AN OVERVIEW OF ΦNIX

David Barstow, Roger Duffey, Stephen Smoliar, Stanley Vestal

Schlumberger-Doll Research
Old Quarry Road
Ridgefield, Connecticut 06877

ABSTRACT

ΦNIX is an automatic programming system being developed for use by petroleum scientists who may not be knowledgable about computers. As a first step, a system has been implemented to assist in the trial-and-error process of developing new models and techniques for quantitative interpretation of well logs. The user interface exploits graphical techniques to enable petroleum scientists to describe their models in the natural concepts of the domain. The resulting specification can be implemented in any of several different target languages. The system is in active use by petroleum scientists, who find that it has significantly reduced the time to get feedback on hypothesized models.

I INTRODUCTION

ΦNIX is an automatic programming system whose development is guided by three basic principles:

- ΦNIX users need not be knowledgable about computers or computer science.
- ΦNIX must be able to produce prototype software quickly enough for the user to cycle many times through the trial-and-error loop of initial specification.
- ΦNIX must be able to produce software that is sufficiently general, robust and efficient for routine use.

We do not believe that the software development process is sufficiently well understood for general-purpose systems based on these principles to be built in the near future. In particular, we believe that the goal of generality has led traditional automatic programming research to ignore significant software development issues. Principal among these are:

- software specification by users who are not knowledgeable about computer science
- software complexity due to size rather than algorithm
- software evolution as a fundamental characteristic of software use

With the hope of addressing these issues, we have adopted an experimental approach in which we select particular "real-world" programming tasks and develop special-purpose systems for them, guided by the principles given earlier. In the rest of this paper, we will try to convey our goals and approach by describing our first experimental system from a user's viewpoint, followed by a brief discussion of specific issues we plan to address in future experiments.

II ϕ_0, AN INITIAL EXPERIMENT

For our first experiment, we have chosen the problem of writing prototype software to support the trial-and-error process with which petroleum scientists develop models and techniques for interpreting the petrophysical data measured by logging tools.* Quantitative log interpretation is usually done on the basis of *models* which are statements of relationships between the measured information and the desired information. Many such models can be described mathematically as equations in which the terms denote the log readings or the desired numeric quantities. The models themselves are the result of a variety of concrete experiments and theoretical investigations.

In developing our initial system, called ϕ_0, we have had two primary concerns:

- Since the user (a log analyst) may be neither experienced in nor comfortable with traditional computer interfaces, it was important for ϕ_0 to "speak the user's language". Most importantly, we did not want to require the user to learn some kind of command

* Oil well logs are made by lowering instruments (called tools) into the borehole and recording the measurements made by the tools as they are raised to the surface. The resulting logs are sequences of values indexed by depth. Logging tools measure a variety of basic petrophysical properties (*e.g.*, the resistivity of the rock surrounding the borehole). Petroleum engineers, geophysicists and geologists are typically interested in other kinds of information which cannot (now) be measured directly (*e.g.*, water saturation, the fraction of the rock's pore space occupied by water as opposed to hydrocarbons). Log interpretation is the process of computing the desired information from the measured information.

language.

- Since the user's data exist on a variety of computing environments with different computational abilities, it was important for ϕ_0 to be able to implement the user's model in a variety of target languages. However, the user should be required, at most, to simply indicate the target environment, with ϕ_0 handling all of the details.

Perhaps the best way to describe ϕ_0 is from the perspective of the user, a log analyst not necessarily knowledgable about computers or computer science. In effect, ϕ_0 provides him/her with a "model editor": facilities for developing and modifying the different aspects of an interpretation model. Each of these facilities includes high level operations, expressed in the natural concepts of the domain.

The attached picture shows the ϕ_0 user interface** during the process of developing a particular interpretation model (an unrealistically simple model has been chosen for the sake of this presentation). Different windows are used to show different aspects of the model.

The window entitled "Geological Model: CORI" contains a diagramatic description of the universe which has been assumed for this particular model: the rock around the well consists of two types of solid (calcite and dolomite) and two types of fluid (water and oil). A geological model corresponds to certain equations which constitute part of the overall interpretation model. In this case, the equations are:

$$1.0 = V_{sol} + V_{flu}$$

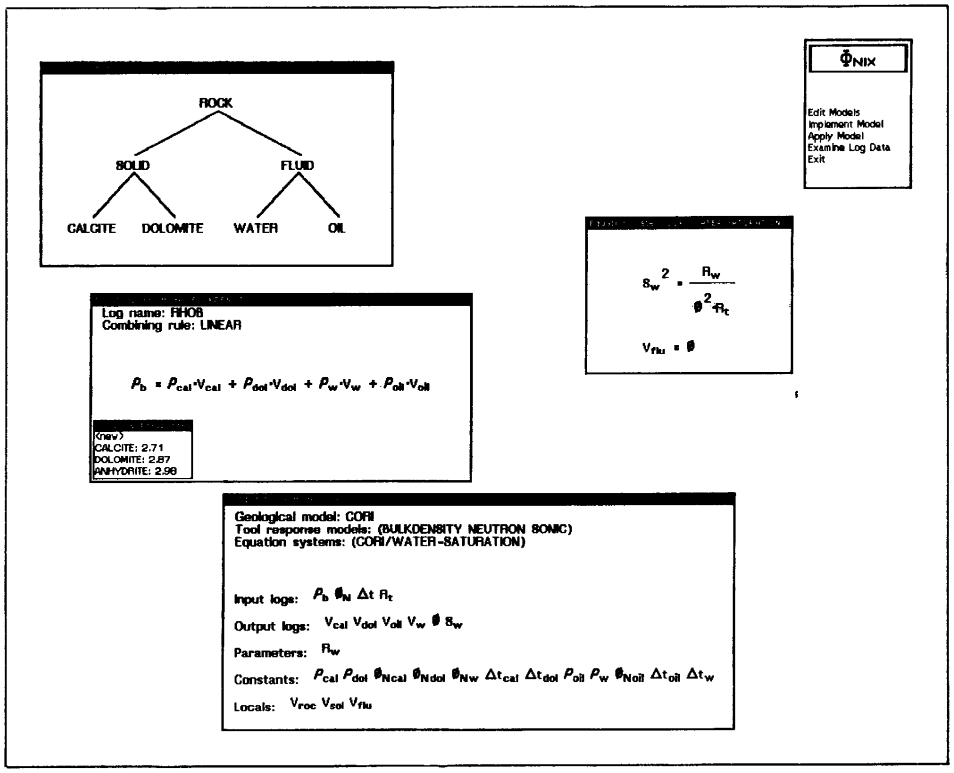
$$V_{sol} = V_{cal} + V_{dol}$$

$$V_{flu} = V_w + V_{oil}$$

where V_x represents the volumetric fraction of material x. By stating this assumption, the user has laid the foundations for the other aspects of an interpretation model. In order to allow the user to develop alternative geological models, ϕ_0 includes a graphics-oriented tree editor. For example, to indicate that there are two types of hydrocarbons present, the user could point (using the mouse) to the OIL node and replace it (through selections from a command menu) with a HYDROCARBON node whose children are OIL and GAS nodes.

The window entitled "Tool Response Model: BULKDENSITY" shows the equation which relates the measured tool response (ρ_b) to the responses the tool would make in the presence of 100% concentrations of the materials which make up the geological model (ρ_x denotes the response of the tool to material x). A table of responses to particular minerals is also shown. ϕ_0 includes an editor for tool response models; for exam-

** ϕ_0 currently runs on a Xerox 1100 Scientific Information Processor which includes a 1024 by 808 black and white pixel screen. ϕ_0 is written in Interlisp-D, a Lisp dialect with a variety of primitives for dealing with bitmaps, windows, menus, and a "mouse" pointing device.



$$S_w^2 = \frac{R_w}{\phi^2 \cdot R_t}$$

$$V_{flu} = \phi$$

Log name: RHOB
Combining rule: LINEAR

$$\rho_b = \rho_{cal} \cdot V_{cal} + \rho_{dol} \cdot V_{dol} + \rho_w \cdot V_w + \rho_{oil} \cdot V_{oil}$$

<new>
CALCITE: 2.71
DOLOMITE: 2.87
ANHYDRITE: 2.98

Geological model: CORI
Tool response models: (BULKDENSITY NEUTRON SONIC)
Equation systems: (CORI/WATER-SATURATION)

Input logs: ρ_b ϕ_N Δt R_t

Output logs: V_{cal} V_{dol} V_{oil} V_w ϕ S_w

Parameters: R_w

Constants: ρ_{cal} ρ_{dol} ϕ_{Ncal} ϕ_{Ndol} ϕ_{Nw} Δt_{cal} Δt_{dol} ρ_{oil} ρ_w ϕ_{Noil} Δt_{oil} Δt_w

Locals: V_{roc} V_{sol} V_{flu}

ple, one editing activity is to add or change the response of a tool in a particular mineral.

Geological models and tool response models are two of the most common ways that log analysts use to describe the equations of an interpretation model. However, not all equations are appropriately described using such techniques. For such cases, ϕ_0 provides an equation editor. The window entitled "Equation System: CORI/WATER-SATURATION" illustrates the editor being used to describe a water saturation equation for use in conjunction with the geological model given earlier. The editing commands are generally oriented to particular subexpressions of the equations. For example, two subexpressions may be swapped by pointing to each with the mouse after selecting the "swap" command from the command window.

The window entitled "Computational Model: CORI" shows how the user may combine together the different aspects of an interpretation model to give a complete model. In this case, the user has selected a single geological model, three tool response models, and the water saturation equations. By putting these pieces together, the user has, in effect, specified a model consisting of eight equations in eight unknowns. The typical use of such a model is to compute, for each depth in a well, the relative concentrations of the minerals of the geological model, given the tool readings at that depth. This use is described in the rest of the computational model window: the input and output logs are the inputs and outputs for each depth; parameters are values the user has chosen for run-time specification; constants are values which ϕ_0 has stored in its knowledge base about log interpretation; locals are terms which appear in the equations but which the user has chosen not to see as outputs.

At this point, the user has finished his/her part of the job; the process of implementing the model as a program in a particular target language is "merely" an exercise in mathematical manipulation and programming and is left to ϕ_0 to complete.***

III DISCUSSION

ϕ_0 is now in active use by petroleum scientists. They have found the rapid feedback to be of great value in their interpretation development activities. Perhaps more importantly, however, they have found that they can concentrate their energies much more directly on the concepts of log interpretation rather than being overly concerned with figuring out the right way to express such concepts in traditional programming terms.

Thus, ϕ_0 seems to have addressed our first two principles quite well. By focusing on pure equational models, however, we have avoided the third principle: the ability to produce software that is sufficiently general, robust and efficient for routine use. Future work on ΦNIX will address this problem directly. This will require research in three general directions:

- Specification: the development of convenient facilities for expressing the real-world considerations that make software much more complex than simple equational models. We are currently exploring the use of an object-oriented specification technique as a way of dealing with such complexities; the initial results look promising [2].
- Programming knowledge: ΦNIX will clearly require access to several types of programming knowledge in addition to the algebraic manipulation knowledge contained in ϕ_0. Chief among these are knowledge about certain aspects of numerical analysis and knowledge about the efficiency trade-offs among different programming techniques. Perhaps the most interesting question concerns the degree to which knowledge about log interpretation itself will be required during synthesis: is the information contained in the specification sufficient for synthesis, or must the synthesizer understand the domain to a greater degree in order to decide among implementation alternatives?
- System development repository: During the synthesis process, ΦNIX will be making a large number of decisions. Both to validate the target code and to facilitate evolution of the code to accomodate changes in the specification, it will be necessary to keep detailed records about the decisions and the motivation for them.

We are hopeful that addressing these issues will help automatic programming research to have a significant impact of the real world on software engineering.

*** Of course, the process of enabling ϕ_0 to do this exercise was a major effort, involving the codification of several kinds of programming knowledge, ranging from techniques for algebraic manipulation to the syntax of three significantly different target languages (LISP, FORTRAN, PROSE). The details of how ϕ_0 works internally are available elsewhere [1]

REFERENCES

[1] Barstow, D., Duffey, R., Smoliar, S., Vestal, S. "An automatic programming system to support an experimental science", Sixth International Conference on Software Engineering, Tokyo, Japan, Sept. 1982.

[2] Smoliar, S. "Specifying logic and control: an object-oriented approach." (in preparation).