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 knowledgeable 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 knowledgeable 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 φ0, 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 φ0, 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 φ0 to "speak the user's language". Most importantly, we did not want to require the user to learn some kind of command
language.

- Since the user’s data exist on a variety of computing environments with different computational abilities, it was important for Φ₀ 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 Φ₀ handling all of the details.

Perhaps the best way to describe Φ₀ is from the perspective of the user, a log analyst not necessarily knowledgeable about computers or computer science. In effect, Φ₀ 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 Φ₀ 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:

\[
1.0 = V_{\text{w}} + V_{\text{oil}}
\]

\[
V_{\text{sat}} = V_{\text{water}} + V_{\text{oil}}
\]

\[
V_{\text{fract}} = V_{\text{water}} + V_{\text{oil}}
\]

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, Φ₀ 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 \(p_x\) to the responses the tool would make in the presence of 100% concentrations of the materials which make up the geological model (\(p_x\) denotes the response of the tool to material \(x\)). A table of responses to particular minerals is also shown. Φ₀ includes an editor for tool response models; for exam-
They have found the rapid feedback to be of great value in their interpretation development activities. Perhaps overly concerned with figuring out the right way to the concepts of log interpretation rather than being can concentrate their energies much more directly on more importantly, however, they have found that they

Of course, the process of enabling $\phi_0$ 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 $\phi_0$ works internally are available elsewhere [1]

III DISCUSSION

$\phi_0$ 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, $\phi_0$ 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 $\Phi\text{NIX}$ 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: $\Phi\text{NIX}$ will clearly require access to several types of programming knowledge in addition to the algebraic manipulation knowledge contained in $\phi_0$. 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, $\Phi\text{NIX}$ will be making a large number of decisions. Both to validate the target code and to facilitate evolution of the code to accommodate 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.

REFERENCES
