**KnowledgeScape, an Object-oriented Real-time Adaptive Modeling and Optimization Expert Control System for the Process Industries**

**Contents**



## Introduction

KnowledgeScape is an object-oriented real-time expert control system for the process industries that has built in adaptive modeling and optimization capabilities. The primary use of KnowledgeScape is the on-line continuous monitoring of plant performance and the calculation of new process set points that maintain and optimize performance as feed and operating conditions vary.

Conceptually, KnowledgeScape is designed around the idea of intelligent software objects that represent real-world plants and processing equipment. These software objects can be connected together representing flow of material from one object to the next or they can be placed within each other representing the concept of groups of

equipment forming a circuit. These concepts are so powerful and unique that they have been patented (United States Patent Number 6,112,120)

One element of intelligence is the ability to predict one's future. This concept is embodied within KnowledgeScape via adaptive on-line neural networks. Reasoning about the past to determine process set points that will better drive the plant to more desired levels is accomplished by a real-time expert system that allows knowledge to be recorded as both crisp and fuzzy rules. Rule based control is centered around the concept of heuristics which are rules of thumb that have been developed over time and represent generalized experiences that can be relied upon in the future.

A subtle and infrequently discussed fact is that the rules themselves represent an inverse model of the process. This leads to the question of what other types of models can be created for the processes and what could be done with them if they are more robust that the heuristic rules. KnowledgeScape uses on-line neural network models to adaptively model the process, which adds to the unique features of KnowledgeScape. The predicted future values of important process variables can be used in the rule system and additionally by a genetic algorithm component of KnowledgeScape to ask the models what would be the best new process set points to achieve the desired control objectives.

This model based optimization ability adds true artificial intelligence to KnowledgeScape in that it creates the basis for learning about the process. By enlarge, rule based expert control are quite static in nature. Knowledge about the process is contained within the rules and represents the knowledge of those who wrote the rules. Models that continually adapt to accurately represent the process as feed conditions and equipment conditions changes add learning to KnowledgeScape.

Crisp and fuzzy rules, adaptive models and genetic optimization all require substantial computing resources, especially if they are on-line and are being used simultaneously. KnowledgeScape uses a distributed hardware architecture that allows computers to be added at will to the control system to meet the computational needs as a systems grows.

**Intelligent Software Objects and their use in KnowledgeScape**

Configuring KnowledgeScape is accomplished by building the flow sheet of the plant or process being controlled using KnowledgeScape's concept of intelligent software object. Figure 1. below is taken from the patent and describes the basic elements of an intelligent software object.

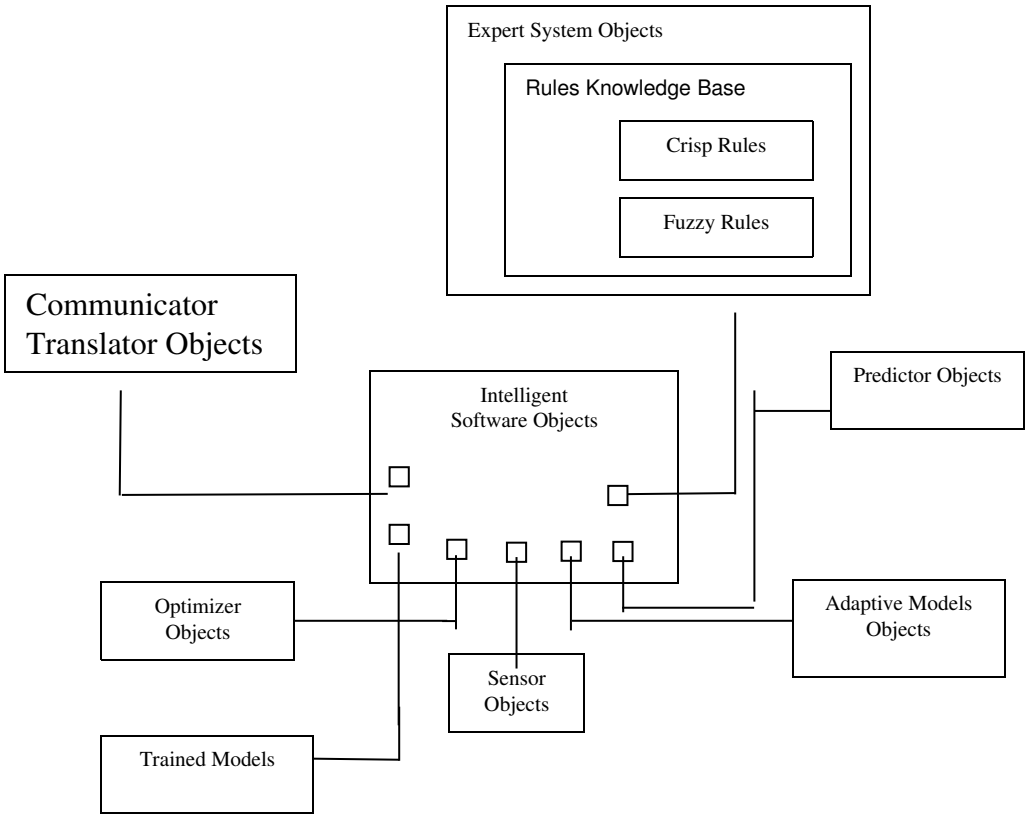



Figure 1. Diagram of an intelligent software object as defined in the KnowledgeScape patent.

Figure 2. shows a screen capture of the configuration window where a flow sheet is represented. Each box or node represents either a piece of processing equipment or, where a node surrounds other nodes it then represents a plant area or section.

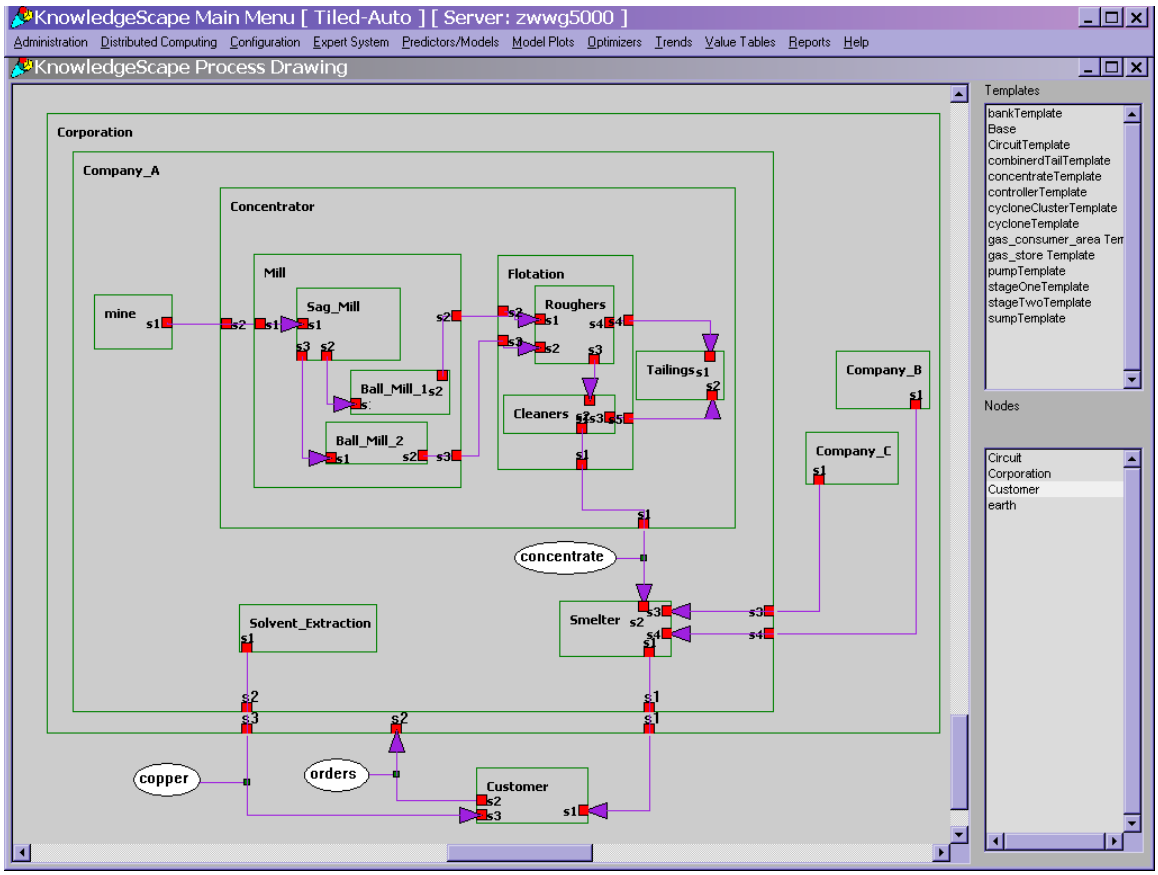



Figure 2. Configuration using intelligent software objects to represent a plant.

The nature of the intelligence associated with each node is shown by the node menu options in Figure 3. By clicking on a node and looking at the menu options for that node it is seen that the options include configuring 1) a predictor/model, 2) an optimizer, and 3) process rules.

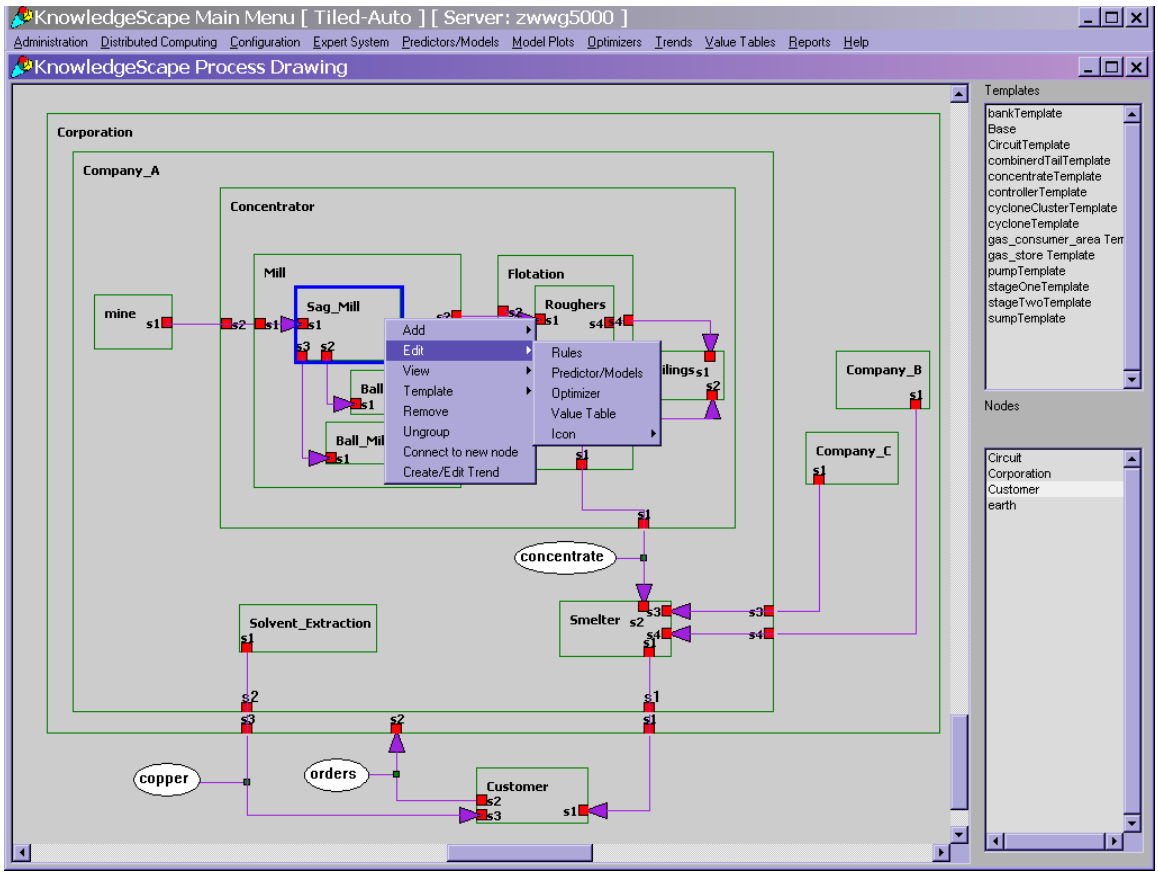



Figure 3. Each node, or intelligent software object has built in artificial intelligence capabilities.

## Artificial Intelligence and Process Control

Human intelligence is characterized by such things as 1) the ability to reason and draw conclusions given an incomplete and sometimes partially wrong sets of facts, 2) given a set of circumstances, or facts, predict future events or conditions, 3) based on experience gained in one environment or set of conditions, create generalizations that are true going from one situation to another and 4) given experience gained in one environment or from one set of conditions abstract generalizations applicable in broader terms. KnowledgeScape has built in artificaial intelligence tools that are integrated in a way so as to achieve performance that can be described as intelligent.

In one mineral processing application the adaptive neural network models were used which the genetic algorithm optimization functions to identify and run the plant at a combination of set points that were novel. The plant manager's statement that, "…it taught us something we didn't already know" really sums up the value of KnowledgeScape as a tool to improve and sustain plant performance.

The point is that the adaptive models had learned more about the cause and effect physical realities of the process than was understood by the operators or the engineers who had written the heuristic control rules contained within the expert system.

Interfacing KnowledgeScape to the Real World

KnowledgeScape is primarily used as a real-time control system, and thus, needs to be interfaced to other systems that actually monitor and control plants. It can be thought of as a supervisory control system that monitors processes through primary stabilizing control systems and then calculates new process set points and sends them to the stabilizing control systems form implementation.

There is no limitation on how many systems can be connected to KnowledgeScape to supply process information. Many systems have three or four of KnowledgeScape's data servers running piping data into and out of KnowledgeScape. Figure 4 shows the configuration screen used to configure one such data server. As the screen shot shows there are many built in drives to KnowledgeScape.

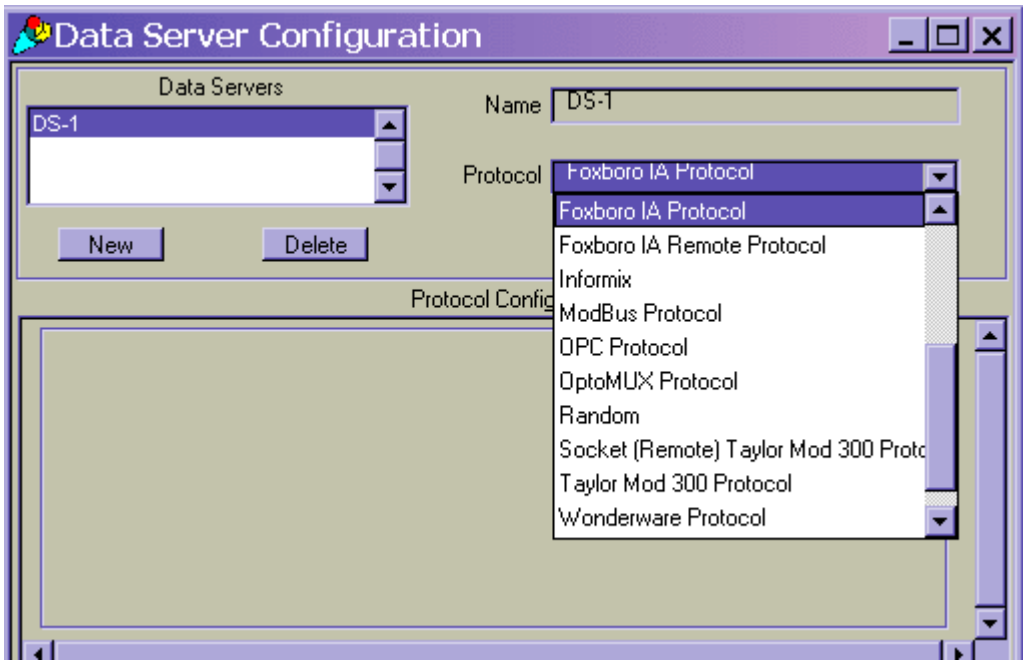



Figure 4. Data Server Configuration within KnowledgeScape

Real-Time Expert Control

The heart of KnowledgeScape is the real-time expert system that allows the designer to encapsulate knowledge about a process with both crisp and fuzzy rules. The ultimate objective being the monitoring of process conditions then based on the rules calculating new process set points. Other uses of the expert rule system might be the creation of smart alarm systems where alarms are monitored, characterized and then reported in logical ways that also might point operators to quick corrective action or provide advanced analysis of causes and effects of the alarms.

Rules in KnowledgeScape

KnowledgeScape is designed to aid the control system designer by compiling rules as they are entered and checking for correct syntax. Rules can also organized according to the flows and hierarchies represented in the process drawing. This schema results in a very logical and easily maintained system.

Basic creation and editing of rules is illustrated in the following screen shots. Because of the object-oriented basis of KnowledgeScape it is possible to interact with each element of KnowledgeScape by simply pointing at it and clicking or by directly going to the system function via the main menu bar. This concept is shown in figures 5 and 6.

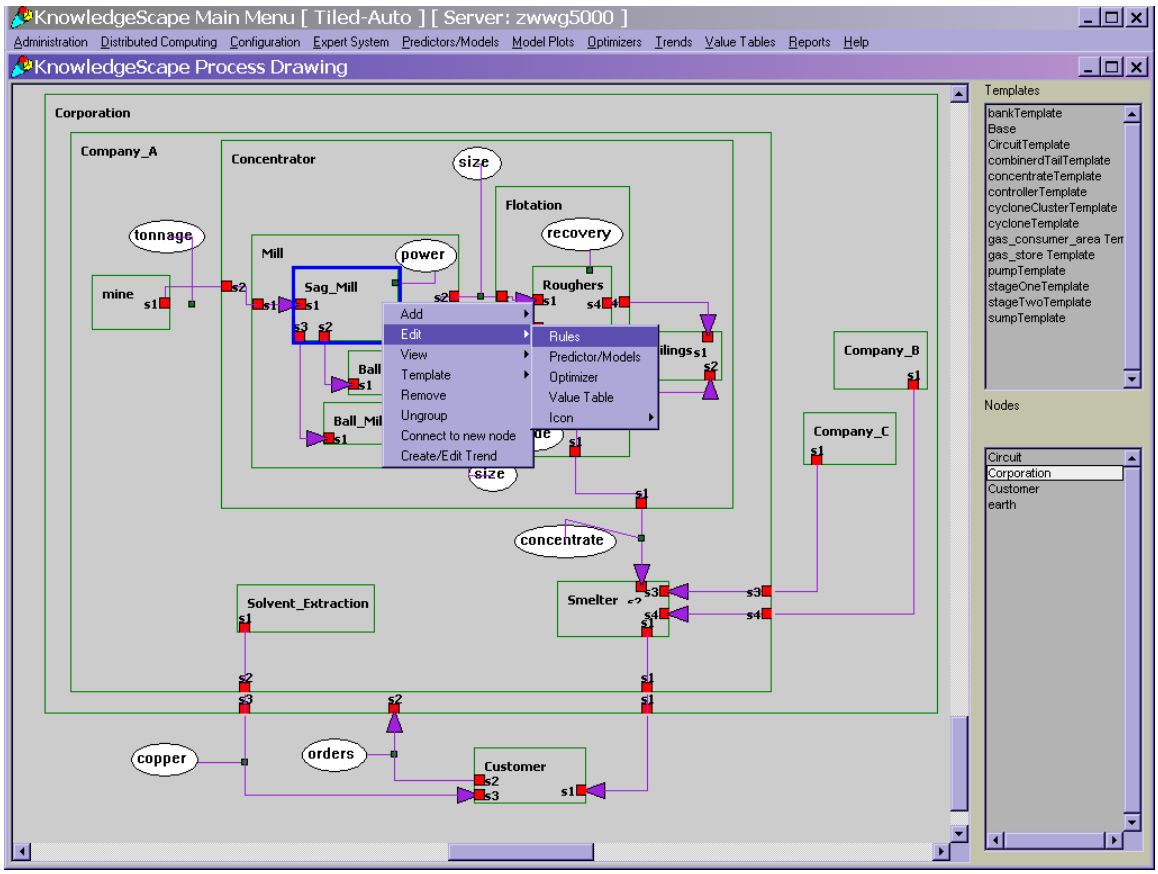


Figure 5. Pointing to an intelligent software object representing a Sag_Mill to create an associated process control rule.

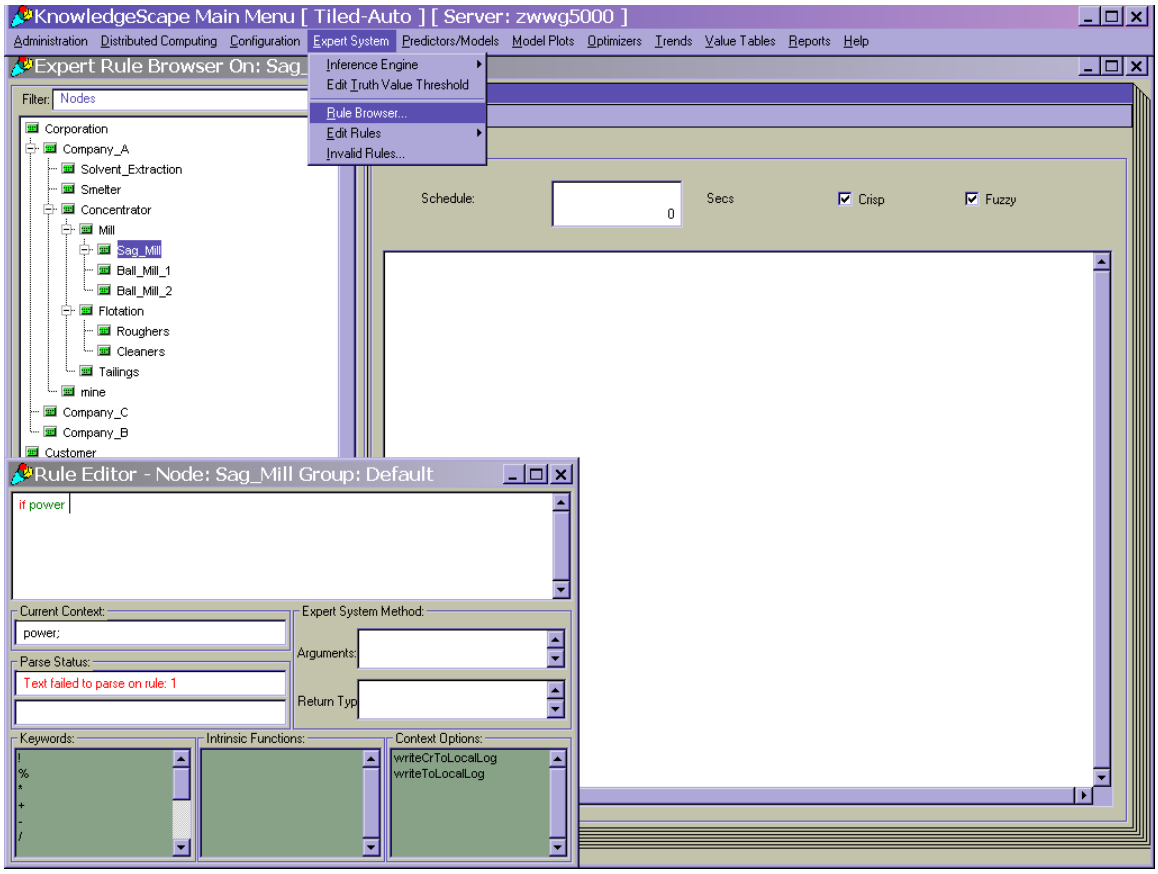



Figure 6. Opening the general rule editor directly from the main KnowledgeScape menu bar.

Specifically, after configuring the process drawing with intelligent software objects or nodes, the next step is to add expert system rules to accomplish the control objectives. The expert system of KnowledgeScape is a flexible environment for creating rules that control the nodes and attributes in the process drawing. It employs a powerful rule syntax within a structured execution environment.

Selecting the **Edit | Rules** menu option of the node's pop-up menu or selecting the Rule Browser from the main menu results in the expert system rule browser opening with the window title indicating the name of the node for which it was opened. The rule configuration user interface contains the node list and the rules window with tabs for each rule group and the rule schedule.

Rules are edited by double-clicking on the rule or by highlighting the rule and selecting *Edit Rule* from the right mouse button pop-up menu. Editing a rule will open the rule editor user interface with the selected rule as the text in the rule text editor widget.

Rules are added to the group by selecting *Add Rule* from the right-click menu. Adding a rule will open the rule editor user interface with the rule text editor widget blank. After accepting the rule, the current rule will be replaced or the new rule will be added to the rule group.

Figures 7 and 8 illustrate the concept of rule groups. Rule groups can be thought of as a bin containing expert system rules. You decide what bin or rule group contains which rules. In other words, rule groups provide an organizational framework that helps you simplify configuration of the expert system by building your control strategy in parts.

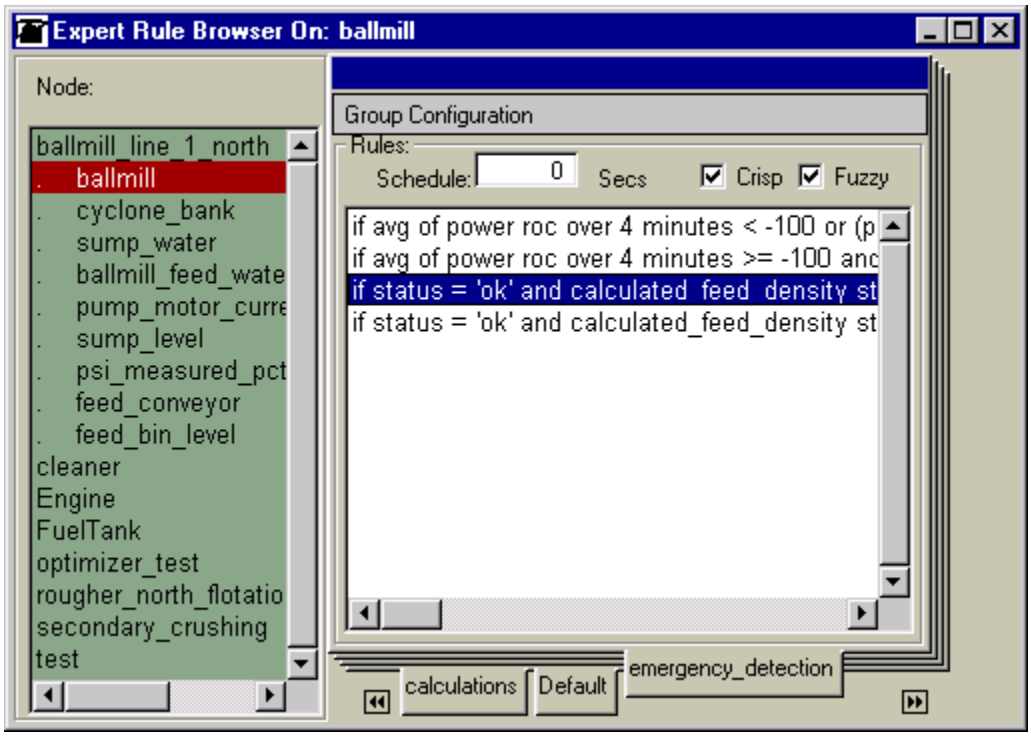



Figure 7. Rule Browser

For example, you can create a node that represents a power generation plant and define a rule group for high demand, low demand, and maintenance. You can then select the high demand rule group and configure rules that ensure the plant operates at peak efficiency during periods of high demand. Then, you can select the low demand rule group and configure other rules that ensure the plant conserves resources during periods of low demand. Later, when additional rules need to be created to ensure environmental discharge permit limits are not exceeded during periods of high demand, you can return to the high demand rule group and make the changes. Because the high demand rules are encapsulated in the high demand rule group, these changes can be made without

considering the details of the rules in the low demand or maintenance rule groups. This organizational concept is shown in Figure 8.


Figure 8. Rule Groups for a node or intelligent software object

Rule groups are used to logically organize the control application by allowing you to work at appropriate levels of detail so that complex process control issues requiring hundreds of rules can be broken into manageable smaller parts.

The actual node based rule browser, shown in Figure 9, contains a Schedule type-in box, Fuzzy check-box, Crisp check-box, and a rule list. The Schedule type-in box displays the frequency at which the expert system inference engine will execute the selected rule group. The value in the type-in box is an integer number of seconds. The Fuzzy check-box controls whether rules that employ fuzzy logic are displayed in the rule list. The Crisp check-box controls whether rules that do not employ fuzzy logic (rules that employ only crisp logic) are displayed in the rule list. The rule list display shows all rules in the selected rule group when the Fuzzy and Crisp check-boxes are selected.

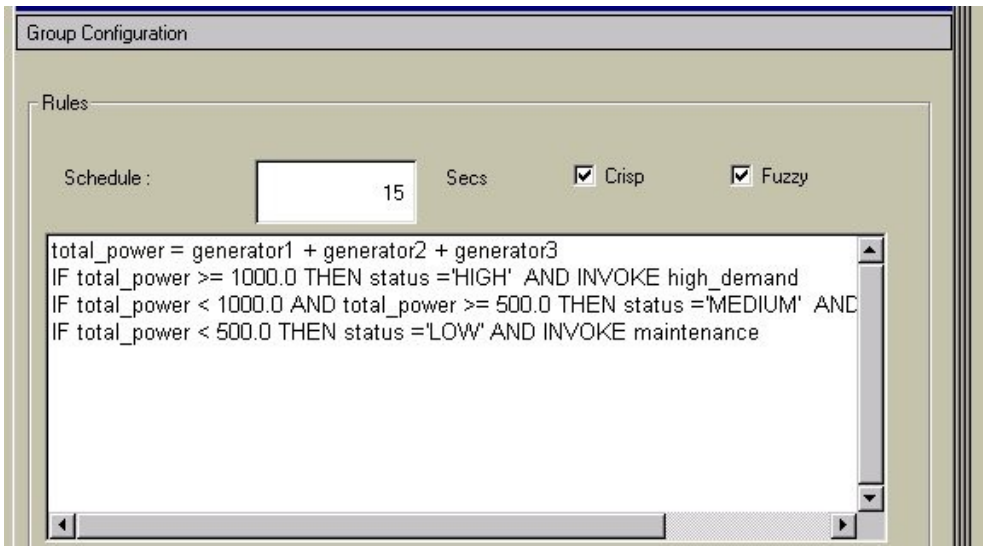



Figure 9. Rule Group Browser

The Rule Editor, shown in Figure 10, is used to configure individual rules and optimizer functions. The Rule Editor user interface window contains several data fields and widgets. These are:

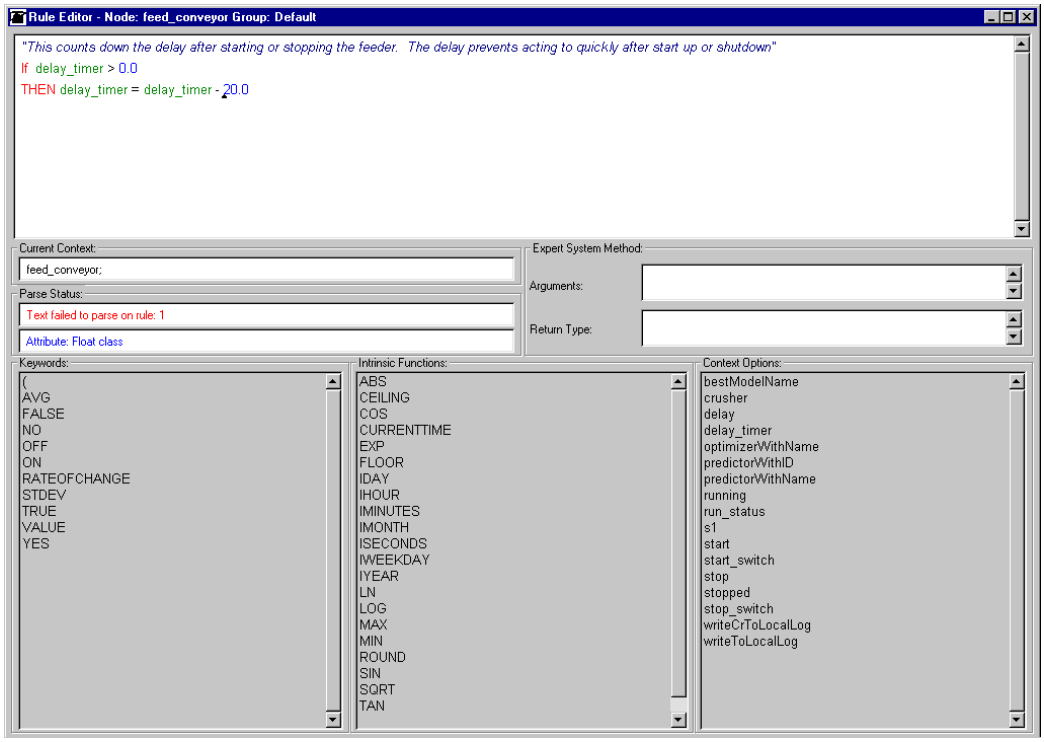



Figure 10. Context Sensitive Rule Editor

The text field at the top of the window provides text-editing capabilities to configure expert system rules and optimizer functions. The user clicks in the text field to position the cursor appropriately and then uses the keyboard or selects entries in the *Keywords*, *Intrinsic Functions*, and *Context Options* lists to configure valid rule and function syntax. As text is added to the rule text editor, KnowledgeScape parses the text and updates entries in the Keywords, Intrinsic Functions, and Context Options lists. When finished, the user accesses the right mouse button pop-up menu options to either accept the rule or function (Accept Rule), accept the rule and close the grammar assistant (Accept and Close), or (Accept as New). The right mouse menu also allows copy, cut, paste, find, replace, and undo for editing the text of the rule.

The Current Context field displays the name of the KnowledgeScape object that provides the context for the rule. As the rule is created the current context will indicate the context of the last typed node or attribute. When configuring functions for an optimizer the current context field will initially display the name of the optimizer. The current context will update as the rule or function is edited. If the text *"tank level"* is entered into the rule text editor widget, the *level* attribute of the *tank* node will become the current context and all operations in the selection lists will be based on that context.

Two indicator fields display the parse status. The upper indicator displays whether the text contained in the rule text editor was successfully parsed. Parsing occurs interactively as each character is typed, and if the current text is invalid or incomplete the upper

indicator will display the statement number that needs to be modified to make the rule valid. The lower indicator displays hints and information about the expected argument types for assignments and conditions, and expected input types to complete the rule or function being edited.

The Expert System Method Arguments field indicates the expected argument types for references to intrinsic and context option functions that accept arguments. For example, if *"COS("* is entered into the rule text editor widget, this field will display *"Number"* indicating that one argument is expected that is a number.

The Expert System Method Return Type field indicates the expected return type for references to intrinsic and context option functions. For example, if *"CEILING( 3.4 )"* is entered into the rule text editor widget, this field will display *"Number"* indicating that the intrinsic function returns a number.

The Keywords list displays all keywords that are valid to complete the rule or function. As text is entered into the rule text editor widget, the keywords list is updated to display only valid entries. If the character *"s"* is typed, only the selections that begin with *"s"* are displayed. Double-clicking on an entry in the keywords list causes the selected entry to be added to the rule text editor.

The Intrinsic Functions list displays all valid intrinsic functions available to complete the rule or function. As text is entered into the rule text editor widget, the intrinsic functions list is updated to display only valid entries. If the character *"c"* is typed, only options that begin with *"c"* are displayed. Double-clicking on an entry in the intrinsic functions list causes the selected entry to be added to the rule text editor.

*The Context* Options list displays valid options available to the current context. Entries in this list include the names of attributes, connections, sub-nodes, functions, etc., within the current context. As text is entered into the rule text editor widget, the context options list is updated to display only valid entries. If the character *"c"* is typed, only options that begin with *"c"* are displayed. Double-clicking on an entry in the context options list causes the selected entry to be added to the rule text editor.

The hierarchy of nodes in the process drawing determines the scope of the node in relation to other nodes. The node's scope determines which nodes and attributes can be accessed by the node's expert system rules.
The scope of a node includes every sub-node, any sub-node of its sub-nodes, and nodes which it is connected to. For example, consider the car, fuel tank, and fuel pump system illustrated in Figure 11.

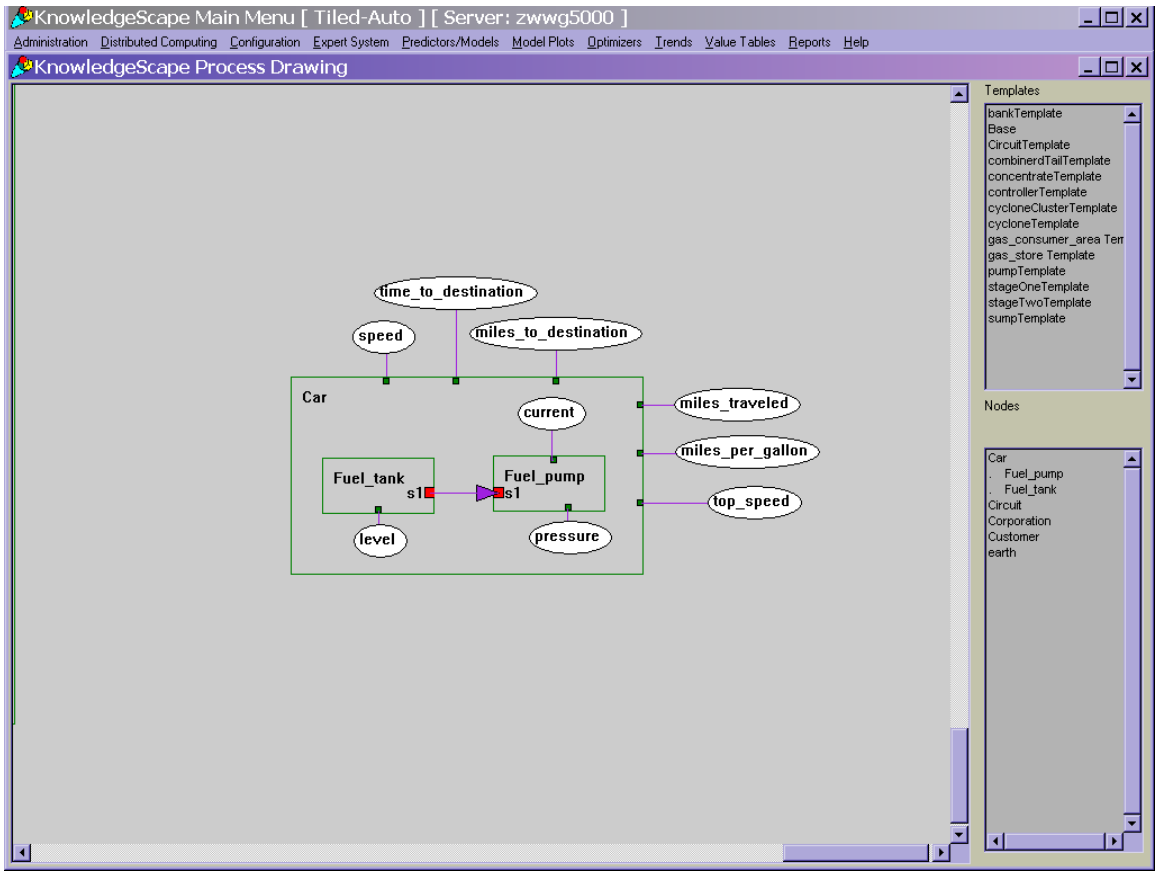



Figure 11. Simple representation of a car in KnowledgeScape

The scope of the car node allows it to "see" or access the fuel tank and its level, and the fuel pump and its pressure and current. Therefore, you can configure rules in the car node that refer to the fuel tank and fuel pump. However, the car node is not in the scope of the fuel tank node. The fuel pump is on the same hierarchy level with the fuel tank and because it is connected to the fuel tank, the fuel pump node is in the scope of the fuel tank node. The fuel pump has only the fuel tank in its scope.

The node hierarchy and its scope determine the rule syntax required to access the attributes of nodes in expert system rules. To access attributes of the selected node, you simply use the attribute's name in the rule syntax. For example, if the car node in the previous figure has attributes named, "time_to_destination," "speed," "miles_to_destination," "miles_traveled," "miles_per_gallon" and "top_speed," you would refer to those attribute directly in the rule syntax.

Examples that illustrate rules that can be configured in the car node are shown in Figure 12.

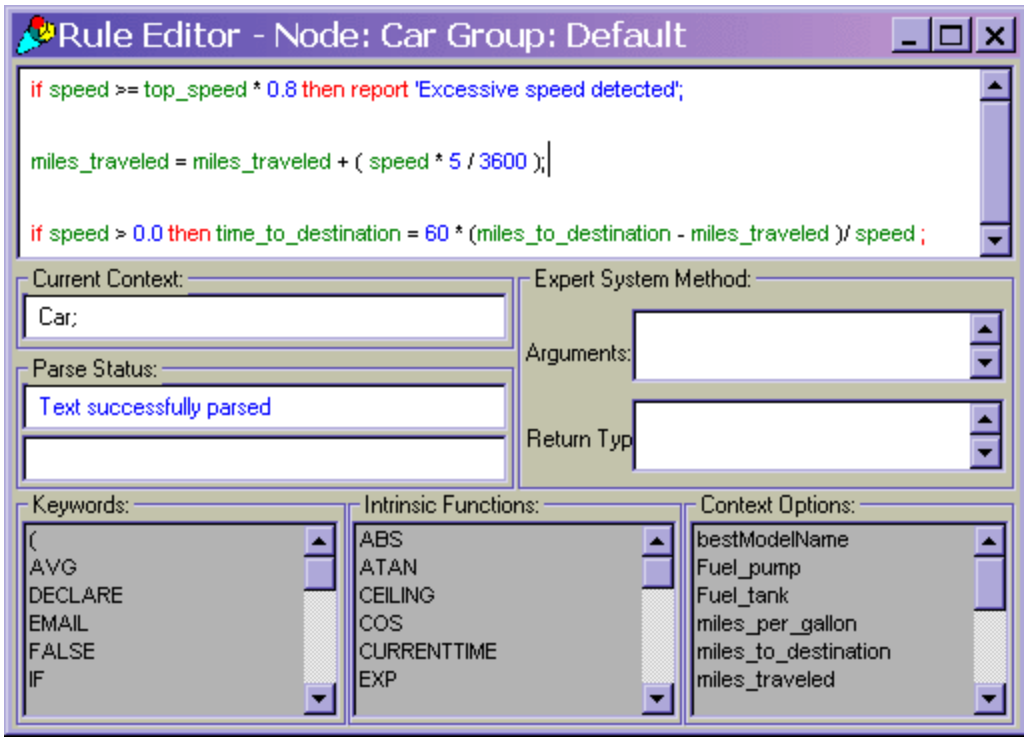



Figure 12. Rule Browser on the Node Car.

To access attributes of sub-nodes of the selected node, you must include the name of the node before the name of the attribute in the rule syntax. For example, consider the following rules that access the fuel tank level, fuel pump pressure, and fuel pump current:

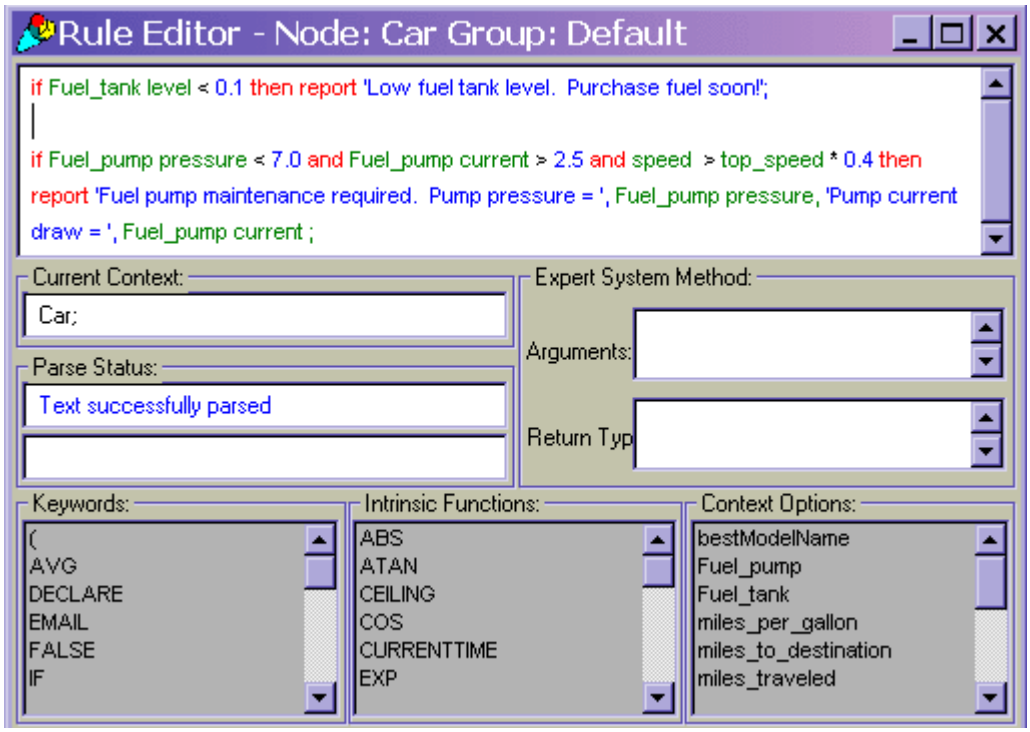



Figure 13.

*IF fuel_tank level > 0.0 THEN miles_per_gallon = miles_traveled / ( fuel_tank level * 13.5 )*

To access the attributes of sub-nodes at levels lower in the hierarchy, you must specify the name of each node in the hierarchy, starting at the first sub-node down to the target sub-node followed by the attribute name. The form of the rule syntax is:

*subnode_1_name … subnode_n_name subnode_n_attribute_name*

The three basic KnowledgeScape fuzzy expert system rule constructs are indicated below:

1. <Action>,
2. IF <Condition> THEN <Action>, or
3. IF <Condition> THEN <Action> ELSE <Action>

Any action that can be configured in an IF-THEN or IF-THEN-ELSE rule can be configured without any conditions. This is analogous to configuring a rule where the condition is always true. Some of the most common types of actions are:

Crisp Equation Actions – actions that calculate values and store the resulting value in an attribute of a node are equations. Equations typically contain an equal sign (=). For example, the following rule has the form of an equation action:

*miles_traveled = miles_traveled + ( speed * 5 / 3600 )*

Fuzzy Equation Actions – actions that calculate values and store the resulting value in an attribute of a node using fuzzy logic mathematics. Fuzzy logic equations contain the IS keyword. For example, the following rule has the form of an equation action:

*speed IS fast*

Report Actions – actions that generate information messages that are reported to information logs. For example, the following rule is a report action:

*REPORT 'Fuel pump maintenance required. Pump pressure = ', fuel_pump pressure, ' Pump current draw = ', fuel_pump current*

Invoke Actions – actions that cause other rule groups to execute. For example, the following rule is an invoke action:

*INVOKE fuel_pump high_pressure_strategy*

Send Actions – actions that send data to an external interface to be communicated to an external system. For example, the following rule is a send action:

*SEND fuel_pump current*

Update Actions – actions that cause an internal program to run, such as one that recalculates histogram values:

*UPDATE histogram_level*

Conditions configured in an IF-THEN or IF-THEN-ELSE rule generate a truth value. The truth value for crisp conditions will either be true or false. The truth value for fuzzy conditions is a value between 0.0 and 1.0.

The two types of conditions are:

Crisp Conditions – conditions, reduced to their most simple form, contain two values separated by a combination of mathematical operators. Valid operators include:

Equal (=) – test if values are equal.

Not Equal (~=) or (<>) – test if two values are not equal.

Greater Than (>) – test if one value is greater than another value.

Greater Than Or Equal (>=) – test if one value is greater than or equal to another value.

Less Than (<) – test if one value is less than another value.

Less Than Or Equal (<=) – test if one value is less than equal to another value.

Statements that refer to values can be inserted for values on either side of the operator. For example, the following demonstrates the syntax of a valid crisp condition:

*miles_traveled >= miles_to_destination*

Fuzzy Conditions – fuzzy logic conditions contain a node attribute name, the fuzzy IS operator, and a fuzzy membership function name. For example, the following condition demonstrates the syntax of a valid fuzzy condition:

*fuel_pump current IS high*

Fuzzy rules always are put at the top of the rules list in the expert system Rule Browser. They have no need for a particular order because they are evaluated simultaneously and then their combined result is calculated. Crisp rules do rely on the order they are inserted into the expert rule browser because they are fired in the order they are listed.

Once rules have been configured in rule groups of the nodes in your process drawing, they need to be scheduled for execution. Often it will be most convenient to create an "administrative" group of rules that is scheduled to run at a regular frequency whose rules invoke the other rule groups for the appropriate action.

Neural Networks in KnowledgeScape

Neural networks are a generic modeling concept based loosely upon a parallel computing model of the human brain. Research has shown them to be capable of modeling any non-linear relationship as long as there is cause and effect information in the modeled data. Many of the screens used in creating neural networks in KnowledgeScape are shown in Figures 14 through 15.

.

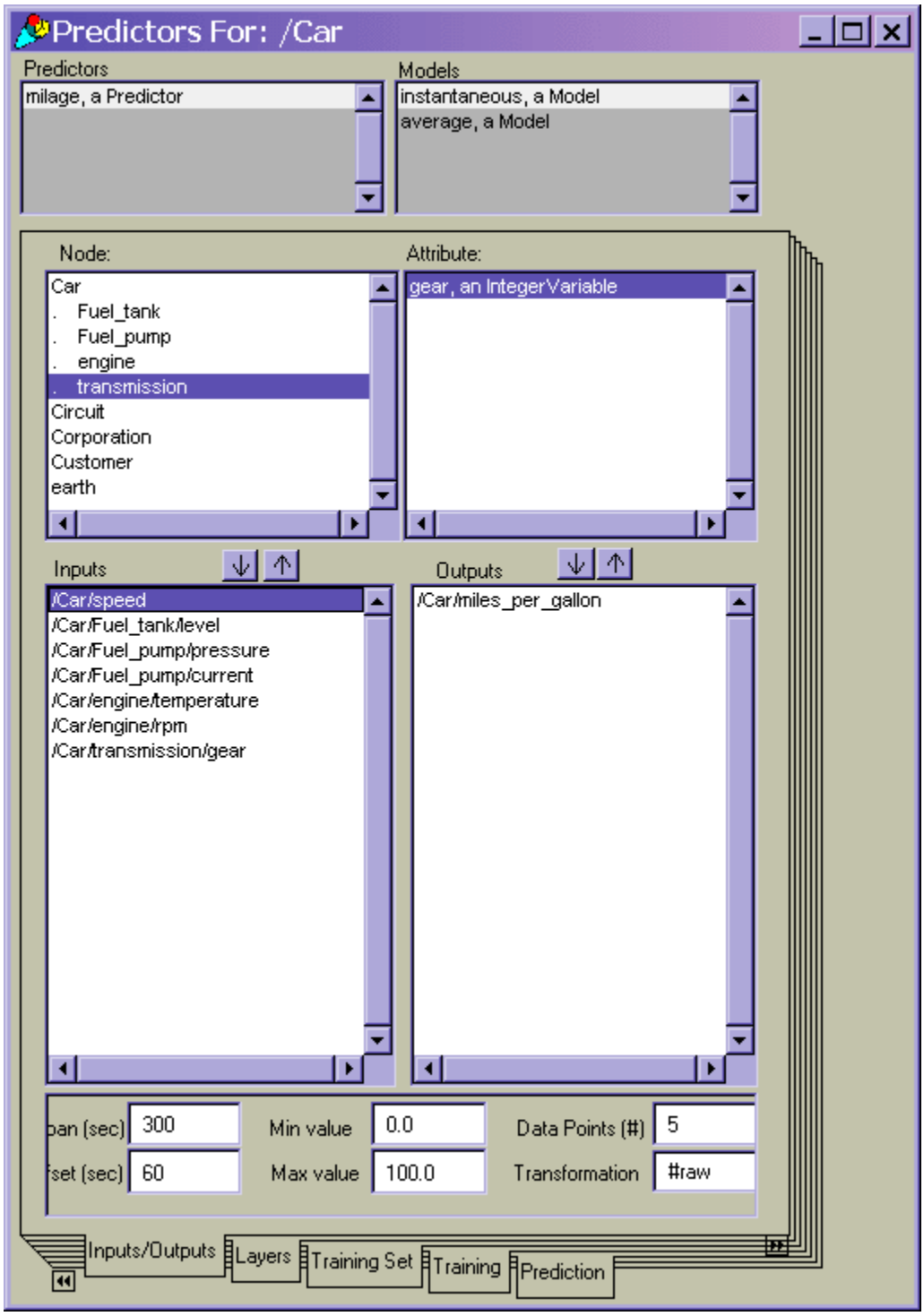



Figure 14. Predictors and Neural Network models in KnowledgeScape

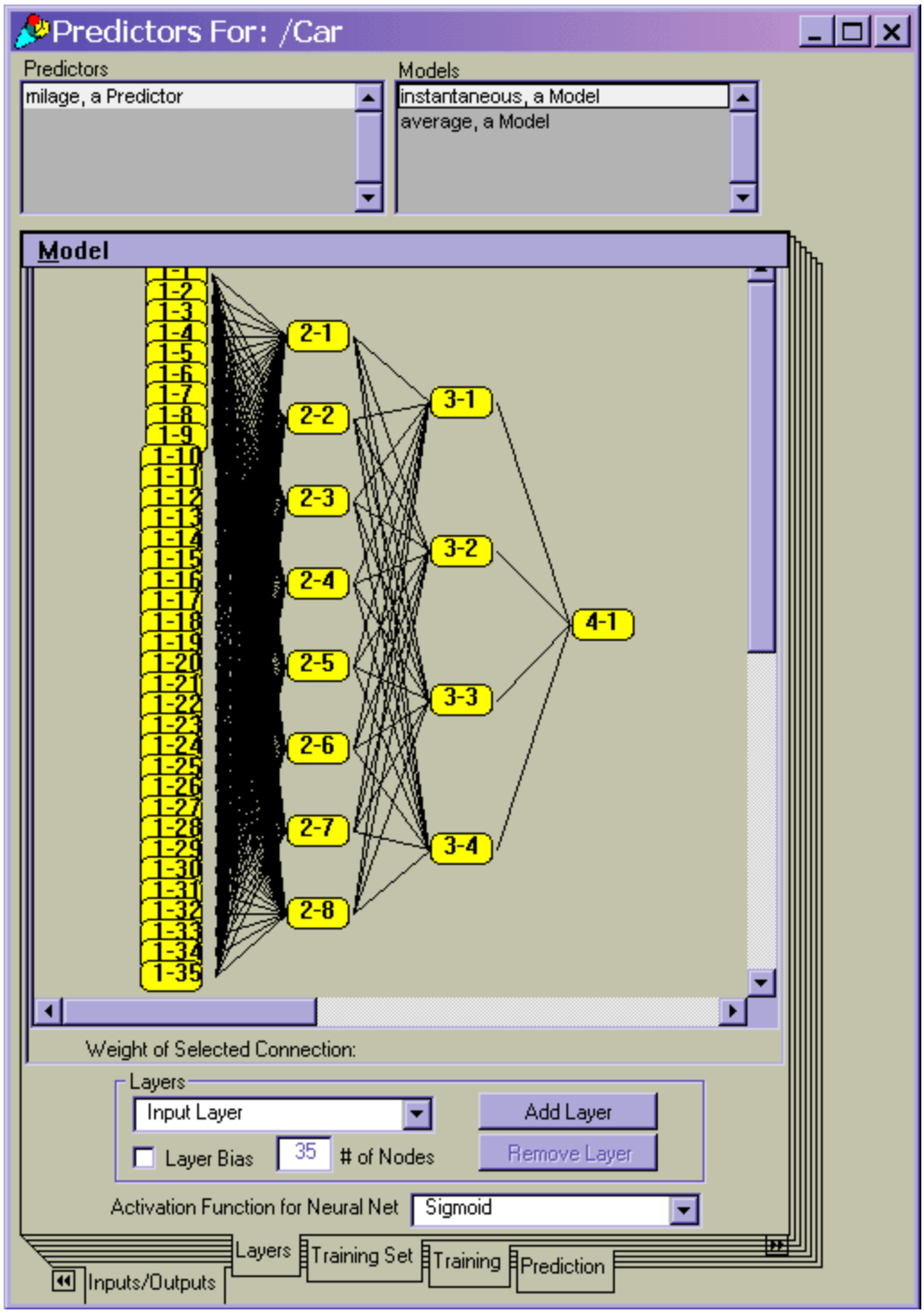



Figure 15. Topology of a Neural Network

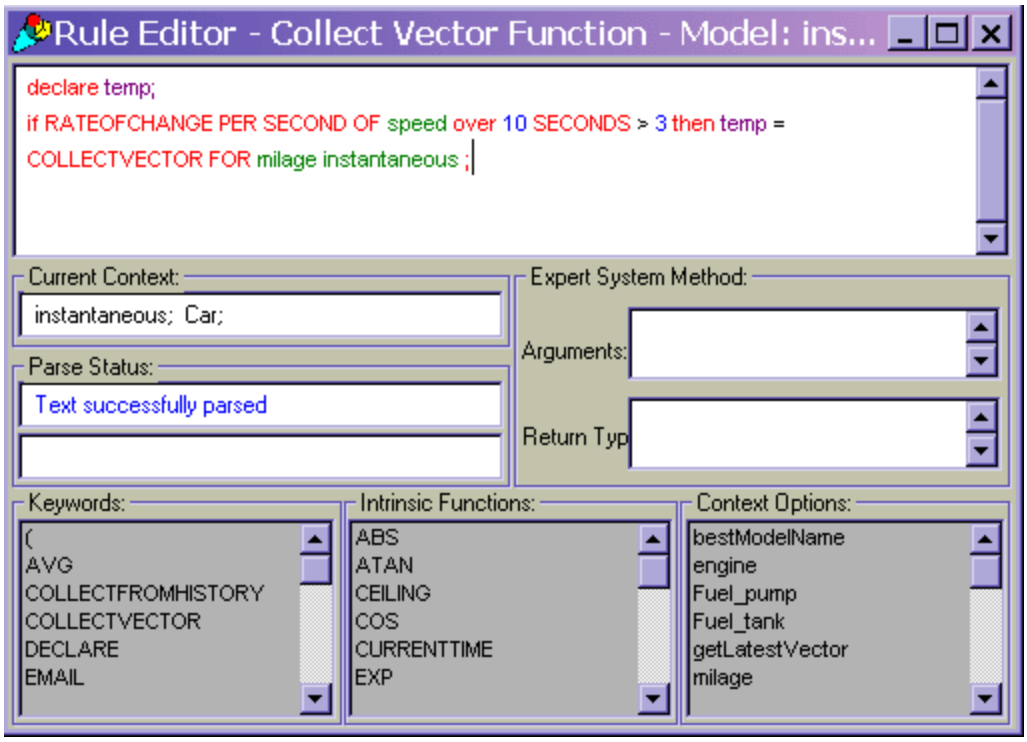



Figure 5. Configuration Settings for a Neural Network

In KnowledgeScape we have created the concept of competitive models that vie to be deemed accurate enough to be selected by the optimization component to be used to predict the process and be used in optimization. This is accomplished by what we call Predictors and their models. A Predictor with its associated models is a collection of neural network models who have been configured to all predict the exact same process variables. Each of the individual models however has a unique design. This concept acknowledges the reality that it is difficult to create one super model that is accurate under all processing conditions.

Each of these models can be set up to predict continuously and then the Predictor automatically monitors their predictions and what actually occurred in the process to determine which model is actually predicting the best

Figures x through y below show KnowledgeScape's predictors and neural network models in action. The first plot illustrates the adaptive nature of these models after a critical process sensor was recalibrated.

Figure x. Adaptive Model Predictions in Real-time

Figure y.

## Genetic Algorithms in KnowledgeScape

With adaptive models that aptly demonstrate their ability to accurately predict future process states given current conditions what additional control benefits can be achieved by using them in constrained optimization? KnowledgeScape uses genetic algorithms to interrogate the trained models as an alternate technology to determine new process set points.

Genetic algorithms "are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. The central theme of research on genetic algorithms has been robustness, the balance between efficiency and efficacy necessary for survival in many different environments. Genetic algorithms are theoretically and empirically proven to provide robust search in complex spaces." (Goldberg, 1989).

KnowledgeScape's use of genetic algorithms is illustrated in the following figures.

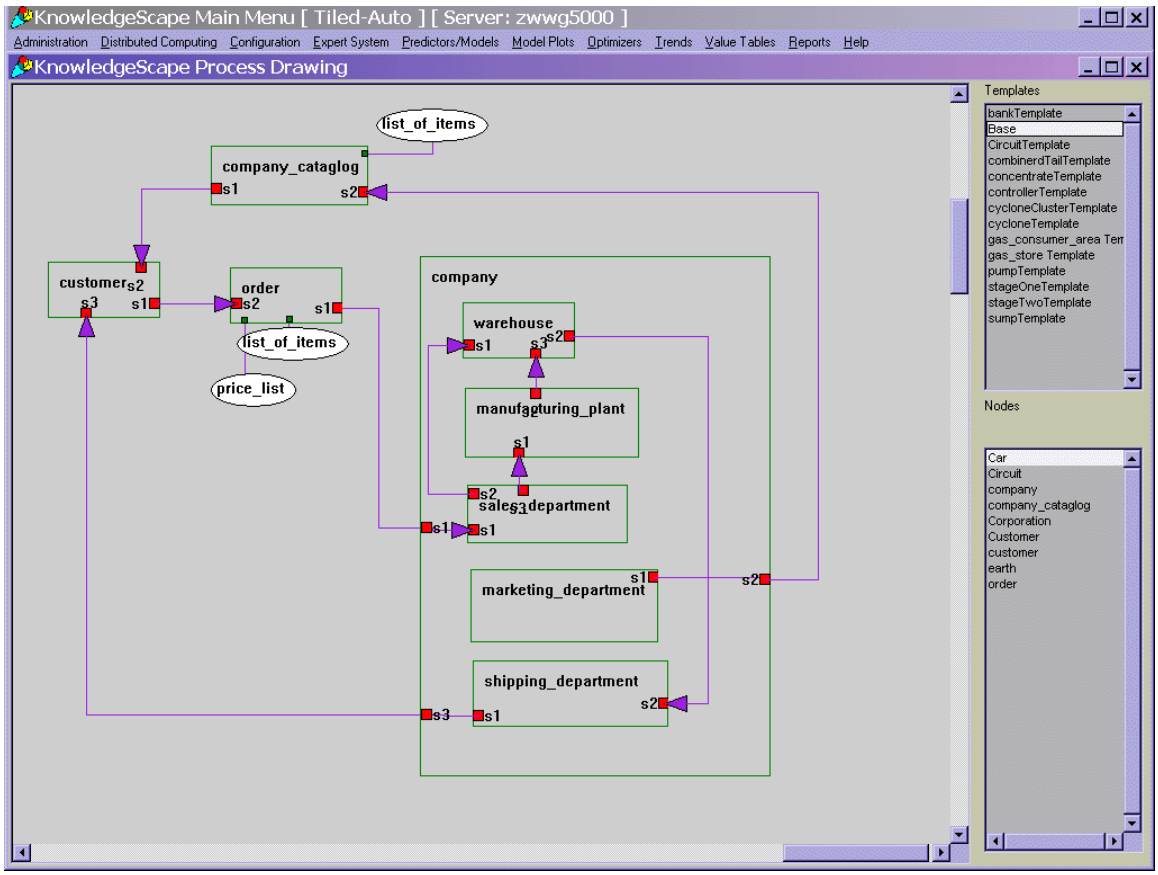
Figure yyy. Basic Configuration Of An Optimizer in KnowledgeScape.

Figure xxx. Creation Of The Fitness Function That is Either Minimized or Maximized By the Genetic Algorithm

## Documentation of Performance – Reports and Email

KnowledgeScape is perhaps the first distributed computer system that has integrated the powerful capabilities of real time expert control, on-line adaptive and competitive neural network models, genetic algorithm optimization into one software system designed to monitor and control any industrial process. Careful reflection on the descriptions of functions of KnowledgeScape suggests that its design can be used in a much broader sense. Specifically, KnowledgeScape can be used to embed intelligence in any process. Process in this sense means any collection of actions that make up a system in the broadest of terms. An order entry process has customers, inventory, ship dates, price lists etc.

KnowledgeScape can be easily used to diagram the process, specify each element of the process and then control the process as attributes of the elements change as the process is used.




Documenting performance is key to the success of any control project. KnowledgeScape has extensive reporting, trending, emailing and statistical analysis capabilities built in. Figures x – y show screen of many of these features.

Histograms are a very powerful way to keep track of performance. It is possible to write rules that govern when data is actually collected and put into a histogram. In this way it is possible to easily track performance as a function of what control strategy is being run or on specific operating conditions.

Figure x. Histogram

Trends of process data, set points, model predictions and optimized set points are commonly used to monitor the process as well as the performance of the control strategy(s).

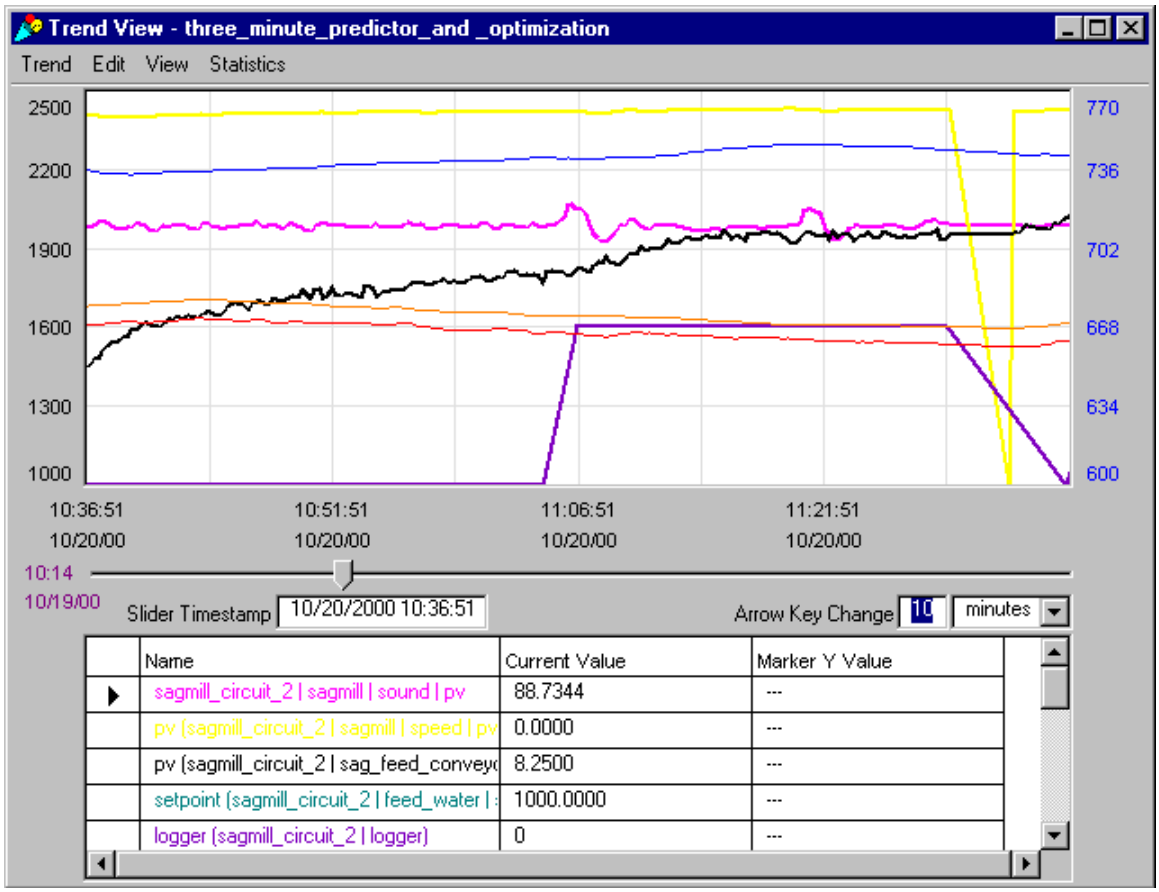



Figure x. Trend

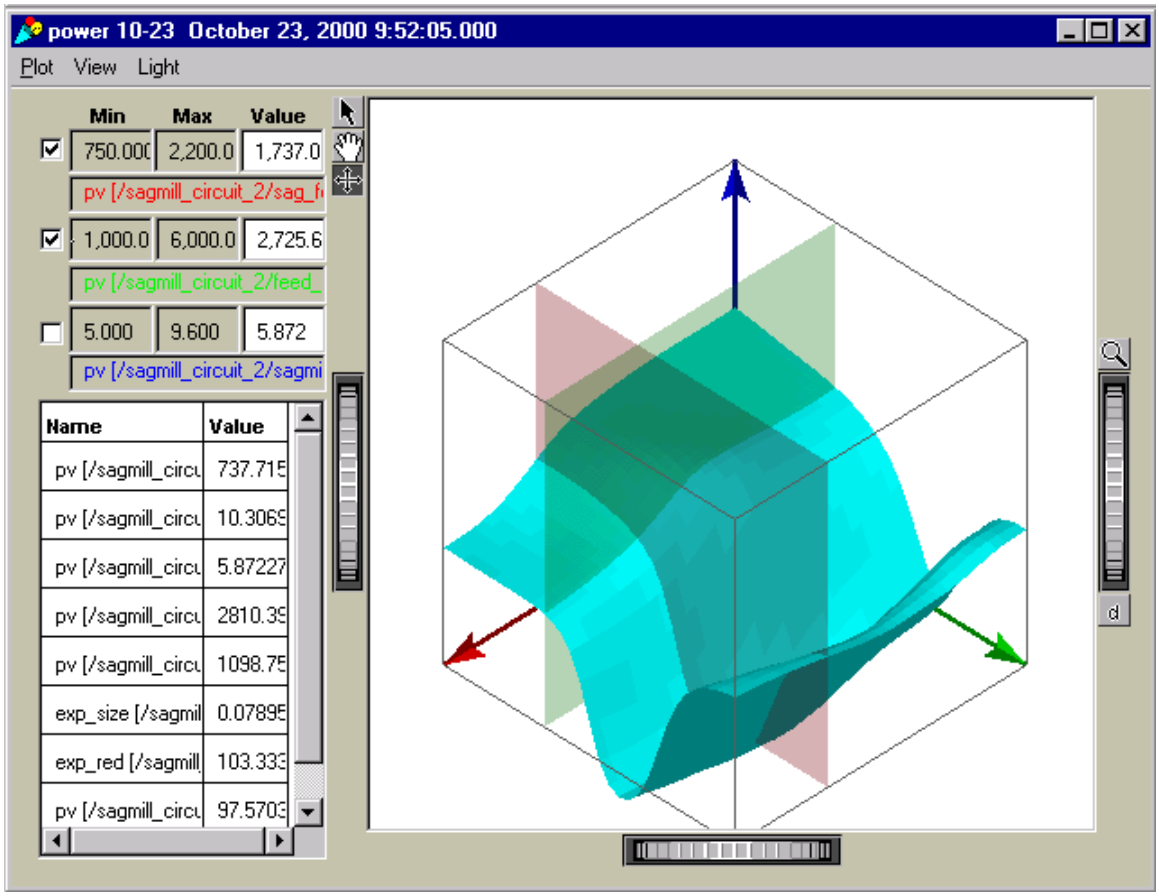
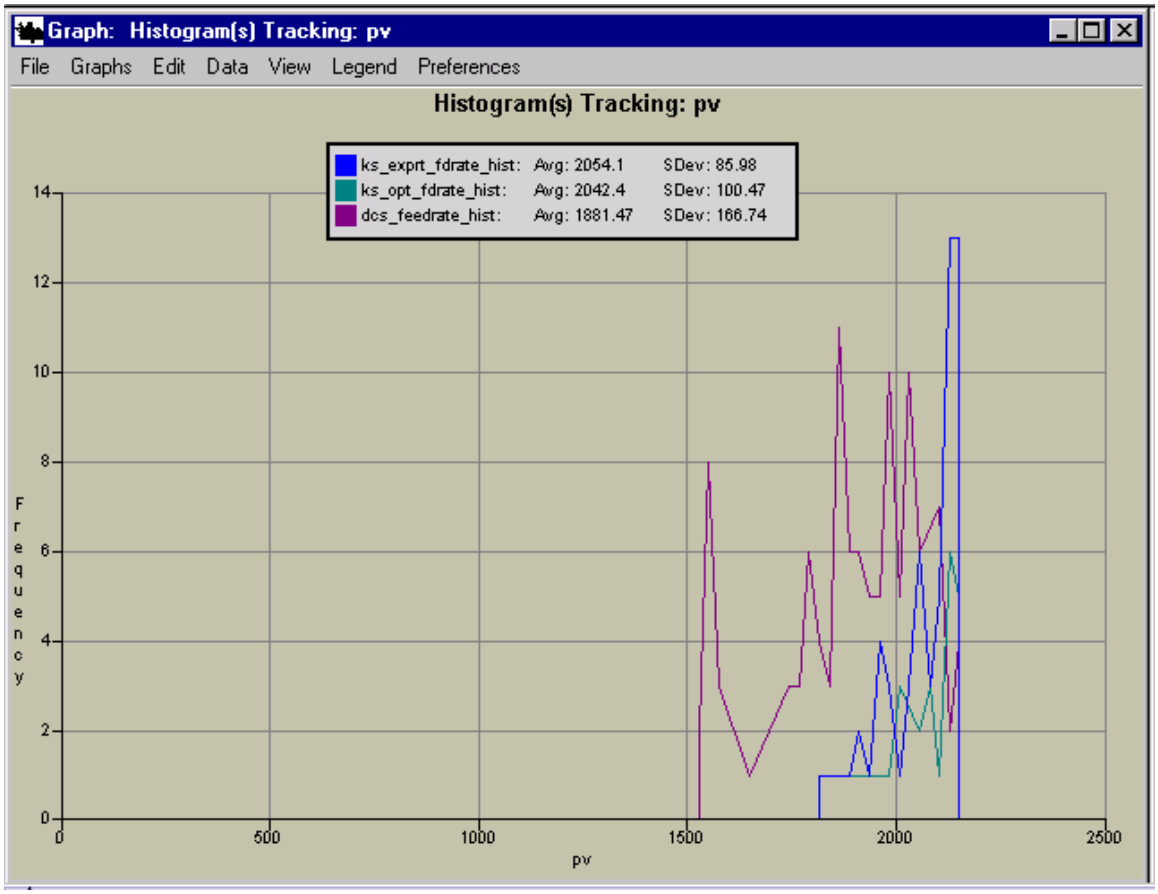



Figure x. Report

Figure x. Email Configuration




Putting it all together

- Configuring an application in KnowledgeScape
- Writing crisp and expert control rules
- Creating adaptive, on-line neural models of the process
- Creating genetic algorithm optimizers for the process

Results in the Minerals Processing Industry

Conclusions

References

Genetic Algorithms in Search, Optimization and Machine Learning, David E. Goldberg 1989

Adaptive Object-Oriented Optimization Software System. United States Patent number 6,112,126 dated August 29, 2000.

# Appendixes

## Keywords

The following is a list of keywords that will be available in the grammar assistant with an example of the use after each keyword. These will be prompted into the keyword window as needed in the rule structure.

**AGO**

Used in conjunction with VALUE keyword to reference the database.

**Syntax**

**VALUE OF *attribute* AS OF *number scale* AGO**

Attribute is a location and name of an attribute. Number is an integer value. Scale can be DATAPOINT(S), WEEK(S), DAY(S), HOUR(S), MINUTE(S), or SECOND(S).

**Examples**

VALUE OF power AS OF 10 MINUTES AGO
VALUE OF load AS OF 5 DATAPOINTS AGO

**AS**

Used in conjunction with VALUE keyword to reference the database.

**Syntax**

**VALUE OF *attribute* AS OF *number scale* AGO**

Attribute is a location and name of an attribute. Number is an integer value. Scale can be DATAPOINT(S), WEEK(S), DAY(S), HOUR(S), MINUTE(S), or SECOND(S).

**Examples**

VALUE OF power AS OF 10 MINUTES AGO
VALUE OF load AS OF 5 DATAPOINTS AGO

**DATAPOINT(S)**

Used in conjunction with VALUE, AVG, STDEV, and RATEOFCHANGE keywords to reference a specific number of datapoints in the database.

**Syntax**

**AVG OF attribute OVER number DATAPOINTS**

Attribute is a location and name an attribute. Number is an integer value.

**Examples**

VALUE OF power AS OF 1 DATAPOINT AGO
AVG OF flow OVER 7 DATAPOINTS
STDEV OF production OVER 25 DATAPOINTS
RATEOFCHANGE PER DATAPOINT OF speed OVER 5 DATAPOINTS

## DECLARE

Used to assert temp names as variables within a rule.

**Syntax**

**DECLARE tempname1, tempname2.... tempnameN**

**Remarks**

This type of rule writing is very powerful, it allows a user to declare local variable that make elaborate equation writing a bit more obvious rather than having lengthy rules that are difficult to read. There are however precautions to take. The local variables store whatever the value for the attribute named but only hold it for this one rule, they also have to be transferred back into an attribute if they wish to be stored. The last rule shows a case when the local are transformed back into a stored attribute value. Saying X =Y/Z will not work.

**Examples**

**declare** X, Y, Z
X= production
Y= tank level
Z= mill speed
IF X >Y THEN
REPORT 'production = ', X AND SEND X ELSE
Production = X + tonnage/(Y *Z)

## INVOKE

Branch rule firing to another location in the database.

**Syntax**

**INVOKE node group**

Node is the node name configured with the desired rule group. Group is the name of the rule group containing the rules desired to be fired.

**Remarks**

- Omit node name in rule syntax if invoked rule group exists on the same node
- After all rules in the newly invoked group have fired the inference engine returns to the rule preceding the original INVOKE call.

**Examples**

INVOKE emergency_rules
INVOKE main_conveyor status

**IS -** IF speed **IS** high AND pressure > 3000 THEN load = VALUE OF load AS OF 5 DATAPOINTS AGO AND REPORT 'Load is High!'

## OF

Used in conjunction with a database accessing keywords such as AVG, RATEOFCHANGE, STDEV, and VALUE.

**Syntax**

**AVG OF attribute OVER number scale**
**RATEOFCHANGE PER scale OF attribute OVER number scale**
**STDEV OF attribute OVER number scale**
**VALUE OF attribute AS OF number scale AGO**

Attribute is a location and name of an attribute. Number is an integer value. Scale can be DATAPOINT(S), WEEK(S), DAY(S), HOUR(S), MINUTE(S), or SECOND(S).

**Examples**
AVG OF power OVER 10 MINUTES
RATEOFCHANGE PER DATAPOINT OF load OVER 5 DATAPOINTS
STDEV OF production OVER 1 HOUR
VALUE OF speed AS OF 4 DATAPOINTS AGO

**OVER OF**
Used in conjunction with a database accessing keywords such as AVG, RATEOFCHANGE, STDEV, and VALUE.

**Syntax**
**AVG OF attribute OVER number scale**
**RATEOFCHANGE PER scale OF attribute OVER number scale**
**STDEV OF attribute OVER number scale**
**VALUE OF attribute AS OF number scale AGO**
Attribute is a location and name of an attribute. Number is an integer value. Scale can be DATAPOINT(S), WEEK(S), DAY(S), HOUR(S), MINUTE(S), or SECOND(S).

**Examples**
AVG OF power OVER 10 MINUTES
RATEOFCHANGE PER DATAPOINT OF load OVER 5 DATAPOINTS
STDEV OF production OVER 1 HOUR
VALUE OF speed AS OF 4 DATAPOINTS AGO

**PER -** power =power +RATEOFCHANGE **PER** DAY OF power OVER 1 WEEK
**RATEOFCHANGE**
Returns the rate of change of an attribute over the specified time period.

**Syntax**
**RATEOFCHANGE PER *scale* OF *attribute* OVER *number scale***
Attribute is a location and name of an attribute. Number is an integer value. Scale can be DATAPOINT(S), WEEK(S), DAY(S), HOUR(S), MINUTE(S), or SECOND(S).

**Examples**
RATEOFCHANGE PER MINUTE OF power OVER 10 MINUTES
RATEOFCHANGE PER DATAPOINT OF load OVER 5 DATAPOINTS

**REPORT / TO**
Writes text messages to the log utility.

**Syntax**
**REPORT '*message*' TO '*log*'**
**REPORT 'message', attribute ,'message' TO 'log'**
Message is the body of script sent to the log writer. Log is the destination log name where the text message will be written. Attribute is a location and name of an attribute

**Remarks**

- Omitting the TO keyword and destination log name will write the text message to the default "REPORT" log.
- Carriage returns are allowable inside text messages.
- Insert attribute values into messages by following the example below.
- If the destination log does not exist it will be created when the REPORT rule is fired.

**Examples**

REPORT 'Knowledgebase Running'
REPORT 'Feed Pressure Too High!' TO 'emergency_log'
REPORT 'Feed Flow =', flow,' gpm.'

**RETURN - RETURN** opt_production (This rule is used in the optimizer, usually the finished function to return the final value that the optimizer chose)

**SEND**

Transmits attribute contents to external data devices via the data server bridge.

**Syntax**

**SEND *attribute***

Attribute is a location and name of an attribute.

**Remarks**

- Attribute must have data server configuration to implement the SEND command.

**Examples**

**SEND setpoint**

SEND valve output

**STDEV**

Returns the standard deviation of the specified attribute.

**Syntax**

**STDEV OF *attribute* OVER *number scale***

Attribute is a location and name of an attribute. Number is an integer value. Scale can be DATAPOINT(S), WEEK(S), DAY(S), HOUR(S), MINUTE(S), OR SECOND(S).

**Examples**

STDEV OF power OVER 10 MINUTES
STDEV OF load OVER 5 DATAPOINTS

**IF / THEN / ELSE**

Conditional clause. Compares two or more conditions for equality. If results are true, the THEN instructional set is used; if the results are false, the ELSE instructional set is used.

**Syntax**

**IF condition THEN action ELSE action**

Condition phrase can be either a fuzzy or crisp comparison. Action can be either a fuzzy or crisp rule.

**Examples**

IF tank level 90 THEN pump speed = 100
IF flow IS high THEN feed_rate IS low
IF pump status = 1 THEN valve status = 1 ELSE valve status = 0
IF message = "open" THEN output = 100 ELSE output = 0

EMAIL / TO / CC / SUBJECT / ATTACH

Automatically send messages anywhere in the world via email. Using the keywords above in the following syntax any PC can be updated by KnowledgeScape via the world-wide-web.

**Syntax**
**EMAIL '*text message*' TO '*somebody@somewhere*' CC '*somebodyelse@somewhereelse*' SUBJECT '*text description*' ATTACH '*drive:\file*'**
Text message is the body of the e-mail script. Somebody@somewhere is the email address of the primary recipient. Somebodyelse@somewhereelse is the address of the secondary recipient. Text description is the subject script. Drive:\file is the location and name of the file intended to be attached to the email message.

**Example**
EMAIL 'Yesterdays average production.' TO 'CEO@headquarters.com' CC 'accounting@headquarters.com' SUBJECT 'Daily plant production' ATTACH 'C:\temp\shift_report.txt'

**UPDATE / USING**
Add data point to histogram data set.

**Syntax**
**UPDATE *histogram* USING *attribute***
Histogram is a name of a histogram. Attribute is a location and name of an attribute.

**Examples**
UPDATE histogram_1_level USING sump level
UPDATE histogram_feed_off USING feed_pv

**VALUE -** IF speed IS high AND pressure > 3000 THEN load = **VALUE** OF load AS OF 5 DATAPOINTS AGO AND REPORT 'Load is High!' AND INVOKE high_load_emerg_rules
**SECOND(S) / MINUTE(S) / HOUR(S) / DAY(S) / WEEK(S)**
Used in conjunction with VALUE, AVG, STDEV, and RATEOFCHANGE keywords to reference the last period of time in the database.

**Syntax**
**STDEV OF attribute OVER number MINUTES**
Attribute is a location and name of an attribute. Number is a integer value.

**Examples**
VALUE OF power AS OF 1 MINUTE AGO
AVG OF flow OVER 5 MINUTES
STDEV OF production OVER 1 HOUR
RATEOFCHANGE OF speed OVER 30 SECONDS

Intrinsic Functions The following are a list of the available callouts with an example rule next to them. The grammar assistant will show all applicable functions as the user writes the rules.

**ABS -** counter= **ABS**(counter-10)

**CEILING**

Rounds a floating point value to the next highest whole integer.

**Syntax**
**CEILING(value)**
Value is either a number or attribute name.

**Examples**
CEILING(1.1) **equals 2**
CEILING(14.0212) **equals 15**
CEILING(-3.2) **equals -3.0**
CEILING(setpoint)

**COS**

Returns the cosine of a number or attribute. Cosine argument is always calculated in radians.

**Syntax**
**COS(value)**
Value is either a number or attribute name.

**Examples**
COS(1.1047) **equals 0.449402**
COS(angle)

**CURRENTTIME**

Returns an integer value equaling the number of seconds from the year 1900.

**Syntax**
**CURRENTTIME(value)**
Value is either a number or attribute name representing the offset in seconds from the current time.

**Examples**
CURRENTTIME(0) **equals the current time without offset**
CURRENTTIME(3600) **equals one hour in the future**

**EXP**

Returns the exponential value of the natural logarithm of e (2.7163...)

**Syntax**
**EXP(value)**
Value is either a number or attribute name.

**Examples**
EXP(2.13) **equals 8.415**
EXP(-5) **equals 0.006738**

EXP(power)

**FLOOR**
Rounds a floating point value to the next lowest whole integer.

**Syntax**
**FLOOR(value)**
Value is either a number or attribute name.

**Examples**
FLOOR(1.9) **equals 1**
FLOOR(9.25) **equals 9**
FLOOR(-3.2) **equals -4.0**
FLOOR(flow)

**IDAY** / **IMINUTE** / **ISECOND** / **IMONTH** / **IWEEK** / **IYEAR** / **IHOUR**
Returns the current time in numerical format as specified by the function name.

**Syntax**
**IDAY()** or **IMINUTE()** or **ISECOND()** or **IMONTH()** or **IWEEK** or **IYEAR()** or **IHOUR()**

**Examples**
IDAY()
IMINUTE()

**LN**
Returns the natural logarithm based on e (2.7183...)

**Syntax**
**LN(value)**
Value is either a positive number or attribute name.

**Examples**
LN(1.5) **equals 0.405**
LN(5) **equals 1.609**
LN(growth)

**LOG**
Returns the logarithm base 10

**Syntax**
**LOG(value)**
Value is either a positive number or attribute name.

**Examples**
LOG(1.5) **equals 0.176**
LOG(5) **equals 0.699**
LOG(growth)

**MAX -** load= **MIN**(100, (tonnage*density))

**MIN/MAX**
Returns the smallest value between two arguments

**Syntax**
**MAX(value1, value2)**
Value1 and Value2 can be numbers, attribute names, or expressions.

**Examples**
MIN(3, 101) **equals 3**
MIN(-6, -3.2) **equals -6**
MIN(-12, MIN( -6, 10)) **equals -12**
MIN(feed, power * 5)

**NUMVALFOR -** production=**NUMVALFOR** (status)
Status is a non-numerical attribute.

**ROUND**
Changes a decimal (floating-point) number to the closest integer.

**Syntax**
**ROUND(value)**
Value can be numbers, attribute names, or expressions.

**Examples**
ROUND(7.9) **equals 8**
ROUND(-6.5) **equals -7**
(ROUND((1.7256)*100))/100 **equals 1.73**
ROUND(tonnage)

**SIN**
Returns the sine of a number or attribute. Sine argument is always calculated in radians.

**Syntax**
**SIN(value)**
Value is either a number or attribute name.

**Examples**
SIN(0.707) **equals 0.650**
SIN(angle)

**SQRT**
Returns the positive square root of a number or attribute.

**Syntax**
**SQRT(value)**
Value can be a number, attribute name, or expression.

**Examples**
SQRT(4) **equals 2**
SQRT(flow)

**TAN**
Returns the tangent of a number or attribute.

**Syntax**
**TAN(value)**
Value can be a number, attribute name, or expression.

**Examples**
TAN(1.14) **equals 2.176**
TAN(angle)

**TRUNCATE**
Drops digits preceding the decimal point on a floating-point attribute or number.

**Syntax**
**TRUNCATE(value)**
Value is either a number or attribute name.

**Examples**
TRUNCATE(7.70984) **equals 7**
TRUNCATE(setpoint)

Context Options

These options are available to you for use in the writing of rules and will be prompted to you as necessary when writing the rules. The majority of the options available to the user in this column will be node names and attribute names. The other options will used in the optimizer. See the chapter on Optimizers for a more detailed description of the options specifically associated with optimizers