CHAPTER 7: ES SHELLS: JESS BASED IMPLEMENTATION

Pages: From 89 to 104

This chapter discusses different expert system shells used in the design and development of expert system and the selection of the appropriate shell for ESQA and its functionality.
Designing an Expert System (ES) from scratch is a time consuming and costlier process. The alternative approach is the use of expert system shells/tools. We have studied the functionality and use of various ES tools in various research works already done in different domains to find out the suitable ES tool for my research work. In this chapter, I discuss the various ES tools and the JESS based implementation techniques/issues in ESOA.

7.1 Expert System Shells

Expert System shells are the tools for construction of expert systems which provides knowledge representation facilities and inference mechanisms. The knowledge engineer must gain detailed knowledge about a particular problem domain from an expert and other information source. Hence, ES shell can be considered as an expert system with all the domain specific knowledge removed and a facility for entering a new knowledge base provided. An ES shell is the software skeleton which provides an inference engine and reasoning techniques and can be customized through user interface to add knowledge of a domain. It results of a tailored expert system that matches user requirements.

Inference methods vary significantly from one domain to another and expert system shells have developed to allow the designer more flexibility during the development of the expert system.

Bourbakis (1993) states some of the advantages that arise from using an ES tool include the following:
• Use of an ES tool can improve the quality and reliability of the resulting expert system
• Tools relieve the expert system builder from having to deal with low level programming.
• It allows the expert system builder to focus on the modelling of the expert system domain.
• Tools offer facilities for the acquisition and modification of the expert system's knowledge.

7.2 ES shell components

The generic components of ES shell: the knowledge acquisition, the knowledge base, the reasoning engine, the explanation and the user interface are shown in figure 7.1.

![Diagram of Expert System Shell](image)

Figure 7.1 Components of ES Shell

There are several approaches in developing expert systems in terms of what kind of ES tools to use. These tools are normally classified as: languages, environments or shells, while some tools may fall between any two categories (Guttorsmen, 1987).
(a) Languages can be either special purpose languages for symbolic programming, such as LISP or PROLOG, or a conventional one, such as PASCAL, C or Java.

(b) Environments contain various types of knowledge representation, inference mechanisms, user interface and development aids. These tools also give access to the underlying language the environment is written in. This enables the developer to incorporate special tasks. KEE, ART and EDSS are all environments.

(c) Shells provide a more specific set of knowledge representation languages and inference mechanisms, geared to handle a particular class of problems.

Both shells and environments differ from the languages in the fact that they already contain control mechanisms that determine how they reach conclusions.

ES shells differ in many aspects such as programming language knowledge, domain knowledge, and development time.

7.3 Expert System shells: Types and Features

Terry (1991) listed the following features in typical ES shells:

(a) Built-in inference engine with choice of chaining methodology: In some shells mixed chaining is considered to be a useful feature in some shells, as it helps the knowledge engineer to either put constrains at the start of the rules or at the end depending on what is required to solve a problem.
(b) Menus and prompts for development and testing: Menus must be easy and clear to use and contain all the functions needed to build the required expert system. It is important that the knowledge engineer becomes familiar with the shell menus and prompts in order to find out whether it will be able to perform the job in hand.

(c) Debugging and value-checking aids: This function will deal with the ability of the shell in the debugging of any input errors, which could be contained in the information and values.

(d) Consistency checking: This function will check that the rules do not contradict each other, as this could lead to false assumptions or wrong conclusions.

(e) Rule - prioritise capabilities: When solving a particular problem a group of rules which relate more closely to the problem will be considered first. By matching the rules to the problem in hand the correct rule to solve the problem will be chosen.

(f) Interface with other operating systems and software: The shell must have all the required interfaces to assist in building the expert system. These interfaces can either be supplied as part of the shell or it must have the ability to communicate with other software, whether by direct access or by exporting and importing functions. The shell should also have the ability to create and display graphics or interface with graphics software programs.

(g) Mechanism for handling input and calculation of uncertainties: This function will allow the shell to deal with missing or incomplete data which could be inserted by the user in the form of input.
(h) Explanation facilities: This function will allow the user to find out why and how the system arrives at its decisions, with the sequence of rules involved.

7.4 Comparison with other development tools

In the survey conducted by Mohan (1990), 24 out of the 37 expert systems developed were developed on IBM PC class of microcomputers. 18 out of the 37 expert systems were developed using expert system shells (that is nearly 50%), 7 were developed using expert systems environments, 7 using artificial languages and 5 using other languages. It indicates the popularity of shells as a tool in developing expert systems.

Guttorsman (1987) divided shells into the following types:

(a) Induction tools

These shells generate rules based on a number of examples given by the developer. The shell then applies certain algorithms to the rules to determine the order the system will take when putting questions to the user. These tools are useful only when the task is simple and well structured. SUPEREXPERT is a typical example of this class.

(b) Simple rule-based tool

"If...Then" rules are used to represent knowledge in the knowledge base. The main weakness of these shells is the inability to sub-divide into various sets that can be arranged into a hierarchy. Typical applications are small rule bases containing from 50 to 200 rules.
(c) Object oriented tools

These tools represent the knowledge as objects which can be given certain values. An object may contain facts, rules or pointers to other objects. The problem must be of a certain complexity to justify the use of complex object-oriented shells. LEONARDO is an example of this class.

(d) Logical tools

These tools use Horn clauses and resolution strategies derived from predicate calculus. They are useful when dealing with very complex problems in terms of logic.

(e) Structured rule-based tools

These shells also use "IF ... THEN" rules but with the facility of arranging rules in a hierarchy. These systems are useful where a large number of rules are involved. CRYS TAL is a typical example of structured rule-based tools.

There are many ES shells some were improved while others were decommissioned. The famous shells are Automated Reasoning Tool for Information Management ART-IM, C Language Integrated Production System (CLIPS), Knowledge Engineering System (KES), Level 5, VAX OPS5 and Java Expert System Shell (JESS).

ART-IM is one of the first expert system shells written in Lisp. It supports three different knowledge representation techniques: rule-based, pattern matching, and frames. As for its inference technique it supports only forward chaining. However, users can write their own backward chaining mechanism. It supports Windows and DOS.
environments. ART-IM was involved in the development of Vertical Launch System Tech Assistant Expert System (VTAEXS). A program designed for in-service engineering for the missiles launch system for the U.S navy fleet. They designed a technical assistant ES for maintenance with fewer resources. This has resulted in a paper-less environment, expertise level assistance, and cost beneficial system (Small and Yoshimoto, 1995).

CLUES, Countrywide Loan Underwriting Expert System, are developed through ART-IM shell. Underwriting a loan is time consuming, expensive and difficult process for loaners. Countrywide has built this system to provide expert support to underwriters. Using loan application, loan customer’s credit report, and appraisal CLUES can process this information to produce an underwriting stating whether the customer is funded, or forwarded for second review by an underwriter. CLUES had been in use since 1993 at most countrywide branches and is processing around 50% of loan applications. This supported anti-bias decisions, served as a training tool, improved customer service, and is cost beneficial to the Countrywide (Talebzadeh et al., 1995).

CLIPS is a widely used expert system shell written in C and developed by NASA. Designed to prevail over Lisp designed shells due to its high portability and ease of integration. However, it lacks backward chaining and frames supports. Nevertheless, it is favoured for its low cost and ease of use. Al-Taani (2005) used the CLIPS shell to build an ES for car failure detection. The system favoured CLIPS because it provides forward chaining, windows deployment, source code availability, and pattern matching. The result was an ES with high performance, and sufficient response time. It has been used for building a prototype medical expert system which distinguishes between migraines and
headaches. And it proved to be time effective for physicians to determine the case from possible diagnosis results (Kopec et al., 2005).

KES is another well-known ES shell. It supports rules and classes, as well as both inference mechanisms. It is a window based development tool. As it is written in C, it has reasonable performance time. It provides a simple GUI and supports ease of integration. In case of product manufactures, there is a Flexibility Manufacturing System (FMS). It is designed to manage manufacturing resources such as time and effort. Ozbayrak and Bell (2003) propose a Knowledge Based Decision Support System (KBDSS) for FMS in tool and operational control. KBDSS has the ability to diagnose malfunctions at the machine, cell, and system levels. The system offered quick expertise level responses, reduced downtime, and improved product manufacturing.

Level 5 is an ES shell. It supports production rules, and backward chaining. Because of its ease of use, simple installation, and reasonably priced, it has become the choice for Abdel khalek and Arrashid (2004) choice for designing their Ground Water and Soil unsuitability for Foundation Claims (GWSFC). Their ES is built to assist in proposing solutions to claims in ground water and soil and aid in decision making for engineers and arbitrators. Quality Feedback Expert System (QFES) is developed through Level 5 shell (Balas et al., 1996). The area of clinical practice guidelines is the source for practice recommendations. Through performance analysis and making available several related resources of different practices the patient receives better diagnosis, and it reduces the medical cost for both clinic and patient.

OPS5 is an ES shell used for large projects. Its knowledge representation includes rules, and pattern matching. And it supports an
efficient forward chaining mechanism based on the Rete algorithm. OPS5 is highly flexible but lacks frame representation, backward chaining, and it is highly machine dependable. The first expert system developed through the OPS5 shell is the famous Expert Configuration (XCON), previously named R1. It was designed to assist in the area of software and hardware configuration at Digital Equipment Corporation. XCON confirms an order and guides its assembly. It has a success rate of 98%. It has increased optimization performance, flexibility, and parts were ordered and configured correctly which increased customer satisfaction.

Tomic et al., (2006) describes JavaDON is an open-source expert systems shell based on the OBOA framework for developing intelligent systems. The central idea of the JavaDON project was to make an easy-to-use and easy-to-extend tool for building practical expert systems. Since JavaDON is rooted in a sound theoretical framework, it is well-suited for building even complex expert system applications, both stand-alone and Web-based ones. JavaDON knowledge representation scheme supports using multimedia elements along with traditional techniques, such as rules and frames. Another important feature of JavaDON is its capability of saving knowledge bases in XML format (in addition to the shell’s native format), thus making them potentially easy to interoperate with other knowledge bases on the Internet. So far, JavaDON has been used to build several practical expert systems, as well as a practical knowledge engineering tool to support both introductory and advanced university courses on expert systems. The paper presents design details of JavaDON, explains its links with the underlying OBOA framework, and shows examples of using JavaDON in expert system development.
Drools is a Rule Engine that uses the rule-based approach to implement an Expert System and is more correctly classified as a Production Rule System. It is a Java based rule engine and is free. The rules are represented in XML format.

Euler is an inference engine supporting logic based proofs. It is a backward-chaining reasoned, enhanced with Euler path detection. It has implementations in Java, C#, Python, Javascript and Prolog.

InfoSapient is an Open Source rules engine that allows members of the business community to enter and maintain business rules using everyday language. The resulting automated decision-making is similar to the way informed human make decisions. Many variables may be considered simultaneously and used to weigh risks and opportunities in order to arrive at the best course of action. Given ambiguous situations that are typically found in business processes, InfoSapient will always arrive at a 'best' course of action. This allows complex decision-making to take place at the precise moment it is needed without depending on the skill and experience of a highly distributed workforce. InfoSapient is written to the 100% Pure Java standard. It supports fuzzy logic representations.

Apache Jena™ is a Java framework for building Semantic Web applications. The Jena Framework includes a rule-based inference engine for reasoning.

JEOPS is free Java rule based system. It adds forward chaining, first-order production rules to Java through a set of classes designed to provide this language with some kind of declarative programming. With that, the development of intelligent applications, such as software agents or expert systems is facilitated.
JLisa is a free Clips like Rule engine accessible from Java with the full power of Common Lisp. With JLisa the following things can be done:

- Execute Clips-like Rules on Java objects
- Build Java agent systems that “reason”
- Build expert systems where Java objects participate
- Build GPS (general problem solvers) in Java
- Perform Business Rule validations
- Create and execute Decision Trees for Java objects
- Perform Logical Queries and Searches on Java objects
- Create Expert and Smart Wizards

JTP is an object-oriented modular reasoning system developed by Gleb Frank in Knowledge Systems Laboratory of Computer Science Department in Stanford University. JTP is based on a very simple and general reasoning architecture. The modular character of the architecture makes it easy to extend the system by adding new reasoning modules (reasoners), or by customizing or rearranging existing ones. The system is implemented in Java. The reason for this choice is that Java is ideally suited for easy integration of object-oriented systems, which facilitates both extending JTP's functionality and embedding JTP in other systems.

OpenRules is a general purpose Business Decision Management System (BDMS) available as an Open Source product. It makes rules-based systems not only less expensive, but more importantly, easier to develop, and manage, and more sustainable. OpenRules include a rule engine known as "OpenRulesEngine" that is used to execute different rule sets and methods using application-specific business objects. OpenRulesEngine can be invoked from any Java application
using a simple Java OpenRules API or the standard JSR-94 interface. OpenRulesEngine is thread-safe and works very efficiently in multi-threaded environments supporting real parallelism. Contrary to many other rule engines, OpenRules' rule engine is stateless. It allows a user to create only one instance of the class OpenRulesEngine, and then share this instance between different threads - there are no needs to create a pool of rule engines.

OpenRules® as a component of complex real-world applications has a proven record of high performance and scalability. These days most Rule Engines demonstrate a high level of performance, but OpenRules® is frequently chosen over major commercial BR products because of the following differentiators:

- High performance
- High scalability for multi-user environments
- Efficient use of memory.

7.5 Comparison of Techniques

To conclude ES shells comparison, I find the use of forward chaining will be simpler, faster, eliminate overhead, continue to maintain information throughout the program of the user input, and interactivity is more supported in forward than backward.

I have found that CLIPS and JESS are the best ES tools for my research work. In the following section I have cited the comparisons of CLIPS and JESS.
7.6 Comparison of CLIPS and JESS

Both CLIPS and JESS are products with a large support on the internet, but CLIPS seems to have a broader audience, probably because it exists longer. This difference in age results in the CLIPS package being more stable and complete, while JESS users will still experience some minor bugs. JESS is constantly updated and the author, Ernest Friedman-Hill, has been very responsive to user/developer feedback and regularly puts out new releases and bug fixes.

Now a day, the choice between JESS and CLIPS depends on the application. If it is web-based, the choice of JESS is a very logical one (which is even supported by the authors of CLIPS). For the more classic applications, CLIPS will probably be chosen because of its reputation of being more stable and having more support.

The future of JESS depends highly on the evolution of the web, the Java programming language and its own future stability. These three conditions make that there is a great possibility that JESS will become more popular and more frequently used. Especially the object-oriented possibilities and the easy integration into Java code makes JESS’ future very promising.

CLIPS, on the other hand, is more likely to implement the new and sophisticated features first as they come out, since it still has the advantage in time. CLIPS has also various extensions and variants (like FuzzyCLIPS, AGENT CLIPS, DYNACLIPS, KnowExec, CAPE, PerlCLIPS, wxCLIPS and EHSIS to name a few) that give it an advantage with respect to support of methods like fuzzy logic and agents.
To summarize, CLIPS is still more complete and stable than JESS, but this might change in the future, since the JESS package is being improved constantly. Besides that, JESS has also the property of using Java, which in the long run might prove to be a big advantage over CLIPS.

Since my research work is related to the web based expert system, I have chosen JESS as my ES tool.

7.7 Java Expert System Shell (JESS)

JESS is a rule engine and scripting environment written entirely in Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. Jess is small, light, and one of the fastest rule engines available. Its powerful scripting language gives one access to all of Java’s APIs. JESS includes a full-featured development environment based on the award-winning Eclipse platform. JESS uses an enhanced version of the Rete algorithm to process rules. JESS has many unique features including backwards chaining and working memory queries, and of course JESS can directly manipulate and reason about Java objects. JESS is also a powerful Java scripting environment, from which one can create Java objects, call Java methods, and implement Java interfaces without compiling any Java code. JESS can be licensed for commercial use, and is available at no cost for academic use.

JESS has some advantages over CLIPS. The Java language implementation makes JESS the choice of developing web based expert system, even if it is also possible to use CLIPS with C++ in a Common Gateway Interface (CGI) script. JESS enables the user to put multiple expert systems in one process. Java threads can be used to run these systems in parallel.
7.7.1 JESS Architecture

![JESS Architecture Diagram]

Figure 7.2 Architecture of JESS

7.7.2 ES with JESS Based Implementation

Chunet al. (2001) presents the design and implementation of a product recommender system on an Internet shopping mall. In e-commerce application, sometimes potential buyers may be interested in receiving recommendations about what to purchase. The mainstream of an automated recommender system is collaborative filtering. In the paper, they present a knowledge-based product recommender system. The knowledge base of product domain and inference engine is implemented by JESS (Java expert system shell) and Java servlet. The system gathers the user's requirements on a particular product by questioning the user and consulting its knowledge base to find the items that best meet the user's requirement.
Ho et al. (2005) introduces a web-based expert system, called class schedule planner (CSP). CSP encapsulates class-scheduling knowledge and gives intelligent scheduling advises to students. CSP has a set of Web forms to collect inputs from the users and then translates the request into facts. It uses JESS (Java expert system shell) as a tool to process the facts and rules to generate feasible schedules for the students. Integration of data from various sources into JESS facts and rules is done by CSP controller and XML translator. The translated result from JESS is integrated with the Web presentation using JSP (Java server page).