CSCU9T6
Rule Based Systems 1

Aims

- To learn about implementing a simple rule based expert system using **e2glite**.

Objectives

By the end of this session, students will be able to:
- Understand the structure of an **e2glite** knowledge base.
- Use an **e2glite** knowledge base from within an HTML document.
- Create a knowledge base for a simple rule based system with **e2glite**.

Remember to register your practical attendance in the usual way!

Further Documentation

There is reference material about **e2glite** as well as general information about rule-based systems at **`http://expertise2go.com/webesie/e2gdoc/`**

Getting Started

- ➢ Navigate to **Groups on Wide**: open the folder **CSC9T6\practicals**. Find the folder **RuleBasedSystems**. Copy this entire folder into the **Web** or **WWW** folder in your own home folder.
- ➢ Open your copy of **RuleBasedSystems**. You should find these five files:
  - o **e2glite.jar** This is a Java Archive (JAR) file containing a set of compiled Java classes implementing a rule based system shell. The Java classes make up an *applet* (more on this later). You will not need to modify this file.
  - o **cars.kb** This is a “knowledge base” file, which provides a body of rules to be used by e2glite. In this case, the rules concern how to diagnose car problems.
  - o **weather.kb** This is another knowledge base which we will be using later on.
  - o **rbs-examples.html** This is an HTML file that is used for running the **e2glite** applet.
  - o **errors.html** This contains explanations of the error codes used by **e2glite**.

Using a Rule-Based System

The car won't start! Our first expert system will help us to decide what to do.

- ➢ Open the file **rbs-examples.html** in a web browser. (You should be able to do this by simply double-clicking on it).

(If you see a message saying that the browser has "restricted the file from showing active content", click on the information bar at the top and choose "Allow Blocked Content".)

There are two rule-based systems examples on this page. We will begin by looking at the first one: the Auto Diagnosis Advisor.

Here are the symptoms displayed by the car. When you turn the key to start the car the engine tumbles slowly but the car does not start. The headlights light up when they are switched on and they go dim when you try to start the car. You can't read the fuel gauge without starting the car.

- ➢ Click the button marked "Start the consultation". Work through the consultation, answering all the questions you are asked. What does the system recommend?
- ➢ A trace of the consultation is displayed in the small debugging window that popped up at the start of the consultation. You may find this useful to look at.
- ➢ Restart the consultation (reload the file), and try giving different answers to the questions. Notice all the different kinds of question prompts that appear. Try to spot examples of the following kinds of prompt:
  - o `YesNo` (radio buttons with yes/no values)
  - o `MultChoice` (radio buttons with text string values)
  - o `Choice` (drop down list with text string values)
  - o `Numeric` (numeric input)

There are also two other types of prompt that are not used in this example:
  - o `ForcedChoice` (like `MultChoice`, but without the "I don't know" option)
  - o `AllChoice` (text string check box allowing multiple values to be selected)

- ➢ Close the file **rbs-examples.html** when you are finished.

Using e2glite from a Web Page

The **e2glite** shell is implemented as a Java *applet*. An applet is a form of Java program that can be embedded into a web page via a special HTML tag. Just like any compiled Java program, the e2glite consists of a set of class files. These class files have all been packaged together as a Java Archive file, **e2glite.jar**.

The most common way to run a Java applet is to invoke it from a web page via an HTML `applet` tag. There are two applet tags in the file **rbs-examples.html**.

- Right-click on the file **rbs-examples.html** and open it with TextPad.

Find the first applet tag. It looks like this:

```
<applet code="e2glite.e2g.class" archive="e2glite.jar"
        width="450" height="300">
<param name="KBURL" value="cars.kb">
<param name="DEBUG" value="true"
<param name="APPtitle" value="Auto Diagnosis Adviser">
<param name="APPSUBtitle" value="an eXpertise2Go Demonstration">
Java-enabled browser required to view this applet
</applet>
```

This tells the browser to load a Java applet, **e2glite.e2g.class** (the `code` attribute), contained in the archive file **e2glite.jar** (the `archive` attribute), which is in the same folder as the HTML document itself. The program is displayed in a space 450 pixels (picture elements) wide and 300 pixels high (the `width` and `height` attributes).

The applet takes a number of *parameters*, each supplied in a `<param>` tag. Each parameter has a `name` attribute and a `value` attribute. The first parameter, `KBURL`, specifies which knowledge base is to be used. Here it has the value "`cars.kb`". The second parameter, `DEBUG`, controls whether or not a debugging window will be displayed when the applet is running. The other parameters are used to customize the appearance of the applet by giving it a title and subtitle. There are other features, such as colours, and text size, that can also be adjusted via parameters, but we have used default values for these. The Reference section of the **expertise2go.com** website gives a full list of the parameters that are available.

- Look at the second applet tag in **rbs-examples.html**. What is the name of the file containing the knowledge base in this applet tag?

The ability to be incorporated into web pages makes **e2glite** very flexible. It is possible to integrate **e2glite** with JavaScript so as to produce dynamic web pages whose content is controlled by the output from an **e2glite** system. This is well beyond the scope of this practical, but if you would like to use this feature in your own projects, you can learn more at the **expertise2go.com** website.

- Close the file **rbs-examples.html** when you are finished.

Looking inside the Auto Diagnosis Advisor

Next we shall look inside the knowledge base file, cars.kb, to see how the Auto Diagnosis Adviser was constructed.

- Open **cars.kb** in TextPad.

The file is divided, using `REM` comments, into three sections: the Rule section, the Prompt section, and the Goal section. Read each of these sections carefully and satisfy yourself that you understand what they mean. Consult the lecture notes or ask a demonstrator if there is anything you do not understand. Close TextPad when you are finished.

Build your own rule based system: the Weather Advisory System

Your final task for this practical is to build a small rule-based system of your own. The Weather Advisory System asks the user various questions about the weather and then recommends what to wear. The knowledge base for this system is kept in the file **weather.kb.** There isn't much there at the moment.

Like the previous system, the Weather Advisory System is run in a web browser through the file **rbs-examples.html**. Don't try to run it yet or you will get an error message. We must add some knowledge to the knowledge base first.

- Open the file **weather.kb** in **TextPad**. (Make sure that you are opening your copy of this file, not the original copy on Groups on Wide.)

Look at the contents of the knowledge base. There are lots of `REM` lines but no rules and no prompts. At the bottom we see that there is a goal: it is to find a value for the attribute `[the recommendation]`.

Let's add our first rule to the knowledge base.

- Carefully type the following rule into the **rule section**:

```
RULE [Is it going to rain?]
If [precipitation] = "expected"
Then [the recommendation] = "Wear a raincoat!"
```

The quote mark " is the symbol above the number 2 on the keyboard.

This rule says that if the `[precipitation]` attribute has the value `"expected"`, then `[the recommendation]` attribute is given the value `"Wear a raincoat!"` The name of the rule is `[Is it going to rain?]`. To find out if precipitation is expected we will add a prompt to ask the user:

- Carefully type the following prompt into the **prompt section**:

```
PROMPT [precipitation] MultChoice
"According to the weather forecast, precipitation is:"
```

```
"expected"
"not expected"
```

This is a multiple choice prompt, which displays the question `"According to the weather forecast, precipitation is:"` and gives the user a list of answers to choose from (including `"expected"` and `"not expected"`). The answer is stored in the attribute `[precipitation]`.

- Save the file **weather.kb**. (Don't close it yet, as we will be making more changes to it.)

- Open the file **rbs-examples.html** in a web browser. Your new knowledge base will be loaded automatically. Try out the Weather Advisory System.

If you have made any typing errors in **weather.kb** you will need to go back to **TextPad**, correct the error, and save the file. Then go back to the browser and reload **rbs-examples.html** to load the new version of the knowledge base.

Now let's make the system a little smarter. Precipitation can be either rain or snow. We will use a new attribute called `[the expected temperature]` to decide which it is going to be. If precipitation is expected, and the expected temperature is greater than 0, then the recommendation is wear a raincoat. However, if precipitation is expected, and the expected temperature is less than 1, the recommendation is wear a warm coat and boots.

- Modify the `[Is it going to rain?]` rule by adding a second condition, `[the expected temperature] > 0` to the premise of the rule. The two conditions should be combined using `and`.

- Next, add a second rule `[Is it going to snow?]` with a premise and conclusion as suggested above.

To find out the expected temperature, the system will ask the user via a `Numeric` prompt.

- Add the following to the prompt section:

```
PROMPT [the expected temperature] Numeric
"The expected daytime temperature (degrees Celsius) is:"
"-50"
"50"
```

This prompts the user to enter a numerical value between -50 and 50. The value will be stored in the attribute `[the expected temperature]`.

As a final touch, we will add a default for the system to recommend if it can't find a recommendation using the rules:

- Type the following into the Goal section:

```
DEFAULT [the recommendation] = "Wear whatever you want!"
```

- Save the file **weather.kb**, then reload **rbs-examples.html** into the browser and try out the improved Weather Advisory System. Fix any errors that you find.

Once you have got the system working, try to experiment with it by adding more rules and prompts that you have thought of yourself. For example, what should you wear on a hot, cloudless day? What about a cold, windy day?

Some tips:

- When writing prompts, ask the user for unambiguous, factual information. For example, ask “What is the expected wind speed?” rather than “Is it windy?”
- When writing rules, avoid checking the same information twice. For example, if you know that precipitation is expected, there is no need to consider whether the sky is clear.

That’s it for this worksheet, but you can carry on playing and experimenting with the Weather Advisory System if you want to. Have fun!

Savi Maharaj