Chapter 3

Development of Prototype

Introduction

In the following sections, a brief introduction is given about the designing of the prototype and the expert system shells. The steps involved in the development of the prototype are discussed. Finally the results and observations are tabulated.

Designing a Expert system prototype

Expert systems require special approaches to systems analysis, especially to the collection of the data (or rather knowledge) on which the system is based. The process of gathering the knowledge to stock the expert system's knowledge-base & knowledge acquisition has proved to be the most difficult component of the knowledge engineering process. It's known as the 'Knowledge Acquisition Bottleneck', and expert system projects are more likely to fail at this stage than any other. Knowledge acquisition almost always involves extracting knowledge from someone expert in the specific field. This process of knowledge elicitation involves a variety of interview and non-interview techniques.

The development of Expert systems requires special approach to software management. The methodologies used to build expert systems have been shaped by the problems with knowledge acquisition, described earlier. For a long time, the favourite development methodology was rapid prototyping. (Cyclical development means more or less the same thing). In the mid-1980s, this approach came under criticism, because it appeared to have all the shortcomings of the unstructured approaches which had been used, with very poor results, in the early days of mainstream software. But the structured systems analysis & design methodologies did not seem to be appropriate, because of the differences between knowledge and data. As a result, special-purpose development methodologies for knowledge engineering were developed. The most well known is KADS, (KADS is a development methodology for knowledge-based systems) which was developed at the
University of Amsterdam, as part of the ESPRIT programme, in co-operation with several European partners. The phase of prototyping can have the following steps.

i. Build a small prototype
ii. Test, improve and expand it
iii. Demonstrate and analyse feasibility
iv. Complete the design

Prototyping systems development focuses on the iterative creation of a new expert system. Rather than going through the whole SDLC process for everything that could be potentially envisioned with the system, a portion of the system is chosen to use to create a prototype. The prototype does not go through extensive requirements analysis and instead focuses on getting something created quickly for immediate use by end-users in order to gather feedback to either modify the prototype or begin the process again with another component. While the prototyping development method gets users involved in the system's development and typically brings about real results quickly, it can be difficult to manage prototyping-based projects due to its differences with SDLC [49]. The primary level analysis and design is illustrated in figure 3.1.

![Figure 3.1: Primary level of analysis and design](image-url)
The process of building an expert system is commonly known as knowledge-engineering. This implies knowledge acquisition from a human or other source and coding it into the knowledge base of the expert system. The main phases in the knowledge engineering process are the dialog process as represented in Fig. 3.2. This is similar to the task of a system designer discussing the requirements of the program with the client, in conventional programming. After acquiring knowledge from the human expert, the knowledge engineer has to explicitly code it into the expert system knowledge-base. After the coding stage, the human expert evaluates the expert system and gives feedback/critique to the knowledge engineer. The knowledge engineer alters the knowledge base in order to reflect the human expert's comments [50, 51].

![Figure 3.2: Basic concept of an expert system](image)

This process will iterate until the human expert finds the expert system satisfactory for the envisaged purpose. Some rule-based systems may display the capability to learn rules by
example (rule induction), creating rules from tables of data. Learning is a complex process, hindered by ambiguities, duplications and inconsistencies in the human expert knowledge.

**Expert System Shell Used**

Expert system shells provide methods of building expert systems without extensive knowledge of programming through mechanisms that (i) input the decisions, questions and rules that are followed (ii) construct a knowledge database that can be manipulated by subsequent parts of the system (iii) verifies possible violations of surface validity and (iv) operates the "inference engine" that operates on the rules, poses the questions to the users, and determines whether a particular decision is valid. Most expert systems also allow the user to halt the processing at any time to query the system why a question was asked, or how a decision was reached [52]. Most expert system shells can now run easily on most current micro-computers and are able to handle the manipulation of a relatively large number of rules and associated questions. Expert system shells are expert system development tools consisting essentially of the expert system without the knowledge base, embodying the inference engine, working memory, and the user interface [53]. An example of the inference engine part of an expert system that deduces new conclusions from known facts is illustrated below [53]:

\[
\text{IF liquid limit=known} \\
\text{AND plastic limit=known} \\
\text{AND plastic limit>liquid limit} \\
\text{THEN soil=non plastic}
\]

Expert systems give advice or solve problems by drawing upon this knowledge stored in the IF/THEN rules. A survey of different expert system shells was conducted. After careful study of the shells two shells for the development of prototype were selected they are ES-Builder Ver 3.0 and CLIPS.
3.3.1 ES-Builder Ver. 3.0 Expert System Shell

ES-Builder Ver. 3 is simple software to build a expert system prototype. It does away with the separate programs model of the previous version and combines everything into one simple interface. The user can create a web site in the ES-Builder Ver. 3.0 Expert System Shell. A full expert systems project can be created in HTML format for electronic publishing. Built-in exporting functions include the flexibility to create a web page for:

- Searching the expert system
- Displaying the Knowledge Base
- Displaying the Decision Tree
- Displaying the full Decision Table
- Listing the Attributes and Values
- Specifying the ES Documentation

Some of the important features of the ES-Builder are as follows

- The **ES-Builder Expert System Shell** is a set of purpose built tools for use in the teaching of the basic concepts of expert systems.
- **ES-Builder** is compatible with all versions of Microsoft Windows from 98 to XP. It has a simple familiar Windows interface which allows the easy entry of expert system data once the research for the expert system has been completed.
- **ES-Builder** allows the user to easily build decision trees of basically unlimited size and complexity. Once conclusions, attributes and values are defined using the decision tree, data entry is simple and intuitive.
- **ES-Builder** uses the Windows tree view component which allows the easy creation of expert system decision trees by point and click addition of attributes, values, and conclusions. Many possible errors in tree construction are precluded by the program which only allows valid branches to be added to any node. The decision tree can be tested easily at every stage of data entry for validity by a simple button click that will report errors and move to the erroneous node.
- **ES-Builder** allows the user to add detailed notes about each attribute, value and conclusion along with a picture or diagram to illustrate each one. This additional data becomes a part of the Expert System web site [54].
Once the **Decision Tree** is completed using **ES-Build**er, the user can print the **decision tree** to a selected printer or choose to export paste the tree in rich text form into another document, e.g. the word processor they are using to write up their assignment.

### 3.3.2 CLIPS

A CLIPS (C Language Integrated Production System) is an expert system tool developed by the Software Technology Branch (STB) at the NASA/ Lyndon B. Johnson Space center. It was released in 1986 for the first time and has undergone continual refinement and improvement ever since. Distributed computing systems hosts a mailing list for CLIPS users at clips@discomsys.com. More information can be found in the official CLIPS web page [55].

A CLIPS is a tool that is designed to make the development of software to model human expertise easier. Apart from being used as a stand-alone tool, CLIPS can also be called from a procedural language, or CLIPS can call procedural code itself. It has been designed for integration with other languages such as C, C++, and Ada. A CLIP represents (human) Knowledge in three ways:

i. Rules for experience-based, heuristic knowledge

ii. Generic functions for procedural knowledge

iii. OO programming, also for procedural knowledge

The CLIPS language looks a lot like LISP. Commands are written between ‘(‘and ‘)’ characters and it uses similar features like atoms, numbers, strings and lists. Adding 7 and 5 would be done by typing: “(+ 7 5)” at the command line. CLIPS will then respond with 12. The CLIPS shell provides the basic elements of an expert system:

i. A fact-list, and instance-list : this is global memory for data

ii. A knowledge-base: which contains all the rules

iii. An inference engine: for controlling the execution of rules

Facts are data that usually designate relations or information like: (is-animal duck) or (animals duck horse cow chicken) or (this is a test). Rules can be applied on these facts in the form of IF-THEN rules. These rules are special because they ‘fire’ only once. Variables and wildcards can be used in the rules; Functions can be defined to manage the rules. The
subsequent section shows how these facts and rules can be implemented. The other choice of ES shell was JESS, which is based on CLIPS. JESS was originally a clone of essential core of CLIPS, but has begun to acquire a Java influenced flavor of its own. Hence, it was found that CLIPS is more suitable for the development.

**Steps in Implementation of prototype**

The detail procedural analysis was carried out. After going through the analysis the procedure which was adopted to develop a prototype is mentioned below.

i. Study of ES-Builder Ver. 3.0 and CLIPS concepts
ii. Discussions/interviews with the Legal experts in field and academicians in Law
iii. The framing of rules using the Legal knowledge and incorporation into CLIPS and Expert system Builder
iv. Testing and Implementation

A series of discussion and the interviews were held with the practicing advocates who are experts in the field of transfer of property act. The discussions consisted of understanding the concept of law/rules of transfer of property act and the intricacies involved in them. The broad outlines of the rules which need to be framed were noted. The discussions and the interviews with the academicians in law were conducted. The interviews were of great help in framing of the rules. Following are the some sample examples of rules which are used in the development of the prototype

**RULE 1:**

IF THE SUBJECT IS ACQUISITION OF PROPERTY

THEN PROPERTY IS ACQUIRED BY TRANSFER | PROPERTY IS ACQUIRED BY SUCCESSION

**RULE 2:**

IF THE PROPERTY IS ACQUIRED BY TRANSFER

AND IF TRANSFER DEED IS AVAILABLE / THEN THE TRANSFER DEED IS REGISTERED | THE TRANSFER DEED IS UNREGISTERED
The rules were framed and thoroughly checked by the legal experts, after which the implementation in the ES-Builder Ver. 3.0 and CLIPS was taken up. The ES-Builder Ver. 3.0 and CLIPS syntax formats in form IF-THEN form and ‘decision tree’ to cross check the findings was done. The testing of the rules was done and then again verified by the Legal Experts and scholars in field of law. The subsequent section explains the implementation in Es Builder and CLIPS.
Figure 3.3: Prototype Expert System in ES Builder 3.0
Decision Tree Generated:

Purchase of Property

---A - Method of Purchase

| ---V - Transfer

| +---A - transfer deed registration

| | ---V - Yes

| | | +---A - Transfer <= 13 years

| | | | ---V - Yes

| | | | | +---A - Record of Rights

| | | | | | ---V - Yes

| | | | | | | +---A - Nil Encumbrance Certificate

| | | | | | | | ---V - Yes

| | | | | | | | | +---A - Up-to-date Tax Paid Receipts

| | | | | | | | | | ---V - Yes

| | | | | | | | | | | +---C - Property can be purchased

| | | | | | | | | | | | ---V - No

| | | | | | | | | | | | | +---C - Property cannot be purchased

| | | | | | | | | | | | | | ---V - No
Figure 3.4: A Part of Sample Decision tree developed in ES Builder 3.0
Clips Implementation:

Figure 3.5: Snapshot of CLIPS WINDOW
Table 3.1  Pseudo code of clips program

;CLIPS code to implement the process of purchase of property.

(deftemplate POP (slot pop))
(deftemplate TREG (slot treg))
(deftemplate TR13 (slot tr13))
(deftemplate RRT (slot rrt))
(deftemplate NET (slot net))
(deftemplate TaxR (slot tr))
;
(deftemplate STI (slot sti))
(deftemplate SPRB (slot sprb))
(deftemplate SSUC (slot ssuc))
(deftemplate SLAM (slot slam))

;main question, divides the flow control into two parts.
;PART 1:Acquisition of property thru transfer.
;PART 2:Acquisition of property thru succession.

(defrule GetPOP
 =>
 (printout t "Is the Property Acquired by Transfer or Succession <t/s>::")
 (bind ?response (read))
 (assert  (pop ?response)))

;***************************************************PART -1***************************************************

(defrule GetTREG
 (pop t)
(defrule GetRRTn
  (tr13 no)
  =>
  (printout t "Is the Nil Encumberance Certificate Available (yes/no):")
  (bind ?response (read))
  (assert (net ?response)))

(defrule GetNETy
  (rrt yes)
  =>
  (printout t "Is the Nil Encumberance Certificate Available (yes/no):")
  (bind ?response (read))
  (assert (net ?response)))

(defrule GetNETy
  (rrt yes)
  =>
  (printout t "Is the Nil Encumberance Certificate Available (yes/no):")
  (bind ?response (read))
  (assert (net ?response)))

(defrule GetNETn
  (rrt no)
  =>
  (printout t "Property cannot be purchased:")
  (halt))

(defrule GetTaxRy
  (net yes)
Table 3.3 Pseudo code of clips program

```
(defrule GetRRTn
  (tr13 no)
  =>
  (printout t "Is the Nil Encumberance Certificate Available (yes/no):")
  (bind ?response (read))
  (assert (net ?response)))

(defrule GetNETy
  (rrt yes)
  =>
  (printout t "Is the Nil Encumberance Certificate Available (yes/no):")
  (bind ?response (read))
  (assert (net ?response)))

(defrule GetNETn
  (rrt no)
  =>
  (printout t "Property cannot be purchased:")
  (halt))

(defrule GetTaxRy
  (net yes)
  =>
  (printout t "Are up\-to\-date Tax Paid Receipts Available (yes/no):")
  (bind ?response (read))
  (assert (net ?response)))
```

The development of a rule-based expert system prototype for tracing the title of property consisted of only few rules of law. The prototype has been tested successfully by the domain experts. In the development of the prototype rules were checked for their consistency. These results can be substantiated with addition of more rules and more modules. Development of composite expert system is discussed in subsequent chapter.