N90-27287

KIPSE1 - A KNOWLEDGE-BASED INTERACTIVE PROBLEM SOLVING ENVIRONMENT FOR DATA ESTIMATION AND PATTERN CLASSIFICATION

Chia Yung Han*, Liqun Wan+, William G. Wee+
*Department of Computer Science
+Department of Electrical and Computer Engineering
University of Cincinnati
Cincinnati, Ohio 45221

ABSTRACT

A knowledge-based interactive problem solving environment called KIPSE1 is presented in this paper. The KIPSE1 is a system built on a commercial expert system shell, the KEE system. This environment gives user capability to carry out exploratory data analysis and pattern classification tasks. A good solution often consists of a sequence of steps with a set of methods used at each step. In KIPSE1, solution is represented in the form of a decision tree and each node of the solution tree represents a partial solution to the problem. Many methodologies are provided at each node to the user such that the user can interactively select the method and data sets to test and subsequently examine the results. Else, users are allowed to make decisions at various stages of problem solving to subdivide the problem into smaller subproblems such that a large problem can be handled and a better solution can be found.

I. INTRODUCTION

One of the major goals in computer-based problem solving is to provide computer users more flexibility and guidance in using the available computing resources. To facilitate user select proper methodologies and decide what computing tools to use in solving problems of a particular problem domain, specialized knowledge of expertise needs to be incorporated. A current research thrust in achieving this goal is to build a knowledge-based problem solving system where numeric processing is integrated with symbolic processing (4, 7, 10).

A problem solving environment (PSE), as broadly stated in (3), is an integrated multitasking system that supports and assists user in the solution of a given class of problems. Normally, within a more established discipline area or domain, collections of numerical programs in various forms and capabilities are available for use. These programs may be in the form of callable subroutines that can be accessed within an application program or through the use of specialized high level languages. There are currently many examples of PSE available in some problem areas. For instance, in the area of statistical computing, there exist packages such as SAS, BMDP, and others. In the arena of CAD/CAM where numeric computing routines for data display and geometric information manipulation are incorporated, there exist systems such as AutoCAD, GeoMod, just to name a few. Although these PSEs employ well-proven techniques of the problem area and may relieve the burden of code writing from the user, knowledge about the numeric algorithms and techniques used in these programs and facilities that allow a user to interactively define the process and examine the various results are not readily available to the user.

To augment PSE with capabilities to incorporate knowledge to manage symbolic manipulations, numerical routines, and problem solving strategies AI techniques are used.

AI techniques, specifically, methodologies from the expert systems technology enhance greatly the capability of PSE which include strategies for finding nondeterministic solutions, techniques for constraint propagation and search, and various knowledge representation schemes such as semantic nets and frames.

Many features are needed in a knowledge-based PSE. Basically, it should contain the following components: a friendly user interface, a bank of numeric programs, a kernel for system control with a variety of data structures for handling different data types, output display for both text and graphics, preprocessor and postprocessor modules for interfacing the various numeric programs, and a knowledge base. In Section II, a PSE architecture for data estimation and pattern classification is presented and its components explained. In Section III, the detailed implementation issues of our system KIPSE1 are discussed. Section IV discusses some features of the KIPSE1 system. Concluding remarks will be given in Section V.

II. A PSE ARCHITECTURE FOR DATA ESTIMATION AND PATTERN CLASSIFICATION

Problem solving process is a highly intelligent behavior. It is domain dependent and user involvement is extensive. For any new problems, the questions the user wants to answer are usually non-routine and many time very difficult (2). To model the general problem solving strategy and to build a system for all problem areas is almost impossible or at least very inefficient. Therefore, only PSE concentrated on a specific application domain is built in practice. As an environment the user involvement issue requires that it be convenience of use with effective user interfaces.

A PSE for data estimation and pattern classification tasks that are common in many areas of application has been proposed by the authors. A data estimation problem is given below to illustrate this. In data estimation, one is often asked to find a mathematical model to best explain the given data set. Many statistical packages, which may reside in many different computer systems, can be used for this purpose. For example, the multiple linear regression P1R routine in the BMDP statistical package and the regression REG routine in the SAS packages. After choosing the package, user still needs to have some statistical knowledge about each model, its assumptions, requirements, and so on, in order to pick the right model. User also needs to know the specific language for each package in order to use these routines. Many pages of numerical output are generated from these routines and a casual user usually needs expert assistance for a meaningful interpretation of the results. The process may repeat many times before an acceptable solution is determined. A system which integrates all these in one environment will greatly increase the productivity of the user in solving the problem. The PSE architecture for data estimation and pattern classification is shown in Figure 1.

The entire problem solving process can be divided in the following steps: problem formation, process generation, process setup, and postprocess. At the first stage, the problem formation includes problem parameter definition, data format conversion, and system initialization. In the process generation step, the strategy for solving the problem is formulated by the user based on the domain knowledge and previous results. Once a process is defined to perform certain task, the system will automatically set up the interface to the application program and initialize the computing processes. During the postprocess stage, useful information from the computing routines are extracted, converted into proper format and stored to the database for later retrieval. Output display routines are invoked to display the results both in text and graphical forms. A rule-based result interpreter is used to interpret the results. User examines the results and makes decision to either redesign the process or stop.

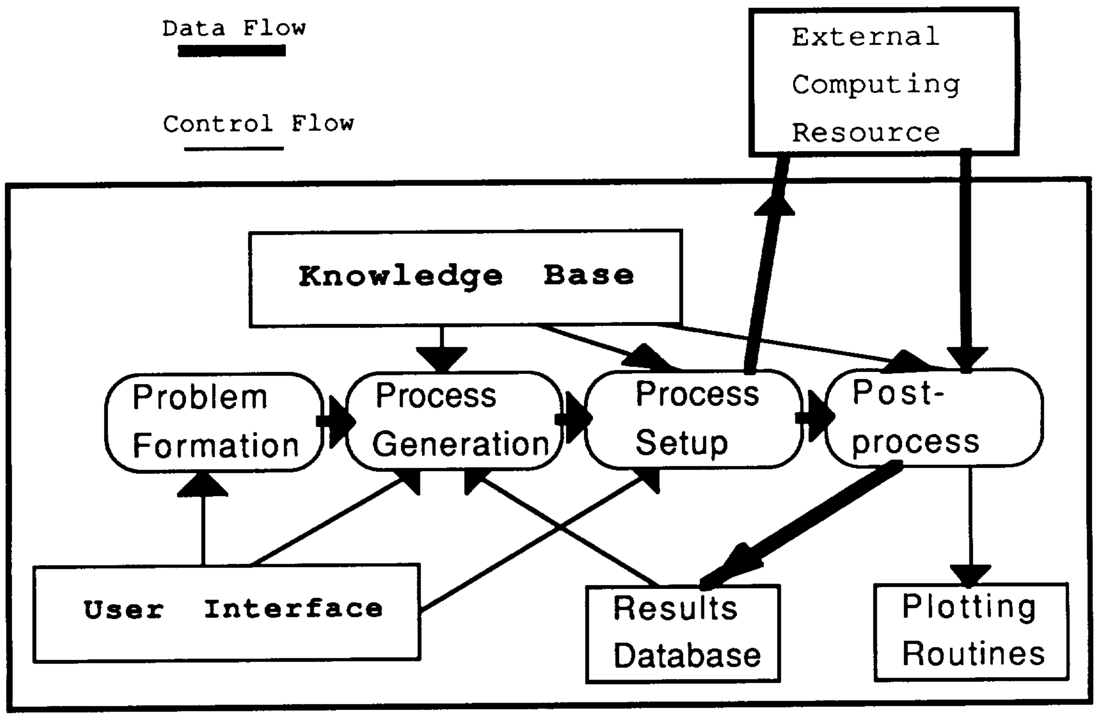


Figure 1. An Architecture for PSE for Data Estimation and Pattern Classification

III. SYSTEM IMPLEMENTATION

It has been determined from our past experiences in designing various expert systems that higher end expert system shells can provide the necessary tools for the implementation of the KIPSE1. The system requirements essentially consist of 1)powerful data structure to represent complex process, 2)flexible knowledge representational schemes to represent domain knowledge, 3)user interface with graphical capability, and 4)interface to other computer systems.

The KIPSE1 system is built on top of the Intellicorp's Knowledge Engineering Environment, the KEE system. Frame-based structure is used to represent the attributes of individual process. This frame-based structure is an object-oriented system where objects are structured as units with slots. Attributes are specified in slots. Slots may contain descriptive, behavioral, or procedural information, and relations are expressed using slot values (5). The KIPSE1 is organized as a hierarchical tree structure where each individual process is incorporated as a node within the structure. As an example, the structure of pattern recognition (PR) process is shown in Figure 2. At each node the control of the problem solving is coded in one of attached procedures associated with that node. Figure 3 shows an example of a procedural method written in Lisp. This procedure is the control procedure for the classifier design process.

The objective of pattern classification problem is to determine to which class a given data sample belongs. The process of designing a pattern recognition system can be divided as data gathering, normalization, data structure analysis, classifier design, and testing (error estimation) (11). The main purpose of the data structure analysis is to explore the data set structure in order to design the classifier in a later stage. The operations in this stage include feature extraction, clustering, statistical tests, and modeling. The classifier design is also called training or learning stage in which data samples are collected from various classes and the class boundaries are determined. There are several common classifier design methods used in this stage. They include linear classifier, quadratic classifier, piecewise linear classifier, and nonparametric classifier.

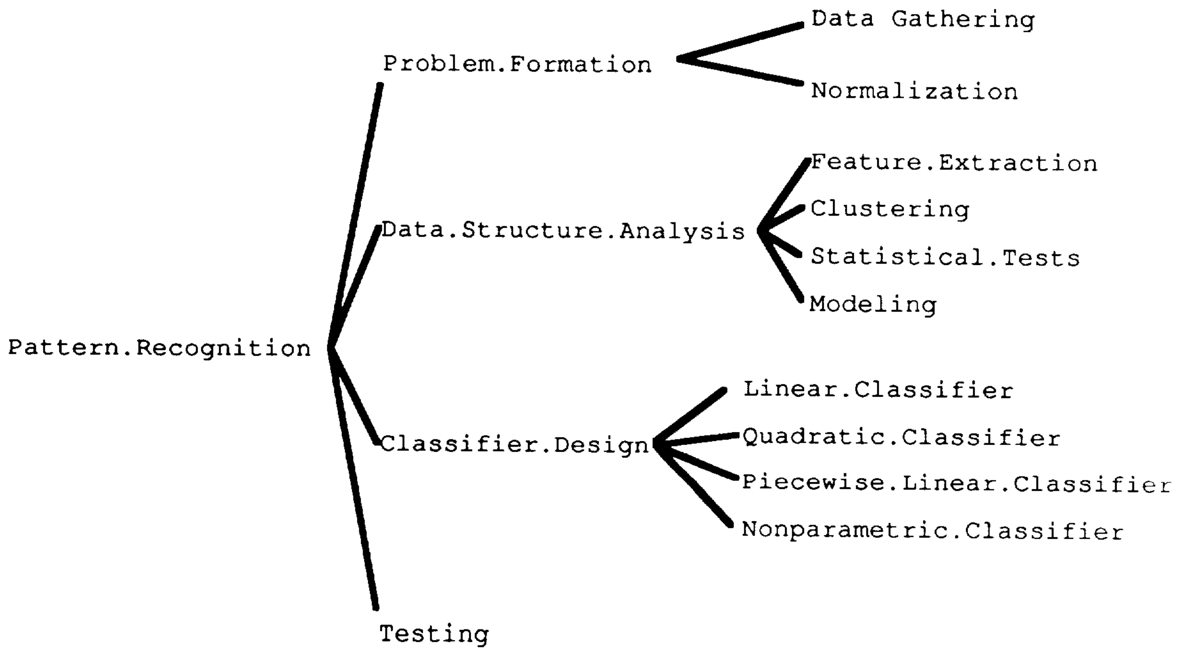


Figure 2. A Structure for the Pattern Recognition Process

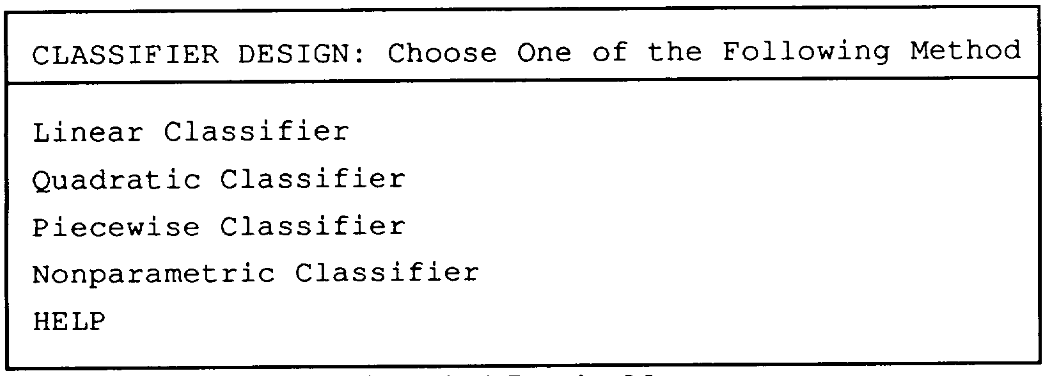
Our philosophy is to provide as many methods as possible to the user to try out at each node. Just as no general data structure is suitable for all problems, there is no general method applicable for all the pattern recognition problems. The best that one can do is to provide a good environment with all available methods so the user can pick up right method for the problem under consideration based on both the nature of that problem and the user's previous experience. A user might even try different methods, compare the results, and select the best one among the various trials. For those users who are not familiar with those methods provided in the system, heuristic rules are provided to help the user find appropriate one to use based on the general nature of the problem and usage requirements and limitations of each method.

The KIPSE1 system is menu-driven. Figure 4 shows the screen menu for the selection of classifier design methods as an example of a stage of running process. Four types of classifier methods, which include linear, quadratic, piecewise linear, and nonparametric classifier are provided to the user. In addition, expert knowledge is available in the HELP option.

Different types of knowledge are imbedded in the environment. In addition to the one of selecting a method as mentioned above there is specific knowledge for helping user to choose right parameters for each selected method, knowledge to help interpret the results, and knowledge for error handling. These knowledge types are coded explicitly in the rule form and is structured as shown in Figure 5.

```
(lambda (self node)
    (let  (process)
          (setf process (pop-up-cascading-menu-choose (make.cascading.menu
              :POP-UP '((" Linear Classifier"  'Linear.Classifier)
                         ("  Quadratic Classifier" 'Quadratic.Classifier)
                                 (" Piecewise Linear Classifier"  'Piecewise.Linear.Classifier)
                                 (" Nonparametric Classifier"  'Nonparametric.Classifier)
              :PARENT  *kee-root-window*
              :TITLE   "CLASSIFIER DESIGN:  Choose One of the Following Method")))
      ; Display the menu on the screen, ask user to choose method
      (cond  ((not (equal  process 'help)) (put.value  self  'process  process))
                      ; Save user selection in slot process
                      (t  (query '(THE PROCESS OF CLASSIFIER.DESIGN IS ?X)
                                                        'Classifier.Design.Help.rule.class)
                     ; Help user find right process using the knowledge provided by the expert
       ))
       (unitmsg (get.value  self  'process) 'control  node)
     )
  )
```

Figure 3. Control Procedure for Classifier Design Process



```
CLASSIFIER DESIGN: Choose One of the Following Method

Linear Classifier
Quadratic Classifier
Piecewise Classifier
Nonparametric Classifier
HELP
```

Figure 4. A Running Menu

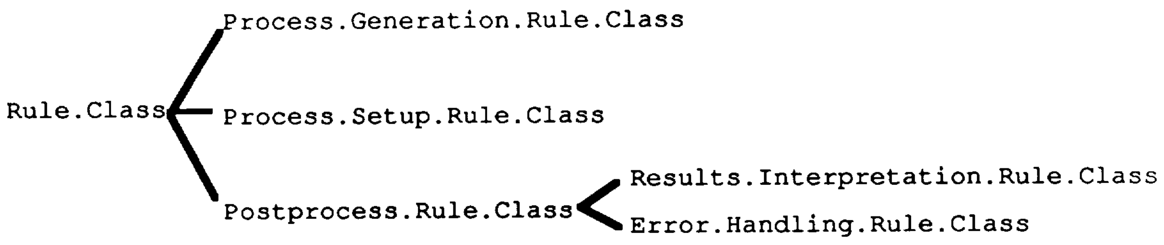

Rule.Class
Process.Generation.Rule.Class
Process.Setup.Rule.Class
Postprocess.Rule.Class
Results.Interpretation.Rule.Class
Error.Handling.Rule.Class

Figure 5. The Structure of the Rule System

Rules are used to represent knowledge. An example of a rule is given here. This is a result interpretation rule used in the clustering process. The purpose of the clustering process in pattern classification or recognition is to find a method for combining objects into groups or clusters such that objects in each cluster are similar. Similarity is defined by a measure of similarity provided by the user. If this measure is set improperly, say, it is too small, many meaningless small clusters will result. In this case the measure needs to be adjusted. From the domain expert it has been determined that the total number of clusters in most cases should be kept under four. A rule that convey this expert knowledge may be included as shown in Figure 6.

```
CLUSTERING.RESULTS.INTERPRETATION.RULE.001

(IF (THE RESULTS OF CLUSTERING IS ?X)
      (LISP (> ?X 4))
 THEN
 (THE PERFORMANCE.EVALUATION OF CLUSTERING IS NOT.ACCEPT)
```

Figure 6. An Example of Rule

IV. SYSTEM FEATURES

In this section, some of the basic features of the system are discussed. They are ease to integrate new software, availability of various methodologies, and user involvement in generating hierarchical decision tree.

An important feature of the system is its extensibility. General procedure is set up to integrate any software to the system. As explained earlier, a new software is defined as a new node in the system tree structure. The process of adding a new software is equivalent to inserting a new node to the proper position of the existing tree structure.

The information associated with each node is categorized into two classes. One is the global information used in interacting with the system. It includes software location, description, usage, results, performance evaluation, and the rule class that provides conditions for accessing the methods. Another category is the local information relating to actual computing process initialization. It includes preprocess, postprocess, result interpretation rule class, and error handling rule class. The general node structure is shown in Figure 7.

As an example, to integrate the method of multiple linear regression, say, the P1R routine of the BMDP package, which resides in a node called 'ucaicv' on the network, the following relevant information is stored in the pertinent node of P1R and shown in Figure 8.

It has long been noted in the field of pattern recognition that the key to problem solving does not rely only on the sophistication of the available algorithms but also lies heavily on the proper representation of the pattern structure of the data (6). Quite a few complex problems can be broken down into simpler problems. Each subproblem becomes easier to be solved. As stated in (1), hierarchical structure from a very important data representation and decision tree are naturally suited for problem solving with such a data representation.

In the KIPSE1 system both the problem and its solving process are also represented in the form of frame of structure. One of the important features of the KIPSE1 is that it provides a mechanism to allow user to break a problem node down into several subproblem nodes. These problem nodes can be organized hierarchically by setting up two special slots:

parent slot and children slot. Each problem node captures the partial decision process and the decision tree represents the problem solution.

Program Location
Program Description
Functional Description
Help Rule Class
Results
Performance Evaluation
Control Procedure
Preprocess Procedure
Process Setup Rule Class
Postprocess Procedure
Results Interpretation Rule Class
Error Handling Rule Class

Figure 7. A General Node Structure

UCAICV
A routine in BMDP statistical package
Multiple Linear Regression
P1R.GENERATION.RULE.CLASS
(results)
(performance evaluation)
P1R.CONTROL
P1R.PREPROCESS
P1R.SETUP.RULE.CLASS
P1R.POSTPROCESS
P1R.RESULTS.INTERPRETATION.RULE.CLASS
P1R.ERROR.HANDLING.RULE.CLASS

Figure 8. An Example of Integrating an External Program to the System

Several strategies are incorporated into the system. They include ad hoc strategies defined by the nature of the problem based on a priori experience, by visualization of input data set, by performance evaluation, and by exploration and trade-off study.

A decision tree solution to a data estimation problem is shown in Figure 9. The original set of 91 data points has been partitioned into four subproblem spaces of 22, 20, 24, and 25 data points. Shown along each node are the regression equation associated with the sample data set, the respective mean squared error, and the decision made to subdivide the problem space.

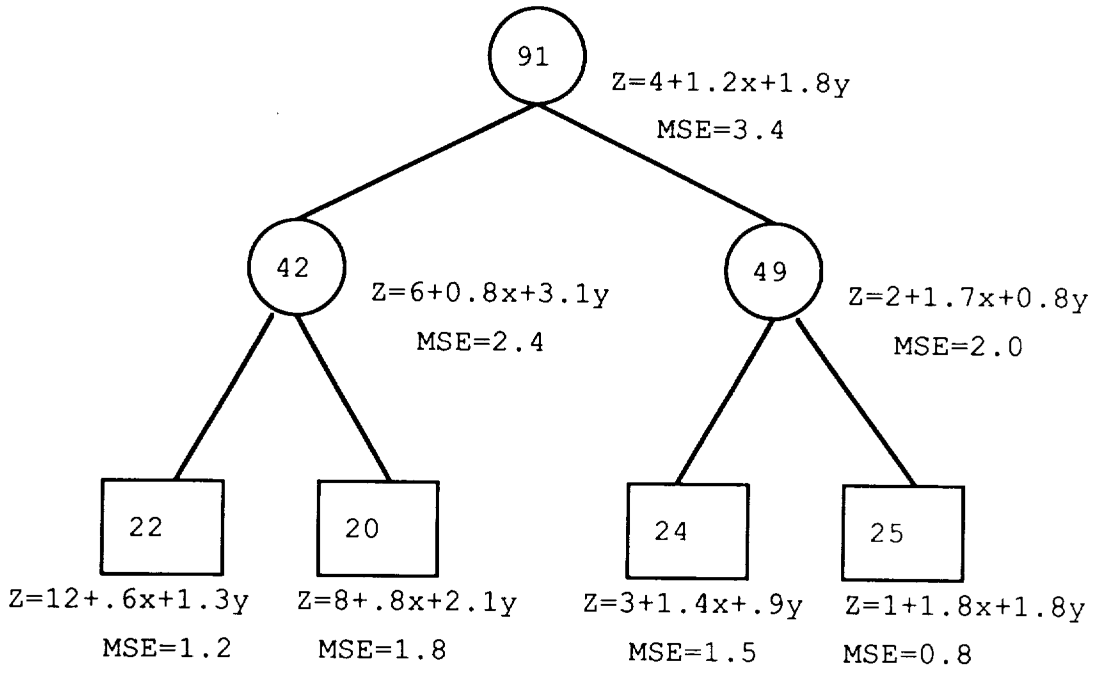


Figure 9. An Example of Solving a Data Estimation Problem Using Decision Tree

V. CONCLUDING REMARKS

A knowledge-based interactive problem solving environment for data estimation and pattern classification has been presented. The entire problem solving process is viewed as a multi-step decision making process. It is a learning process that at each step some kind of exploratory actions take place. The KIPSE1 provides this powerful problem solving capability. It is easy to use, easy to expand, and special knowledge is include to assist the user throughout the problem solving process. The system has been implemented on top of a powerful commercial expert system development shell. Its flexible knowledge representation and useful interface facilities have provided means to organize the numeric computing tools, to incorporate symbolic knowledge, and thus create a powerful problem solving environment.

VI. REFERECES

[1] Dattatreya, G. R. and Kanal, L. N., "Decision Tree in Pattern Recognition," *Progress in Pattern Recognition 2,* Kanal and Rosenfeld, eds, North-Holland, 1985, pp. 189-239.

[2] Feldman, S., "The How and Which of Problem Solving Environment," *Problem Solving Environment for Scientific Computing*, Ford and Chatelin, eds, Elsevier Science Publishing Co., 1987, pp. 23-32.

[3] Ford, B. and Iles, R. M. J., "The What and Why of Problem Solving Environments for Scientific Computing," *Problem Solving Environment for Scientific Computing*, Ford and Chatelin, Eds, Elsevier Science Publishing Co., 1987, pp.3-22.

[4] Gladd, N. T. and Krall, N. A., "Artificial Intelligence Methods for Facilitating Large-Scala Numerical Computations," *Coupling Symbolic and Numeric Computing in Knowledge-Based Systems*, Kowalik, J. S., editor, North-Hollad, 1987, pp.123-136.

[5] Intellicorp's KEE version 3.0 manuals, 1986.

[6] Kanal, L. N., "Interactive Pattern Analysis and Classification Systems: A Survey and Commentary," Proc. IEEE, vol.60, Oct. 1972, pp.1200-1215.

[7] Kitzmiller, C. T., Kowalik, J. S., "Coupling Symbolic and Numeric Computing in Knowledge-Based Systems," AI Magazine, Summer, 1987, pp.85-90.

[8] Lusk, E. L. and Overbeek, R. A.,"Automated Reasoning and Knowledge Based Design in the Scientific Programming Environment," *Problem Solving Environment for Scientific Computing*, Ford and Chatelin, eds, Elsevier Science Publishing Co., 1987, pp.83-98.

[9] Machura, M., "Issues in the Design of Problem Solving Environments," *Problem Solving Environment for Scientific Computing*, Ford and Chatelin, eds, Elsevier Science Publishing Co., 1987, pp.263-280.

[10] Tompkins, J. W., "Using an Object-Oriented Environment to Enable Expert System to Deal with Existing Programs, *Coupling Symbolic and Numeric Computing in Knowledge-Based Systems*, Kowalik, J. S., editor, North-Holland, 1987, pp.161-168.

[11] Young, T. Y. and Fu, K. S., eds., *Handbook of Pattern Recognition and Image Processing*, Academic Press, 1986.