Expert System Creator: A visual tool for expert systems construction

D. Pop, V. Negru, C. Sandru
University of the West
Timisoara, Romania
**Goals**

- Help domain experts to easily build expert systems for decision support, diagnosis, fraud detection etc;

- Create a graphical tool for:
  - representation of domain knowledge,
  - conversion between different representation forms,
  - integration within „host projects“,
  - support for expert systems shells (CLIPS, JESS etc.),
  - database integration (RDBMSs),
  - automatic construction of expert systems from available data sets (using data mining and learning algorithms).

- The tool should support all the phases of development of a software project:
  - analysis and design;
  - developing and implementation;
  - testing, debugging and tuning.
Knowledge Representation

**Rules Set**

- Easily built by human experts
- Important, mature and large systems are already in use: Mycin, Garvan ES1
- Production rules, association rules, rules with exceptions

**Sample:**

```
Rule VeryGood
    (Price ?P&(<= ?P 1000))
    (Quality Medium)
    (Service Good)
=>
    (class VERY-GOOD)
```
Knowledge Representation

**Decision Table**

- Automatic correctness analysis
- Compact form
- Easy to visualize and understand

**Sample:**

<table>
<thead>
<tr>
<th></th>
<th>VERY GOOD</th>
<th>GOOD</th>
<th>MEDIUM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Price</strong></td>
<td>&lt;= $1000</td>
<td>&lt;= $3000</td>
<td>&gt; $3000</td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td>Medium</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td><strong>Service</strong></td>
<td>Good</td>
<td>Good</td>
<td>-</td>
</tr>
</tbody>
</table>
Knowledge Representation

**Decision Tree**

- Procedural or object oriented programming code can be easily generated
- Automatically built from large datasets using data mining algorithms

**Sample:**

```
Decision Tree:

Price
  <= $1000  <= $3000  > $3000
  |        |        |
  Quality Quality Quality
    Medium Good  Good
      VERY-GOOD GOOD MEDIUM
```
The Equivalence of Representation Forms

Translation algorithms

A. Production Rules ⇔ Decision Tables
   • Easy
   • Pruning

B. Decision Tables ⇔ Decision Trees

Decision Table ⇒ Decision Tree
1. table has no rows nor attributes ⇒ the tree is a leaf node
2. at least one attribute, one row and two distinct classifications are present ⇒
   2.1. select one attribute
   2.2. create a decision node associated with this attribute
   2.3. for each distinct value of the attribute
       - create a new table by removing the selected attribute’s row
       - add a new arc with the parent identified by the decision node and the child
         obtained from recursively processing the above table
Expert System Architecture
Rules Set Designer

✓ Production rules
✓ Project-based: a project may contain one or more modules;
✓ Structured designer using blocks to group entities in higher level structures;
✓ Exports to CLIPS/JESS file format;
✓ Integrated debugger and profiler;
Rules Set Designer

A Visual Tool for EXPERT SYSTEMS Construction
Rules Set Debugger
Rules Set Profiling

- Helps user to find time-consuming rules;
- Traces the following:
  - fired rules along with their activation time
  - facts which triggered the rule
- Saves trace data in log files for later use;
- Opens log files and displays their content (system activity).
Rules Set Profiling
Decision Tree Designer

✓ Project-based; a project may contain one or more trees;
✓ Structured designer which uses block structures;
✓ Imports definition files using the **Dictionary Manager** module;
✓ Generates tree code in C++/Java language;
✓ Integrated debugger and profiler;
✓ Shows every rule encapsulated in a tree diagram;
✓ Two working modes: specification mode and implementation mode
✓ **Scores** to prioritize node’s importance.
Decision Tree Designer

A Visual Tool for EXPERT SYSTEMS Construction
Decision Tree Profiling

- Helps you find time-consuming rules;
- Traces fired rules along with the node's values;
- Opens / saves trace files;
- Visualizes trace files.
Decision Tree Profiling

- Starting time: Wed Aug 23 18:11:47 2000, Rule=RepairDistributor (Score: 0)
  - Conditions
  - Engine.State (Status: 1)
    - Engine.State=STATE_ROTATE
  - Engine.Spark.State (Status: 1)
    - Engine.Spark.State=STATE_SPARK
  - Actions
    - AskMessageBox("Repair distributor wire")

- Starting time: Wed Aug 23 18:11:55 2000, Rule=Ok (Score: 0)
  - Conditions
  - Engine.State (Status: 0)
    - Engine.State=STATE_OK
  - Actions
    - AskMessageBox("Engine is OK")

- Starting time: Wed Aug 23 18:12:08 2000, Rule=Ok (Score: 0)

- Starting time: Wed Aug 23 18:12:13 2000, Rule=NoDiagnose (Score: 0)

- Starting time: Wed Aug 23 18:12:18 2000, Rule=ChargeBattery (Score: 0)

- Starting time: Wed Aug 23 18:12:22 2000, Rule=ChargeBattery (Score: 0)

- Starting time: Wed Aug 23 18:12:30 2000, Rule=RepairDistributor (Score: 0)
  - Conditions
  - Engine.State (Status: 1)
    - Engine.State=STATE_ROTATE
Decision Table Designer

✓ Project-based; a project may contain one or more tables;
✓ Imports definition files using the Dictionary Manager module;
✓ Generates C++/Java code for the decision table;
✓ Integrated debugger;
✓ **Scores** to prioritize the rules
Decision Table Debugger
Verifies your decision table for correctness and redundancies and handles duplicated and/or ambiguous rules;

Verifies the completeness of the table and may generate the missing rules for you
### Decision Table Analyzer

<table>
<thead>
<tr>
<th>No</th>
<th>Score</th>
<th>PreAction</th>
<th>Condition</th>
<th>OK</th>
<th>RepairIgnition</th>
<th>ChangeBattery</th>
<th>RepairDistributor</th>
<th>NoDiagnose</th>
<th>RepairBattery</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td></td>
<td>SCORES</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>Engine.St</td>
<td>==STATE_OK</td>
<td>==STATE_R...</td>
<td>==STATE_NO...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>Engine.Spark.St</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>Engine.Battery.St</td>
<td>==STATE_NO...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
<td>==STATE_N...</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>ACTIONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Duplicated Rules**

- Rule 3[ChangeBattery] ↔ Rule 4[RepairDistributor]
- Rule 3[ChangeBattery] ↔ Rule 6[RepairBattery]
Dictionary Manager

- **Dictionary** contains the following:
  - user-defined functions (from C/C++ header or Java files),
  - user-defined types (classes and structures) (from C/C++ header or Java files),
  - user-defined constants (from C/C++ header files),
  - external variables (from C/C++ header files),
  - internal variables;

- **Dictionary Builder** – imports C/C++ header and Java files

- Together with Code Generator ensures integration of expert systems into “host projects”
Dictionary Manager

A Visual Tool for EXPERT SYSTEMS Construction

Expert System Creator:

University of the West, Timisoara
Expert System Creator: A Visual Tool for EXPERT SYSTEMS Construction

**Code Generator Module**
- Help users to embed constructed expert system into “host projects”
- Exports decision tables/trees to C++/Java languages
- Exports rule-based systems to CLIPS/JESS shells
- Communicates with Dictionary Module
- Used by Debuggers and Profilers
- Customization

![Image of Code Generator Module with settings and options]

Select trees and then use the Generate button to generate code.
Database Integration

- Decision Table/Decision Tree – use DBMiddleware API to access any relational database management system

- Rules Set – use RulesSet DataBase (Decision Frame DataBase) subsystem

RulesSet DataBase subsystem is composed of:

- DBRepository – collects and stores all the information about the data managed by DBMSs
- DBWizard – GUI tool allowing the creation and maintenance of the DBRepository
- DBEngine – fetches/uploads the data from/to the database
- DBMiddleware – assures database autonomy, being responsible for any database specific task
Applications

- Classification Systems
- Diagnose Systems: Medical diagnostic decision support
- Fraud Detection Systems
- Decision Support Systems:
  - Accurate data
  - User friendly interface
  - Reliable knowledge base
Future Research and Extensions

- Integration with Fuzzy logic engines
- Growing decision trees from large sets of data
- Others…