HOW TO USE EXPERTPRIZ

1. What is ExpertPRIZ? What is it for?

ExpertPRIZ is an integrated shell which helps you
- to solve different computational problems and
- to build applications in Windows 3.0 environment.

ExpertPRIZ can be used in fields where there is a need for combining
- complicated mathematical **computations** and
- **database queries** with
- employing **empirical knowledge** and
- **decision making**.

ExpertPRIZ includes the following parts:
Solver - a problem solver, computing unit;
Expert - an inductive expert system shell;
Data - a dBase-compatible database system.

Both **Solver** and **Expert** contain their own knowledge representation tools and inference engines. **Data** provides functions for database queries and includes a database editor. All these parts can be used independently of each other, but the power of ExpertPRIZ lies in their interaction.

Solver has its own **input language** for specifying computational problems and concepts which can be used in problem specifications. Solver is the most important part of ExpertPRIZ; speaking about the input language of ExpertPRIZ, the input language of Solver is always meant.

In order to understand how ExpertPRIZ and, in particular, Solver works we start with some basic concepts.

2. Basic concepts

Here you find explanations of some terms used most frequently when speaking about **Solver** of ExpertPRIZ.

Basic concepts of **Expert** see "Expert Editor".

Basic concepts of **Data** see "Database subsystem Data".

Object is any entity with name and specified properties; it may have a complicated internal structure which may remain unknown to the user. An object can be defined by the user even if he does not precisely know its structure and properties.

Concept is a specification of properties of a class of objects of some kind. Concepts are used for specifying objects. The object that is specified by means of a prototype concept acquires all the properties of this concept and contains all its components, too. A set of concepts stored in a file is called
concept base (or conceptual knowledge base).

Every **problem** can be regarded as a certain situation describable by means of objects and relations between them. In order to solve a problem with ExpertPRIZ you have to describe it. **Problem specification** is a text in the input language of ExpertPRIZ where objects and relations between the objects are specified.

Relation is a specification showing how to derive values of some objects from the values of other objects.

Problem model is an extended representation of problem specification, which is used in carrying out computations. ExpertPRIZ has only one problem model - that of the current problem. The model is extended step by step according to the specification: new objects and relations will be added to the existing model.

The **input language** of ExpertPRIZ consists of statements which can be of the following types: specifications, problem statements, commands and comments. **Problem statement** is a statement indicating the objects for whose values or analytical expression for dependencies of these numeric objects are to be found. **Commands** are intended to perform some actions without choosing them "manually" from the menu.

Application is a set of files consisting of knowledge bases filled with particular knowledge, databases, problem specifications, graphical input-output forms, etc. - all of them prepared for solving problems from a certain field. In addition, an application contains the shell - ExpertPRIZ.

3. Windows of ExpertPRIZ

After starting ExpertPRIZ there are three main windows -**Workspace**, **Concept Base** and **Log** - in the frame window of ExpertPRIZ. During the working session some other windows and dialog boxes can appear and disappear according to the actions performed.

The **Workspace** window is a standard edit window for preparing input text (problem specification) for Solver. In this window the commands of the **File** and **Edit** menus can be used for working with files and editing text.

The **Concept Base** window allows to work with concepts. In this window you can see the list of the names of the concepts stored in the concept base currently open. For creating new concept bases and opening existing ones use the **New** and **Open** commands of the **File** menu. In this window you can show and edit concept descriptions (an edit window for the concept appears); add, delete and rename concepts, and also choose concepts for using in problem specification.

The **Log** window is an edit window for the protocol of the working session with ExpertPRIZ. How detailed the protocol will be depends on the **Viewing** options chosen from the **Options** menu. Results of computations are also written in the **Log** window. The size of the Log buffer is 4 Kbytes. When the buffer becomes full, ExpertPRIZ saves the contents of it in the file currently open. If no file is opened in the **Log** window, then the system asks about it. It is also possible to write the protocol to the end of an existing text file. For that purpose you have to open this file in the **Log** window.

You can change the **input focus** between these windows by **mouse click** in the window you wish to activate, by pressing **Ctrl+F6**keys, or by choosing the window in the **Window** menu.

You can move the windows inside the frame window of ExpertPRIZ, change their sizes, make them iconic, etc. - for all that use the standard facilities of Windows 3.0.

4. The menu commands of ExpertPRIZ

4.1. The File menu

Here the standard **File menu** commands are presented.

New - associates the active edit window with a new (untitled) text file. If the Concept Base window is active, makes an empty concept base file.

Open - if the active edit window is either Workspace or Log, opens text file in the active window; if the Concept Base window is active, then opens concept base file.

Save - saves the file currently open in the active edit window. If the Concept Base window or the Concept window is active, saves the concept description.

Save As - saves the file currently open in the active edit window (Workspace or Log) under a new name. If the Concept Base window is active, saves the current concept base under a new name. In case the Concept window is active, allows to give a new name to the currently edited concept and save it.

Print - allows to print the contents of the active window.

Printer Setup - allows to set up printing options.

Exit - quits the working session with ExpertPRIZ.

4.2. The Edit menu

The **Edit menu** contains standard commands for the edit windows:

Undo - undoes the last change in the text currently edited.

Cut - cuts the selected text and copies it to the clipboard.

Copy - copies the selected text to the clipboard.
Paste - pastes the text from the clipboard to the current position of the caret.
Delete - deletes the selected text.
Select All - selects the whole text in the buffer of the active edit window.

4.3. The Model menu

The **Model menu** enables to carry out some actions on the problem model.

Clear - clears the existing problem model. After executing that command the problem model is empty and you can start building a new one.

Reset - deletes the values of all objects existing in the current problem model. This enables to solve different problems or the same problem with different initial values on the same problem model.

Show - shows the contents (objects, equations, relations, etc.) of the current problem model in the Log window.

4.4. The Run menu

These commands allow to start computations and consultations with expert systems, and evoke special editors.

Perform - Solver executes the statements that are selected in the Workspace window text.

Expert - starts a consultation course with an expert system.

Expert Editor - evokes Expert Editor.

Data Editor - evokes Data editor.

Form Editor - evokes Form Editor.

4.5. The Options menu

Here you can change some options of ExpertPRIZ.

Equations - allows to change the parameters (initial approximations, precision, maximal number of iterations) used in solving equations.

Limits - allows to change some system limits: maximal number of objects in the problem model, relations, constants, etc.

Viewing - allows to change the following options:

Output Precision - output fraction length of numerical values;

Show Algorithm - output of problem solving algorithm in the Log window;

Show Results - output of results of computations in the Log window;

Show Compile - output of compiled input text and error messages (if any) in the Log window;

Show Explanations - output of expert system explanations (tracing the path in the decision tree) in the Log window;

Check Consistency - checking the consistency of arithmetical equations and removing the clashing ones.

4.6. The Window menu

This menu is a standard MDI window menu of Windows. Its commands are related to displaying the child windows inside the frame window.

Cascade - displays the windows of ExpertPRIZ in a cascade.

Tile - displays the windows of ExpertPRIZ in a tile mode.

Arrange Icons - arranges the icons.

The next lines in the menu display the currently open child windows and the names of the files associated with these windows. Among them you can choose the window you wish to activate.

4.7. The Help menu

This menu has the following commands providing some help about using ExpertPRIZ:

Index,
Basics,
Keyboard,
Commands,
Procedures,
Glossary,
Using Help,
About ExpertPRIZ.

5. The commands of input language

About the **input language** of ExpertPRIZ see "Input Language".

The **commands** of input language are intended to perform some actions without choosing them "manually" from the menu.

For example, before solving a new problem you have to delete or reset the model of the previously solved problem. This can be done by the appropriate command of the Model menu. In some cases it would be more convenient to have this done automatically, without the user's interruption. This is why the commands are included to the input language.

If ExpertPRIZ finds a command when processing an input text (problem specification), it executes the command immediately, and processing of the input text continues.

Command of the input language is a statement beginning with an exclamation mark ("**!**") followed by the command name (in lower-case letters) and parameters, if the command has any.

The commands are following.

!clear [y|n]

for deleting the current problem model.

Possible forms of that command:

!clear - asks confirmation about clearing;

!clear y - clears the model without any questions;

!clear n - does nothing.

It is recommended to start problem specification texts with the command !clear y.

!reset

resets the problem model (deletes the values of all the objects existing in the current model).

!show

displays the contents of the current problem model in the Log window.

!file <filename> [#<entry name>]

reads input from a file. If the name of the file <filename> does not contain extension, the default extension "**.TXT**" is added. One file can contain several parts of input text which can be executed separately. The beginning and the end of an entry part in the file are text lines in the form: **#<name>**.

Some examples:

!file problem #Car - processes the input text of the entry named Car from the file PROBLEM.TXT. The text is processed from the text line that follows the line "#Car" to the line of the same form, or up to the end of the file (if the end of the entry is not marked).

!file problem - the whole text from the file PROBLEM.TXT is executed.

!expert <knowledge base name>

starts consultation course with an expert system. The parameter determines the expert knowledge base file name. If the file name does not contain extension, the default extension "**.EXP**" is added.

Example:

!expert demo - starts the demonstration according to the expert knowledge file DEMO.EXP.

!use <concept base name>

opens the concept base. In case the concept base is not found, makes a new (empty) one with the given name and opens it. If the concept base name does not contain extension, the default extension "**.CPT**" is added.

Example:
!use demo - switches to the concept base DEMO.CPT.

!concept <concept name>

saves the text following this command as a new concept under the given name. The text is considered as a concept description up to the next line beginning with "!", or to the end of the selected text (if the text is in the Workspace window) or file (in case the input comes from a file). The end mark of the concept description (line beginning with "!") is skipped.

The concept will be saved to the concept base currently open.

For example, the following input lines describe a concept square and add it to the current concept base:

```
!concept square
* Square  a - side; d - diagonal; S - area; p - perimeter.
d^2 = 2*a^2
S = a^2
p = 4*a
!
```

6. How to use Solver in problem solving

Suppose you have the following computational problem. You wish to buy 14 details. The price of each detail is $129. The tax you have to pay for that purchase is 7% of the cost of the details. You wish to calculate the total cost of the purchase.

After starting ExpertPRIZ the **Workspace** window is activated. You can see the caret blinking in that window. It means that the **Workspace** window has the input focus and you can start typing the problem specification text.

First of all, you can use Solver as you use a pocket calculator - type the arithmetical expression for calculating the needed value (let X denote the total cost):

```
X = 14*129+0.07*14*129
?
```

The line "?" is a problem statement, which asks Solver to carry out all the possible computations. You can also type "? X" which asks to compute the value of X, but so far you have only the object X, and hence the problem statements "?" and "? X" are equivalent in this case.

After typing the two input lines in the **Workspace** window, select this text using **mouse**, or choosing **Select All** from the **Edit** menu, or pressing the **F2** key. Now you have the problem specification text selected, and you can choose the **Perform**command from the **Run** menu (or press the **F3** key) to get Solver compute the answer.

The result will be displayed in the **Log** window in the form:

--> X = 1932.42

If you have no answer in the **Log** window, check the **Viewing**options from the **Options** menu: **Show Results** has to be switched on. It is also recommended to have the **Show Compile** option switched on - in case of errors the error messages will appear in the **Log** window.

Now you have got the answer to your problem, but the way of specifying it was not very intelligent, and gives no help if you have similar probelm with another number of details, price etc. For that purpose let us specify the same problem once again. Type the following lines in the **Workspace** window:

```
!clear y
nDetails = 14
Price = 129
TaxRate = 7
DetailsCost = nDetails * Price
Tax = TaxRate/100 * DetailsCost
Total = DetailsCost + Tax
? Total
```

The first input line is the **!clear** command by which the model of the previous problem will be deleted.

The next three input lines are equations giving initial values to the objects nDetails (which denotes number of details), Price (price of a detail) and TaxRate (tax %).

The three equations that follow specify some more objects (DetailsCost, Tax, Total) and relations between these objects. The last input line is a problem statement which asks Solver to compute the value of the object Total (the total cost you wish to know).

Select the text starting with the **!clear** command and choose **Perform** from the **Run** menu, or press **F3**.

The answer is displayed in the **Log** window:

--> Total = 1932.42

Now you can easily solve similar problems with different prices, amounts and tax rates: just change the values of the objects nDetails, Price and TaxRate in the **Workspace** window, select the problem specification text (beginning with !clear) and perform it.

If you are going to use this input text during next working sessions, you can save it in a file and use it just when you need it.

In ExpertPRIZ, it is also possible to find **analytical expressions**for dependencies of some objects on some other objects. For example, you can ask Solver to find an analytical formula, which expresses the dependence of Total on nDetails and Price. Just type another problem statement in the form:

? nDetails Price -> Total
and ask Solver to perform it. The result will be:

--> Total = 1.07*nDetails*Price

A question "Add generated equations to model?" will also appear. In some cases it is necessary to add generated in that way equations to problem model, but in our case you can answer "No".

7. How to use a concept

In case you have to solve problems about total costs of some details more often, it makes sense to describe the situation as a concept.

For that purpose choose the concept **(untitled)** in the list of concept names in the **Concept Base** window and open it. After that the **Concept** window appears and gets the input focus, too. You can start typing the description of the concept.

It would be nice to begin with some comments (comment is a text line beginning with an asterisk ("*"):

* nDetails - number of details
* Price - price of a detail
* TaxRate - tax percent
* DetailsCost - cost of details without tax
* Tax - total tax
* Total - total cost
DetailsCost = nDetails * Price
Tax = TaxRate/100 * DetailsCost
Total = DetailsCost + Tax

As you can see, the three equations are the same as in problem specification text. If you still have the text in the **Workspace** window, you can copy it and paste it in the **Concept**window.

Now the situation is described by means of 6 objects (DetailsCost, nDetails, etc. - called the components of the concept) and 3 equations giving the relations between the components.

You can save the concept using **Save** or **Save As** from the **File**menu. As your conept is still untitled, both the commands will ask for a name for the concept. You can give a name - let it be "cost" - to it, and close the **Concept** window. The concept cost is saved in the current concept base.

Now you can specify the same problem using the concept cost:

```
!clear y
Dolls cost nDetails = 14 Price = 129 TaxRate = 7
? Dolls.Total
```

As usual, the first input line is the **!clear** command. The next line specifies an object Dolls which is of type cost. It
means that the object Dolls acquires all the components and the relations of the concept cost. The initial values of components can be given on the same text line.

The problem statement "? Dolls.Total" asks Solver to compute the component Total of the object Dolls.

After executing the problem (select the text and perform it) you can see the answer in the **Log** window:

```
--> Dolls.Total = 1932.42
```

Using the concept cost, it is also possible to solve different problems which can occur in the same situation.

For example, you can compute tax rate, knowing the total cost, price of a detail and number of details:

```
!clear y
Spoons cost Total = 1023.35 Price = 97 nDetails = 10
? Spoons.TaxRate
```

The answer will be:

```
--> Spoons.TaxRate = 5.5
```

Another example: tax rate and the total tax you have to pay are given, and you wish to know the cost of details. The concept cost can be used once again:

```
!clear y
Screws cost TaxRate = 2 Tax = 21
? Screws.DetailsCost
```

After performing this problem specification, the result appears in the **Log** window:

--> Screws.DetailsCost = 1050

8. A more complicated problem

Suppose you have to buy 24 pairs of knives and forks. The price of a knife is $12, and you have to pay 2% of additional tax. A fork costs twice as a knife but you get them tax-free. You are interested in the costs of knives and forks, and in the total cost of the purchase.

In the **Workspace** window, you can give the problem specification using the concept cost you have defined:

```
!clear y
nPairs = 12
Knives  cost  nDetails = nPairs  Price = 12  TaxRate = 2
Forks  cost  nDetails = Knives.nDetails  TaxRate = 0
Forks.Price = 2*Knives.Price
Knives.Total + Forks.Total = TotalCost

? Knives.Total  Forks.Total  TotalCost
```

As usual, start with the **!clear** command. Let a separate object (nPairs) denote the number of pairs - then the value of it can be easily changed in case you wish to repeat the computations with a new value.

The next line specifies an object Knives as a cost, which has the number of details equal to nPairs, Price equal to 12, and TaxRate being 2.

An object Forks is specified by means of the concept cost, too. It is also shown that the number of forks is the same as the number of knives, and tax rate is 0%.

The equation Forks.Price=2*Knives.Price expresses the fact that a fork costs twice as much as a knife.

The last equation shows the relation between the costs of knives and forks, and the total cost.

Finally, the problem statement asks Solver to compute the three values you are interested in.

Having performed the problem, you will see the result in the **Log** window:

```
--> Knives.Total = 293.76
--> Forks.Total = 576
--> TotalCost = 869.76
```

As you can see, Solver gets the problem specification as an input, and gives you the results. What is behind that? - a question may occur. Look at the problem model Solver has built according to your specification: choose the command **Show** from the **Model** menu. The contents of the problem model - all the objects and relations between them - are shown in the **Log** window.

Model is an extended representation of problem specification: you give an equation nPairs=12, Solver adds to the model a numerical object nPairs and an equation nPairs=12; you describe an object Knives using the concept cost, Solver adds to the model a structural object Knives with the numerical components nDetails, Price, etc., and also adds the equations Knives.nDetails=nPairs, Knives.Price=12, etc.

When Solver gets a problem statement, it tries to find the way the computations have to be carried out - the algorithm of solving the particular problem. After the algorithm is found, Solver performs the computations.

You can also have a look at the algorithm: switch the **Show Algorithm** option on from **Viewing** of the **Options** menu. After that, solve the same problem you have specified in the **Workspace** window once again. Before displaying the computational results, an algorithm appears in the **Log** window. This algorithm shows you which relations (in our case all the relations are given by equations) and in which order were used in computing the result.

Now you have acquired the basic skills of using Solver. The last example presented here will show you how to use a more complicated concept. Suppose you have the same problem of buying knives and forks. You can specify the following concept (let the name of it be "buy"):

```
* Knives - a lot of knives
* Forks - a lot of forks
* nPairs - number of the pairs
* Total - total cost
Knives  cost  nDetails = nPairs
Forks  cost  nDetails = nPairs
Total = Knives.Total + Forks.Total
```

After that you can use this concept to specify the problem:

```
!clear y
To_Ann buy Knives.Price=12 Knives.TaxRate=2 Forks.TaxRate=0
To_Ann.Forks.Price = 2*To_Ann.Knives.Price
To_Ann.nPairs = 24
? To_Ann.Total
```

In this specification, compound names consisting of three names are used. For example, the name To_Ann.Forks.Price denotes the component Price of the component Forks of the object To_Ann.

The result will be displayed in the **Log** window:

--> To_Ann.Total = 869.76