Chapter 7: Methodology and System Design

The chapter presents the methodology adopted for the present research work which includes development of an expert system, the proper choice of an expert system shell, system design, and the knowledge base development of the system and the technical method of the evaluation of the system.
Methodology and System Design

7.1. Introduction

The thrust area of the present research work is mainly focus in the background of study, system designs and development of prototype of an expert system. The system background starts with the compilation of different disease symptoms which appears throughout the life span of the rice plant from agricultural experts, plant pathologists and also previous literatures. The acquired knowledge of the rice plant diseases are represented to develop expert system with a selected expert system shell.

An Agricultural expert system relies on accumulation and integration of related knowledge and information from various diverse sources. The common sources to gather information that the different stakeholders require for decision making are Agriculture specialists and raw experiences of the experts. Since, in recent years, tools, technologies and applications of information technologies have emerged as efficient and effective measures for upgradation of the whole agricultural fields, ranging from scientific studies to farmers help. In such an era of IT revolution, integration of expert system acts as a powerful tool for the stakeholders of agricultural production has extensive potential.
7.2. Choices of Expert System Shell

The program which we use to build an expert system is an expert system shell. It is an SDE (Software Development Environment). This can be visualised as if a word processor is used to make a report or letter similarly expert system shell is the instrumental for developing an expert system. A typical expert system shell consists of basic components of expert system like some form of knowledge representation scheme and a built in inference mechanism. A shell is associated with a prescribed method for building applications by configuring and interacting with these components. In addition, the shell often contains facilities for producing clear interfaces for the use of the application. But it must be noted that this type of shell gives no assistance in eliciting and organising the knowledge. To do his job a knowledge acquisition tool is must.

There are many expert system shells for developing an expert system. To select an appropriate expert system shell for the development of an expert system, we have discussed the four expert system shells namely: ESTA, CLIPS, JESS and Drools. ESTA is an expert system shell which stands for Expert System for Text Animation. It has the explanation facilities of the questions in the knowledge base and for given advices. This shell contains the rules
represented in its own syntax for its knowledge base. The inbuilt facilities to write the rules that build the knowledge base are consisted in this shell. "C Language Integrated Production System (CLIPS)" is also another shell for developing an expert system. It has the rule-based, object-oriented and procedural programming features. The inference engine uses the RETE algorithm, only provides forward chaining, LISP-like syntax - all expressions are enclosed within round brackets.

We have also considered the shell JESS (Java Expert System Shell) for the development of the system and it has the following features: Inference engine uses the RETE algorithm, provides forward and backward chaining, extends CLIPS syntax, knowledge represented as rules [106].

Drools5 is also an expert system shell we have considered for our expert system and it also provides the following features: more than a classic expert system shell - provides a platform for integration of processes and rules, consists of four modules: Drools Guvnor-knowledge base repository, Drools Expert-rule engine, Drools Flow-work of modelling, Drools Fusion-event processing/temporal reasoning. It only provides forward chaining, inference engine uses a RETE-based algorithm. We have selected the expert system shell JESS for the development of our system after we have studied the

above shells because it has the some more features than the other shells [25].

7.3. Java Expert System Shell (JESS)

JESS (Java Expert System Shell) is hereby used as an interpreter for the Jess language, a rule-based language for specifying expert systems so as to implement rules for modelling the knowledge base. It is a scripting language entirely written in Sun Microsystems's Java Language. It also supports the development of rule-based expert systems which can be tightly coupled to code written in the powerful portable java language. JESS language is therefore a declarative language [106].

Architecturally, JESS is a production system of executing a rule-based program. A JESS document is usually created in text editor, including the windows platform editor and Notepad. Though it can work alone and could be run on the windows command promptly, but functionally incorporated into a Java program.

JESS file is saved with a “.clp” extension usually adjacent to the normal “.txt” extension. It contains a JAR file which links to the Java IDE environment. In Java IDE environment, program codes are usually written for defining specific functions. As soon as the Jess is referred in the code, it would run predefined instructions subject to
user's input from the Java Interface. JESS run and manipulated on the Java interface.

Various expert systems developed by using JESS are accessible. A study was conducted by developing "Building a Computer-Based Expert System for Malaria Environmental Diagnosis: An Alternative Malaria Control Strategy," using NetBeans 5.5 for the knowledge base and Microsoft SQL server is used as the Database engine for their system [107]. Here, the JESS file is called in the Net Beans environment and the Database. Similarly, "An Expert System for Endocrine Diagnosis and Treatments using JESS" is also an expert system which is developed using JESS [34]. It used to perform its functions, facts and procedures using JESS. Another system "SMARTVIEW - An Intelligent Expert System Tool using Java and JESS Framework," is developed using Java and JESS [108]. In this system, it deals with the issues of the management processes required to effectively understand and utilize the valuable facts as they should be used. At the same "Decision Support Systems for Launch and Range Operations using JESS" is also an expert system which discusses details of the knowledge base and inference engine launch and range virtual test bed [109]. "Signature-based network intrusion detection system using JESS (SNIDJ)" is also a rule based expert system developed by using Java Expert System Shell (JESS) [110]. JESS is recently used in the development of "Employing a Java
Expert System Shell for Intelligent Support in Exploratory Activities [111]". The work represents issues around the integration of a framework for the development of interactive exploratory activities (DANTE) and a rule engine and scripting environment (JESS).

From the above survey, we have developed our expert system using Java Expert System Shell (JESS) for the knowledge base development with Netbeans 7.0 and SQL is used as the Database engine for the system. The JESS file is called in the NetBeans Environment and the Database also.

7.4. The Rete Algorithm

JESS, a rule-based expert system shell, in the simplest terms, means that a set of if-then rules continuously apply to a set of data in the knowledge base. JESS, therefore, employ an advance form of well-known algorithm called the Rete Algorithm to match the rules against the facts in the knowledge base. Dr Charles L. Forgy, Carnegie Mellon University had designed the Rete Algorithm which is an advance pattern matching algorithm [112]. This algorithm is a very well-organised algorithm for matching facts against the patterns in rules.

It is required to define a rule set of knowledge base which is consisting of one or more rules for the complete implementation of
the Rete Algorithm. There is some knowledge represented in every rule in the rule set. The if-then statement is usually used for the representation of rules in the knowledge base. The Rete algorithm is sometimes called the Rete Rules because if-then rules are suitable for this algorithm. The rule engine required a complete rule-set for further processing of the rules. Each rule in the rule set is matched by the rule engine with given facts to make a decision whether to fire the rule or not. This process is called the pattern matching process and takes place repeatedly. The list of facts may be modified in each cycle: new facts may be added to the knowledge base or old facts may be removed from the knowledge base. Such types of changes make the previously unsatisfied patterns to be satisfied. In addition, the set of rules satisfied must be maintained and updated during each cycle. The actions of rules modify only a few of facts in the list in most of the cases and we call this temporal redundancy. If the rule engine examines each rule to control the search for all the facts even though most of the rules are not modified then it will slow down the process. The unnecessary computation can be avoided by computing only the changes necessary for the newly added or removed facts. This work is perfectly performed by the Rete algorithm.
7.4.1. Implementation

A network of nodes is used for the implementation of the Rete algorithm. The network of nodes is designed in order to save the state of the matching process from cycle to cycle and recomputed the changes only for those modified facts. The matching process state is updated only after the facts are added or removed. The matching process will be faster if the number of process of added or removed facts.

7.4.2. The Inference Cycle

An inference cycle which is consisting of three phases: match, select and execute has to be a part of each rule. The conditions of the rules in matching phase are matched against the facts determining which rules are to be executed. The rules having the conditions for firings are met are put in a list known as agenda for firing. One of the rules in agenda for firing is selected to execute or fire. The selection approach may depend on the selection criteria, for example, priority, regency of usage and specificity of the rule. The selected rule from the agenda for firing is fired by carrying out the actions in the right hand side of the rule. The action may be any of an assertion, executing a user defined function or built-in function or executing a decision table, or otherwise.
7.4.3. Building of the Rete Network

The Rete network consists of nodes representing patterns in the conditions of the rules shown in the Figure 7.1. The nodes are behaving like filters, testing the incoming tokens, and giving only those which pass the test. This network consists of two parts which are alpha network and beta network. Alpha networks are consisting of alpha nodes and each node has one input which defines intra-elements. Beta network consists of beta nodes and each node takes two inputs to define inter-element conditions.

The Rete network in the Rete algorithm starts with the root node called Rete Node and the root node is followed by Kind nodes. Each fact type should have a kind node in the network. After this, alpha nodes are constructed for each pattern and connected to the corresponding kind node. Let us take an example. There are two patterns $x>y$ or $y>z$ of a condition, hence two alpha nodes will be
created: one for \(x>y\) and one for \(y>z\). Each alpha node is connected with a memory known as alpha memory and this is used to remember the facts matched. These alpha nodes are then joined by beta nodes and only two inputs are accepted by beta nodes. So if there exist three alpha nodes then the first two nodes will be joined by one beta node. The output of this beta node and the third alpha node then will be joined by another beta node. In this way, beta nodes support partial matching and each beta node has a memory to store joined patterns.

A token is created by assertion of each fact and the tokens initially enter the root node. The network then divides a branch for each token type. After this, each kind node gets a copy of the token and performs SELECT operation to select token of that kind. A copy of the token node it received is delivered by the kind node to the alpha node. The alpha nodes perform a PROJECT operation and extract components from the token that match with the variables of the pattern after receiving the token. Thus, alpha nodes evaluate the conditions and beta nodes then determine the possible cross product for a rule. Thus the actions in the rule will be executed in this way [113].
7.5. System Design

In the design of the proposed expert system, the model of the proposed system is built to capture the dynamic behaviour of the system using Unified Modelling Language (UML). There are five diagrams available to model dynamic nature of a system in UML and Use Case diagram is one of them. A Use Case diagram is used for modelling the design of the system.

Figure 7.2: Use Case Diagram of the System

A Use Case diagram is a system which graphically explains the interactions between the system, the external system and the user. Use Case diagram plays a major role in design of the system as it acts as a strategy in constructing the structure of the system. It also describes who will use the system and in what way the user interact.
with the system. We used the use case diagram in the designing of our system is to portray the user, the set of use cases for the system and the relations between the user and use cases.

Here, the three main Use Cases are introduces which extend, include or use other Use Cases.

- Input Information;
- View Decisions;
- Exit System.

**The User:** The user is one of the clients who make use of the system and must be able to make decisions. "A user might be a person, a company or organization, a computer program, or a computer system - hardware, software, or both."

**Input Data:** The interface where the users are going to enter the input data into the system based on questions about their problems in the rice plant. Here the main input is the symptoms of the diseases. The system then responds based on the correlation between users’ input data and its foreknown intelligence. This uses another Use Case called Diseases Details.

**View Decisions:** This is an interface that enables the user of the system to view the system response through a GUI (Graphical User
Methodology and System Design

Interface). All possible decisions of the system have been stored in a database external to the system and this is for code efficiency.

**Exit System**: The user of the system can decide when to leave the application in the event of getting enough information or otherwise.

### 7.6. Sequence Diagram of the Proposed System

A Sequence diagram represents the graphical visualization of sequences of communication happening between object modules i.e. sequence of method invocation of object modules which results in accomplishing some tasks. The importance in a sequence diagram is on the sequence of communications. It is a structured representation of behaviour as a series of sequential steps over time during the process of a request and the respective response. It is used to represent work flow, message passing and how the system cooperates over time to achieve a result. The sequence diagram for this system is given in the following Figure 7.3.
7.7. Building of the System

During the course of the system development, knowledge base development is the most important part. Knowledge base defines the quality of the expert system. In our system, knowledge base development is taken care with the help of domain specific expert with JESS. In order to build a comprehensive expert system, the following four steps were recommended for our system development [75].
7.7.1. **Steps**

i. Identify the problem and detect if there is a need for the application. Determine if expert systems are a viable solution to the problem.

ii. Select the expert system shell JESS according to the flexibility, power, and speed of execution needed and the ability of the developer himself.

iii. Create a prototype. Demonstrate the credibility and feasibility of the prototype.

iv. Refine and extend the prototype. Customise the user interface.

7.8. **Knowledge Base Development**

The idea behind building an expert system is that it can enable many people to benefit from the knowledge of one person - the expert. An expert system simulates the judgment and behaviour of a human that has expert knowledge and experience in a particular field. Knowledge Base development is the most important part in the expert system development. The quality of the expert system is determined by its knowledge base. Knowledge Base development to build up this expert system with the help of domain specific expert is developed with JESS.
7.8.1. Modelling Of the Knowledge Base

In order to implement the rules for modelling the knowledge base, we are using the Java Expert System Shell (JESS). JESS is a rule-based language for specifying expert systems which can be tightly coupled to code written in the powerful, portable Java language. Jess is architecturally a production system executing a rule-based program and is a scripting language written entirely in Sun Microsystems’s Java Language. Jess language is therefore a declarative language. A JESS document is commonly initiated in text editor, as well as the windows platform editor, Notepad. Although it can work alone and could be run on the windows command prompt, it is usually incorporated into a Java program for functionality. A Jess file is normally saved with a “.clp” extension in opposition to the normal “.txt” extension. It comprises a JAR file which links the JESS to the Java IDE environment. As soon as the Jess is referenced to the code, it would run predefined instructions subject to user’s input from the Java Interface. The JESS is typically run and manipulated on the Java interface. Program codes are usually written for specific functions in Java environment.

7.9. The Database

An important objective in database design is to develop an efficient database structure so that data can be stored, accessed, and
modified easily. Much of the work in creating an effective database is
in the modelling. It is the application domain that determines how the
database should be modelled in order to be successful. OO structure
allows each disease and each symptom to be constructed as a
different object, and the database modelled as a collection of these
objects. This structure gives more flexibility to each object to have
whatever features (i.e. attributes or fields) required to identify it
while maintaining the integrity of the whole system. The database
consists of the main classes: Diseases and Symptoms. Each disease
object in the database includes the following data: DiseaseID,
Disease_Name. Each symptom object includes the following fields:
SymptomID, Symptom_Name and picture related to the symptom.
Note that some data listed above are known and saved in the
database and some data are inferred by the ES.

7.10. The DFD of the System

The inference engine of the developed expert system is
responsible for executing rules in the knowledge base. The rules from
the knowledge-base is retrieved by the inference engine and then
converts the rules to Abstract Sequence Trees (ASTs), and then
provides them to its rule interpreter for execution. The Inference
Engine interpreter traverses the AST, executing actions specified in
the rule along the way. This process is depicted in Figure 7.4.
7.11. Technical Method of the System Development

The technical methods of the developed system involved the design and implementation of components namely the User component, User interface component, Application component (JESS) and Database component (SQL) as shown in fig.7.5. The Graphical User Interface (GUI) component of the system is implemented using the Java Programming language. We have used the Java Expert System Shell (JESS) and the Java IDE of Netbeans 7.0 to implement the Application component and the implementation of database component is using SQL.
The expert system for rice plant disease diagnosis is developed using Netbeans 7.0; JESS (Java Expert System Shell) for the rule/knowledge base and SQL was used as the Database engine. The JESS file is called in the Netbeans environment and the Database also. When JESS rules are executed in JAVA, JESS library files are also to be loaded into the class path of Java. In JESS two JAR files (JESS.jar, jsr94.jar) are present. These two jar files should be included in order to execute a Java file which has JESS commands embedded in it. Instance of JESS Rule engine will be created in Java code and instance of JESS can then be reused. JESS Rule Engine is having Special API to execute rules.

For creating an instance of JESS rule engine,

```
engine = new Rete();
engine.reset();
engine.batch("agri.clp");
```

---

Here, Rete engine = new Rete() is an instance of Rete engine by which JESS is embedded with Java and engine.batch("agri.clp"); is to load the diagnosis rules. After implementing JESS in Java, the symptom data is loaded in the knowledge database.

database = aDatabase;
engine.addAll(database.getSymptoms());

After completing the above processes, the run command is given and then all the rules matching the facts will be fired.

engine.resetToMark(marker);
engine.run();

Here, engine.resetToMark(marker); is to remove all the previous data and engine.run(); is to fire the rules that apply to the given symptoms.

The developed expert system provides a simple way inputting the symptoms and predisposing factors based on the user's description. In addition to the above inputs, the picture of the particular symptoms are also integrated to get the details about the particular disease and to provide the treatment advice from the system. This system allows a user to browse the cases that are similar to the requested case and to compare the symptoms of the requested case with the stored ones. Then the user will choose the
most similar case and browse its information to emphasize his/her choices. The system will process user’s choice and make a suitable decision.

**Figure 7.6: Main Menu of Rice Plant Disease Diagnosis Expert System**

The above Figure 7.6 shows the main menu of the rice plant disease diagnostic expert system which has these menu options;

- **File**
  - **Agent**: The causing agents of the diseases are stored.
  - **Disease**: the diseases occurred in the rice plant in their life span are stored.
  - **Symptom**: All the possible symptoms of the diseases are stored.
  - **Exit**: To exit the system.

- **Diagnosis**: the diagnostic tool for the rice plant diseases is stored.

- **Help**: Help Contents about the system.
Figure 7.7: First Stage Diagnosis of Rice Plant Disease

The first stage of diagnosis for the rice plant diseases is shown in the above Figure 7.7. In this stage, depending on the selection of symptom by user the system provides related picture of the plant having that disease. After the confirmation of the disease the user clicks on the “search” button, the JESS platform performs the necessary knowledge evaluation to determine the expected result and displays the suggestion. The suggested output is shown in the next Figure 7.8.
Figure 7.8: Result of First Stage Diagnosis

The result of the first stage diagnosis of the rice plant disease diagnosis in the above figure is shown in the Figure 7.8. The result includes the disease corresponding to the selected input and related picture. Here, in this Figure 7.8, the result shows the brown spot disease and the causing agent of the disease. If there is more than one disease related to the same symptoms and input in the result of the first stage diagnosis, we can go for further diagnosis of the disease by supplying the complete details of the symptoms and the related parameters of the symptom by clicking on the "search by description" button. Also, we can get the details of the disease by
selecting one of the rows in the generated output and click the "disease detail" button. By selecting the generated row of the result of the above diagnosis and clicking the "disease detail" button, we will get the full details of the brown spot disease corresponding to the selected symptom. The selection of the generated row of the result and clicking the "disease detail" button is shown in the Figure 7.9.

![Figure 7.9: Evaluation of the Disease Detail](image-url)
Figure 7.10: Disease Detail and Control Measures after Diagnosis

Figure 7.10 depicts the result of the diagnosis which includes the disease details like disease name, symptom details, predisposing factors, casual organisms, the picture of the disease and the required control measures of the disease. Here, the Figure 7.10 also shows the disease details of brown spot disease of the rice plant and the photograph of the brown spot disease.
Figure 7.11: Selection of a symptom and its corresponding picture

The above Figure 7.11 shows the selection of the symptom “Lesions turn brown to greyish white then dry” and the related picture of the symptom. After selection of this symptom, the “search” button is clicked and the generated output is shown in the Figure 7.12.
Figure 7.12: Result generated by the above diagnosis

The result of the diagnosis in the above figure is shown in the Figure 7.13. In this output, the two diseases are found against the selected symptom and the related picture. The first disease is the "Narrow Brown Spot" disease which is caused the fungi and the second disease is the "Bacterial Leaf Streak" which is caused by the bacteria. By selecting the second row and go to the "disease detail" button, we will get the complete details of the "Bacterial Leaf Streak" disease and which is shown in the Figure 7.13. And also we can get the complete diagnosis of the disease by clicking the "Search by Description" button and then by supplying the complete details of the...
symptom and related parameter. The process is shown in the Figure 7.15.

Figure 7.13: Selection of the disease to get the details of the disease

Figure 7.13 shows the selection of “Bacterial Leaf Streak” disease generated in the Figure 7.12 and go for the complete details of the “Bacterial Leaf Streak” disease by selecting the “disease details” button.
Figure 7.14: Disease Details of Bacterial Leaf Streak

Figure 7.14 shows the disease details of the "Bacterial Leaf Streak" disease which is generated from the above process in the Figure 7.13.
Figure 7.15: Diagnosis of the Disease with various Input Factors

From the result generated shown in the Figure 7.12, we have selected the “search by description” button. By clicking this button, we are going for the complete diagnosis of the disease by supplying the complete symptom details and the related parameters which is shown in the Figure 7.15.
Figure 7.16: Disease Details of Narrow Brown Spot Disease after Diagnosis

Figure 7.16 depicts the result of the diagnosis which includes the disease details like disease name, symptom details, predisposing factors, casual organisms, the picture of the disease and the required control measures of the disease. Here, the Figure 7.16 also shows the disease details of narrow brown spot disease of the rice plant and the photograph of disease.