EXPERT SYSTEM FOR TREE SELECTION IN URBAN HORTICULTURE

By

RANJIT RAVI IYER

A THESIS PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2005
Copyright 2005

by

Ranjit Ravi Iyer
This work is dedicated to my advisor, my parents and my late grandfather.
ACKNOWLEDGMENTS

I consider earning my master’s degree as reaching one of the significant milestones in my life. The last three years of my academic career have been extremely stimulating and have earned me much academic maturity. I consider my thesis work to be a very big contributing factor towards the same. I would like to profusely thank my advisor, Dr. Howard Beck, collaborating with whom has been a great privilege. His ideas and advice have been the pillars that have helped me achieve my research objectives. His immense passion for technology, and intellectual insights were great motivating factors in my work. I would also like to extend my appreciation to committee members, Dr. Edward Gilman and Dr. Gregory Kiker for their encouragement and guidance to complete this work. Finally, I take this opportunity to thank my parents, and my late grandfather, who have been a guiding force in all my endeavors.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
</tbody>
</table>

## CHAPTER

1. **INTRODUCTION** ........................................................................................................ 1

2. **DECISION SUPPORT SYSTEMS AND EXPERT SYSTEMS** ................................. 3
   - 2.1 Types of Decision Support Systems ............................................................... 4
   - 2.2 Expert Systems ....................................................................................................... 5
   - 2.3 Components of an Expert System ......................................................................... 7
   - 2.4 Building an Expert System ............................................................................... 9
   - 2.5 Applications of Expert System ........................................................................... 10
   - 2.6 The Southern Trees project .............................................................................. 11

3. **LITERATURE REVIEW** ........................................................................................... 14

4. **THE TREES WEB SITE** ....................................................................................... 18
   - 4.1 Features of the Trees Web site ......................................................................... 18
   - 4.2 The Trees Database ............................................................................................ 18
   - 4.3 The Tree Selector .............................................................................................. 19
   - 4.4 The Trees Expert System .................................................................................. 24

5. **THE RULE EDITOR** .............................................................................................. 26
   - 5.1 Features of the Rule Editor .............................................................................. 26
   - 5.2 The Rule Object Model (ROM) ......................................................................... 27
   - 5.3 Components of the Rule Editor .......................................................................... 30
   - 5.4 Building a Knowledge Base using the Rule Editor ............................................. 32

6. **THE TREES EXPERT SYSTEM** .......................................................................... 38

7. **SUMMARY AND FUTURE WORK** ........................................................................ 42
APPENDIX IMPLEMENTATION TOOLS ................................................................. 44
  Programming Language .................................................................................. 44
  Object-Oriented Database Management Systems ............................................ 44

LIST OF REFERENCES ....................................................................................... 47

BIOGRAPHICAL SKETCH ................................................................................. 49
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-2.</td>
<td>Southern Trees Startup Screen</td>
<td>12</td>
</tr>
<tr>
<td>4-1:</td>
<td>Trees Listed by Common Name</td>
<td>19</td>
</tr>
<tr>
<td>4-2:</td>
<td>Details of the “Baumannii Horsechestnut” Tree</td>
<td>20</td>
</tr>
<tr>
<td>4-3:</td>
<td>The Tree Selector Launched from the Trees Web Site</td>
<td>21</td>
</tr>
<tr>
<td>4-4:</td>
<td>The &quot;Flower&quot; Tab Allows Specifying Attributes of Flowers</td>
<td>21</td>
</tr>
<tr>
<td>4-5:</td>
<td>The &quot;Size/Shape&quot; Tab Allows Specifying Size and Shape Attributes of Trees</td>
<td>22</td>
</tr>
<tr>
<td>4-6:</td>
<td>Trees Recommended by the Tree Selector</td>
<td>22</td>
</tr>
<tr>
<td>4-7:</td>
<td>Attribute Details for a Recommended Tree</td>
<td>23</td>
</tr>
<tr>
<td>4-8:</td>
<td>Architectural Diagram of the Tree Selector</td>
<td>23</td>
</tr>
<tr>
<td>5-1:</td>
<td>The Rule Editor</td>
<td>28</td>
</tr>
<tr>
<td>5-2:</td>
<td>The Rule Object Model</td>
<td>29</td>
</tr>
<tr>
<td>5-3:</td>
<td>Architecture of the Rule Editor</td>
<td>31</td>
</tr>
<tr>
<td>5-4:</td>
<td>The Fact Editor</td>
<td>32</td>
</tr>
<tr>
<td>5-5:</td>
<td>The Message Editor</td>
<td>33</td>
</tr>
<tr>
<td>5-6:</td>
<td>The Report Editor</td>
<td>34</td>
</tr>
<tr>
<td>5-7:</td>
<td>The Question Editor</td>
<td>34</td>
</tr>
<tr>
<td>5-8:</td>
<td>The Fact Editor Displaying the Facts used by both Example Rules</td>
<td>35</td>
</tr>
<tr>
<td>5-9:</td>
<td>A Question about Water Drainage</td>
<td>36</td>
</tr>
<tr>
<td>5-10:</td>
<td>Loose-clay Rule</td>
<td>36</td>
</tr>
</tbody>
</table>
Abstract of Thesis Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Master of Science

EXPERT SYSTEM FOR TREE SELECTION IN URBAN HORTICULTURE

By

Ranjit Ravi Iyer

December 2005

Chair: Howard Beck
Major Department: Agricultural and Biological Engineering

Existing decision support tools that are part of the desktop based Trees project, needed to be ported to the Trees Web Site, thus providing a set of Web-based decision support tools for the tree selection problem. An expert system generator tool, a tree selector application and an expert system application have been developed as part of this research project. These applications form part of a larger suite of online decision support tools, hosted by the Trees Web Site. The expert system generator tool, called the Rule Editor, can be used by urban horticulturists and agriculturists for developing and maintaining expert systems. It provides for the creation and maintenance of a knowledge base that includes facts, messages, questions, reports and rules which together describe an expert system for a particular decision problem. The Tree Selector and the Trees Expert System are applications that give advice to urban tree managers on selecting the best tree for planting at a planting site.
CHAPTER 1
INTRODUCTION

An expert system is a kind of decision support system or a software application, which lends advice to its users, helping them make better decisions. Such software systems, work through domain specific rules in reaching a conclusion to a decision problem. Most expert systems are developed with a close synergy between the domain experts, who provide the domain specific expertise, and the knowledge engineer who builds the application.

The objectives of this research project are twofold – to develop an online expert system generator tool to be used by urban horticulturists and agriculturists (domain experts) for rapidly building and maintaining the knowledge base of an expert system application, and to develop two Web-based decision support applications; the Trees Expert System and the Tree Selector, to advise urban tree managers on tree selection. The Tree Selector searches an extensive database of trees suitable for planting in an urban environment according to attributes such as the environmental conditions, size/shape of trees, fruit, flower and foliage characteristics, specified by the expert manager (the user of the application). The Trees Expert System is designed for advising novice tree managers on choosing a suitable tree for planting. Both applications lend advice to tree managers on tree selection in an urban environment. They form part of a larger effort undertaken for building a suite of Web-based decision support tools that could be used by urban horticulturists, and agriculturists to build and maintain expert systems, and by urban and suburban tree managers and utility companies to search for, and get advice on planting
the best tree at a particular planting site. The motivations behind the expert systems research are to:

- Create an expert system development environment that could be used directly by experts in an agricultural discipline (in this case, horticulture) that didn’t require extensive knowledge of computer science or expert system technology.
- Create an online (Web-based) tool that is accessible through a Web browser.
- Address and resolve some of the issues associated with traditional expert system applications which include maintainability, scalability, persistent and durable storage of knowledge, eliminating inaccurate and inefficient means of knowledge sharing between the domain expert and the software engineer.

The main contributions of this research project have been the successful redesigning and porting of the Tree Selector and the Trees Expert System, which are the two Web-based decision support tools for advising tree managers on tree selection. This project also includes the development of the Rule Editor, as a high-level, Web-based expert system generator tool that can be used by urban horticulturists and agriculturists in developing, maintaining and upgrading the knowledge base of expert systems.

The rest of the thesis is organized in the following way. Chapter 2 introduces concepts of Decision Support Systems and Expert Systems. Chapter 3 provides an overview of related work in the field. Chapter 4 describes features of the Trees Web site, including the Trees database, the Tree Selector and additionally provides motivation for the need of an expert system for advising novice managers on trees. Chapter 5 provides a detailed description of the Rule Editor. Chapter 6 describes the architecture of the Trees Expert System and Chapter 7 concludes and discusses future work in this research area.
CHAPTER 2
DECISION SUPPORT SYSTEMS AND EXPERT SYSTEMS

Decision support systems (DSS) are a class of computerized information systems that support decision-making activities. DSS perform their analyses on data and knowledge, stored in a data repository and a knowledge base, to arrive at a decision or conclusion [1].

**DSS components** A typical decision support system consists of the following components or subsystems: Data management subsystem, model management subsystem, knowledge-based subsystem, user interface subsystem, and the user of the DSS.

A Database management system, captures/extracts data for inclusion in a DSS database, updates (adds, deletes, edits, changes) data records and files, interrelates data from different sources, retrieves data from the database for queries and reports, provides comprehensive data security (protection from unauthorized access and recovery capabilities), performs complex data manipulation tasks based on queries, tracks data use within the DSS and manages data through a data dictionary. The model management subsystem is the analog of the database management subsystem and it comprises a model base (for storing models), a modeling language, and a model execution framework. The knowledge management subsystem stores knowledge in logical form (using a knowledge representation language) and provides facilities for solving complex unstructured and semi-structured problems. The expertise is provided by an expert system or other intelligent systems that form part of the DSS. Finally, a graphical user interface provides for all the communication between a user and the decision support system.
2.1 Types of Decision Support Systems

**Data-driven DSS** is a type of DSS that emphasizes access to and manipulation of a time-series of data [2]. File systems accessed by query and retrieval tools provide the most elementary level of functionality. Data warehouse systems are a kind of data driven DSS that allow the manipulation of data by computerized tools tailored to a specific task. Data-driven DSS with On-line Analytical Processing (OLAP) provides the highest level of functionality and decision support that is linked to analysis of large collections of historical data.

**Model-driven DSS** emphasize access to and manipulation of a model, for example, statistical, financial, optimization and/or simulation models. Simple statistical and analytical tools provide the most elementary level of functionality. Some OLAP systems that allow complex analysis of data may be classified as hybrid DSS systems providing both modeling and data retrieval and data summarization functionality. In general, model-driven DSS use complex financial, simulation, optimization or multi-criteria models to provide decision support. Model-driven DSS use data and parameters provided by decision makers but they are not usually data intensive; that is very large data bases are usually not need for model-driven DSS.

**GIS-based DSS** deliver decision support information, using GIS (geographic information system) tools. General-purpose GIS tools are programs such as ARC/INFO, MAPInfo and ArcView [3]. that have extensive functionality for analysis of geographic and spatial data. Internet development tools have matured as well, making it possible to develop fairly complex GIS-based programs that users can use through the World Wide Web.
Knowledge-based DSS can suggest or recommend actions to decision makers based on data structures that mimic the logic of decision making. These DSS are thought of as having specialized problem-solving expertise. The "expertise" consists of knowledge about a particular domain, understanding of problems within that domain, and "skill" at solving some of these problems. A related concept is Data Mining. It refers to a class of analytical applications that search for hidden patterns in a database or sifting through large amounts of data to produce data content relationships. Other examples include expert systems, and other artificial intelligence applications. Knowledge based DSS are also known as intelligent DSS.

2.2 Expert Systems

Knowledge-based decision support systems make use of knowledge or facts about the real world in arriving at decisions. Most knowledge-based DSS make use of an Expert system component as one of their subsystems. Expert systems are part of a general category of computer applications within the field of artificial intelligence. An expert system’s main objective is to provide expert advice in specialized situations. An expert system could be defined as an interactive computer-based decision tool that uses both facts and heuristics (rules of thumb which have been proposed to explain how people make decisions, come to judgments and solve problems, typically when facing complex problems or incomplete information) to solve difficult decision problems based on knowledge acquired from an expert [4].

Traditional decision support systems such as spreadsheets are very mechanistic in the way they solve problems. They operate under mathematical and Boolean operators in their execution and arrive at one and only one static solution for a given set of data. Calculation intensive applications with very exacting requirements are better handled by
traditional decision support tools or conventional programming. The best application candidates for expert systems are those dealing with expert heuristics for solving problems. Expert Systems (ES) differ from conventional computer programs too. Some of the typical characteristics of conventional computer programs are as follows:

- Data and information are processed as a direct conceptual resource.
- Algorithms, mathematical formulas and sequential procedures use data to solve problems to achieve a solution.
- The main objective is to create an efficient solution by designing an efficient algorithm.
- The control mechanism of the system and the knowledge of the system are intertwined as part of the instruction sequence.
- The subject knowledge or the domain rules form part of the program instructions. Hence, to alter the domain rules or knowledge, a programmer needs to edit the program to make the necessary changes.

In contrast, the characteristics of an ES are as follows:

- The main conceptual source of ES is knowledge.
- ES perform symbolic processing of knowledge as compared to algorithmic procedures that operate on data. Knowledge is represented by symbols such as letters, words or numbers that represent objects. Domain expertise is stored as rules, which also forms part of the knowledge base.
- The objective of an ES is to create an efficient knowledge representation to support the reasoning, or inference mechanism.
- Maintenance and scalability are easier, because of the separation of the knowledge base and the control structure.
- Expert systems are capable of working with inexact data. They allow the user to assign probabilities, certainty factors or confidence levels to any or all input data. This feature closely represents how most problems are handled in the real world. An expert system can take all the relevant factors into account and make a recommendation based on the best possible solution rather than the only exact match.
- The ability to explain their reasoning processes is another key feature of expert systems. Such explanation facilities provide the user with a means of understanding the system behavior.

Expert systems are typically written in special programming languages like LISP, CLIPS, PROLOG, etc. These languages are characterized by a syntactic structure that more conveniently supports the data structures used in expert systems.

### 2.3 Components of an Expert System

The fundamental building blocks of an Expert system are the knowledge base and the inference engine [5].

The knowledge base contains both factual and heuristic knowledge. Factual knowledge is that knowledge of the task domain that is widely shared, typically found in textbooks or journals and commonly agreed upon by those knowledgeable in the particular field. Heuristic knowledge is the less rigorous, more experiential, more judgmental knowledge. The knowledge base is the nucleus of the expert system structure. A knowledge base is created by knowledge engineers, who translate the knowledge of real human experts into knowledge or data, in a computer understandable form. This is known as knowledge representation. It implies a systematic means of encoding what an expert knows about a knowledge domain, in an appropriate medium, and procedures for automatically manipulating these data structures. Knowledge can be represented as predicate calculus or formal logic, production rules, semantic networks (knowledge is organized through nodes in a graph rather and relationships between the facts are represented by the edges between the nodes in the graph). Most ES represent knowledge as rules and therefore the knowledge base is often referred to as the rule base. Rules are if-then expressions. The “if” part of the rule, lists a set of conditions in some logical combination. If the “if” part of the rule is satisfied, consequently, the “then” part is
concluded and actions such as asserting or retracting facts or truth statements, are taken accordingly.

The mechanism that performs the search and reasoning in rule-based systems is called the inference engine. Based on the facts present in the knowledge base, the inference engine selects rules and executes them. Three techniques are identified when inferring facts or conclusions from the knowledge base; forward chaining, backward chaining and hybrid chaining. Forward chaining involves checking the condition part of a rule to determine whether it is true or false. If the condition is true, then the action part of the rule is also true. This procedure continues until a solution is found or a dead end is reached. Forward chaining is also commonly referred to as data-driven reasoning.

Consider the following set of rules to demonstrate a forward chaining evaluation technique:

Rule 1: IF A and C THEN F
Rule 2: IF A and E THEN G
Rule 3: IF B THEN E
Rule 4: IF G THEN D

Suppose it needs to be proved that D is true, given A and B are true. We start with Rule 1 and go on down till a rule that "fires" is found. In this case, Rule 3 is the only one that fires in the first iteration. At the end of the first iteration, it can be concluded that A, B and E are true. This information is used in the second iteration. This time Rule 2 fires adding the information that G is true. This extra information causes Rule 4 to fire, proving that D is true. Backward chaining is the reverse of forward chaining. It is used to backtrack from a goal to the paths that lead to the goal. Backward chaining is very good
when all the outcomes are known and the number of possible outcomes is not large. In this case, a goal is specified and the expert system tries to determine what conditions are needed to arrive at the specified goal. It is a goal-driven backward chaining of rules to try to prove a hypothesis made by the system. Consider the aforementioned set of rules to demonstrate a backward chaining evaluation technique. Here, the strategy to prove that D is true would be as follows. First, we find a rule that proves D. Rule 4 does so. This provides a sub-goal to prove that G is true. Now Rule 2 comes into play, and as it is already known that A is true, the new sub-goal is to show that E is true. Here, Rule 3 provides the next sub-goal of proving that B is true. But the fact that B is true is one of the given assertions. Therefore, E is true, which implies that G is true, which in turn implies that D is true. A combination of forward chaining and backward chaining is called hybrid chaining. Hybrid chaining may be used to model strategies used by the human experts and to employ complex inferences, when a large problem domain is involved.

2.4 Building an Expert System

Knowledge engineers work closely with the domain experts, who provide the knowledge and rules for solving problems in a specific problem domain. The typical steps in the building process, involve, defining a problem domain and problem specification, knowledge acquisition by the knowledge engineer from the domain expert, developing and deploying the expert system.

Expert systems can be built using a piece of development software known as a "tool" or a "shell." Though different styles and methods of knowledge engineering exist, the basic approach is the same: a knowledge engineer interviews and observes a human expert or a group of experts and learns what the experts know, and how they reason with
their knowledge. The knowledge is then translated into rules and stored in the knowledge base to be used by an inference engine that performs the reasoning on the knowledge. Finally, the knowledge engineer also determines how to integrate the use of uncertain knowledge in the reasoning process, and what kinds of explanation would be useful to the end user after a final decision is reached.

**Figure 2-1. Components and the Development Process of an Expert System**

### 2.5 Applications of Expert System

The spectrum of applications of expert systems technology to industrial and commercial problems is so wide as to defy easy characterization. The applications find their way into most areas of knowledge work [6]. They tend to cluster into several major classes. Some of them are listed below:

- **Diagnosis and troubleshooting of devices and systems:** This class comprises systems that deduce faults and suggest corrective actions for a malfunctioning device or process. Medical diagnosis was one of the first knowledge areas to which ES technology was applied. There are probably more diagnostic applications of ES than
any other type. The diagnostic problem can be stated in the abstract as: “given the evidence presenting itself, what is the underlying problem/reason/cause?”

- **Planning and Scheduling:** Systems that fall into this class analyze a set of one or more potentially complex and interacting goals in order to determine a set of actions to achieve those goals, and/or provide a detailed temporal ordering of those actions, taking into account personnel, material, and other constraints. Examples involve airline scheduling of flights, personnel, and gates; manufacturing job-shop scheduling; and manufacturing process planning.

- **Financial Decision Making:** Expert system programs can be used to assist bankers in determining whether to make loans to businesses and individuals. Insurance companies could use expert systems to assess the risk presented by the customer and to determine a price for the insurance. A typical application in the financial markets is in foreign exchange trading.

- **Process Monitoring and Control:** Systems falling in this class analyze real-time data from physical devices with the goal of noticing anomalies, predicting trends, and controlling both optimality and failure correction. Examples of real-time systems that actively monitor processes can be found in the steel making and oil refining industries.

- **Design and Manufacturing:** These systems assist in the design of physical devices and processes, ranging from high-level conceptual design of abstract entities all the way to factory floor configuration of manufacturing processes.

### 2.6 The Southern Trees project

Expert systems find wide applications for solving decision problems in the agriculture and horticulture domain [7, 8, 9]. Deciding on the best tree to plant at a particular planting site, could be one such decision problem that tree managers often encounter.

Previous work by Beck et al. [10] resulted in the development of the Southern Trees project which is a collection of several decision support system applications designed for solving the aforementioned decision problem of tree selection. Some of the features of this software are as follows:
• The Southern Trees software contains an extensive database of trees, and a rich
description of the attributes of each tree. All trees were searchable by their scientific
name and common name.

• It was developed under the Windows environment and distributed on a CD-ROM.

• It contains an expert system application that offers novice tree managers,
recommendations on tree selection. The expert system works by asking questions
about the environmental and special conditions, the attributes of fruits, flowers, uses
of the trees, etc and makes its recommendations based on the answers provided by
users. Help messages are provided for most of the questions asked, and reports are
generated in the end along with the recommended trees, to explain the reasoning
behind arriving at the recommendation. Hence the knowledge in the Southern Trees
Expert System could be categorized as facts, messages, reports, questions, and the
rules themselves.

• The knowledge in the Southern Trees Expert System is represented as rules written in
the CLIPS programming language [11]; a functional language that is a dialect of the
LISP programming language, used historically for many artificial intelligence
application development.

• The expert system application uses the CLIPS expert system shell [11]. to maintain
the knowledge base and perform reasoning on that knowledge.

Figure 2-2. Southern Trees Startup Screen
Although the Southern Trees software has been distributed and used among trees managers there still are some problems and constraints. Some of these problems are described as follows:

- The database of trees may grow over time as new urban trees are discovered. It may be difficult to update the database built into the Southern Trees CD-ROM to make it available to managers. Any such update would warrant a new distribution of CD-ROM. Web-based delivery more recently replaced CD-ROM delivery in situations where the application changes frequently because the database is maintained and accessed from a central location.

- Since the production rules of the Southern Trees Expert System were written in the CLIPS language and stored as “.clp” files on the local file system any modifications to the problem solving strategy or domain rules would require those to be communicated by the domain expert to the knowledge engineer, who would then directly modify the code to incorporate those changes. This is certainly not a good software engineering practice, as modifying code could have potential implications in some other areas of code section. More importantly, it is difficult for the expert to read and understand rules in CLIPS syntax if the expert wishes to perform those changes without the intervention of the knowledge engineer. A clean separation between the code and the domain rules should be achieved. At the same time, it would be useful to provide an online tool for the domain experts, to maintain, upgrade and even create the knowledge base of an expert system themselves, reducing the need for the interdependence of the domain expert and the knowledge engineer.

- Insufficient backward compatibility support between the older and newer versions of the Windows operating system, has led to installation problems of the CD-ROM software on newer machines.

- With the widespread availability and use of the Internet, the prospect of providing an online expert system application looks more attractive. This will for one, achieve platform independence. Also, making the tools available over the Internet would provide easier and faster access and cater to a wider audience.
The primary objectives of the research project are to 1) develop a Web-based expert system application for use by tree managers for solving the tree selection decision problem, and 2) to develop an online tool for efficient expert system generation, to be used by urban horticulturists and agriculturists (domain experts). The following proprietary software systems were surveyed as possible expert system generator tools.

**Business rule studio (BRS)** [12]. BRS is a software tool used by corporate managers, enterprise experts and software developers who wish to define and manage business rules as a separate unit owned by management and corporate experts, which are then deployed as part of the application software in co-operation with the software developers. The rules are stated in a spread-sheet like format. The objects or entities are mainly customers, orders and product details which render the application specific to a business domain. BRS is not an expert system. It comprises a Rule Author for capturing business policies in the form of objects and rules, a rule engine for executing these policies and a rule repository to manage and store the rules. The application is tied to the Visual Basic programming language to be used as a client application.

**Visual rule studio (VRS)** [13]. The Visual Rule Studio was also considered as a potential expert system generator. This tool requires the rules to be coded using VRS’ Production Rule Language (PRL), which have to be learned by domain experts. The rules are compiled into either, a standard exe, an ActiveX control or an ActiveX document. The compiled rule set is then referenced from within a client application such as Visual
Basic. VRS is a desktop, rule development environment for Windows NT, Windows 98 and 95.

**ILOG JRules** [14]. ILOG JRules is a general purpose expert-system generator that combines rule-based techniques and object oriented programming to help add rule-based modules to business applications. The product primarily targets business applications, which have business logic embedded within production code and aims to achieve a separation of the two. JRules provides a proprietary rule-engine developed in the C++ language. JRules uses the Rete algorithm [15] to perform pattern matching in the rules. The JRules engine can be used in all types of Java applications including applets running inside a browser. JRules also supports representation of rules in an XML [16] format. The application renders itself as a solution mainly in the domain of banking, Customer Relationship Management, insurance, telecommunication, etc.

**JESS expert system shell** [17]. As mentioned earlier, expert systems can be built using a piece of software, known as an expert system shell. One important feature of expert systems is the way they separate domain specific knowledge from more general purpose reasoning and representation techniques. The expert system shell encompasses the latter. The shell will typically provide the inference engine (and knowledge representation scheme), a user interface, an explanation system and sometimes a knowledge base editor. The Trees Expert System developed as part of this research project uses the JESS expert system shell. JESS was written entirely in the Java language. It supports the development of rule-based expert systems which can be tightly coupled to code written in the powerful, portable Java language. Using JESS can provide Java
applets and applications, the ability to "reason". JESS is typically used as a rule engine that efficiently applies a set of if-then rules to a set of data or facts (the knowledge base).

JESS supports both, forward chaining and backward chaining methods of rule execution. It uses the Rete algorithm for evaluating the rules. An inference engine that determines all applicable rules before applying them would intuitively perform linearly with the number of rules. The more rules that are added, the slower the inference engine would perform. The Rete algorithm was developed to provide a production system whose performance varied less than linearly. More precisely, the Rete algorithm reduces the number of iterations in a rule loop when performing pattern matching. Thus, given a significant number of rules, the performance of the Rete algorithm is unaffected when more rules are added. JESS tries to determine whether a fact in the knowledge base, matching the pattern of a particular rule’s condition part, exists. If it is found, the rule is placed on an “agenda” to be fired or executed. As the rule executes, some facts in the knowledge base are altered in a way, which leads other rules to be placed on the “agenda” to be fired. The contrary is also true. A fired rule could cause some other rules, presently on the “agenda” to be retracted.

JESS can be easily embedded within a Java application. Each individual jess.Rete object (one of the classes that JESS comprises) is an independent reasoning engine. A single program may include several independent engines. JESS can also be used in a multi-threaded environment. The jess.Rete class internally synchronizes itself using several synchronization locks. The rules that are loaded into JESS from within a Java application are stored in a file with a “.clp” extension. To guard the correctness of the
rules, the Rule Editor supports generation of JESS code, after the rules are constructed by the domain expert using the editor.

The aforementioned commercially available expert system generator packages were found unsuitable for meeting the research objectives, due to the following reasons.

- None of the assessed tools provide support for generating expert systems for the trees-domain and mainly focus on business applications.

- This research project aims at making the development tools, freely available over the internet. All of the surveyed software packages are proprietary and hence only available at a cost.

- JESS is used as the expert system shell for the Trees Expert System, because of the advantages that it offers, as described earlier in this chapter. The Southern trees expert system uses the CLIPS inference engine, and JESS inherits most of the formers’ features. Currently, there exists no tool for generating expert systems that work with JESS as the inference engine. All of the surveyed applications use proprietary rule engines.

- Most of the surveyed software systems represent domain rules in special languages that have to be learned by the domain expert. The Rule Editor does not make any such assumptions and aims to provide an abstract and simple view of the rules to the domain experts so there is a very short learning curve incurred, before using the tool.

- The Expert system generator (Rule Editor) aims to support generation of rules in RuleML [18], which is an XML based file format that is fast becoming the de-facto standard for exchanging rules over the internet. The surveyed tools either do not support representation of rules in XML, or use non-standard XML representation of the rules.
CHAPTER 4
THE TREES WEB SITE

The Trees Web site has been designed to provide all the decision support tools from the Trees project over the internet to urban horticulturists, agriculturists and urban trees managers. The Web site is located at http://orb.at.ufl.edu/TREES. The Tree Selector and the Trees Expert System, developed as part of this project, are also hosted on the Trees Web site.

4.1 Features of the Trees Web site

The Trees Web site offers a suite of online tools that are grouped according to the users of the tools and their potential applications. The categories are listed as follows:

- **Tree Indexes:** The Trees Web site offers extensive database of trees and rich description of each tree, listed and searchable by scientific names and common names.

- **Tools for Novices:** The two tools provided for novice users are the Tree Identification and the Tree Expert System that helps novice managers make decisions on tree selection and identification.

- **Tools for Experts:** Site Analysis and Tree Selector, are the two tools offered by the Web site for expert managers in helping them choose the right trees.

- **Related Sites:** The Trees Web site also points to external Web site that are sources of information about the best practices for growing trees in an urban environment and growing quality trees in a nursery.

4.2 The Trees Database

The Trees Web site connects to an extensive database of trees that are searchable by common names and scientific names. An example listing of all trees by common name
is shown in the following Figure 4-1. Similar searches could be conducted for trees, based on their scientific names.

Figure 4-1: Trees Listed by Common Name

4.3 The Tree Selector

The Trees Web Site makes a very large database of trees available to managers. It also offers rich details about every tree in the database. Although there is a vast amount of tree information available, it may still be a difficult problem for a tree manager to decide on the tree that is best suited for planting at a particular site. The Tree Selector is a tool designed for expert tree managers to help them make a decision on choosing the right trees. The tool allows expert managers to specify attributes of fruits, foliage, flowers, size/shape, trunk/branches, and environmental conditions such as hardiness zone as
search criteria possess, marks them as the search criteria. The Tree Selector searches for and presents trees from select states, based on those search criteria. An instance of the Tree Selector as launched from the Trees Web site is shown in Figure 4-3. Clicking on a particular tree displays a wealth of details about that tree, such as its uses, native origin, hardiness zone, life span, and much more. Figure 4-2 shows details about the “Baumannii Horsechestnut” tree.

Figure 4-2: Details of the “Baumannii Horsechestnut” Tree

Search attributes are organized under several categories, which are represented by the different tabs on the Tree Selector. These can be observed in the screenshots shown in Figure 4-4 and Figure 4.5. Figure 4-6 shows a snapshot of the recommendations made by the Tree Selector after performing a query based on the search attributes as indicated by
the user of the application. Figure 4-7 displays details of one of the trees chosen from the list of recommended trees.

Figure 4-3. The Tree Selector Launched from the Trees Web Site

Figure 4-4. The "Flower" Tab Allows Specifying Attributes of Flowers
Figure 4-5. The "Size/Shape" Tab Allows Specifying Size and Shape Attributes of Trees

Figure 4-6. Trees Recommended by the Tree Selector
Figure 4-7. Attribute Details for a Recommended Tree

![Diagram](image)

DATABASE OF TREES

JAVA SERVLET

INTERNET

FLASH CLIENT

Figure 4-8. Architectural Diagram of the Tree Selector
An architectural diagram of the Tree Selector application is shown in Figure 4-8. The user interface of the Tree Selector has been developed using Macromedia Flash [19]. The application works by gathering user search criteria and building a query string to be sent to a Servlet program [20]. (a Java program running on the server side, in a client-server application framework). The Servlet program performs a search on the database of trees, with the attributes specified by the user of the application and returns the recommended trees, back as an XML string. The XML string contains the common name and scientific name of all the recommended trees. The XML string is parsed by the Flash application and displayed in a form as shown in Figure 4-6. Details of each resultant tree can be viewed by clicking on a tree present in the results list, as shown in Figure 4-5.

4.4 The Trees Expert System

The underlying assumption in using the Tree Selector is that the manager knows the attributes of trees, and environmental conditions at the planting area. Hence the Tree Selector may not be suitable for novice managers, who may not possess knowledge about the details of the attributes of a tree and so forth. The novice does not know what tree attributes are important for particular planting sites. For example they may not know how to evaluate the soil drainage at the planting site and they may not know that it is important to choose trees that tolerate wet soil if the soil is wet. This necessitates development of an expert system to ask the user very specific questions about the planting site. The question and answer session is designed to guide the user through a thorough site analysis and to assert tree attributes on a facts list that ensure that only trees suited for the site appear on the recommended trees list. One such expert system was developed as part of the Trees project in the early 90’s. Some of the problems associated with the same, were discussed towards the end of Chapter 2. Towards solving those
problems and providing an easier experience for experts to build and maintain their own expert systems work was undertaken to build the Rule Editor and the Trees Expert System; the former to be used by domain experts and the latter to be made available to tree managers by hosting it on the Trees Web site.
CHAPTER 5
THE RULE EDITOR

The need for an expert system for advising novice users on tree selection was underscored in the previous chapter. The user of the expert system is anyone interested in planting a tree in an urban environment, and that could be a city arborist, utility company, urban forester, landscape architect, home owner, etc. The term “tree manager”, may apply to any of these categories of users and will be used throughout this thesis, unless stated otherwise. This thesis work aims at providing a Web-based expert system capable of giving advice, and which can also be created, maintained and gracefully upgraded by domain experts themselves. The latter aim is motivated by the need to resolve problems in the existing desktop based expert system developed as part of the Southern Trees project, as discussed in chapter 3. This necessitates a two pronged approach towards achieving our objectives; developing an expert system generator to be used by horticulturists and agriculturists (domain experts) and a Web-based expert system application to be used by novice tree managers.

5.1 Features of the Rule Editor

The Rule Editor is an internet based expert system generator tool, used for rapidly creating expert systems and maintaining and scaling existing ones. The Rule Editor is available from the following location on the internet: http://orb.at.ufl.edu/LyraEditor/ruleeditor.html. An instance of the Rule Editor is shown in Figure 5-1. Some of the features of the Rule Editor are as follows:
1) All of the knowledge used by an expert system build is modeled and stored as objects in an object-oriented database, by the Rule Editor.

2) Authorized users can easily create and maintain a knowledge base of rules, facts, messages, reports and questions. Any subsequent changes made to the knowledge base are consolidated in a database. Thus new domain rules and strategies could be easily incorporated in existing expert systems.

3) The Rule Editor provides a facility for categorizing the rules of an expert system under logically separate modules. This makes the rules more easily manageable.

4) The Rule Editor is a template-based editor, in which items necessary to formulate rules could be picked and chosen other sub editors. This reduces the amount of typing of names by hand, which could potentially lead to misspellings and incorrect references to facts, messages, reports and questions, within a rule.

5) The Rule Editor supports automatic generation of JESS code that represents the domain rules in a form suitable for loading and executing in the JESS inference engine. Hiding the intricate details of the rules and providing for automatic code generation, offers a more easily understandable view of the rules of the expert system to the horticulturists.

6) The Rule Editor supports arithmetic, relational, Boolean and unary operators for constructing the “if” condition of a rule. It is fairly easy to extend this set and provide support for new operators if the need arises in the future.

7) The Rule Editor also attempts to help build expert systems in a relatively short period of time, since it avoids the knowledge-transfer phase between domain experts and knowledge engineers.

5.2 The Rule Object Model (ROM)

The Rule Editor maintains the knowledge of the expert system (facts, messages, reports, questions and rules) as objects or entities in an object-oriented database system.

An object-oriented design approach employed by the Rule Editor has several advantages.

Some of the concepts and benefits of object-oriented design are listed below:

- Objects - Packaging data and functionality together into units within a running computer program; objects are the basis of modularity and structure in an object-oriented computer program.

- Abstraction - The ability for a program to ignore some aspects of the information that it is manipulating, i.e. the ability to focus on the essential.
Encapsulation - Ensures that users of an object cannot change the internal state of the object in unexpected ways; only the objects’ own internal methods are allowed to access its state. Each object exposes an interface that specifies how other objects may interact with it. Other objects will not know and thus rely upon this object’s internal representation.

Polymorphism - Instead of subroutine calls, object-oriented languages can make message sends; the specific method which responds to a message-send depends on what specific object the message is sent to. This gains polymorphism, because a single variable in the program text can hold different kinds of objects as the program runs, and thus the same program text can invoke different methods at different times in the same execution.

Inheritance - Organizes and facilitates polymorphism and encapsulation by permitting objects to be defined and created that are specialized types of already-existing objects - these can share (and extend) their behavior without having to re-implement that behavior.

Figure 5-1. The Rule Editor

The Rule Object Model uses Unified Modeling language principles [21] to describe all the objects (knowledge), and their complex interrelationships. For example, a question
can be thought of as an object having a title, the question text and a collection of answers as its attributes. An answer could also be thought of as an object, with its attributes being, the answer text and a collection of actions to be taken for that answer. An action could again be any of the components of the knowledge base; facts, messages, etc which are also objects in the system. Thus it is evident that questions, answers and actions are tightly interrelated. There are many such relationships between all the objects of the system, as described by the layout of the ROM.

Figure 5-2. The Rule Object Model

An object-oriented database is used to store objects in a secondary storage medium (external disk) while preserving their state (the values of the attributes of an object at the time the object needs to be saved to a persistent storage) so they can be recreated with the exact same state at a later point in time. This is a critical reason for
using an object-oriented database for maintaining the knowledge base of an expert
system. From Figure 5-2 of the Rule Object Model for the Rule Editor, it is evident that
most of the objects (constituting the knowledge base) of the system are interrelated,
which is described by the references that each object maintains, to other objects. The
database maintains these references and thus the interrelationships of all the objects of the
knowledge base, even after they have been written out to a backing store. The Rule
Object Model provides a high level impression of the way objects look like, in an actively
running system or after all knowledge is saved to disk.

5.3 Components of the Rule Editor

The Rule Editor comprises four sub-editors - Fact Editor, Message Editor, Report
Editor and Question Editor, to create and maintain facts, messages, reports and questions,
respectively. The entire knowledge base of an expert system comprises the domain rules,
and the above mentioned four knowledge components.

**Fact editor** Facts are statements of truth about the real world. Facts are
represented as attribute-value pairs. The Rule Editor categorizes the facts into three types
– Boolean, Existence and Multi-valued. Boolean facts take on either a true or a false
value, existence facts are just attribute names with no value that indicate either the
presence or absence of a truth statement about the world in the knowledge base, and
multi-valued facts are those which could take on several possible values, each of which
could be of type string, integer or floating point. Facts appear as part of conditions
necessary for rules to fire. Facts are also asserted as actions, by rules and questions.
Figure 5-4 shows the Fact Editor preloaded with some of the facts used by the Trees
Expert System. The highlighted fact in the Fact Editor, “maxheight” describes the
maximum height (in feet) that a tree could grow up to.
**Report and message editor** An Expert System maintains information about the answers that users provide, in response to questions asked and substantiates its final conclusion by describing the logical sequence of intermediate steps that led to it with the help of reports. Help messages are also provided to assist users for answering questions. The Rule Editor supports the Report Editor and the Message Editor to create new reports and messages. Examples of a report and a message from the database of the desktop Trees project are shown in Figure 5-5 and Figure 5-6.

**Question editor** The Question Editor is used to build all the questions that could potentially be asked by the expert system to its users. The Question Editor allows for creation of questions, providing help messages for a question (by picking help messages from the Message Editor), creation of all possible answers for a particular question, and the actions that need to be taken in response to a particular answer. The action consists of either, asserting or retracting facts, asserting reports or messages or asking another
question. A question from the Trees expert system that asks for the planter for the zone at the planting site is described in Figure 5-7.

Figure 5-4. The Fact Editor

5.4 Building a Knowledge Base using the Rule Editor

This section cites two rules (written in the English language when conceived) that are used by the desktop Trees expert system and illustrates the process of identifying the knowledge components within those rules - these knowledge components are the building blocks for every rule in the expert system. It further explains the procedure for entering
and storing those knowledge components. Rules are then created using the templates provided by the Rule Editor and by using the previously created knowledge components.

The two rules that are considered in this example are the following:

1) If the user indicated that the soil at the planting site is of type clay and the density of the soil is loose, then ask a question to confirm if the user really means that the soil is loose. Experts feel that many clay soils in urban areas are compacted.

2) If the user indicated that the soil density is loose and the soil types are either sandy or loamy, then ask user how fast does water drain through the soil.

Figure 5-5. The Message Editor

Both rules make use of facts about the soil types and the soil densities to evaluate their “if” conditions, and ask a question as part of their actions. The facts could be “soil-type” and “soil-density” as is evident from the English description of the rules, each being a multi-valued fact with values such as (loam, sandy, clay) and (loose, compacted) respectively. Having culled out the facts, it remains for the domain expert to formulate
the questions and store them in the question bank, so they could be pulled from the database and asked, when either rule “fires”.

Figure 5-6. The Report Editor

Figure 5-7. The Question Editor
After the facts and questions are stored in the Fact Editor and the Question Editor both the rules could be easily constructed using the templates provided by the Rule Editor and by picking facts and questions from their respective editors to add them as part of conditions and actions of the rules. Finally, since the Rule Editor supports automatic code generation, JESS code could be generated to represent the rules in a form that could be loaded into the inference engine during the course of the expert systems’ execution.

Figure 5-8 describes the two facts used in this example as stored in the Fact Editor. Figure 5-9 shows a question on water drainage along with the permissible answers and actions associated with each answer. Figures 5-10 and 5-11 display the two rules as constructed and stored in the rule base of the Rule Editor. The code snippet in Figure 5-12 shows the JESS code generated for Rule 1 and Rule 2.

Figure 5-8. The Fact Editor Displaying the Facts used by both Example Rules
Figure 5-9. A Question about Water Drainage

Figure 5-10. Loose-clay Rule
Figure 5-11. Loam-sand-or-loam Rule

(defrule loose-clay
 ?var1 <- (soil ?sl1)
 ?var2 <- (density ?dy2)
 =>
 (if (and (eq ?sl1 CL) (eq ?dy2 LOOSE))
 then
 (assert (question TREES loose clay))
 (halt))

(defrule loose-sand-or-loam
 ?var1 <- (density ?dy1)
 ?var2 <- (soil ?sl2)
 =>
 (if (and (eq ?dy1 LOOSE) (or (eq ?sl2 SA) (eq ?sl2 LO)))
 then
 (assert (question TREES water drainage))
 (halt)))

Figure 5-12. JESS Code Generated by the Rule Editor
CHAPTER 6
THE TREES EXPERT SYSTEM

The justification need for an expert system and in particular, the need for it to be available on the Internet was introduced in Chapter 2 and Chapter 4. The Trees Expert System, developed as part of this project is Web based expert system capable of providing advice on tree selection, to novice tree managers. A combination of the Rule Editor presented in the previous chapter and an application framework presented in this chapter, together lead to the development of Trees Expert System hosted on the Trees Web Site.

![Diagram of the Trees Expert System](image)

Figure 6-1. Architectural Diagram of the Trees Expert System

An architectural diagram of the Trees Expert System is shown in Figure 6-1. The Trees Expert System application consists of the following building blocks:
• Database: An object-oriented database, stores facts, questions, messages, reports and rules as knowledge of an expert system, constructed using the Rule Editor. This is also the repository for all the searchable trees used by the expert system.

• Expert System shell: An instance of the JESS expert system shell that maintains facts in its knowledge base, and executes rules thus providing the reasoning capabilities of the expert system.

• Client (Flash): The client application presents a user interface that asks questions and accepts answers. The client application also presents the final list of trees and reports as the recommendations of the Trees Expert System.

• Middleware (Java Servlet): The middleware application communicates with all the other components of the expert system, namely, the database, the expert system shell and the client application. Apache Tomcat [22]. has been used as the Web server, for this project.

An instance of the Trees Expert System is shown in Figure 6-2. The application works by asking questions, taking appropriate actions for an answer, executing rules and finally presenting its recommendations the user of the expert system. The working of the Trees Expert System is described by the following steps:

1) The client application sends a request to the Servlet program, asking for the first question to be asked, as soon as the expert system starts up.

2) The Servlet program retrieves a question from the database, and sends it across to the client application.

3) The client application presents the question to the user and accepts an answer, and sends it back to the servlet program. The Servlet program takes actions that correspond to the answer provided for the question asked. The actions include asserting facts, reports, messages, or questions into the knowledge base of the expert system shell. Occasionally, an action would load a set of rules into the expert system shell and have the inference engine execute them based on the fact list present in the knowledge base. The rules execute, and they too may assert facts, messages, reports and questions. If there are any remaining questions in the knowledge base, after the rule executes, the procedure follows from step 2.

4) An absence of newly asserted questions in the knowledge base after the rules execute, marks the end of the question-asking phase. The expert system formulates a database query by using the facts present in the knowledge base of the expert system shell. The query results are sent to the client application, along with the reports generated thus far as a result of all the questions asked. The reports substantiate the inference
engine’s reasoning process; the intermediate decision paths that were taken to arrive at the final list of trees as its recommendation.

Figure 6-2. The Trees Expert System Launched from the Trees Web Site

A recommended list of trees and an explanatory report generated by the Trees Expert System is shown in Figure 6-3. The veracity of the recommendations offered by an expert system has to be verified before claiming the system to be effective. This verification can best be performed by the domain expert. The expert system that was part of the Trees project was extensively tested by the subject expert (Dr. Edward Gilman) and a similar testing procedure will be employed to prove the accuracy of the recommendations made by the Trees Expert System.
Figure 6-3. Recommended Trees and an Explanatory Report
CHAPTER 7
SUMMARY AND FUTURE WORK

The following online tools and applications were developed as part of this research project and have been presented in this thesis.

1) Tree Selector – The Tree Selector has been designed to be used by expert tree managers. The application searches a database of trees based on the search criteria provided, and produces a list of trees as its recommendation that are suitable for planting at a particular planting site.

2) Rule Editor – The Rule Editor is an expert system generator, to be used by urban horticulturists and agriculturists for rapidly building and maintaining the knowledge base of expert systems. This online tool is used as the first step in converting the desktop based expert system into a Web-based expert system. All of the knowledge base that is part of the desktop based Trees expert system has been successfully inputted into the Rule Editor and stored in an object-oriented database system. To ease the migration of the knowledge base by avoiding to manually entering the knowledge into the Rule Editor, the messages, reports and questions stored in database of the desktop expert system were first written out as XML files. Parser programs were then developed, which consumed these XML files and populated the sub-editors of the Rule Editor with respective knowledge components (messages, reports, questions). Finally, all of the rules that are part of the desktop based expert system were entered into the Rule Editor. This was the only time consuming activity in the whole migration step. Once all the knowledge was migrated, JESS code was generated for all the rules. This completed the process of generating the expert system or in this case, migrating the knowledge base of the desktop Trees expert system.

3) Trees Expert System – Developing the Web-based Trees Expert System was the second step in porting the desktop Trees expert system to the Trees Web site. Once the knowledge base was stored in the Rule Editor, and JESS code generated for the rules, the Trees Expert System was developed by implementing the client-server architecture described in Chapter 6. The Trees Expert System achieves the objective of providing a Web-based DSS tool, that provides recommendations on suitable trees for planting purposes, to novice tree managers.

Some of the future works in this area include:

1) Enhancing the Rule Editor to provide for representing rules of an expert system in RuleML, an XML based language that is soon becoming the de-facto standard for representing and exchanging rules across different expert system applications. The
RuleML standard has not fully matured, yet. The RuleML community is working on establishing standard tags for representing rules. Once such an agreement on the tags is established, custom translators could be written to convert the rules represented in RuleML to any language as required by the expert system shell that is used by the application, to process the rules.

2) Incorporating features into the sub-editors of the Rule Editor in an effort to make the Rule Editor more generic so it could support development of expert systems across most problem domains. Examples include, providing a facility to indicate units for the values of facts, in the Fact Editor. Help messages in the form of text as well as images could be provided on each sub-editor to guide the user.

3) Verifying the correctness of the recommendations made by Trees Expert System by conducting rigorous testing. This could be performed by urban horticulturists and agriculturists who are experts in the problem domain. A brute force approach to this verification problem would be to test all possible sequences of questions and answers and verifying the correctness of the recommendations produced for each. However, this would be a very tedious process to perform manually, since there could potentially be an exponentially large number of test cases. Writing automated scripts to explore all possible question-answer sequences and logging the recommendations for each would seem like a better approach. After logging all the recommendations samples could be drawn from those and checked for the correctness of the advise given by the expert system for certain question-answer sequences. Still, other strategies from the area of automated testing could be employed to verify the correctness of the system.
APPENDIX
IMPLEMENTATION TOOLS

Programming Language

The Rule Editor has been developed using the Java programming language and the Trees Expert System has been built using Macromedia Flash on the client side and Java Servlet on the server side. The choice of the Java language for the development of the tools is due to the following advantages that it offers.

- Java is an object-oriented language, simplified to eliminate language features in earlier object-oriented languages like C++, that cause common programming errors.

- Java source code files are compiled into an intermediate format called byte codes which can then be executed by a Java interpreter, installed on any operating system environment, thus making all applications written in Java, portable across different machines.

- Java is a general purpose programming language with a number of features that make the language well suited for use on the Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on a computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

Object-Oriented Database Management Systems

The Rule Editor and the Web-based Expert System are designed, based on the client-server model. Most of the client-server applications make use of a Relational Database Management System (RDBMS) as their data store, while using an object oriented programming language for development of client applications. Since the Rule Editor aims at storing all the knowledge (facts, messages, reports, questions, and rules) as objects, an object-oriented database [23]. is more suitable to use as the data repository. Using a Relational database would cause certain inefficiencies because objects must be
mapped to tuples in the database and vice versa instead of the data being stored in a way that is consistent with the programming model. An object-oriented database management system (OODBMS) is the result of combining object oriented programming principles with database management principles. Object oriented programming concepts such as encapsulation, polymorphism and inheritance are enforced as well as database management concepts such as the ACID properties (Atomicity, Consistency, Isolation and Durability) which lead to system integrity, support for an \textit{ad hoc} query language and secondary storage management systems, which allow for managing very large amounts of data. Some of the advantages of an OODBMS as compared to a relational database that lead to the formers’ choice for the Expert System project are stated follows:

- **Composite objects and relationships**: Objects in an OODBMS can store an arbitrary number of atomic types as well as other objects. It is thus possible to have a large class which holds many medium sized classes which themselves hold many smaller classes, ad infinitum. In a relational database this has to be done either by having one huge table with lots of null fields or via a number of smaller, normalized tables which are linked via foreign keys. Having lots of smaller tables is still a problem since a join has to be performed every time one wants to query data based on the "Has-a" relationship between the entities. Also, an object is a better model of the real world entity than the relational tuples with regards to complex objects. The fact that an OODBMS is better suited to handling complex, interrelated data than an RDBMS means that an OODBMS can outperform an RDBMS by ten to a thousand times depending on the complexity of the data being handled.

- **Class hierarchy**: Data in the real world usually has hierarchical characteristics. The ever popular Employee example used in most RDBMS texts is easier to describe in an ODBMS than in an RDBMS. An Employee can be a Manager or not, this is usually done in an RDBMS by having a type identifier field or creating another table which uses foreign keys to indicate the relationship between Managers and Employees. In an ODBMS, the Employee class is simply a parent class of the Manager class. Complex interrelationships between objects of the application are still maintained even after being made persistent in the ODBMS.

- **Circumventing the need for a query language**: A query language is not necessary for accessing data from an ODBMS unlike an RDBMS since interaction with the database is done by transparently accessing objects. It is still possible to use queries in an ODBMS however.
• No “impedance mismatch”: In a typical application that uses an object oriented programming language and an RDBMS, a significant amount of time is usually spent mapping tables to objects and back. There are also various problems that can occur when the atomic types in the database do not map cleanly to the atomic types in the programming language and vice versa. This "impedance mismatch" is completely avoided when using an ODBMS.

• No primary keys: The user of an RDBMS has to worry about uniquely identifying tuples by their values and making sure that no two tuples have the same primary key values to avoid error conditions. In an ODBMS, the unique identification of objects is done behind the scenes via OIDs and is completely invisible to the user. Thus there is no limitation on the values that can be stored in an object.

• One data model: A data model should typically model entities and their relationships, constraints and operations that change the states of the data in the system. With an RDBMS it is not possible to model the dynamic operations or rules that change the state of the data in the system because this is beyond the scope of the database. Thus applications that use RDBMS systems usually have an Entity Relationship diagram to model the static parts of the system and a separate model for the operations and behaviors of entities in the application. With an ODBMS there is no disconnect, between the database model and the application model because the entities are just other objects in the system. An entire application can thus be comprehensively modeled in one UML diagram.
LIST OF REFERENCES


BIOGRAPHICAL SKETCH

Ranjit Ravi Iyer, hailing from Bombay (Mumbai), India, finished his schooling from Saraswati Education Society High School, Thane. He pursued a diploma in electronics engineering from Vivekananda Education Society’s Polytechnic, Mumbai. He then studied electronics engineering and acquired a bachelor’s degree from VJTI (Veer Mata Jijabai Technological Institute, Mumbai). He is currently working towards completion of his master’s degree program in agricultural and biological engineering, and computer information sciences and engineering.