Detection and Identification of Bacterial and Parasitic Pathogens in Fish Husbandry using Web-based Expert System

GRADUATE PROJECT TECHNICAL REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Lakshmi Pinnenti
Spring 2006

Committee Members

Dr. Mario Al. Garcia
Committee Chairperson

Dr. LongZhuang Li
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

Proper disease diagnosis is essential for understanding infectious diseases in fish husbandry. Unfortunately, many of these methods are poorly understood or not utilized by aquaculturists often because of the lack of contact with epidemiologists / veterinarian who are willing to investigate fish diseases. Nowadays expert systems are becoming popular in disease diagnosis in humans, animals and in agriculture. Web-based Detection and Identification of Diseases in Fish Husbandry using Java Expert Shell System (WDDIFHJIES) was implemented to prevent fish loss. The moribund fish is evaluated morphologically for external symptoms of the disease and later subjected to microbial and biochemical tests. The results obtained from the observation will be sent to the expert system, where they are converted into facts using the constructs provided in JESS. These facts are then matched against the rules contained in the knowledge base expert system. If any of these rules match against the facts then the rule is fired and the corresponding action will be taken to mitigate the crop loss.
# TABLE OF CONTENTS

Abstract ......................................................................................................................... i

Table of Contents ......................................................................................................... iii

List of Figures ............................................................................................................... vi

1. Background and Rationale .................................................................................... 1
   1.1 Introduction ........................................................................................................ 1
   1.2 Importance of Disease Diagnosis in Fishes ...................................................... 2
   1.3 Previous Expert Systems .................................................................................. 3
   1.4 New Web Based Expert System Design ......................................................... 3

2. Narrative ................................................................................................................ 5
   2.1 A Simple Expert System .................................................................................. 5
      2.1.1 User Interface .......................................................................................... 7
      2.1.2 Knowledge Base ..................................................................................... 7
      2.1.3 Knowledge Acquisition ......................................................................... 7
      2.1.4 Inference Mechanism ............................................................................ 8
   2.2 Rule Based Expert System ............................................................................... 8
      2.2.1 Rule Base ............................................................................................... 10
      2.2.2 Working Memory .................................................................................. 10
      2.2.3 Inference Engine ................................................................................... 10
      2.2.4 Execution Engine .................................................................................. 11
   2.3 Domain Background ....................................................................................... 11
      2.3.1 Pond Inspection ...................................................................................... 12
      2.3.2 Inspection of Internal Anatomy of a clinically diseased fish ................. 12
LIST OF FIGURES

Figure 2.1: A simple Expert System ................................................................. 6
Figure 2.2: Forward and Backward Chaining .................................................... 8
Figure 2.3: A Typical Rule Based System ......................................................... 9
Figure 2.4: Development of Rule Based System .............................................. 14
Figure 3.1: Eclipse Work Bench ..................................................................... 19
Figure 3.2: Layers of JDBC Architecture ...................................................... 21
Figure 3.3: Architecture of WDIDFHJES ....................................................... 22
Figure 3.4: Model-View-Controller pattern for Web Application .................... 26
Figure 3.5: User Interface of WDIDFHJES ...................................................... 27
Figure 3.6: LOGIN table in Data perspective ................................................. 28
Figure 3.7: FEEDBACK with data in Data perspective ................................. 29
Figure 4.1: Output of WDIDFHJES ............................................................... 35
1. BACKGROUND AND RATIONALE

1.1 Introduction

Fish inhabit an alien world, and their exploitation for food or for sport has, until recently, been virtually dependent on the hunting the wild stocks. It's only over the last twenty years that culture methods for few species have evolved. There has been an unprecedented growth in commercial farming of the fishes under intensive stocking densities. This has been manifested throughout the world, and blue revolution, no less significant than the green revolution which led to self-sufficiency in rice and grain for much of the developing nations, is now underway.

Aquaculture is growing more rapidly than any other animal food producing sector (10-15% versus 1-3%). Aquaculture production, excluding aquatic plants, reached 51.4 million tons by quantity (US$ 60.0 billion by value) in 2002, from 35.5 million tons (US$ 52.1 billion) in 2000. Its contribution to global supplies of fish, crustaceans and molluscus increased from 3.9 percent of total production by weight in 1970 to 27.3 percent in 2000, according to FAO's State of World Fisheries and Aquaculture 2002 report [SOFIA 2002]. For millions of people around the world, fish is a dietary mainstay (protein source). FAO figures for 2000 show that fish provided around 19 percent of total animal protein supplies in Africa, 21 percent in China and 23 percent in Asia. At the country level, the profile of fish in meeting nutritional needs can be even higher.

Fish pathology is one of the main scientific bases upon which this expansion in aquaculture industry has been dependent, and our knowledge of the subject has expanded remarkably over the past decade. This was attributed due to the technological breakthroughs that were achieved by the experts and researchers in the field of pathology and
molecular biology. Diagnosis, identification and taking proper preventive steps required considerable expertise in and in biology, physiology, microbiology and immunology. Understanding the disease causing agent and the etiology of the disease requires wide knowledge of environmental constraints of aqueous, exothermic existence, the physiology of survival and growth in such conditions, the epizootiology and characteristics of various pathogens that cause infections, the response of the host, and the methods by which they may be controlled.

1.2 Importance of Disease Diagnosis:

Most of the crop loss around the world (>80%) was attributed due to the disease break out that occurred due to lack of proper disease diagnosis and identification at the early stage and the failure to take necessary preventive methods to overcome the disease. With the increase in investment in aquaculture and closer examination of factors that contribute to the risks faced by the aquaculturist, the concept of integrated health prevention measures had developed in recent years. Improved expertise and facilities in disease diagnosis have led to the identification of several unknown pathogens and disease conditions [Hedricks 1998]. Consequently, greater efforts are now being made to diagnose and control disease conditions with the use of several new diagnostic tools that utilize the biotechnological and microbial procedures [Thorburn 1999]. The high cost and limited trained man-power to handle the disease diagnosis has crippled the industry as many simple, identifiable and curable disease need to rely on human-expert as there is no proper alternative easy to use diagnostic tool.
1.3 Previous Expert Systems:

The rapid development of cost effective internet application are revolutionizing information technology and creating unprecedented opportunities for developing traditional decision support systems and expert systems [Power, 2000]. A number of expert systems are reported in the literature [Potter et al., 2000] but most of them are prototyping systems. It seems that research and development of Web-based expert systems are still in their early stages. Most of the expert systems we currently have were built for few selected fish species like tilapia and salmon and for pest control management [Wilhoit 1991]. The main disadvantages of these systems are being stand alone systems. Hence a user or an organization needs to have the expert system on each of the system. This will be both cost prohibitive and difficult to maintain as all the systems need to be installed with the require software and any changes made in the application needs to be updated in all the systems.

1.4 New Web Based Expert System Design:

The main advantage of Web based expert system is the universal accessibility and worldwide availability of centralized applications and data. The only software needed to access and execute the applications is a web browser. This satisfies the real basic need for the designing of an expert system that is availability of expert at all time in the form of a Web-based expert system.

This project work was aimed at making a more exhaustive rule based expert system which can identify most of the bacterial and parasitic diseases of fishes. The out
come of this project would be a disease diagnostic tool, which is easy and cost effective for fish farmers in the developing nations
2. NARRATIVE

The project deals with the implementation of a Web based expert system for the detection and identification of bacterial and parasitic pathogens in fish husbandry. This project is aimed at developing a rapid and cost efficient disease diagnostic tool for fish farmers in the developing nations. This web-based intelligent system can mimic human fish disease expertise and diagnose a number of fish diseases with a user-friendly interface. It contains a large number of fish diseases as rules, which are used to conduct online disease diagnosis. The system has been tested with farmers in South India under the supervision of an extension worker from a private organization.

2.1 A simple expert system:

An expert system is a computer based system that simulates the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field. Typically, such a system contains a knowledge base containing accumulated experiences from a domain expert or through expert’s publications in literatures. Most expert systems have certain basic components: a user interface, a knowledge base, and an inference mechanism as shown in figure below. [Martin 1988]
Figure 2.1: A Simple Expert System [Martin 1988]
2.1.1 User Interface:

A user interface is a set of commands or menus through which a user communicates with a computer system. A command-driven interface is one in which user enters commands. A menu-driven interface is one in which the user select command choices from various menus displayed on the screen. The user interface is one of the most critical parts of any system because it determines how easily a user can make the program do what was intended to do. The expert system responds accordingly to the users input. The expert system even accepts and learns new knowledge from the user. [Martin 1988]

2.1.2 Knowledge Base:

The expert system is also known as knowledge base system as it holds the knowledge about the domain of the expert system. [Martin 1988]. In a rule-based expert system, a knowledge base typically includes definitions of attributes and rules along with control information. Knowledge base format is specific for a given expert system shell or other software in an expert system.

2.1.3 Knowledge Acquisition:

The process of obtaining the appropriate information from an expert and then presenting it in a manageable form is called Knowledge Acquisition. This is typically done by organizing the knowledge into IF-THEN rules or some other form of knowledge representation. Knowledge systems have traditionally been constructed by knowledge engineers, who are people interviewing domain experts and formalizing their knowledge. [Martin 1988].
2.1.4 Inference Engine:

This is the software component in an expert system that provides the reasoning mechanism. In rule based expert system, this will be implemented either as forward chaining or as backward chaining as shown in figure 2.3.

![Forward Chaining Diagram]

```
If a = 1 && b = 2 then
   c = 3
If c = 3 then
d = 4
```

![Backward Chaining Diagram]

Figure 2.2: Forward and Backward Chaining

2.2 Rule Based Expert System:

Expert systems whose knowledge is represented in rule form are called rule-based systems. The expert system works by parsing the data and applying the rules accordingly. A typical rule based expert system consists of inference engine, rule base, working memory. The inference engine in turn consists of a pattern matcher, an agenda and an execution engine. These components are show in the figure 2.3. [Friedman 2003]
Figure 2.3: A Typical Rule Based System [Friedman 2003]
2.2.1 Rule Base:

The rule base stores all the rules that are required for the expert system to work in the particular domain in which the expert system is being built. The rules can be stored as plain text or strings but for efficient processing they are converted into some form by the rule compiler. In addition, the rule compiler may reorder the premises or conclusions of a rule, either to make it more efficient or to clarify its meaning for automatic execution. Depending on the particular rule engine, these changes may be invisible to the programmer. [Friedman 2003]

2.2.2 Working Memory:

Every problem in a domain has some unique data associated with it. This data may consist of the set of conditions leading up to the problem and are called facts base. This is the only part of an expert system that is volatile, or subject to rapid changes. This data, specific to the problem needs to be input by the user at the time of consulting the expert system. The working memory holds all the facts that the rule base system is working on. It holds all the current facts the system has concluded up to that point. Typically, the rule engine maintains one or more indexes, similar to those used in relational databases, to make searching the working memory a very fast operation. [Friedman 2003]

2.2.3 Inference Engine:

The inference engine is a critical component that controls the whole process of applying the rules to the working memory to obtain the outputs of the system. [Friedman 2003] The rules in rule base are compared with the facts in the working memory using
pattern matcher as shown in the above figure 2.3. The activated rules along with the previous rules that got activated are ordered to form an agenda. This process of ordering the rules is called conflict resolution. The first rule of the agenda will be fired and this process will continue. Since this process keeps repeating, rule engines use advance techniques to avoid most of the redundancy. In particular, the results from the pattern matcher and the conflict resolution will be saved across the cycles. This will result in enhanced performance of the rule engine as only new work needs to be processed. [Friedman 2003]

2.2.4 Execution Engine:

The execution engine in rule based expert system is the component that is responsible for executing the fired rule’s action part. Advanced and latest rule engines offer a complete programming language that one can be used to define the action when a given rule is fired. [Friedman 2003]

2.3 Domain Background:

The success of an expert or veterinarian in determining the status of the pathogen depends upon proper sampling and testing procedures. The following are the common and most recommended procedures for disease diagnosis.

- Pond Inspection
- Inspection of Internal Anatomy of a Clinically Diseased Fish
- Biochemical Tests
- Disease treatment and Preventive Measures
2.3.1 Pond Inspection:

This process will aid in observing the culture conditions, water quality and also fish behavior. This type of inspection includes collecting the data about the farm size, type of farming (extensive, semi-intensive and intensive), type of the fish, age, water quality (like dissolved oxygen, transparency, ammonia), and fish behavior. Change in fish behavior includes erratic swimming, poor feeding, and excess mucus secretion. All of these indicate the need for further investigation.

2.3.2 Inspection of Internal Anatomy of a Clinically Diseased Fish:

This is the primary and most critical part in diagnosis. Affected fish can be examined for external lesion near the fins and tail, dislocation of the scales on the surface, white patches on the gill, swollen eyes and other external deformities. Later the fish sample should be taken and slightly anaesthetized to see the internal anatomy especially with reference to stomach, viscera and muscle tissue. In most cases with the information that is obtained from the above two procedures, it is possible to determine the cause of the disease. If further diagnosis is required to determine the cause of disease, then fish sample needs to be analyzed for biochemical and bacteriological test.

2.3.3 Biochemical Test:

In case of bacterial infection an accurate and through diagnosis of the disease and its cause is sometimes impossible without a biochemical test. Straining techniques like Grams strain and other confirmatory tests need to done before coming to conclusion about the causative agent.
2.3.4 Disease Treatment and Preventive Measures:

After following all the above procedures an expert can come to a conclusion about the cause of disease and suggest the appropriate type of treatment based on other prevailing farm conditions like stage of the crop, season, and age of the fish.

When considering developing a web-based expert system for fish disease diagnosis, it is vital to have a thorough investigation of the domain problems. In this case, problems inherited in disease diagnostic processes need to be considered and methods need to be explored on how to deal with them in a computer based system.

2.4 WDIDFHJES as a rule based expert system:

The WDIDFHJES was developed like a typical rule based expert system. The development of WDIDFHJES application comprises of the following steps as shown in figure 2.4.

- Knowledge Engineering
- Structuring data
- Testing
- Interface building
- Writing the rules
- Iterative Development
Figure 2.4: Development of a Rule Based System [Friedman 2003]
2.4.1 Knowledge Engineering:

Fish disease experts with many years of experience have developed a body of knowledge, which they can use to make correct diagnosis. Many techniques have been developed for knowledge acquisition (KA). Some commonly used approaches are interviews, observations, books, journals and articles. KA is seen as a crucial problem concerning the success of an expert system and has always been regarded as the bottleneck in developing any expert system [Hart, 1989]. This bottleneck is mainly caused by communication difficulties between the knowledge engineer (KE) and the expert, the inability of the expert to describe expertise, and the inability of the KE to obtain expertise. In Knowledge engineering for WDIDFHJES all the data necessary for the successful implementation of WDIDFHJES was collected. The different disease symptoms like external morphology, internal anatomy and microbiological test of moribund fish of various bacterial and parasites was collected from various peer review journals and books.

2.4.2 Structuring Data:

In this phase the knowledge collected during the previous phase of WDIDFHJES was examined and proper data structures were developed using defrule construct of JESS to implement the rules clearly and directly. This process resembles the object-oriented analysis phase of the object oriented software engineering process. [Friedman 2003]

2.4.3 Testing:

An application can not be robust, modular and scaleable unless it is tested at different phases of the development. Testing in WDIDFHJES was done on most of the
individual rules to make sure that they collectively run without any bugs or errors. In case of the java classes that were developed for GUI interface of WDIDFHJES, test cases were developed with different input and each java class was unit tested using JUnit testing.

JUnit framework provides a simple way to explicitly test specific areas of a Java program. It is extensible and can be employed to test a hierarchy of program code either singularly or as multiple units. Using a testing framework is beneficial because framework forces to explicitly declare the expected results of specific program execution routes. When debugging it is possible to write a test case which expresses the result one is trying to achieve and then debug until the test is passed. By having a set of test cases that test all the core components of the project it is possible to modify specific areas of the project and immediately see the effect the modifications have on the other areas. JUnit promotes the idea of first testing then coding, in that it is possible to setup test data for a unit which defines what the expected output is and then code until the tests pass.

JUnit 2005

2.4.4 Interface Building:

The usefulness of any rule base expert system depends a lot on the interface building. In WDIDFHJES the interface building describes how an expert system analyzes the symptoms and converts them into facts and compares those with the rules in the rule base before identifying the disease.
2.4.5 Writing the Rules:

After completing the above phase of development, WDIDFHJES has well defined and tested data structures and interfaces. The rules of WDIDFHJES will be developed based on our own rules that we obtain from studying various literatures from books and peer reviewed journals. In the initial phase the rule were written in plain English which made it much easier to convert then into rules that can be scripted into the expert system language used by WDIDFHJES. Since each rule is independent of the other, rules were developed iteratively. Care was taken to ensure that this way of writing rule will not cause any unstructured rule-based program by writing WDIDFHJES rules as modules. WDIDFHJES rules were divided into separate modules like environmental observation module, water quality monitoring module, internal anatomy observation module, bacteria diseases and biochemical test module and recommendations module.

2.4.6 Iterative Development:

WDIDFHJES was developed as flexible rule base expert system that can incorporate new rules into the rule base. This is very important because new diseases and pathogens tend to attack the fishes with increase in intensity of farming. A process of continuous feedback from farmers and experts was done using dynamic web page feed back with the help of Java Server Pages (JSP) and Java Servlets. This process not only helps the KE to represent the domain knowledge correctly, but also helps the expert to clarify the reasoning processes. All domain knowledge is analyzed and represented in developing new rules or refining the existing rules for better rules in the rule base.
3. SYSTEM DESIGN

3.1 System Requirements:

The rule based Web expert system for Detection and Identification of Diseases in Fish Husbandry was developed using Java Expert Shell System (WDIDFHJES). Jess is a rule engine and scripting language developed at Sandia National Laboratories in Livermore, California in the late 1990s. Since JESS was developed from Java 2 Standard Edition (J2SDK) it has several advantages compared to other expert systems. It has access to all the powerful Java APIs for graphics, database access, Java Beans, Networking and Remote Method Invocation (RMI) etc. It also has the advantage of developing industrial robust expert systems using Java 2 Enterprise Edition (J2EE) with Java Server Pages, Servlets and Enterprise Java Beans APIs. This project to a certain extent was implemented using those technologies with the help of open source Eclipse Workbench using Apache Tomcat plugins as shown in figure 3.1 and with IBM WebSphere Studio Application Developer version 5.0. Java 2 Runtime environment was used to run the JESS applet. WDIDFHJES application was implemented using Java Application server Apache Tomcat version 5.5, the official reference implementation for the Java Servlet and Java Server Pages technologies and IBM Websphere Application Server version 5.0.

3.1.1 IBM WebSphere Studio:

The WebSphere Studio product family is a collection of tools built on an open application development framework. The base, called the WebSphere Workbench, is a robust offering by IBM of the Eclipse framework [Eclipse 2005]. This framework allows
application development tools to be plugged in a very dynamic fashion. This allows the product to grow and evolve both with a development team and with changing technology. WebSphere Studio and Eclipse provide a role-based organization to the Integrated Development Environment (IDE). There are distinct perspectives organized around the specific tools and views need to perform a specific task or series of tasks. It is quite common that these perspectives are structured around the type of artifacts that are being developed or tested, such as Java code, Web assets, XML documents, data access, J2EE components, and debugging. Developers can easily adapt and create their own perspectives to reflect the tasks that they perform and the way in which they perform them.

![Eclipse Workbench](image-url)

**Figure 3.1:** Eclipse Workbench
3.1.2 Database Connectivity:

JDBC technology is an API that provides cross-DBMS connectivity to a wide range of SQL databases and access to other tabular data sources, such as spreadsheets or flat files [Sun Micro Systems 2005]. The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manager ensures that the correct driver is used to access each data source.

A JDBC driver translates standard JDBC calls into a network or database protocol or into a database library API calls that facilitates communication with the database. This translation layer provides JDBC applications with database independence. If the back-end database changes, only the JDBC driver need be replaced with few code modifications [Dustin 2001]. The location of the driver manager with respect to the JDBC drivers and the java application is shown in Figure 3.2
3.1.3 IBM Cloudscape:

IBM Cloudscape provides a fullfeatured, robust, database server that was developed from the open source Apache Derby database [IBM 2005]. IBM Cloudscape provides full transactional capability that is simple to deploy and reduces the cost of Web-based and embedded applications. Like many robust relational enterprise databases such as IBM's DB2 and Oracle, Cloudscape supports on-line backup and crash recovery
as well as advanced features like Unicode support/internationalization, encryption, and multiple low-overhead connections. It also supports for stored procedures, functions and triggers.

3.2 WDIDFHJES Architecture:

The WDIDFHJES architecture consists of several phases like requesting the user about the occurrence of the problem, the nature of the problem and the prevailing water quality conditions as show in the figure 3.3. This feedback will be sent to the expert system component of WDIDFHJES where it will be converted to facts by constructs provided in JESS. These facts are compared with the rules in rule base and the matching rules are then fired to take appropriate action.

![Diagram](image)

**Figure 3.3: Architecture of WDIDFHJES**
3.2.1 Writing Rules:

The rules for WDIDFHJES were written using the `defrule` construct of JESS. The content for the rules were obtained by reviewing literature in peer reviewed journals and from books on fish diseases. A total of nearly 300 rules that can identify most of the fish diseases in culture condition were stored in the rule base of WDIDFHJES. A simple rule construct in jess can be described as follows.

```lisp
(defrule fish4
  (condition 2)
  ?x <- (condition ?number)
  =>
  (printout t " Take all necessary steps like doing" crlf)
  (printout t "water exchange, applying sanitizer, limestone " crlf)
  (printout t "and reduced over feeding"crlf )
  (retract ?x))
```

This rule will not fire just because the function call would evaluate to true. Instead, Jess will try to find a fact in the working memory that looks like (Condition 2). Unless we have previously asserted such a fact, this rule will not be activated and will not be fired.

3.2.2 Writing Facts:

The facts of WDIDFHJES corresponding to different symptoms that the user will enter through GUI interface. The facts of the WDIDFHJES will determine which rules are to be fired and when. The GUI interface was developed in such a user friendly manner that user doesn’t need to know any of the JESS commands. JESS commands were invoked through buttons. This buttons in turn will run the JESS commands through
Action Listener Adapter classes in Java. A simple fact construct in jess can be described as follows:

```
(deffacts
   newpigmentation
   (the pigmentation is pale)
)
```

Here the fact, the new pigmentation is pale is added to the fact list and the appropriate rule corresponding to this fact will be fired by the expert system.

3.2.3 Matching Rules and Facts:

WDIDFHJES has a predetermined set of rules, representing different types of disease symptoms, while the facts, representing the symptoms of moribund fish, keep changing frequently with each test case. These rules and facts will be supplied to the JESS rule engine that performs pattern matching. JESS uses the fast and efficient Rete algorithm for pattern matching. The strength of Rete is that it uses a set of memories to retain information about the success or failure of pattern matches during previous cycles. The Rete algorithm involves building a network of pattern-matching nodes. JESS uses many different kinds of nodes to represent the many different kinds of pattern-matching activities. When a pattern is matched to a set of facts the rule corresponding to the facts is fired or executed. [Friedman 2003]

3.3 WDIDFHJES as Web Application:

The WDIDFHJES was implemented as fat-client enterprise application using Java Applet, JSPs and Servlets using Model-View-Controller Model-2 pattern. The purpose of the Model-View-Controller (MVC) pattern is to separate user interface logic from business logic. By doing this it is possible to reuse the business logic and prevent changes
in the interface from affecting the business logic. MVC pattern consist of the following three components

- Model
- View
- Controller

3.3.1 **Model:** The *Model* is responsible for keeping the data or the state of the application. It also manages the storage and retrieval of the data from the data source. It notifies all the *Views* that are viewing its data when the data changes.

3.3.2 **View:** The *View* contains the presentation logic. It displays the data contained in the Model to the users. It also allows the user to interact with the system and notifies the Controller of the user’s actions.

3.3.3 **Controller:** The *Controller* is the one that coordinates the *View* and the *Model*. It instantiates the *Model* and the *View* and associates the *View* with the *Model*. Depending on the application requirements, it may instantiate multiple *Views* and may associate them with the same *Model*. It listens for the user’s actions and manipulates the *Model* as dictated by the business rules. Figure 3.3 shows the relationship between the components of the MVC pattern.
Figure 3.4: Model-View-Controller pattern for Web Application

The View components of WDIDFHJES consist of an html form for user login and a JESS GUI interface that is embedded in a JSP. The dynamic web page that is generated from the JSP page allows the user to interact with WDIDFHJES application. Figure 3.5 shows the main user interface of the WDIDFHJES application.
The user request for interacting JESS and the database was delegated through Java servlets. These Java servlets act as controllers and delegate the requests to appropriate Model components like JESS engine and Java Beans.

The Model components of WDIDFHJES that implement business logic consist of Rete engine and reusable Java Bean components. Several Java Bean objects were created for user database connectivity, password encryption, automated email, form data and user information. Any change in the state of Model as result of user request is passed back to the user through View.
3.4 WDIDFHJES Databases:

The data about the users and feedback submitted by the users are stored in IBM Cloudspace in the form of tables. The tables that were used in WDIDFHJES application are described in the following section.

3.4.1 LOGINUSER Table:

This table is used to store the username and password of the farmers who want to use this application. Email field of the LOGINUSER table is selected as a primary key. When user enters the username and password in a html form, it sends the request to the server to check whether or not both values match in the LOGINUSER table. Figure 3.6 illustrates the LOGINUSER table that was used in WDIDFHJES application.

![Figure 3.6: LOGINUSER table in Data perspective](image_url)
3.4.2 FEEDBACK Table:
This table is used to store the feedback that farmers would like to submit about a disease problem that WDIDFHJES is unable to detect. The data from this database is later automatically submitted to an Expert and developer through email. This will facilitates in iterative development and in enhancing the rule base of WDIDFHJES application. Figure 3.7 illustrates the data that was collected into the FEEDBACK table in WDIDFHJES application.

![Image of the FEEDBACK Table]

**Figure 3.7:** FEEDBACK table with data in Data perspective
3.5 Security Issues of WDIDFHJES Application:

Web application development is very different from other environments. Web browser and the nature of HTTP pose security pitfalls not found in traditional client-server applications. Lack of special precautions over computer network between user’s browser and the server are subject to eavesdropping. Strong encryption must be used to protect sensitive data traveling over computer networks. The Secure Sockets Layer (SSL) protocol is a de facto standard for protecting web-based network traffic. The SSL protocol protects data from alteration and disclosure while it is in transit. It also gives users some assurance that they are communicating with the sites they think they are. There is also provision in the SSL standard for client certificates which allow the server to authenticate the end user.

WDIDFHJES was developed keeping in mind about the security risk that might occur as result of eavesdropping, password cracking and buffer overflow. WDIDFHJES was implemented with SSL over the HTTP protocol. Digital certificate associated with public and private keys were generated using the Java Security APIs and Java Keytool. In WDIDFHJES, a pair of public and private keys was generated and added to the keystore using the keytool -genkey command. The -genkey option creates a public/private key pair and then wraps the public key in a self-signed certificate.

The security vulnerability that occurs as result buffer overflow was prevented by developing the WDIDFHJES using Java programming language. All the user data is stored in a secured IBM Cloud Space database. The password that is supplied at login is encrypted and sent to the database by using Secure Hash algorithm-1 (SHA-1) using a Java Bean class.
4. EVALUATION AND RESULTS

4.1 Testing the WDIDFHJES Application

Several testing methods were used in evaluating the WDIDFHJES Web application. Testing of WDIDFHJES was done in two phases once during the development phase and the other during the post development phase. During the development phase care was taken while writing the rules by dividing the rules into modules. This prevented any unstructured rule based program. The following are different types of testing done measure to the performance of WDIDFHJES application.

4.1.1 Unit Testing:

Unit Testing of WDIDFHJES was done using JUnit testing. Eclipse v. 3.1 comes with JUnit built into the Workbench. This allows to quickly create test case classes and test suite classes to write test code. All the core java classes that were used in this project were tested using several test cases. By having a set of tests that test all the core components of the WDIDFHJES project it became possible to modify specific areas of the project and immediately see the effect the modifications have on the other areas. The test cases were built with the needed imports and extensions for JUnit to run.

4.1.2 Stress Tests

Modularity and scalability of the application can be known thorough stress test. WDIDFHJES was subjected to stress test by entering different disease symptoms as facts. WDIDFHJES did reasonably well with all the possible symptoms and it was able to diagnose most of the diseases effectively during this test phase. The test cases were
conducted in India with a help of an extension worker. The results of those test cases were show in the table 4.1

<table>
<thead>
<tr>
<th>Test Case No</th>
<th>Symptoms</th>
<th>Fish Species</th>
<th>Human Expert Diagnosis</th>
<th>WDIDFJHGES Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1. Water depth is 1-1.5m  2. Water temperature in the noon is 25-30 C  3. Water color is blue green  4. Sechi disc reading is less than 30 cm</td>
<td><em>Catla catla</em></td>
<td>High blooming of the pond due <em>Anabena, Nostoc, Oscillatoria</em> species  Suggestion: Exchange water 20 -30% for a week</td>
<td>Might be due to algal blooms of <em>Anabena, Oscillatoria</em> species. Suggestion: Algal Blooms are also responsible for consuming much of the oxygen. Do 20 -30% water exchange during sunny days and reduce the feed to 20-30%</td>
</tr>
<tr>
<td>2</td>
<td>1. Water Quality under optimal condition  2. Fish is fresh water fish  3. Fishes are having chronic mortality  4. Frenzied tail chases behavior particularly when being fed.  5. Posterior and tail region of the fish fingerlings turn dark  6. Mishappen skulls and twisted spines</td>
<td><em>Labeo rohita</em></td>
<td>The disease is caused by a sporozoan parasite <em>Myxosomer cerebralis</em></td>
<td>The disease is caused by a sporozoan parasite <em>Myxosomer cerebralis</em></td>
</tr>
<tr>
<td>3</td>
<td>1. Water Quality under optimal condition  2. Fish is fresh water fish  3. Mortality is acute Fish look dark and swollen  4. Severe necropsy and</td>
<td><em>Labeo rohita</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hemorrhagic viscera</td>
<td>The bacteria is <em>Flexibacter columnaris</em> and the diseases is Columnaris disease</td>
<td>The bacteria is <em>Flexibacter columnaris</em> and the diseases is Columnaris disease</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Water depth is less than 1 m</td>
<td>2. Water temperature in the noon is greater than 30 °C</td>
<td><em>Catla catla</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The pond has shallow water depth. Pump water in to the pond to maintain a level of 1-1.5 m</td>
<td>This high temperature is due to rapid heating of water due to shallow water depth. Increase water depth to min of 1 meter to minimize diurnal fluctuations. To prevent aquatic weed infestations and other management problems, avoid shallow water ponds with less than 1 meter</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1.3 Integration Testing:

Integration testing is a coherent extension of unit testing. Several units that have already been tested are combined into a component and the interface between them is tested. Integration testing is one of the most vital testing phases in the software development process as it identifies problems that occur when units are combined. WDIDFJHES system was subjected to integration testing when it was running on Apache
Tomcat server as well as on IBM Websphere Application server. Different disease symptoms were manually entered on `expertSystem.jsp` and the rules that got fired were observed. This test was done several times by restarting the server to ensure that the whole application works well repeatedly without any problem.

4.2 Executing WDIDFHJES Application

The execution and running of the WDIDFHJES application was designed in a user friendly way keeping in mind that the users will be mostly farmers who have few or no knowledge about JESS and Java. This application can be executed by just clicking the right mouse button. WDIDFHJES application consists of a login screen where the user enters a valid name. A dynamic Web page is generated with a greeting message and the expert system is ready for the user input. The file `expertSystem.jsp` has an applet embedded in it and invoking the right button will start the application. When the user clicks the ‘Fish Diagnosis’ button that will automatically load the built in `Disease.clp` file. This file contains all the rules that represent different fish diseases. WDIDFHJES waits for the user input in the form of choices and the user need to enter a valid choice using the radio buttons. The user input is stored internally and the facts that user gave as inputs will be matched with the rules. Once WDIDFHJES has enough facts to conclude about the right disease it will display on the applet text area as shown in the figure 4.1, the identified diseases, treatment and preventive measure. Since expert systems can’t solve all the problems in a given domain it would be a good design consideration to have the user communicating with an expert. WDIDFHJES application has a feedback form in the file `feedback.jsp` where the user can post about the problems of the domain and the performance of the WDIDFHJES application. This will aid in iterative development of
the application by adding new rules into the rule base of WDIDFHJES from the user input. This is very important as new diseases and symptoms will be observed with increasing intensity of farming.

Figure 4.1: Output of WDIDFHJES
5. FUTURE WORK

The WDIDFHJES application has some scope for enhancement in the future. The following are the some of the ideas that can be incorporated into this project in future.

5.1 Enhanced Knowledge-base

The diagnosis of fish disease by WDIDFHJES application depends a lot on the built in rules that were created through knowledge engineering. It would have been better if we have included some other experts besides our own knowledge that we gain through experience. This would have refined our existing rules and also add new rules to our rule base because there would be some intuitive diagnostic skills that experts might have gained through experience which are hard to find in books.

5.2 Developing a Thin-Based Client

The WDIDFHJES was implemented as a fat client web based application to have the advantages of less processing power on the server, and the maximum control over the user interfaces as we can develop the application using all the Java APIs. Nevertheless some issue need to addressed in future with regard to downloading the client software due to poor network speed and frequent updates on the client side software etc. In future this application needs to be modified as thin client web application if we are looking for a more distributive Web based expert system with several multimedia images incorporated in the database for better user interface.
5.3 WDIDFHJES as Web Services:

The WDIDFHJES can be implemented as a web service application so that it can be a component of an integrated animal health expert system or as component of some other large software that were aimed at improving the fish husbandry.
6. CONCLUSION

WDIDFHJES project was an effort in implementing the first Web based disease diagnostic tool for fish farmers using a rule based expert system. It was able to eliminate the requirement of an expert by efficiently diagnosing the most common fish diseases that were found in fish husbandry. With several tests conducted in south India with the help of an extension worker it was proved to be a reliable fish disease diagnostic tool. Since this application was developed using JESS and J2EE technologies scalability and other performance issue will be effectively addressed in future that comes with enhanced rules, database interaction, multimedia user interface, and automated KE mechanism.
ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my project committee chairperson Dr. Mario Garcia, for his guidance, expertise in the area of Expert Systems, support, and valuable insights all along the project, right from the proposal phase. I also thank the other committee members Dr. David Thomas and Dr. Long-zhuang Li for their encouragement and valuable feedback during various stages of the project.
6. BIBLIOGRAPHY AND REFERENCES


APPENDIX A: EXPERT SYSTEM ON WEB

This section describes the process that enables an expert system running on the web using Java applet. To achieve this process a browser needs to have Java 2 Runtime Environment. The following are the some of the important class that needs to be created for user interface using java awt and swing package.

**ConsoleApplet:** A simple Applet that has *ConsolePanel, Rete, Thread* and *Main* class as instance variables. This applet creates the necessary GUI with the help of *ConsolePanel* that severs as the basic for interview style experts systems. *Rete* class is the one that does the reasoning and is the central class of Jess Library. The instance variable *Thread* class is used to run the parser loop. *Main* class is used to drive the *Rete* class.

**ConsolePanel:** This is a basic question-and-answer dialog graphic user interface (GUI). The *ConsoleApplet* class has an instance variable of this class. This class is a Panel containing input and output text areas for a *Rete* to use. It uses the *TextReader, TextAreaWriter* and *Button* classes to turn textual data or into I/O streams.

**TextAreaWriter:** A simple Writer, suitable for constructing a *PrintWriter*, which uses a *TextArea* as its output. This class is a convenient way to write a GUI in which Jess prints its output to a text widget.

**Rete:** This is the reasoning engine and the central class in the Jess library. It executes the built Rete network, and coordinates many other activities. Rete is basically a facade for all the other classes in the Jess.

To run the JESS on the web we need to combine all the above classes beside the library class in the JESS into an archive file that bundles the applet’s files into a single jar file. This can be achieved by running the following command
jar cf jess *.class.

In the browser insert the applet class ConsoleApplet along with the jar file using the archive attribute <APPLET> tag. The code that implements this is shown below.

<applet archive=jess.jar code=jess.ConsoleApplet height=700 width=750>
</applet>
APPENDIX B: FISH DISEASE RULES

The following project requests the user to select from the menu's
and will arrive at a conclusion and recommendations.

Project By: Lakshmi Pinneti and Dr. Mario Garcia
Date: 05/02/05

(de rule fish1
  (the color is ?color1)
  => (printout t crlf crlf crlf)
  (printout t crlf
   "***********************************************************************
   crlf crlf)
  (printout t "This project is to diagnosis the disease in fish and give the " crlf
  (printout t "appropriate recommendation to the disease" crlf crlf)
  (printout t "This project is developed by: Lakshmi Pinneti and Dr. Mario Garcia" crlf)
  (printout t crlf
   "***********************************************************************
   crlf crlf)
  (printout t "Did you check the Physio-Chemical (Water Quality) Parameters" crlf)
  (printout t "1. Yes" crlf)
  (printout t "2. No" crlf)
  (printout t "Enter Your Choice:")
  (assert (quality(read)))
)

(de rule fish2
  (quality 2)
  ?x <- (quality ?number)
  => (printout t crlf
   "***************************************************************************
   crlf crlf)
  (printout t "Go and Check the Water Quality Parameters and run the Water Quality
Diagnostic tool" crlf)
  (printout t "before you use this Disease diagnostic tool." crlf)
  (printout t crlf
   "***************************************************************************
   crlf crlf)
  (retract ?x)
)

44
(defrule fish3
  (quality 1)
  (?x <- (quality ?number))
  => (printout t crlf
  "******************************************************************************
  (printout t "Do the water parameters are under optimal condition" crlf)
  (printout t "that is recommended for your culture species: " crlf)
  ( printout t "1. Yes" crlf)
  ( printout t "2. No" crlf)
  ( printout t "Enter Your Choice:"
  ( assert ( condition(read)))
  (retract ?x)
  )

******************************************************************************

(defrule fish4
  ( condition 2)
  (?x <- (condition ?number))
  => (printout t crlf
  "******************************************************************************
  (printout t " Take all necessary steps that were mentioned in the other " crlf)
  (printout t "diagnostic tool (Water Quality diagnostic tool like doing" crlf)
  (printout t " water exchange, applying sanitizer, limestone " crlf)
  (printout t " and reducing over feed etc." crlf)
  (printout t " Now we go ahead and analyse your animal condition" crlf)
  (printout t " Choose choice one to continue... " crlf)
  ( assert ( type(read)))
  (retract ?x)
  )

******************************************************************************

(defrule fish5
  ( condition 1)
  (?x <- (condition ?number))
  => (printout t crlf "******************************************************************************
  (printout t "What kind of fish are you culturing" crlf)
  ( printout t "1. Warm Water Fish" crlf)
  ( printout t "2. Cold Water Fish" crlf)
  ( printout t "Enter Your Choice:"
  ( assert ( type(read)))

45
(retract ?x)
)

(defrule fish6
 ( type 1)
 ?x <- (type ?number)
 => (printout t crlf "******************************" crlf crlf)
 ( printout t " What is the water environment of the Fish"crlf )
 ( printout t " 1. Marine Water "crlf )
 ( printout t " 2. Fresh Water "crlf )
 ( printout t "Enter Your Choice:" )
 ( assert ( host(read)))
(retract ?x)
)

(defrule fish6a
 ( type 2)
 ?x <- (type ?number)
 => (printout t crlf "******************************" crlf crlf)
 ( printout t " What is the water environment of the Fish"crlf )
 ( printout t " 1. Marine Water "crlf )
 ( printout t " 2. Fresh Water "crlf )
 ( printout t "Enter Your Choice:" )
 ( assert ( host(read)))
(retract ?x)
)

(defrule fish7
 (host 1)
 ?x <- (host ?number)
 => (printout t crlf "******************************" crlf crlf)
 ( printout t "What is the condition of fishes (Moribund or Dying) "crlf )
 ( printout t " 1. Moribund "crlf )
 ( printout t " 2. Dying "crlf )
 ( printout t "Enter Your Choice:" )
)
(assert (state(read)))
(retract ?x)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule fish8
  (state 1)
?x <- (state ?number)
=>
(printout t crlf "******************************************************************************** crlf crlf"
(printout t "Are the fishes swimming Normally or Erratically "crlf")
(printout t "1. Normal "crlf")
(printout t "2. Erratically "crlf")
(printout t "Enter Your Choice:"
(assert (swim(read)))
(retract ?x)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule fish9
  (swim 1)
?x <- (swim ?number)
=>
(printout t crlf "******************************************************************************** crlf crlf"
(printout t "There is no sign of Fish lices but there could be some other" crlf)
(printout t "please enter choice 2 to continue for further diagnosis" crlf)
(printout t crlf "******************************************************************************** crlf")
(assert (swim(read)))
(retract ?x)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule fish10
  (swim 2)
?x <- (swim ?number)
=>
(printout t crlf "******************************************************************************** crlf crlf"
(printout t "How long have you identified this behavior "crlf")
(printout t "1. Recently (less than a week) "crlf")
(printout t "2. Earlier (more than a week) "crlf)
(printout t "3. Chronic (continuous from the beginning) "crlf)
(printout t "Enter Your Choice:"
(assert (since(read)))
(retract ?x)
)

************** another rule **************

(defrule otherrule
(swm yes)
?x <- (swim ?choice)
=>
(printout t crlf ""crlf crlf)
(printout t "How long have you identified this behaviour "crlf)
(printout t "1. Recently (less than a week) "crlf)
(printout t "2. Earlier (more than a week) "crlf)
(printout t "3. Chronic (continuous from the beginning) "crlf)
(printout t "Enter Your Choice:"
(assert (since(read)))
(retract ?x)
)

**************

(defrule fish11
(since 1)
?x <- (since ?number)
=>
(printout t crlf ""crlf crlf)
(printout t "Did it occur concurrently with poor "crlf)
(printout t "water quality parameters in the pond: "crlf)
(printout t "1. Yes" crlf)
(printout t "2. No" crlf)
(printout t "Enter Your Choice:")
(assert (occur(read)))
(retract ?x)
)

**************

(defrule fish12
(occur 1)
?x <- (occur ?number)
=>
(printout t crlf "**************************************************************************** " crlf
crlf)
(printout t "Did it subside by improving water quality as suggested by the Water Quality
Diagnostic Tool" crlf)
  (printout t "1. Yes" crlf)
  (printout t "2. No" crlf)
  (printout t "Enter Your Choice:")
  (assert (improve(read)))
(retract ?x)
)

****************************************************************************

(defrule fish13
  (improve 1 )
?x <- (improve ?number)
=>
(printout t crlf "**************************************************************************** " crlf
crlf)
(printout t "The problem was due to poor water quality" crlf)
  (printout t "like high ammonia, low D.O and high organic load. " crlf)
  (printout t "Keep the water in good condition for future prevention" crlf)
(retract ?x)
)

****************************************************************************

(defrule fish14
  (improve 2 )
?x <- (improve ?number)
=>
(printout t crlf
"**************************************************************************** " crlf crlf)
(printout t "Does the fish secrete excess mucus and are there any " crlf)
(printout t "tiny hard to find red lesion? (Choose Yes or No)" crlf)
  (assert (argulus(read)))
  (retract ?x)
)

(defrule argulus
  (argulus yes)
?x <- (argulus ?choice)
=>

49
Diagnosis

The problem was due to Argulus. This is caused by a parasitic crustacean.

Treatment

Short bath treatment with potassium permanganate in a concentration of 10 mg/Liter for 30 min.

Since the effect depends on the temperature the bath temperature should be above 10 C.

You can also use prolonged bath treatment in weak trichlorophon at a concentration rate of 0.2 mg/Liter for 6 hours to cure the fish.

Since this Argulus propagate by laying eggs near the aquatic vegetation it is suggested to remove the unwanted aquatic weeds as a preventive measure.

Suggestion

Apply 100-200 kg of salt per acre by dissolving the salt solution into the water.