the objects out of which it is composed.

A dial, like the other graphical icons, is thus a dynamic object created out of a mixture of more basic elements which can be instantiated to meet the needs of particular applications. It is capable of responding to a variety of commands (messages) to perform specific actions These messages make possible a very powerful generic interface ability which has been exceedingly useful in building Steamer.

Once satisfied with the visual characteristics of a diagram, the user then must tie the components to an underlying simulation or real-time interface. We refer to this process as *tapping*. By selecting an object and clicking on the "tap" in the graphics editor display, you would be provided with a



Internal Structure of the Dial Icon. Each icon is a message-receiving object composed by mixing together flavors. A subset of the 8 flavors used in a dial icon and the 40 instance variables and 122 messages that the flavors contribute is represented in this figure.

Figure 8.

pop-up menu to facilitate the association of the icon to variables in the mathematical simulation. Here you can specify not only the variable(s) whose value(s) will be reflected by the icon, but also the variable(s) in the math model which can be changed as a result of clicking on the icon. The editor provides a variety of mapping options to simplify translation from an underlying variable type (say logical) to appropriate messages to the object (say "ON" or "OFF"). In addition to associating icons with the mathematical simulation, one can also associate diagrams and the icons which compose them with what we term *model augments*. An augment allows additions to correct inadequacies of the simulation model, it supports the writing of more complex tapping code, and it provides stand-alone simulations for diagrams.

An example of the first augment is depicting flows. Flows are not represented in the Steamer math model. It is quite easy though to tie a section of pipe to a function in the augment which computes flow rate based on things which are represented. Typically one would check to see if there is a flow path through the section of pipe and then flow the pipe in proportion to the speed of the appropriate pump. We also have found augments valuable as a mechanism for implementing various mini-labs within Steamer to demonstrate important physical principles involved in the propulsion domain. In many cases the model augment provides the total mathematical model for a mini-lab and runs independent of the large simulation model.

The editor then is a complete system for constructing diagrams and interfacing them to an underlying mathematical model. It has gone through a long period of evolution and refinement. In its present state, it is very usable by computer-naive individuals. In fact, most of the diagrams in Steamer have been originated or refined by a retired boiler technician with more than 20 years of propulsion plant experience, but with no previous computer experience. In a number of tryouts of Steamer in Navy schools, we have found that a short period of training is all that is required for instructors to begin to be able to use the editor productively.

The only problematic thing is tapping diagrams into the math model. Our subject matter expert now has no problem if an appropriate variable representing a component is available. However, for more complicated tappings like those involved in pipes, where one must write a bit of LISP code or where a stand-alone model is required, we must provide programming support.

The graphics editor is an extremely general and powerful tool with potential applications ranging far beyond Steamer. In designing and implementing the graphics editor we have capitalized on the very flexible object-oriented *Flavors System* of Zetalisp. This editor has been used to create all of the Steamer diagrams. We have also used it in some quite different domains to explore its generality. For example, in collaboration with one of our colleagues at UCSD, Dave Rumelhart, the editor has been used to build a graphical interface to a number of parallel distributed models of cognition. In many ways the problems facing a researcher when implementing such models are much the same that we face with Steamer. When there is a complex dynamic system which needs to be understood, researchers can benefit from graphical views of that system at various hierarchical levels. Having a powerful tool like the editor available when actively developing a simulation model can be incredibly valuable. In addition, there are a wide variety of other potential applications of the graphics editor. For example, it would be very valuable for process control applications where one might tie icons to a real-time interface rather than an underlying simulation.

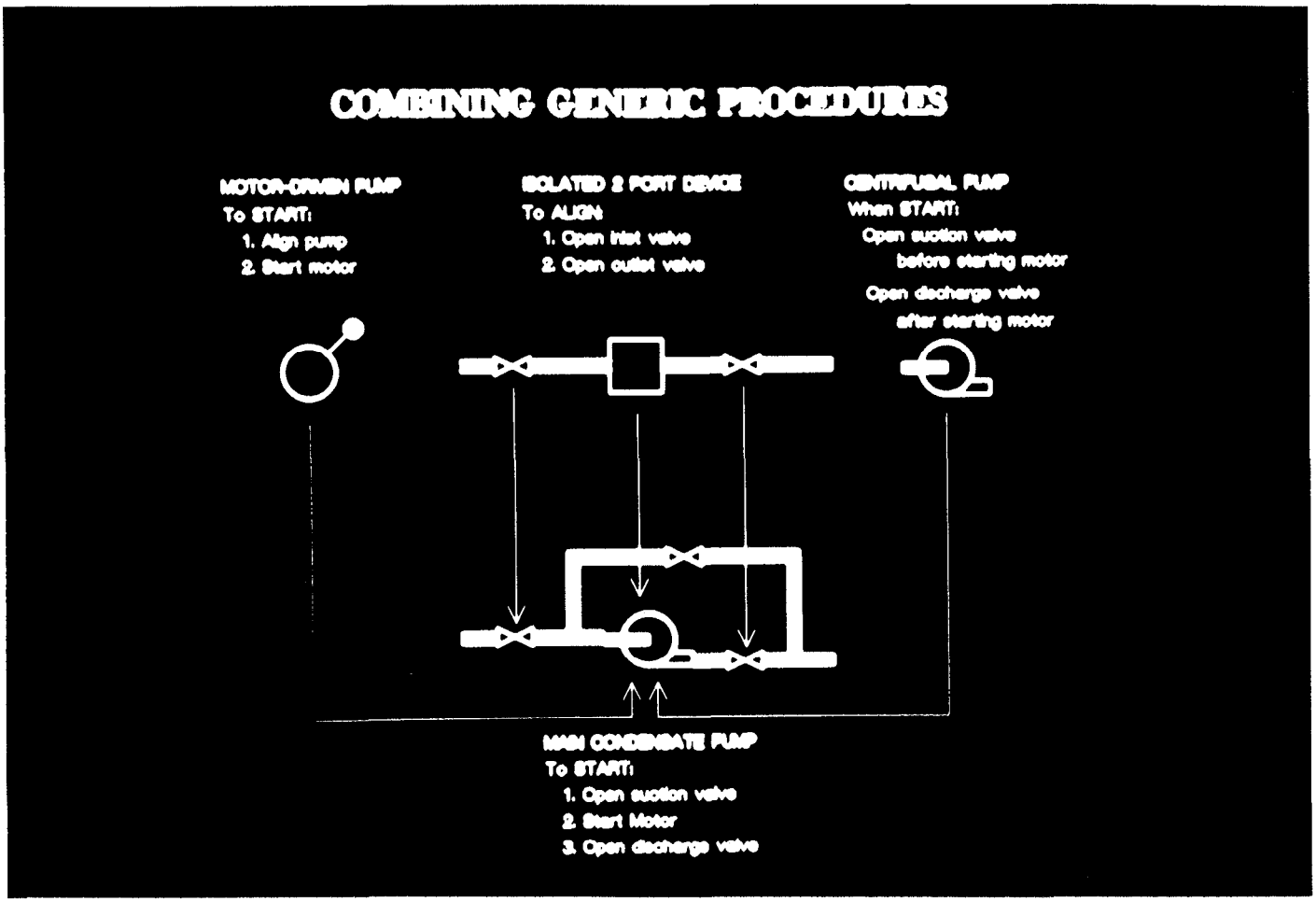
Where is the AI in Steamer?

One might view Steamer as being fundamentally a Cognitive Science rather than an AI research enterprise since we are primarily concerned with how people understand and reason about complex dynamic systems and how interactive graphical interfaces might support the development of useful mental models. On the other hand, we think Steamer is a most important AI application because it called for a very careful look at what aspects of AI technology are ready for application to the design and implementation of computer-based instructional systems. One of the most important AI technologies is the programming environment within which we work. The support provided by this *exploratory programming environment* has made it possible to successfully pursue the construction of a system like Steamer. The system could not have been constructed without the powerful programming tools which AI programming environments make available. Our work has been accomplished within the ZetaLISP environment (Weinreb & Moon, 1981). For an excellent description of a similar exploratory programming environment see Sheil's (1983) recent article.

Of course, using AI tools, either those associated with programming environments or even our use of a truth-maintenance system (McAllester, 1982) for maintaining the consistency of a database of assertions about plant state and student knowledge, doesn't make a project into an AI venture. Most of the genuine AI aspects of Steamer derive from our interest in knowledge representation. We have been very concerned with how one might represent the knowledge involved in the propulsion domain. Much of this has involved efforts to elicit and represent the types of models that human experts use in reasoning about propulsion plant components and procedures (Williams, Hollan, & Stevens, 1982). Considerable effort, thought, and code have gone into issues of representation. We have been concerned with representing the information required to adopt various perspectives on the plant and to maintain a flexible model of the state of the plant and of the student. This is very much the current focus of our efforts. What might be perceived as slowness in getting to this portion of the development is a necessary result of not building a toy system. It would have been easy not to fully complete the earlier graphical phases of the project or to settle for something less than good coverage of the complete propulsion plant or a less general graphics editor.

We would like to give you some examples of the knowledge representation aspects of the project that involve non-graphical aspects of our domain. When you spend from 50 million to a few billion dollars for a Navy ship, you also get an extensive users manual. One form of this manual contains a list of procedures, called the Engineering Operational Sequencing System [EOSS], for operating the plant. This set of manuals, which would fill a good sized book case, contains all of the procedures needed to run the ship's propulsion system. We have been attempting to represent these procedures in a form such that they can be executed and explained at different hierarchical levels. In some sense we face similar problems when creating the views needed to adequately cover the graphical representation of the propulsion domain. We have come to a similar solution: the creation of an editor. This editor, a *Procedures Editor,* makes available sets of generic components, generic procedures, and engineering principles. It allows for the composition of procedures not by writing down their steps, but by performing mappings from abstract generic components and procedures to particular instances.

Figure 9 depicts an example of the types of information that might be involved in the composition of a procedure. The identification of abstract devices and generic components is very much a research activity. That activity requires a deep knowledge of the domain, an appreciation for the generic models that experts use, and the ability to iteratively refine, extend, and reformulate those models which sufficiently cover the EOSS procedures. This is complicated because no such principled process was followed in constructing the existing EOSS procedures. They are a mixture of engineering constraints, rules of thumb, and historical accidents. We are attempting to identify the types of generic components which range from very abstract and general objects (*e.g.,* two-port devices) to less general components (*e.g.,* positive displacement pumps). The editor permits the user to instantiate particular instances of generic procedures and



Composition of a Procedure from Generic Components. We are exploring techniques for composing procedures from generic components and for providing principled explanations.

Figure 9.

to associate the steps that are derived with various underlying engineering principles. The editor provides considerable support for this process. For example, it detects collisions in orderings of steps and allows their resolution. Currently we are discussing what would be the most convenient form of interface to provide for the editor. In some ways a graphical interface seems appealing. In fact, some of the information could be gathered with and support the process of using the graphics editor. For example, knowledge about abstract pumps with suction and discharge valves could be represented. When the user places a pump into a diagram with the graphics editor, the system could assist in making an identification of the associated suction and discharge valves.

An initial version of the procedures editor has recently come to life and has started to meet our needs for putting procedures into the system. It will make it possible to represent procedures in such a way that intelligent use can be made of them by providing the necessary representations to permit the system to adopt different viewpoints on portions of the propulsion system. One might, for example, view a particular pump as an instance of a *positive displacement pump* or as a component of a *pumping station.* We are developing a frame-based representation system that supports multiple perspectives and permits an integration of the vast amount of structural, functional, topological, and graphical information contained within Steamer.

We have also been experimenting with a growing number of interpreters, which we call *presenters,* for allowing Steamer to talk about the propulsion system. Thus, a presenter might take an object and discuss how it is connected to other objects in the system in terms of its physical connections, energy connections, or information connections. Procedures also are objects in the representational system and can be talked about from a variety of perspectives. For example, the system can discuss the components of a procedure, salient procedural fragments which occur in many other procedures (*e.g.,* the securing of an isolation valve or the establishment of a flow path), or the engineering principles which jointly conspire to constrain the ordering of the steps within a given procedure. Considerable work remains to be accomplished

in this area, but currently, we think the procedures editor may turn out to be as valuable and powerful as our graphics editor

In summary, we think there are a number of factors which distinguish Steamer from traditional AI efforts. Most derive from our fundamental cognitive science perspective on creating a training system that will help students come to understand and reason about a complex dynamic system. The effort, at this point, has been dominated not by attempting to build an expert system to replace a person, but rather by a deep commitment to graphics, to interactive graphical forms of support for understanding, to a concern with how to make more and more of the propulsion domain inspectable, and with the creation of generic tools (like the graphics editor) that allow nonprogrammers to create a significant portion of interactive inspectable training systems. These efforts have forced us to be very concerned with explanation as a primary goal rather than something that might be tacked onto the end of an expert system to help it recruit faith in its inferences The types of principled explanations that we need can only derive from a rich and powerful representation of the generic components and procedures that seem to allow a human expert to parse and understand the tremendous complexity of the propulsion domain

Directions for the Future

We are moving in essentially five directions in the future:

- We plan to continue the development of an integrated representation system to tie together the many different types of knowledge currently represented
- We will work on providing Steamer with a consistent interface to create a central point of view and a consistent means of controlling the growing variety of things the system can do and that can be done to it. In short, we want to provide Steamer with a *style* and *consistent personality*
- We will pursue the extension of the generative and reactive aspects of Steamer In particular, we will be providing the diagrams with a form of instructional augmentation containing the information needed to support mixed-initiative interactions with the diagrams. We need to be able to represent consistently the important things that a diagram might reveal to a student, to provide a mechanism for posing questions to the student which might be answered not by typing but by doing things to the diagram, and to provide mechanisms for monitoring a student's behavior while attempting to answer questions. For example, in the *Make-up and Excess Feed diagram* (Figure 3) there are important control relationships between when and how far valves open, based on the levels of the tanks in the system The system might, for example, ask the student to manipulate the diagram to get the Excess-Feed Valve fully open. In our initial evaluation of the system, we have found that having a student put forth hypotheses in this way results in the crossing of an important instructional and motivational threshold
- We plan to pursue the implementation of an expanded student model Presently we have only explored a very limited differential model which notates a student's knowledge of engineering principles. There is much that remains to be done here.
- Finally, in a related project, NPRDC's group plans to pursue a generalization of the graphics editor to include the ability to construct a simulation model interactively. The general notion is to provide default behaviors for objects analogous to the current provision of default visual characteristics This *Behavior Editor* will allow the user to critique the behavior associated with an object in much the same way he or she can currently critique graphical characteristics We would like to see the extent to which a nonprogrammer can construct a simulation program by means of the same sorts of interactions that currently allow a user of our graphics editor to construct a graphical interface

Conclusions, Concerns, and Counsel

We have tried to provide a brief overview of our current and projected efforts with Steamer. More importantly, we have attempted to point out what we see as the underlying ideas in the project:

- Mental Models;
- Graphical Interfaces to Interactive Inspectable Simulations;
- Conceptual Fidelity; and
- Implementation Philosophy

We think we are just now seeing the introduction of the requisite hardware, software, and cognitive theory to permit principled instructional applications We are quite convinced that the major constraints on progress currently are our knowledge of cognition and the methods with which to represent the vast amount and myriad types of domain knowledge needed to support instruction We are also convinced that we will see more and more Steamer-like systems in the future They provide the opportunity for a qualitatively different and superior form of training which focuses on providing the interactions needed to build up useful mental models and understandings of complex dynamic physical systems Such systems can also increase the amount of supervised practice available to students by orders of magnitude

The great promise of the application of artificial intelligence software and hardware technologies to the solution of problems in a variety of domains is a refrain heard often these days. We are both encouraged and fearful of the attention AI is currently receiving. We are encouraged because we too see tremendous promise in the technology and

in the types of explicit computational accounts of cognition emerging from AI and the other cognitive sciences. There is a danger that real developments, substantive though they may be, will fall far short of inflated expectations. Such a turn of events could result in a backlash against AI similar to that suffered by computer-based instruction in the past decade.

It does not require an intellectual historian to see the parallels between the current interest in AI and early interest in computer-based instruction. It was some two decades ago that people first began to advocate the potential of computer technology to provide and improve instruction. Unfortunately, most actual instructional applications of the technology have fallen far short of its promise. We have argued elsewhere that one primary reason is that it is perhaps too easy to see the potential of the technology for instructional applications. It seems to require very little exposure to computation before most people are aware of its instructional potential. Without an appreciation for how much tacit knowledge underlies good instruction performed by human instructors, how difficult it is to make that tacit knowledge explicit, and what kinds of software and hardware are required to support its delivery, it is easy to visualize the potential without really knowing what might be required to make it a reality. As we mentioned earlier, within computer-based instruction we are just now beginning to see the kinds of hardware, software, and most importantly, the explicit computational formulations of cognitive theory essential to principled instructional applications. Without these requisite tools and theories, the types of instructional applications have and will continue to fall far short of the technology's potential. We also are particularly fearful that important research breakthroughs will be used inappropriately to sell aspects of the technology which, while still ripe for further research progress, are not yet ready for application. This could result in a slowing down of important research and development at a time when it should be accelerated.

We also would like to give some counsel on the application of AI to other instructional domains based on our experience with Steamer. In addition to warning about overzealous and uninformed advocates, we would like to put forth the following observations and recommendations:

1 We think it is tremendously important not to rush into applications. Premature application of a fragile technology will instill the same kinds of negative attitudes that premature applications of computer-based instruction did

2 It is exceedingly important to provide sufficient basic research funding to make possible the development of a strong technological base to support future applications. Here we commend a recent letter to the editor in *Science* to your attention Lindamood (1983), puts forth the novel thesis that the primary impact of the Japanese Fifth Generation Project may be *managerial* rather than *technological.* In particular he focuses on the nature of funding of that research endeavor and points out what may be the primary fault of US research funding: *overspecifying and overmanaging the research endeavor* This is typically done in the hopes of obtaining early application results but seems more likely to result not in early application but only poor science.

3 There is a real need for additional centers of excellence for providing a strong scientific and technological base It is vital that mechanisms for transitioning the more promising results also be established as part of these centers. Interdisciplinary centers with ties to academia, industry, and the military are particularly important.

4. Very careful consideration should be given to the choice of a few well-funded initial demonstration systems. Such systems are very important for establishing the credibility of the best research ideas and can serve as important guide posts for subsequent efforts. Given the shortage of experienced people in AI and cognitive science, this seems a particularly wise approach.

5. Finally, much thought and research needs to be addressed to methods for evaluating these new AI-based instructional systems. There are some tremendously hard problems involved Here we want to include not only the traditional view of evaluation but also evaluations of how these systems effect the communities within which they are placed. Powerful technologies sometimes have unanticipated beneficial or detrimental consequences An evaluation of an application of this technology that considers only effects on the problem the technology is intended to solve may miss important consequences of the use of the technology

References

Lindamood, G. E. (1983) Japanese Computer Project Science: 221:1008.

McAllester, D. A (1982) Reasoning utility package user's manual AI Memo 667, Massachusetts Institute of Technology.

Papert, S (1980) *Mindstorms: Children, computers, and powerful ideas.* New York: Basic Books.

Sheil, B (1983) Power tools for programmers *Datamation* 29:131-144.

Weinreb D & Moon D (1981) *LISP Machine Manual* Cambridge, MA: MIT Artificial Intelligence Laboratory

Williams, M , Hollan, J., & Stevens A. (1983) Human reasoning about a simple physical system. In Gentner, D., & Stevens, A. (Eds.) *Mental Models* Hillsdale, New Jersey: Erlbaum

---

A complete series of technical reports and papers concerning Steamer are available from the author. *Mailing address:* Future Technologies, NPRDC, San Diego, CA 92152. *Arpanet address:* HOLLAN@NPRDC.