

Faculty of Computer Science and Information Technology
Universiti of Malaya
50603 Kuala Lumpur

Perpustakaan SKTM

Expert System: Mushroom Species

Nur Fuhaizah Binti Mohammad Fawzi
WEK010214

Supervisor: Dr. Rukaini Haji Abdullah
Moderator: Dr. Sameem Abd Kareem

Perpustakaan Universiti Malaya



A511275168

Sesi 2003/2004

Abstract

Expert Systems are computer programs that use artificial intelligence to solve problems within a specialized domain that ordinarily requires human expertise. A very important aspect of helping mycologist with diagnosis of the mushroom health effects is the identification itself by using only the visible features of the mushroom. Their identification, although very important, is often difficult. This thesis addresses this problem by providing an expert system that takes into consideration various factors about the mushroom, and presents identification as a possible solution. In order to achieve the purpose, the system uses Prolog, a logic programming. In this paper, chapter one outlines the introduction of the system. Chapter two describes the literature review. Chapter three discusses more details methodology. System analysis is described in chapter four. Chapter five describes the system design. Implementations are shown in chapter six and testing in chapter seven. Finally, chapter eight gives the conclusion.

Acknowledgement

I wish to thank all those whose help, advice, assistance and input made this thesis possible. The long hours and hard work put in are deeply appreciated.

The completion of this project owes special thanks to the lecturers in the Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, especially my supervisor, Dr. Rukaini Haji Abdullah and moderator, Dr. Sameem Abd. Kareem for the invaluable support, planning and guidance.

I am also indebted to Dr. Noorlidah Abdullah from the Institute of Biological Sciences, Faculty of Science, and also other staff there for their information and generous assistance during the project.

My thanks and gratitude also goes to my friends and family who are kind enough to help during the making of this project and made the long researching hours more enjoyable.

To these and others who may have been left out inadvertently, I wish to record my sincere appreciation. It is hope that this report will be a fitting testimony to what I gain from completing this task.

Table of Contents

Contents	Page
Abstract	ii
Acknowledgement	iii
Table of Contents	iv
Table List	vii
Figure List	viii
Chapter 1 Introduction	36
1.1 Project Overview	37
1.1.1 What is an Expert System?	1
1.1.2 Why Mushrooms?	2
1.2 Objectives	3
1.3 Project Scope	4
1.4 Target User	5
1.5 Approach and Schedule	
1.5.1 Project Steps	6
1.5.2 Project Schedule	7
Chapter 2 Literature Research	41
2.1 Domain Studies	43
2.1.1 Domain Background	8
2.2.2 Existing System Review	10

2.2	Technology Review	
2.2.1	Development Models	13
2.2.2	System Architectures	15
2.2.3	Application Platforms	16
2.2.4	Programming Languages	22
2.2.5	Authoring Tools	27
2.2.6	Database	29
Chapter 3	Methodology	
3.1	Software Development Life Cycle (SDLC)	
3.1.1	System Development	32
3.1.2	System Operation and Support	34
3.2	Information Gathering Methods	35
3.3	Conclusion on Tools and Technology	
3.3.1	System Type	36
3.3.2	Application Platform	37
3.3.3	Knowledge Engineering	38
3.3.4	Authoring Tools	39
3.3.5	Knowledge-base	39
Chapter 4	System Analysis	
4.1	System Requirements Analysis	
4.1.1	Functional Requirements	40
4.1.2	Non-Functional Requirements	40
4.2	Tools and Technology Proposed	
4.2.1	Software	41
4.2.2	Hardware	42
4.3	Run Time Requirements	
4.3.1	Hardware Specifications	43
4.3.2	Application Software Specification	43

Chapter 5	System Design	
5.1	System Functionality Design	44
5.2	Knowledge-base Design	47
5.3	Interface Design	49
Chapter 6	System Implementations	
6.1	Overview	56
6.2	Define the classes and instances	56
6.3	Define the rules and object communication	59
6.4	Design the interface	62
Chapter 7	Testing and Evaluation	
7.1	Overview	66
7.2	System Validation	
7.2.1	Validate Results	67
7.2.2	Validate Reasoning	69
7.3	User Acceptance	
7.3.1	Ease of Use	70
7.3.2	Clarity of Question	70
7.3.3	Clarity of Explanation	71
7.3.4	Presentation of Results	71

Chapter 8	Discussions	
8.1	Problems and Solutions	72
8.1.1	Problems	72
8.1.2	Solutions	73
8.2	Advantages and Disadvantages	
8.2.1	Advantages	74
8.2.2	Disadvantages	75
7.3	Future Plans	76
Appendix A	Glossary of Expert System Terms	77
Appendix B	Glossary of Common Mycological Terms	79
Appendix C	Illustrated Figures of the Development Models	80
Reference		82

Figure List

Figure	Table List	Page
Table		Page
Figure 1.1: Expert System Block Diagram		1
Table 1.1: Comparison of Human Expert and an Expert System		3
Table 2.1: Existing Expert Systems		10
Table 2.2: Software Process Model Comparison		13
Table 2.3: Application Platforms Comparison		16
Table 2.4: Programming Language Comparison		22
Table 2.5: Prolog Comparison		26
Figure 3.2: Incremental Development Model		33
Figure 5.1: System Architecture		43
Figure 5.2: A class		45
Figure 5.3: Instances of the class		46
Figure 5.4: Knowledge base architecture		47
Figure 5.5: System Structure of the Integrated Knowledge Base		48
Figure 5.6: Creation Screen Template		49
Figure 5.7: Welcome Screen		50
Figure 5.8: The Order Input Dialog		50
Figure 5.9: The Macroscopic Features Input Dialog		51
Figure 5.10: The Microscopic Features Input Dialog		52
Figure 5.11: The Output Display Dialog		53
Figure 5.12: The Help Dialog		54
Figure 5.13: The Template Interface		55

Figure List

Figure	Page
Figure 1.1: Expert System Block Diagram	1
Figure 1.2: Project Scope	4
Figure 1.3: Project Steps	6
Figure 2.1: Mushroom Features	8
Figure 2.2: Two-tier Architectures	15
Figure 2.3: Three-tier Architectures	15
Figure 3.1: A System Life Cycle	32
Figure 3.2: Incremental Development Model	33
Figure 5.1: System Architecture	43
Figure 5.2: A class	45
Figure 5.3: Instances of the class	46
Figure 5.4: Knowledge-base Architecture	47
Figure 5.5: System Structure of the Integrated Knowledge Base	48
Figure 5.6: Question Screen Template	49
Figure 5.7: Welcome Screen	50
Figure 5.8: The Order Input Dialog	50
Figure 5.9: The Macroscopic Features Input Dialog	51
Figure 5.10: The Microscopic Features Input Dialog	52
Figure 5.11: The Output Display Dialog	53
Figure 5.12: The Help Dialog	54
Figure 5.13: The Database Interface	55

Figure 6.1: Table Designer	57
Figure 6.2: Database Designer	57
Figure 6.3: Table	58
Figure 6.4: Application Builder	58
Figure 6.5: ODBC Data Source Administrator	59
Figure 6.6: Dialog Editor	62

Expert systems or knowledge-based systems are computer programs that are derived from a branch of computer science research called Artificial Intelligence. They allow the scarce and expensive knowledge of experts to be explicitly stated into computer programs and made available to others who may be less experienced. They range in scale from simple rule-based systems with the data to very large scale, integrated developments taking many person-years to develop. They usually have a set of inference rules which form the knowledge base and a dedicated inference engine, which provides the decision making. This contrasts with conventional programs where domain knowledge and execution control are closely intertwined such that the function is explicitly stored in the program. This explicit separation of the knowledge from the control mechanism makes it easier to examine knowledge, incorporate new knowledge and modify existing knowledge.

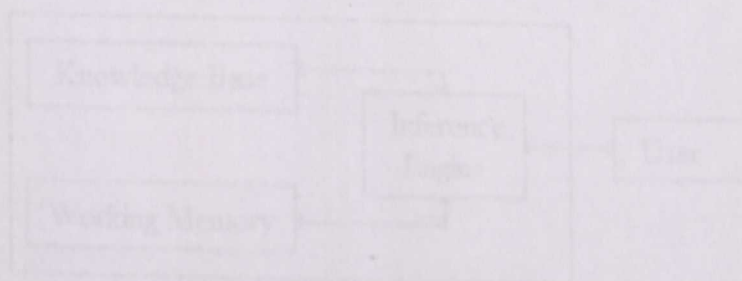


Figure 1.1: Expert System Block Diagram

1. Introduction

1.1 Project Overview

1.1.1 What is an Expert System?

Expert systems or knowledge-based systems are computer programs that are derived from a branch of computer science research called Artificial Intelligence. They allow the scarce and expensive knowledge of experts to be explicitly stored into computer programs and made available to others who may be less experienced. They range in scale from simple rule-based systems with flat data to very large scale, integrated developments taking many person-years to develop. They typically have a set of if-then rules which forms the knowledge base, and a dedicated inference engine, which provides the execution mechanism. This contrasts with conventional programs where domain knowledge and execution control are closely intertwined such that the knowledge is implicitly stored in the program. This explicit separation of the knowledge from the control mechanism makes it easier to examine knowledge, incorporate new knowledge and modify existing knowledge.

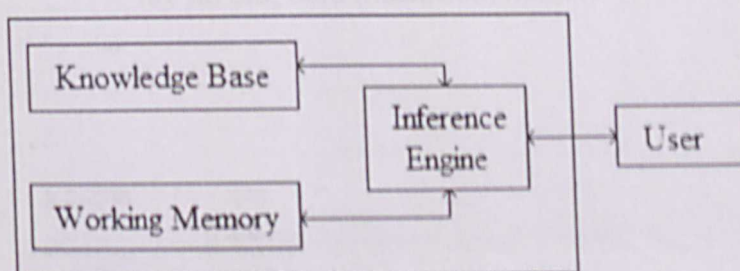


Figure 1.1: Expert System Block Diagram

1.1.2 Why mushrooms?

A person other than a mycologist or other agricultural expert is usually faced with two major problems:

1. The inability to identify many of the mushrooms because no reliable illustrated guide exists and there are thousand of mushrooms.
2. The puzzling fact that sometimes seemingly identical mushrooms pictured and described in various guide books has different names.

These two major problems will occasionally lead to mushroom poisoning as the victim, who was attempting to identify edible species, has mistakenly picked toadstool or poisonous mushroom. Keeping this in mind, developing a stand-alone automated system which would encode a mycologist's knowledge on how to identify a specific type of mushroom using only the visible features is necessary so that any lower-level expert and also non-expert can use the system and get reliable help for accomplishing this task.

“There are old mushroom hunters, and bold mushroom hunters, but there are no old, bold mushroom hunters”.

1.2 Objectives

To understand the project, the author has established five main objectives. These objectives focus on permitting users to understand why the system is build. The five objectives are:

- 1. As a stand alone identification system for the specific knowledge domain perhaps with monitoring by a human expert.
- 2. To provide decision support for a high-level human expert.
- 3. To allow a high-level expert to be replaced by a subordinate expert aided by the expert system.
- 4. To provide management education for decision makers.
- 5. For distribution of up-to-date scientific information in a readily accessible and easily understood form, to agricultural researchers and advisers.

The table below shows the advantages of building an expert system that supports the objectives given;

Table 1.1: Comparison of Human Expert and an Expert System

Factor	Human Expert	Expert System
Time availability	Workday	Always
Geographic	Local	Anywhere availability
Safety	Irreplaceable	Replaceable
Perishable	Yes	No
Performance	Variable	Consistent
Speed	Variable	Consistent (usually faster)
Cost	High	Affordable

1.3 Project Scope

In order to build an effective expert system, the task domain needs to be narrowed down significantly. Figure 1.2 shows the scope focus of the project. The focus is on the Basidiomycotina and then Hymenomycetes branch of the mushrooms. The project is centered only on species that are edible and with health values that exist in Malaysia. A glossary of the terms used in this section is given in Appendix B.

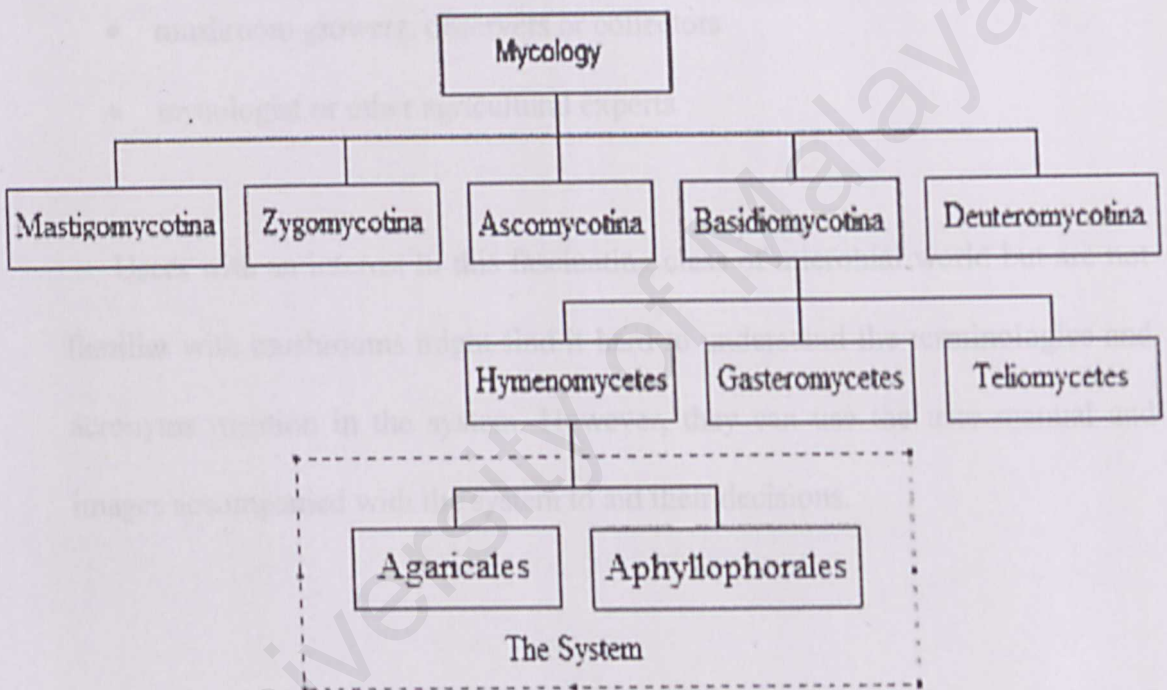


Figure 1.2: Project Scope

The author decided to choose the mentioned scope focus because they are common mushrooms, described by naturalist as the umbrella-shaped type fungus with discoid cap supported on a central stalk. Other funguses have different features, grow on different habitats and different climate, which are not commonly seen or cultivated in Malaysia.

1.4 Target User

The system is designed for users who have at least basic knowledge about mushrooms, for example:

- undergraduate students or researches of microbiology, plant science or plant pathology
- mushroom growers, observers or collectors
- mycologist or other agricultural experts

Users with an interest in this fascinating class of microbial world but are not familiar with mushrooms might find it hard to understand the terminologies and acronyms mention in the system. However, they can use the user manual and images accompanied with the system to aid their decisions.

1.5 Approach and Schedule

The following section 1.5.1 shows each major step of the research, the tasks to be performed and the resources needed to complete the steps. The subsequent section 1.5.2 shows the time needed for each task and a schedule for their completion

1.5.1 Project Steps

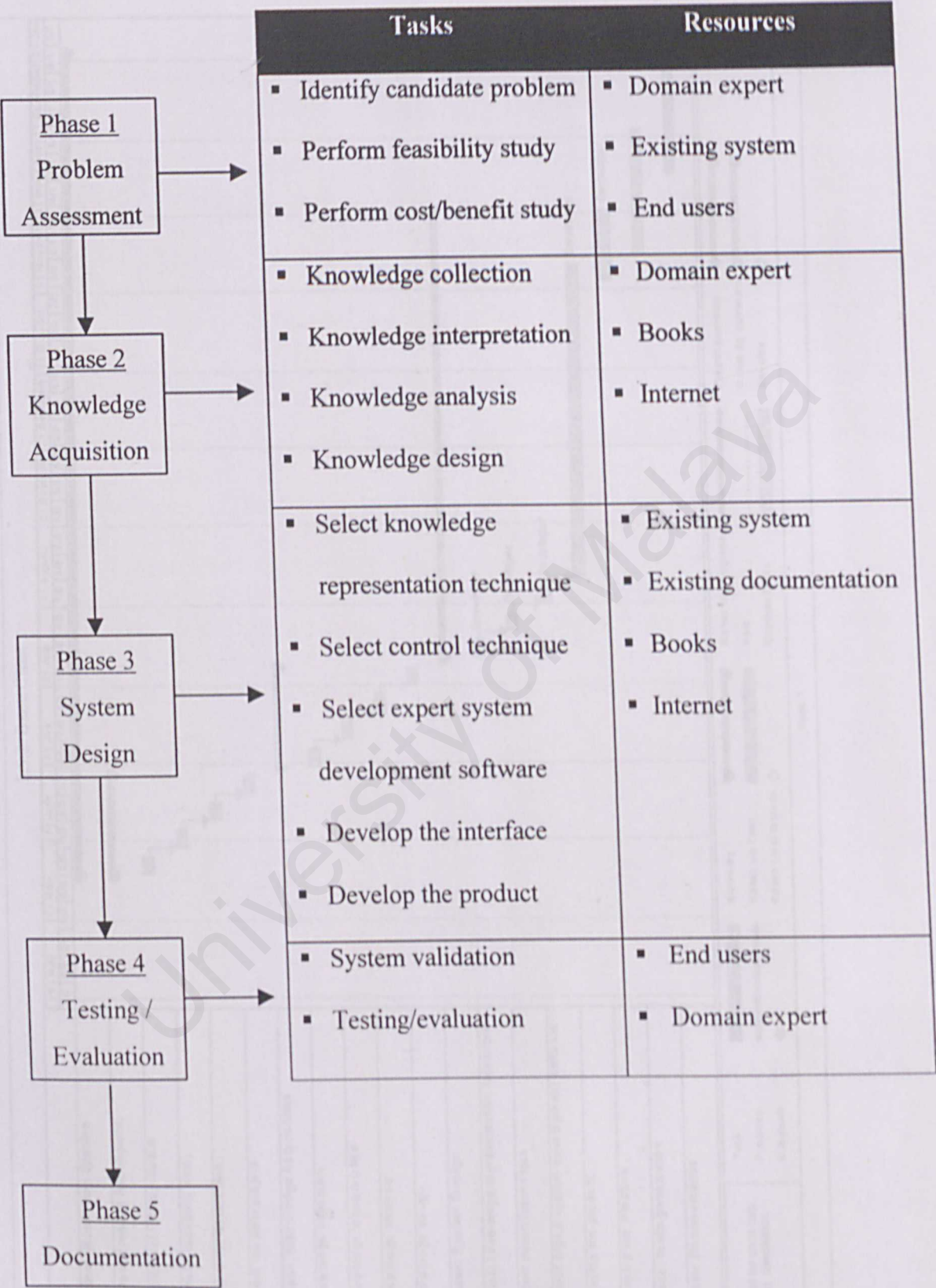
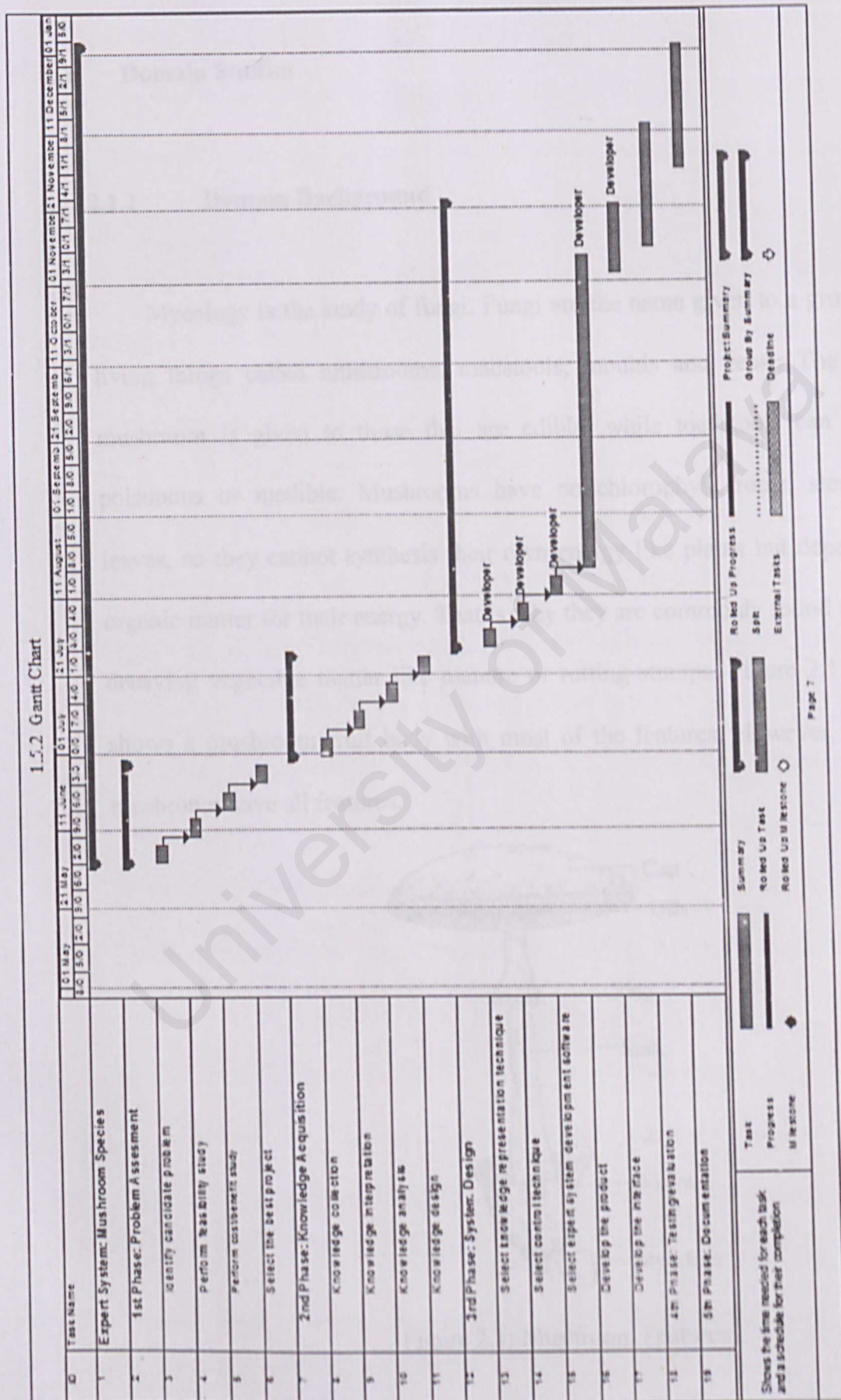


Figure 1.3: Project Steps

1.5.2 Gantt Chart



2. Literature Research

2.1 Domain Studies

2.1.1 Domain Background

Mycology is the study of fungi. Fungi are the name given to a group of living things called mushrooms, toadstools, moulds and yeast. The term mushroom is given to those that are edible, while toadstools can mean poisonous or inedible. Mushrooms have no chlorophyll, roots, stems or leaves, so they cannot synthesis their own energy like plants but depend on organic matter for their energy. That is why they are commonly found around decaying vegetable matter like manure or rotting stumps. Figure 2.1 below shows a mushroom fruit body with most of the features. However, not all mushroom have all features.

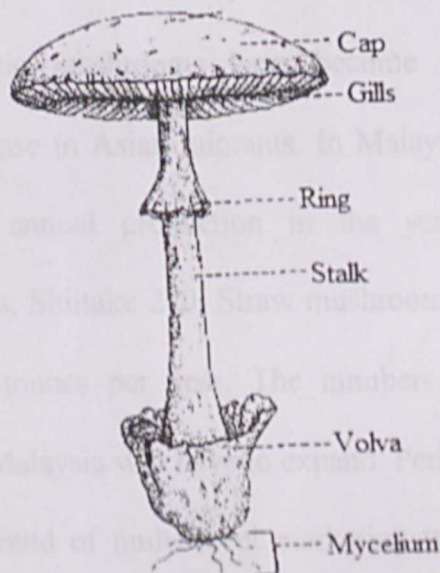


Figure 2.1: Mushroom Features

The growth of mushrooms is favoured by cool, moist and weather, so sometimes they appear in yard after rainfalls. It all starts when the spores are released from the gills. These spores are dispersed by varying methods, when the condition are right, the spores germinate, sending out tiny threads called hyphae or single hypia. In order for it to develop, it has to join with another hypia to form a network of thread called mycelium. The mycelium forms a hyphal knot which grows and develops into a pinhead. In turn the pinhead gradually grows and develops into a full mushroom fruit body ready to disperse new spores.

Egyptian Pharaohs considered mushroom such a delicacy that ordinary people were forbidden to eat it. However, commercial growing has been much more recent phenomenon when the French began to grow field mushrooms in caves in the 17th century. From France they spread to England and finally to Australia. In Australia, commercial production started in 1930's using disused railway tunnels, a far cry from today's sophisticated and modern facilities. Exotic mushrooms have become more available in Australia with the increase in Asian migrants. In Malaysia itself, the major mushroom cultivation annual production in the year 2000 is Oyster mushrooms 1500 tonnes, Shiitake 210, Straw mushrooms 150 and other 25, with a total of 1885 tonnes per year. The numbers are increasing and mushroom growers in Malaysia will have to expand. Perhaps this has been so because of the latest trend of multi level marketing where they show that mushrooms have got medicinal and nutritional values, so as people in Malaysia become more and more health conscious.

2.1.2 Existing System Review

Expert systems have been commercially successful since the late 1970's. Many species identification systems existed but with different domain and techniques, as shown in Table 2.1:

Table 2.1: Existing Expert Systems

System	Domain	Language	User	Platform
Douglas-fir Cone and Seed Insects	Insect pests that infest douglas-fir cones and seeds	ACQUIRE [®] , ACQUIRE [®] SDK	Seed orchard managers, cone and seed collectors	Windows
Whale Watcher	Whales of the Canadian coastal waters	ACQUIRE [®] , ACQUIRE [®] SDK	All	Windows
An Interactive Guide to Massachusetts Snakes	Snakes of Massachusetts	HTML	All	Windows
Common Conifers of the Pacific Northwest	Conifers of the Pacific Northwest	Unknown	Forestry students, resource professionals	Macintosh, IBM computers

A web-based automated identification facility with mushroom as a domain exists at <http://www.agarics.org/>. It comprises:

1. A static database of species of fungi with their sizes, colours and other properties, plus photographs, read at site start-up from an XML file.
2. A property fuzzy-matching library, written in Java. This allows a database of records, each comprising a number of property values, to be searched for the best match to a candidate record. Wide varieties of properties are available, such as numeric values, single and multiple selections and colours. The way that matching operates can be carefully tuned for each property type and each individual record.
3. Extensions to the above library, giving extra properties specific to fungi. These include properties like smell, gill shape etc.
4. A servlet for drawing dynamic graphs and charts in a Web application. Uses Chart2D, a free Java chart library from Source Forge. Currently unused in the live site, due to limitations of ISP's server.
5. A set of static and dynamic (JSP) Web pages to obtain input from a Web client, drive the matching process and present the results to the Web client.
6. An open-source relational database, MySQL is used to persist site visitor tracking data but not for the fungi data.
7. A Fungi Data Feedback for user to add to or correct the information in the fungi database. Information that is entered will not be added straight away. It will be sent to the site administrator and added once it has been checked.
8. Links to other related mushroom sites

Another related system, which demonstration is available at <http://mycosoft.co.uk/home.htm>, is called An Illustrated Key for the Identification of Fungi, Wild Mushrooms and Toadstools. The stand-alone version is for use on PCs running Windows with the minimum specification of 640 X 480 screen, 16MB RAM, 16 bit colours and 4X CD-ROM. However, there is no information on how the system was build.

Research has also been made at the Department of Psychology in University of Alberta. It is a Biological Computation project name "Of Mushrooms and Machine Learning: Identifying Algorithms in a PDP Network" by Michael R. W. Dawson and David A. Medler. The purpose of the experiment was to determine if an artificial neural network could learn to identify correctly a mushroom as edible or not. In particular, they were interested in seeing whether after the network converged, users could determine the rules that it used to classify mushrooms. However, the outcome was not clearly stated.

All systems mention before in this section are web-based because the author did not have the opportunity to experience with a stand-alone system.

2.2 Technology Review

All reviews mention in this section are based on resources (software or hardware) that are available and accessible at the Faculty of Computer Science and Information Technology, University of Malaya.

2.2.1 Development Models

Below are the comparisons of various development models applicable to the system. Illustrated figures of the models is given in Appendix C.

Table 2.2: Software Process Model Comparison

Models	Benefits	Drawbacks
Waterfall model	<ul style="list-style-type: none">• Simple, familiar to most developers, easy to understand• Easy to associate measures, milestones and deliverables with different stages	<ul style="list-style-type: none">• Does not reflect how software is really developed• Does not reflect the back-and-forth, iterative nature of problem solving• Not applicable for many types of development
V model	<ul style="list-style-type: none">• Better spells out the role of different types of testing• Involves user in testing	<ul style="list-style-type: none">• Extensive testing may not always be cost-effective• Some of the same drawbacks as waterfall

Models	Benefits	Drawbacks
Prototyping/ Evolutionary model	<ul style="list-style-type: none"> • Promotes understanding of problems before trying to implement solution • Reduces risk and uncertainty • Involves user in evaluating interface 	<ul style="list-style-type: none"> • Prototyping can use a lot of resources, especially if the prototype fails completely and must be scrapped • In systems where problems are understood or user interface is simple and straightforward, extra time spent in prototyping is not warranted
Incremental development	<ul style="list-style-type: none"> • Customer training can begin early • Frequent release allows problems to be fixed quickly • Expertise can be applied to different release • Reduces time when customer receives some product 	<ul style="list-style-type: none"> • Customer may not be satisfied with an incomplete product or with frequent changes to system • Problems may not be easily decomposable • Changes may have to be made to complete parts in order to work with new parts
Spiral development	<ul style="list-style-type: none"> • Risk analysis preceding each phase • Allows for changing requirements • Allows prototyping 	<ul style="list-style-type: none"> • Once the risk cannot be mitigated, the project is terminated. • Not effective for large-scale projects

Below are the comparisons of two major system architectures applicable.

1. Two-tier

- Simple, distributed broker, fault tolerant and flexible load balance

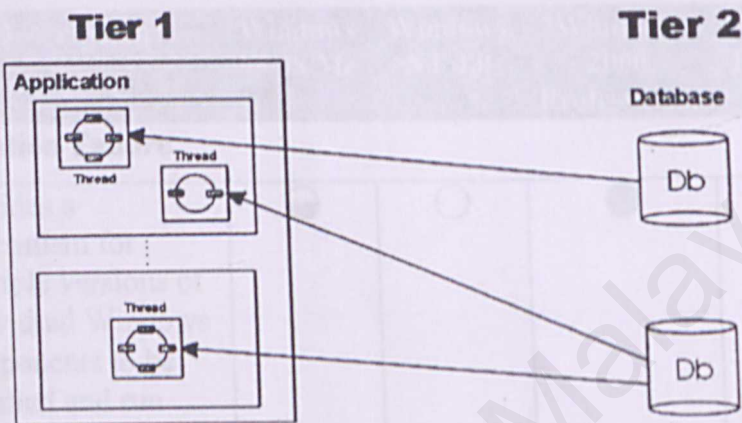


Figure 2.2: Two-tier Architectures

2. Three-tier

- Complex, monolithic broker, single point failure and rigid / difficult load balancing

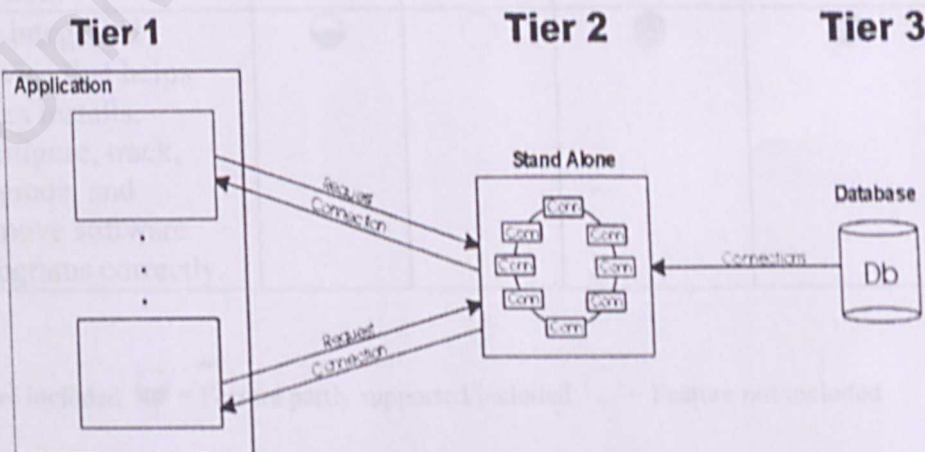

















Figure 2.3: Three-tier Architectures

2.2.3 Application Platform

Below are the feature comparisons of various application platforms (operating systems) available.

Table 2.3: Application Platforms Comparison

Feature	Description	Windows 95/98/Me	Windows NT 4.0	Windows 2000 Professional	Windows XP Professional
Reduces Application Failure					
Side-by-Side DLL Support	Provides a mechanism for multiple versions of individual Windows components to be installed and run "side by side."				
Windows File Protection	Protects core system files from being overwritten by application installations. If a file is overwritten, Windows File Protection will restore the correct version.				
Windows Installer	An integrated service that helps users installs, configure, track, upgrade, and remove software programs correctly.				

 = Feature included
  = Feature partly supported/included
  = Feature not included

Feature	Description	Windows 95/98/Me	Windows NT 4.0	Windows 2000 Professional	Windows XP Professional
Stays Up and Running					
Built on New Windows Engine	32-bit computing architecture and a fully protected memory model				
System Restore	Enables users and administrator to restore a computer to a previous state without losing data. System Restore, by automatically creating identifiable restores points	 In Windows Me			
Device Driver Rollback	When certain classes of new device drivers are installed, the OS will maintain a copy of the previously installed driver, which can be reinstalled if problems occur.				
Device Driver Verifier	Stress tests for device drivers.				
Dramatically Reduced Reboot Scenarios	Eliminates most scenarios that forced users to reboot in Windows NT 4.0 and Windows 95/98/Me.				
Scalable Memory and Processor Support	Supports up to 4 gigabytes (GB) of RAM and up to two symmetric multiprocessors.				

= Feature included
 = Feature partly supported/included
 = Feature not included

Feature	Description	Windows 95/98/Mc	Windows NT 4.0	Windows 2000 Professional	Windows XP Professional
Enhances Windows Security					
Internet Connection Firewall	A firewall client that can protect small businesses from common Internet attack	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Encrypting File System (EFS) with Multi-user Support	Encrypts each file with a randomly generated key. The encryption and decryption processes are transparent to the user	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/> No support for use with multiple users	<input checked="" type="radio"/>
IP Security (Insect)	Helps protect data transmitted across a network. IPSec is an important part of security for virtual private networks (VPNs), which allow organizations to transmit data securely over the Internet	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Kerberos Support	Provides industry-standard and high-strength authentication with fast, single sign-on to Windows 2000-based enterprise resources.	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Smart Card Support	Integrates smart card capabilities into the operating system, including support for smart card login to terminal server session	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

☒ = Feature included
 ☐ = Feature partly supported/included
 ☐ = Feature not included

Feature	Description	Windows 95/98/Me	Windows NT 4.0	Windows 2000 Professional	Windows XP Professional
Simplifies Desktop Deployment					
Increased Application Compatibility	Can specify if the application needs to run in either a Windows NT 4.0 or Windows 95/98/Me compatibility mode	N/A			
User State Migration Tool	Migrate a user's data and application/operating system settings from an old computer to a new computer			 As a resource kit tool with no support	
Support for Latest Hardware Standards	Supports UDF 2.01, formatting of DVD-RAM drives with the FAT32 file system. DirectX® 8 API support included, and supports IrDA, USB and IEEE1394.	 Support for some standards listed		 Support for some standards listed	
Unattended Installation	The ability to specify greater number of options and allows for greater degree of security	 Support for subset of options	 Support for subset of options	 Support for subset of options	
System Preparation Tool (SysPrep)	Clone computer configurations, systems, and applications				
Setup Manager	A graphical wizard that guides administrators in designing installation scripts				
Remote OS Installation	Can be installed across the network				
Multilingual Support	Allows users to easily create, read, and edit documents in many languages				

= Feature included
 = Feature partly supported/included
 = Feature not included

Feature	Description	Windows 95/98/Me	Windows NT 4.0	Windows 2000 Professional	Windows XP Professional
Improves Desktop Management					
Group Policy	Simplify the administration of users and objects by letting administrators organize them and assign the same settings	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Improved Help and Support Services	The Help and Support Centre combines from previous versions of Windows with content from the World Wide Web	<input checked="" type="radio"/> Subset of features in Windows Me	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Automatic Updates	Automatically downloads critical and security updates when the user is connected to the Internet	<input checked="" type="radio"/> Subset of features in Windows Me	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Microsoft Management Console (MMC)	Provides a centralized and consistent environment for management tools	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Windows Management Instrumentation (WMI)	Provides a standard infrastructure for monitoring and managing system resources.	<input checked="" type="radio"/> Subset of features	<input type="radio"/>	<input checked="" type="radio"/> Subset of features	<input checked="" type="radio"/>
Safe Mode Start-up Options	Allows Windows to boot the system at the most basic level, using default settings and minimum device drivers	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>

Feature	Description	Windows 95/98/Mc	Windows NT 4.0	Windows 2000 Professional	Windows XP Professional
Increases User Efficiency					
Fresh Visual Design	Fresh visual design, common tasks have been consolidated and simplified, and new visual cues added	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Adaptive User Environment	Adapts to the way you work	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Improved Handling of File Associations	If you are trying to open a file that is not associated with any program, Windows can send you to a Web page from which to download or purchase the right program	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Context Sensitive Task Menus	When a file is selected in Windows Explorer, a dynamic menu appears	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Integrated CD Burning	Integrated support for burning CDs on CD-R and CD-RW drives	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Troubleshooters	Help users and administrators configure, optimize, and troubleshoot numerous functions	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

☒ = Feature included
 ☐ = Feature partly supported/included
 ☐ = Feature not included

2.2.4 Programming Languages

Below are the feature comparisons of various programming language available.

Table 2.4: Programming Language Comparison

Feature	Description	Java	C++	Visual Basic
Static / Dynamic	Requires or not variable to be declared as specific type	Static	Static	Static
Generic Classes	The ability to parameterize a class with specific data types	No	Yes	No
Inheritance	The ability for a class or object to be defined as an extension or specialization of another class or object	Single class, multiple interfaces	Multiple	None
Feature Renaming	The ability for a class or object to rename one of its features that it inherited from a super class	No	No	No
Method Overloading	Parametric polymorphism, the ability for a class, module, or other scope to have two or more methods with the same name	Yes	Yes	No
Operator Overloading	The ability for a programmer to define an operator (such as +, or *) for user-defined types	No	Yes	No
Higher Order Functions	Functions that can be treated as if they were data objects	No	No	No
Lexical/ static closures	Bundling up the static scope surrounding the function with the function itself, so that the function carries its surrounding environment around with it wherever it may be used	No	No	No

Garbage Collection	A mechanism allowing a language implementation to free memory of unused objects	Mark & sweep / generational	None	Reference counting
Uniform Access	Does not exhibit any notational differences between accessing a feature regardless of whether it is an attribute or a function	No	No	Yes
Class Variables / Methods	Owned by a class, and not any particular instance of a class	Yes	Yes	No
Reflection	The ability for a program to determine various pieces of information about an object at run-time	Yes	No	No
Access Control	The ability for a modules implementation to remain hidden behind its public interface	Public, protected, "package", private	Public, protected, private, "friends"	Public, private
Design by Contract	The ability to incorporate important aspects of a specification into the software that is implementing it	No	No	No
Multithreading	Pattern matching constructs capable of recognizing the class of languages known as regular languages	Yes	No	Yes
Pointer Arithmetic	The ability for a language to directly manipulate memory addresses and their contents	No	Yes	No
Language Integration	The ability to integrate with other languages	C, some C++	C, Assembler	C, C++
Built-In Security	A language implementation's ability to determine whether or not a piece of code comes from a trusted source	Yes	No	No

All the languages mention above are imperative and object oriented programming. Another type is logic programming which is more synonym with Prolog (Programming in Logic), designed to give a practical implementation of the language of Predicate Calculus. A common type of Prolog available is LPA WIN-PROLOG, which is bundled with the following functions and utilities:

- **Automatic Configuration:** the same files run on all Windows version
- **Multiple Document Interface:** any number of program edit windows can simultaneously opened in an MDI-standard development environment
- **Fully Programmable GUI:** large library of GUI functions, providing the creation and control of windows, dialogs, controls, menus, fonts and more
- **Rich Graphics Facilities:** powerful graphics predicates give the ability to create charts, diagrams, as well as graphical buttons and tools; graphics facilities include vector, polygon, bitmap, icon, metafile and cursor display control, together with scaling and scrolling functions
- **Dynamic Link Libraries:** as well as the 32-bit MASM interface, WIN-PROLOG can load and access code in DLLs written using standard Windows development languages, like Visual C/C++ and Visual Basic
- **Direct Windows API interface:** virtually any Windows API function, or third-party DLL function, can be directly called from WIN-PROLOG, without the need to resort to C/C++ programming
- **Dynamic Data Exchange:** ready-to-go DDE interface allows direct communication between Prolog and Visual Basic, Microsoft Word, Excel or any other DDE-aware Windows application

- **Language Interfaces:** Ready-built interfaces to C/C++, Visual Basic, Databases (ODBC), Delphi and 32-bit MASM
- **Comprehensive Help:** a fully cross-referenced version of the Technical Reference manual is supplied as a Microsoft Help file, providing complete on-line documentation of all system predicates and functions
- **True 32-bit Implementation:** up to 4Gb (4096Mb) of memory is directly addressable, without complex internal segmented addressing
- **Small Memory Requirements:** as little as 4Mb of memory: as much space as possible is made available for use by user's applications code
- **Quintus Prolog Compatibility:** the system was designed from the outset with QP compatibility as a key objective
- **64-bit Arithmetic:** full-featured, efficient double precision built-in floating point math library complements the 32-bit integer arithmetic
- **Incremental and Optimised Compilation:** all the flexibility of a traditional interpreter is combined with the runtime speed of fully compiled code
- **Source Level and Box Model Debuggers:** these make full use of windows and other GUI features to make program testing and debugging as easy as possible
- **Operating System Control:** full featured access to the operating system gives Prolog programs full control of files, directories, environment variables, time and date, and allows other applications to be executed
- **User-definable System Hooks:** many events, such as errors, spy points, timers and messages can be directly programmed in Prolog

- **Special Data Types:** A true string data type supports efficient text manipulation, and four linked data types efficiently support compound terms.
- **Sophisticated Data Compression:** Lempel/Ziv data compression and decompression routines are built in, and are used both for saving/loading system files, and for general user-specified applications.
- **Full Range of Options:** as well as Programmer and Developer editions of WIN-PROLOG, the flex expert system toolkit and Prolog++ object-oriented toolkit options are available for use.

Other types of prolog available are briefly described in Table 2.5.

Table 2.5: Prolog Comparison

Name	Free	Operating System	Features
SWI Prolog	Yes	Windows, Linux/Unix, MacOS X, BeOS	Comprehensive built-in predicates, machine-independent saved-states, multi-threading, small and fast
Sicstus Prolog	No	Solaris, Linux	Character handling, break pointing debugger, exception handling, cross-reference, determinacy checker, GNU Emacs/XEmacs interface
Visual Prolog	No	Windows, Linux and SCO	Graphical development environment, compiler linker, debugger
Quintus Prolog	No	Solaris, Linux	Embeddable, portable, debugger, library, client/server, X windows interface
Amzi Prolog	Yes	Windows, Linux, Solaris, HP/UX	Internet, multiple session, database and Unicode support, portable

Below are overviews of related authoring tools applicable to the system.

1. Microsoft® Help Workshop

Help Workshop is a program that you use to create Help (.hlp) files, edit project and contents files, and test and report on help files. Help Workshop takes the information in the project (.hpj) file to combine the topic (.rtf) files, bitmaps, and other sources into one Help file that can be viewed using the Microsoft® Windows Help program. Help Workshop includes the following files and documents:

- Help Workshop (Hcw.exe and Hcrtf.exe). Enables you to edit project and contents files, and to compile, test, and create reports for Help files.
- Help Author's Guide (Hcw.hlp). Describes how to author and compile Help files by using Help Workshop.
- Hotspot Editor Version 2.0 (Shed.exe). Enables you to create a graphic that has multiple hotspots.
- Multi-Resolution Bitmap Compiler version 1.1 (Mrbc.exe). Enables you to create bitmaps that have different resolutions. You can combine these bitmaps into a single graphic to compensate for differences between the aspect ratios of the bitmaps and the aspect ratio supported by a user's display.

2. Adobe® Photoshop®

Adobe® Photoshop® is for professional image-editing that delivers a comprehensive environment for professional designers and graphics producers to create sophisticated images for print, the Web, wireless devices, and other media. Main features include:

- Work more efficiently - file Browser, layers, options bar, history palette, customizable workspace, context-sensitive menus
- Edit images with ease - Colour correction, healing brush, selection tools, precision masking, clipping paths, sharpening controls, edge smoothing, contact sheet generation, web photo display
- Enjoy unlimited creative options – Painting and drawing tools, layer and colour effects, filters, pattern maker, transformation tools
- Create compelling Web designs - Slicing, optimization tools, rollovers palette, transparency, quick GIF animations, link generation
- Enjoy precise typographic control - Editable text, formatting, spelling checker, convert to Shapes

2.2.6 Database

2.2.6.1 Database Model

1. Hierarchical model

The hierarchical data model organizes data in a tree structure, with hierarchy of parent and child data segments. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses Parent Child Relationships. These are a 1:N mapping between record types, done by using trees. In a hierarchical database the parent-child relationship is one to many.

2. Network model

The basic data modelling construct in the network model is the set construct, consisting of an owner record type, a set name, and a member record type. A member record type can have that role in more than one set, hence the multiparent concept is supported. An owner record type can also be a member or owner in another set. Intersection record types may exist, as well as sets between them. Thus, the complete network of relationships is represented by several pair wise

sets; in each set some record type is owner and one or more record types are members. Usually, a set defines a 1:M relationship, although 1:1 is permitted

3. Relational model

A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organised in tables. A table is a collection of records and each record in a table contains the same fields. Certain fields may be designated as keys, which mean that searches for specific values of that field will use indexing to speed them up.

Where fields in two different tables take values from the same set, a join operation can be performed to select related records in the two tables by matching values in those fields. Often, but not always, the fields will have the same name in both tables.

2.2.6.2 Microsoft® Access versus Microsoft® Visual FoxPro

The author has chosen to compare Microsoft® Access and Microsoft® Visual FoxPro because other database such as Oracle or MS SQL Server is used in large companies as well as on the internet for handling multi million numbers of records. Second, the added functionality with these larger databases makes it expensive.

Microsoft[®] Access is a good database for small applications such as recipes. Microsoft[®] Visual FoxPro was designed for small to medium size applications, often handling a million records. In tests with equivalent hardware and file sizes on a Windows 98 system, FoxPro consistently performs faster than Access 2000. Updates show the largest discrepancy in the area of performance, with Access often requiring twice as long to complete the same task. FoxPro also produces a much better and more reliable backup than the Access process of simply copying the file. Even if the FoxPro data is partially corrupted, it is much easier to restore than a corrupted Access database file.

FoxPro can handle much larger files than Access. A 100Mb database in FoxPro will show no signs of overload, compared to the similar database in Access. FoxPro also features an impressive versatility that enables it to interact with a number of interfaces, including command-line clients, web browsers and various programming interfaces such as C++, Perl, Java, PHP, and Python. Users can use a pre-packaged client or write a custom application.

It is true that the Microsoft[®] ActiveX Data Objects Library (ADO) has made Access more flexible in the foreign data market. ADO permits you to retrieve data regardless of its location, and then present that data in a common interface: the browser. On the downside, learning ADO requires much time and effort, even for the competent developer or programmer.

3. Methodology

3.1 Software Development Life Cycle (SDLC)

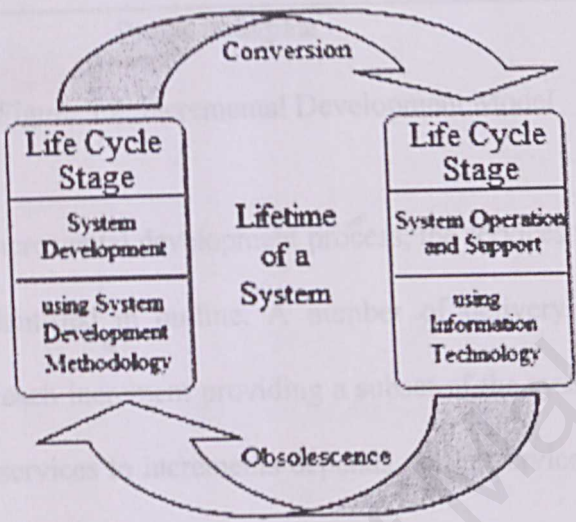


Figure 3.1: A System Life Cycle

3.1.1 System Development

General principles used to underline the system development are:

- 1. Use a problem solving approach
- 2. Establish phases and activities
- 3. Do not be afraid to cancel or revise scope
- 4. Design system for growth and change

The system development process is based on a hybrid model called incremental development, which combines the waterfall and evolutionary model advantages.

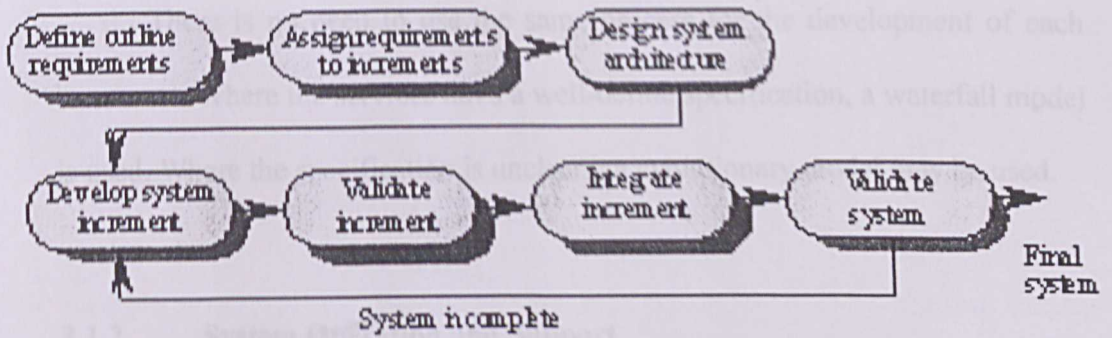


Figure 3.2: Incremental Development Model

In an incremental development process, the services to be provided by the system are identified in outline. A number of delivery increments are then defined, with each increment providing a subset of the system functionality. The allocation of services to increments depends on the service priority. The highest priority services are delivered first.

Once the increments have been identified, the requirements for the services to be delivered in the first increment are defined in detail and that increment is developed using the most appropriate development process. During that development, further requirements analysis for later increments can take place but requirements changes for current increment are not accepted.

Once an increment is completed and delivered, it can be put into service. Users can experiment with the system which helps them clarify their requirements for later increments. As new increments are completed, they are integrated with existing increments so that the system functionality improves with each delivered increment.

There is no need to use the same process for the development of each increment. Where the services have a well-defined specification, a waterfall model is used. Where the specification is unclear, an evolutionary model may be used.

3.1.2 System Operation and Support

System operation and support consists of the following ongoing activities:

1. Assisting users, regardless of how well the users have been trained and how good the end-user documentation is
2. Fixing software defects (bugs), that slipped through the software testing
3. Recovering the system from failure, that is to restore a system's files and databases, and restarts the system
4. Adapting the system to new requirements, which may include new technical problem or new technology requirements.

3.1.3 Cross Life Cycle Activities

Cross life cycle activities are activities that overlap many or all phases of the methodology. These activities include fact-finding, documentation and presentation, feasibility analysis, and process and project management.

3.2 Information Gathering Methods

1. Sampling of existing documentation, files and databases

Collect and review documents that describe the problems including thesis, research papers and other system studies and design documentation. This documentation may include operation manual or project documentation, various type of flowcharts and diagrams, and also design documentation such as input, output and databases All documentations collected are analyzed to determine the information currency, without discarding outdated documentation. Additional fact finding is performed to verify and update the facts collected.

2. Research and site visits

Thoroughly research the problem domain. This includes reference books, computer journals and also exploring the internet. These sources provide information on how others have solved similar problems, plus to learn whether software packages exist to solve the problems.

3. Interviews

Interviews are used to achieve any of the following goals: find facts, verify facts, clarify facts, identify requirements, generate enthusiasm, get the end user involves, and solicit ideas and opinions.

3.3 Conclusion on Tools and Technology

3.3.1 System Type

The author has chosen to develop a stand-alone system as there are some problems related to internet development such as:

1. The dependability of the system, including security, safety and reliability is hard to maintain as everyone in the world can access it and due to the increasing number of cyber crimes.
2. Need to cope with emerging technology, such as new versions of programming languages, browsers, and operating systems
3. Need to provide decentralized support and training for users, though the internet itself can be used for this purpose
4. Problems related to communications speed encountered with the use of multimedia and large database. Example, large files may take tens of seconds to download over modems typically used at home, which, although not a long time, can be frustrating when it occurs repeatedly.

A stand-alone system can also be networked so that several people at once can have access to the files, so a web-based system is not actually necessary

3.3.2 Application Platforms

Windows 2000 Professional is the chosen platform because it is proven to be powerful and flexible enough to perform every task that even Windows XP is built on the proven code base on it. Additional features in Windows XP are not actually necessary for building the system. Windows 2000 is a full 32-bit operating system, which eliminate the problems that have plagued older system. It is able to run most Windows 95/98/NT programs and provides compatibility with older, 16 bit code including DOS.

Windows 2000 uses Win32, which support 32-bit flat addressing and includes Application Programming Interface (API) functions that support thread-based multitasking and security. The Win32 API functions are contained in Dynamic Link Libraries (DLLs), which each program has access to when executing. Dynamic linking has some very important benefits such as; DLLs prevent disk space from being wasted by the significant amount of duplicated object code and the dynamic linking approach makes the emulation of other operating systems an easier task.

Windows 2000 also supports two form of multitasking; process based and thread based. One other thing to know is that it support several file systems, including FAT (File Allocation Table), FAT32 (enhanced, 32-bit FAT) and NTFS (NT File System).

3.3.3 Knowledge Engineering

The main language selected is Prolog as it is found to be suitable for the development of expert system type program for several reasons:

1. Rule based – Rules can work largely with each other and can produce some sort of sense even when incorrect. It also can be incrementally updated.
2. Declarative – Make program design very like program specification, this makes rules amendable to verification by inspection, any assignment of variables is affected by unification, no explicit decision or branches.
3. Explanation – A prolog program can be made to explain its own reasoning in a very straightforward manner.
4. First order logic – Mathematically sound vehicle for reasoning and modeling problem areas, helps in formulating logically consistent rules
5. Top down design – Prolog encourages top-down design when writing program, which is exactly the design method of conventional software engineering. Thus, Prolog comes with a sound design methodology which facilitates the construction of expert system rule-bases.

The type of Prolog selected is LPA WIN-PROLOG as it has numerous built-in function as well as additional toolkits, which are not freely provided by other type of Prolog.

3.3.4 Authoring Tools

Both of the authoring tools mention before in section 2.2.5 will be used to aid the building of the expert system.

Microsoft® Help Workshop is essential for building Windows help file to aid the end users. This help file can easily be access by LPA WIN-PROLOG through the 32-bit Windows help subsystem.

Adobe® Photoshop® is not really crucial but will certainly aid the process of editing the mushroom or any related images that will be used in the system.

3.3.5 Knowledge-base

Microsoft® Visual FoxPro will be used as the database or knowledge base as it is proven to be more reliable and functional then Microsoft® Access, as mention in section 2.2.6.2. The versatility of Microsoft® Visual FoxPro will also make it easier to be integrated with LPA WIN-PROLOG. Hierarchical model will be used as the preferred database model as it permits inheritance.

4. System Analysis

4.1 System Requirement Analysis

4.1.1 Functional Requirements

- The user shall be able to search for the specific type of mushroom according to the visible features entered.
- The system shall provide appropriate explanation for the mushrooms' medicinal and nutritional values, if any.
- Every identification process, including the final output is accompanied by suitable images to aid the user's decision.
- An extra menu is available for officially authorized experts that will enable them to key in newly found mushroom, modify or delete existing one in the database.

4.1.2 Non-Functional Requirements

1. Product Requirements

- The system shall be reusable and portable enough to operate on any environment

- The system shall be efficient enough not too sacrifice too much performance such as memory, speed or disk space.
- The system shall be dependant and reliable enough to operate without catastrophic failure

2. Organisational Requirements

- The system development process and deliverable documents shall be delivered according to the schedule, with the specified format.

3. External Requirements

- The system shall not disclose any personal information about the users or experts that is working with the system.

4.2 Tools and Technology Proposed

4.2.1 Software

Below is the list of the software and application that is used to build the expert system:

- Platform – Microsoft® Windows® 2000 Professional
- Knowledge engineering – LPA WIN PROLOG 4040
- Knowledge base / database – Microsoft® Visual FoxPro 6.0

- Help / user manual – Microsoft® Help Workshop 4.03
- Image editing – Adobe® Photoshop® 7.0

4.2.1 Hardware Specifications

4.2.2 Hardware

Below is the description of the minimum components and configuration that comprises the technical environment in which the system will operate:

Below is the description of the components and configuration that is used to build the system:

- Chip set and bus - Intel Pentium III 800 MHz, Intel Mobile 440 BX PCIset, 64 bits DRAM, 4 Mbits Flash EPROM, 66 MHz AGP, 33 MHz PCI.
- Memory – 256 MB SDRAM, 66 MHz clock speed
- Connectors – parallel, IDE, SVGA, PS/2, microphone-in and headphone/speaker jack, USB.
- Speaker – Sound Blaster, 16 bit, 2.5 ohm, 500 Mw
- Video – 256 bit, 8.0 MB
- Display – TFT, 1024 X 768, 65 536 colours maximum

4.3 Run Time Requirements

4.3.1 Hardware Specifications

Below is the description of the minimum components and configuration that comprises the technical environment in which the system will operate:

- Intel Pentium 350 MHz or any AMD processor
- 64 MB RAM
- 5 MB Hard Disk space for program installation
- 24X CD ROM drive (for installation from a CD only)
- VGA monitor supporting 800x600 graphics

4.3.2 Application Software Specifications

Below is the description of the minimum software and application in which the expert system will operate:

- Microsoft® Windows 98, Windows NT 4.0

5. System Design

5.1 System Functionality Design

The representation various types of mushrooms in a computer system starts when the mycologist formulates a set of keys that identify mushrooms based on their features. That knowledge will be incorporate into the knowledge-based system which consists of an underlying intelligent program called the inference engine. The inference engine examines the current knowledge in the knowledge-base and combines it with accumulated facts to derive additional facts and ultimately, the conclusion.

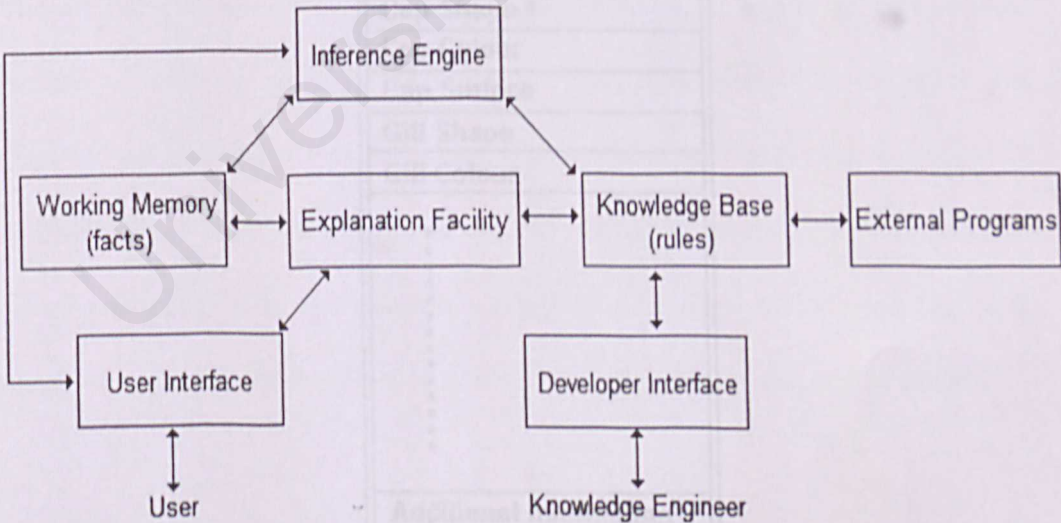


Figure 5.1: System Architecture

The author chooses to represent the mushroom-identification knowledge with the frame-based knowledge representation scheme. Frame hierarchies are similar to object-oriented hierarchies. They allow data to be stored in an abstract manner within a nested hierarchy with common properties automatically inherited through the hierarchy. This avoids the unnecessary duplication of information, simplifies code and provides a more readable and maintainable system. Each frame or instance has a set of slots that contain attributes describing the frame's characteristics. These slots are analogous to fields within records (using database terminology) except that their expressive power is greatly extended. Frames inherit attribute-values from other frames according to their position in the frame hierarchy. This inheritance of characteristics is automatic, but can be controlled using different built-in algorithms.

CLASS: Agaricales
Common Name
Cap Shape
Cap Colour
Cap Surface
Gill Shape
Gill Colour
<div><div></div></div>
Additional Information

Figure 5.2: A Class

5.2 Knowledge-base Design

The knowledge-base contains classes and instances mention before in section 5.1 and it uses simple database architecture. The major steps involved in query processing in such architecture are:

1. Tuples are selected by the DBMS from fact database.
2. While reading data in (a), the tuples are converted into Prolog assertions stored in the main memory buffer.
3. The communication interface calls the related heuristic.
4. The Prolog program reads the assertions and by use of its rule base performs inferences.
5. The result is written in the main memory buffers.
6. The data in (e) are appended to the facts stored in secondary storage.

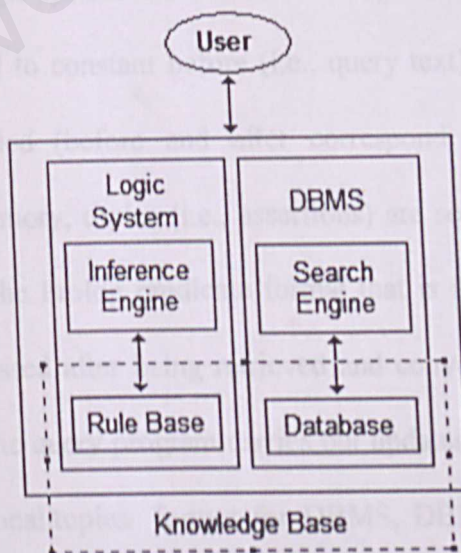


Figure 5.4: Knowledge-base Architecture

Figure 5.5 shows a pictorial version of the interfaces involved in realizing the procedure outlined in Figure 5.4.

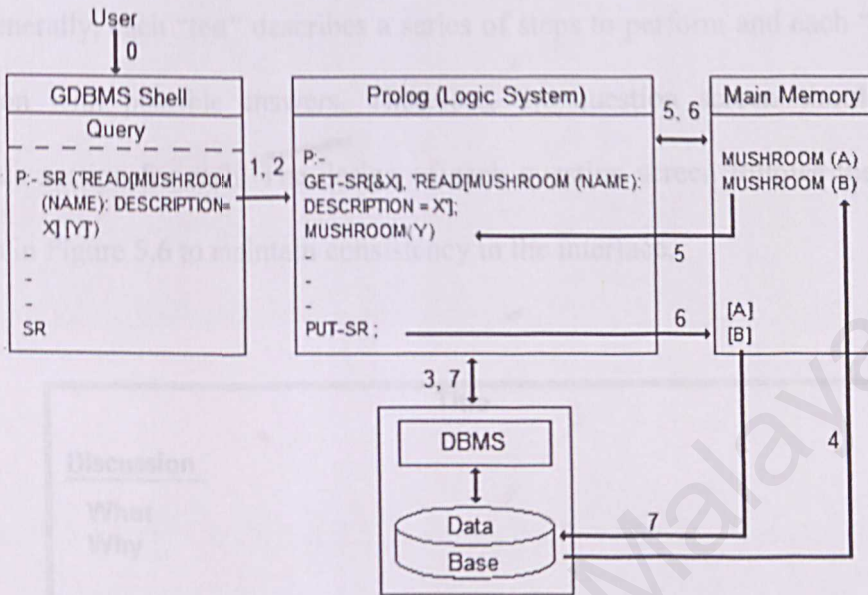


Figure 5.5: System Structure of the Integrated Knowledge Base

The GDBMS shell, upon receiving the query from the user, precompiles the query into Prolog form and activates Prolog to execute the precompiled programs (interface sequence 1 and 2). The logic system and DBMS exchange data via parameter variables. These variables are bound to constant before (i.e., query text) and after (i.e., response data) the DBMS is called (before and after correspond to sequences 3 and 7, respectively). In main memory, tuples (i.e., assertions) are seen at the top and bottom. The top portions shows the Prolog predicate format that is input to the logic program (sequence 5) to be processed after being retrieved and converted from relations in the database (sequence 4). The query program carries out updates and writes new tuple data (sequence 6) in the relational tuples format for DBMS, DBMS is called (sequence 7) for appending the main memory tuples to the relation files.

5.3 Interface Design

Generally, each “test” describes a series of steps to perform and each “results” a question with possible answers. Therefore, the question screen should contain separate section for each. The design of each question screen follows the template shows in Figure 5.6 to maintain consistency in the interface.

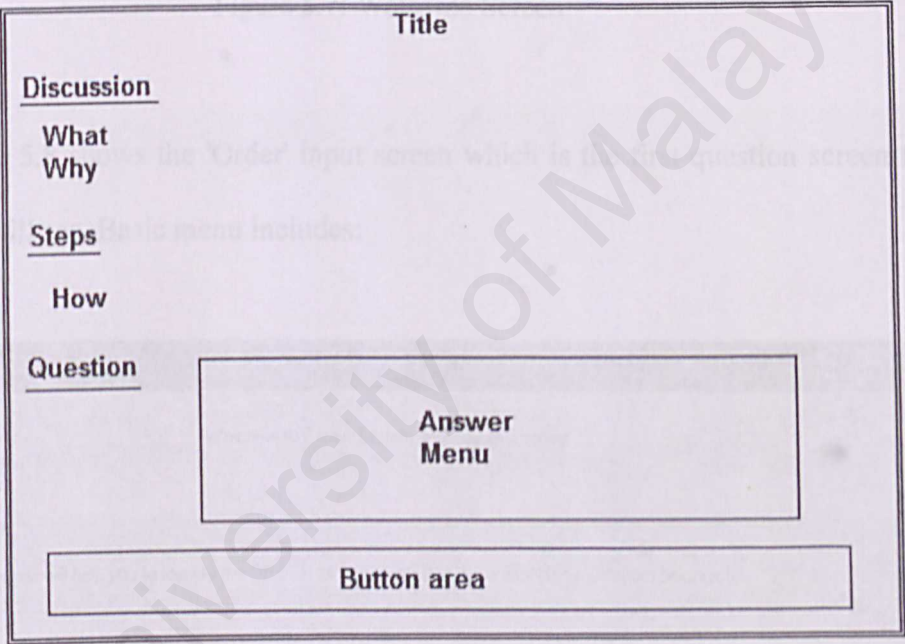


Figure 5.6: Question Screen Template

Figure 5.7 shows the welcome screen which is use to display general information about the program to the user. The program will start automatically when the user clicks the 'OK' button.

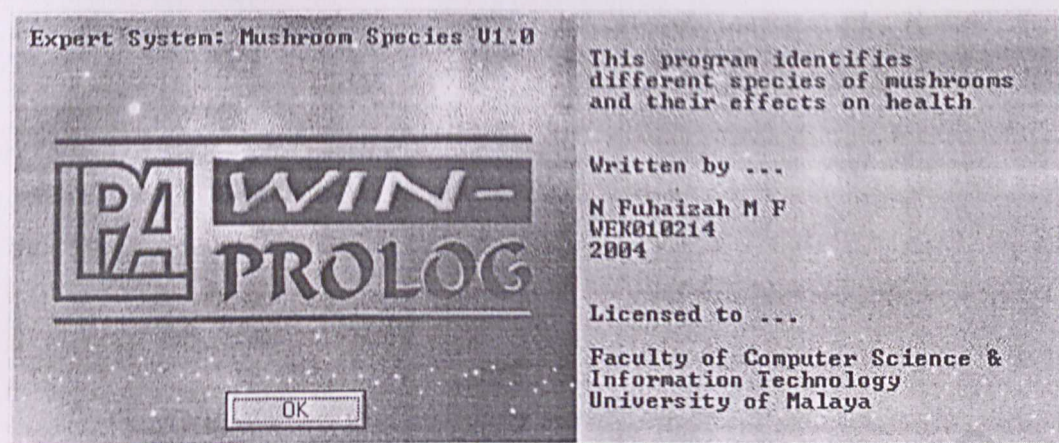


Figure 5.7: Welcome Screen

Figure 5.8 shows the 'Order' input screen which is the first question screen that the user will see. Basic menu includes:

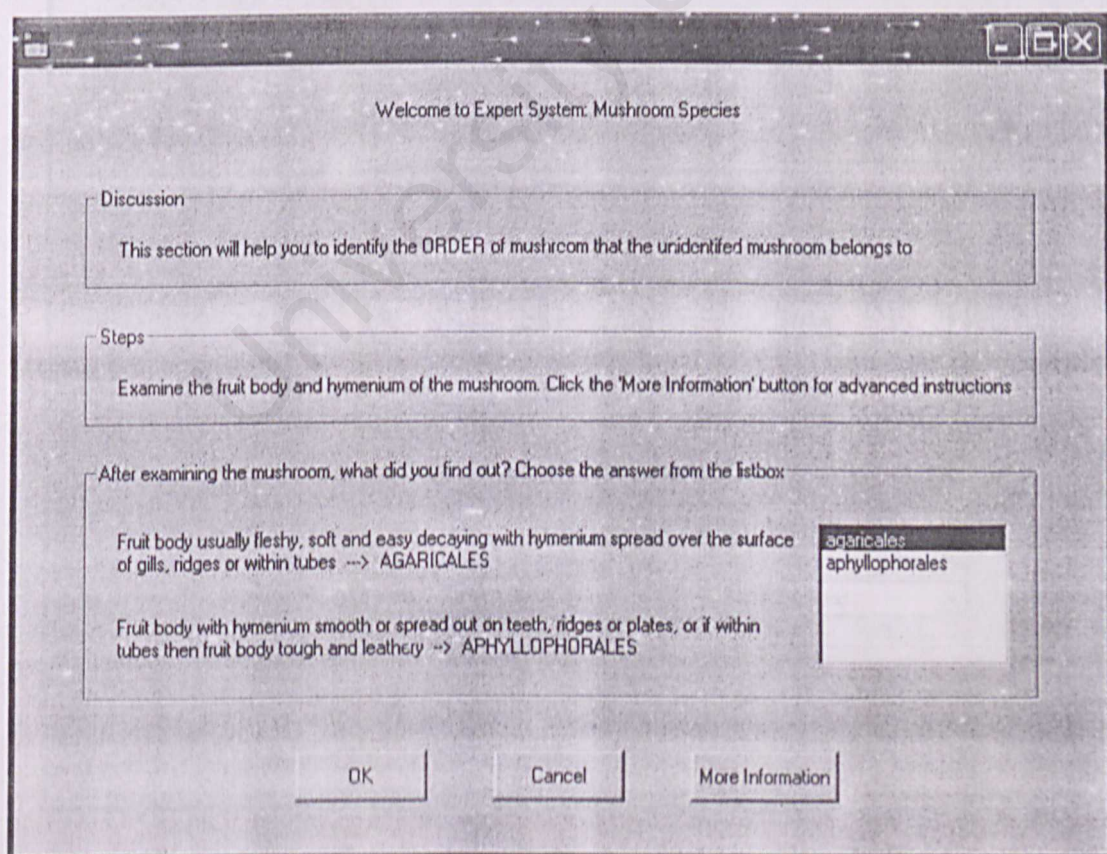


Figure 5.8: The Order Input Dialog

1. Users can select the appropriate features from the list box based on the description given.
2. Users have the option to exit or view the help file to aid the identification at any time.

Figure 5.9 shows the most important question screen. It is use to obtain the mushrooms' visible features from the user. Basic menu includes:

Discussion

The system will now check the macroscopic (visible) characteristics of the unidentified mushroom

Steps

Examine the general features (cap, gill, stem) of the mushroom. Use 'More Information' for advanced instructions

What Macroscopic Features does the mushroom have?

Cap Shape	convex	Stem Surface	smooth
Cap Colour	white	Veil	none
Cap Surface	dry	Ring	none
Gill Shape	adnate	Volva	none
Gill Colour	white	Smell	none
Stem Shape	equal	Distribution	solitary
Stem Colour	white	Habitat	ground

Continue Cancel More Information

Figure 5.9: The Macroscopic Features Input Dialog

1. Users can select the appropriate features from the combo boxes. This approach is easier and avoids typing errors or illegal answers.
2. Users have the option to exit or view the help file to aid the identification at any time.

Figure 5.10 shows the final question screen. Basic menu includes:

Discussion

Finally, the system will check the microscopic characteristic of the unidentified mushroom

Steps

Make a spore print and then examine the spores using a microscope. Use 'More Information' for advanced instructions

After examining the spores, what did you find out?

Spore Print: white

Spore Features: ellipsoid, globose, hyaline, oblong, oval, ovoid, smooth

Search Cancel More Information

Figure 5.10: The Microscopic Features Input Dialog

1. A combo box and list box is used again for selecting the appropriate features.
2. Users have the option to exit or view the help file to aid the identification at any time.

Figure 5.11 shows the output display produce from the all the question screen. Basic menu includes:

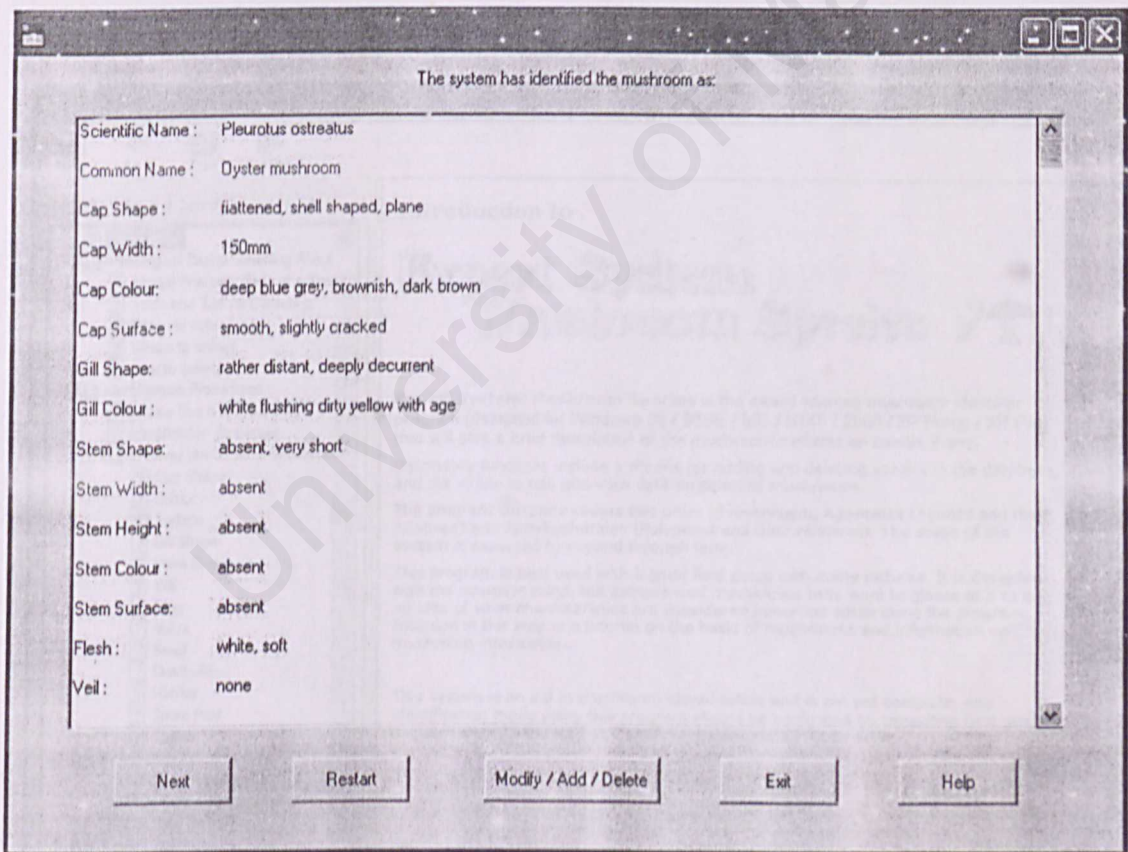


Figure 5.11: The Output Display Dialog

1. All the features entered by the user and inferred by the system, for reference.

2. If there are two or many species that share the same features, users can view other species by pressing the 'Next' button.
3. The 'Modify/Add/Delete' button is for opening and organizing the mushrooms record in the database.
4. Users have the option to exit or restart the system at any time.

Figure 5.12 shows the help display produce when the 'More Information' button is clicked. It is basically used to display related help information about the system and the identification process to aid the user.

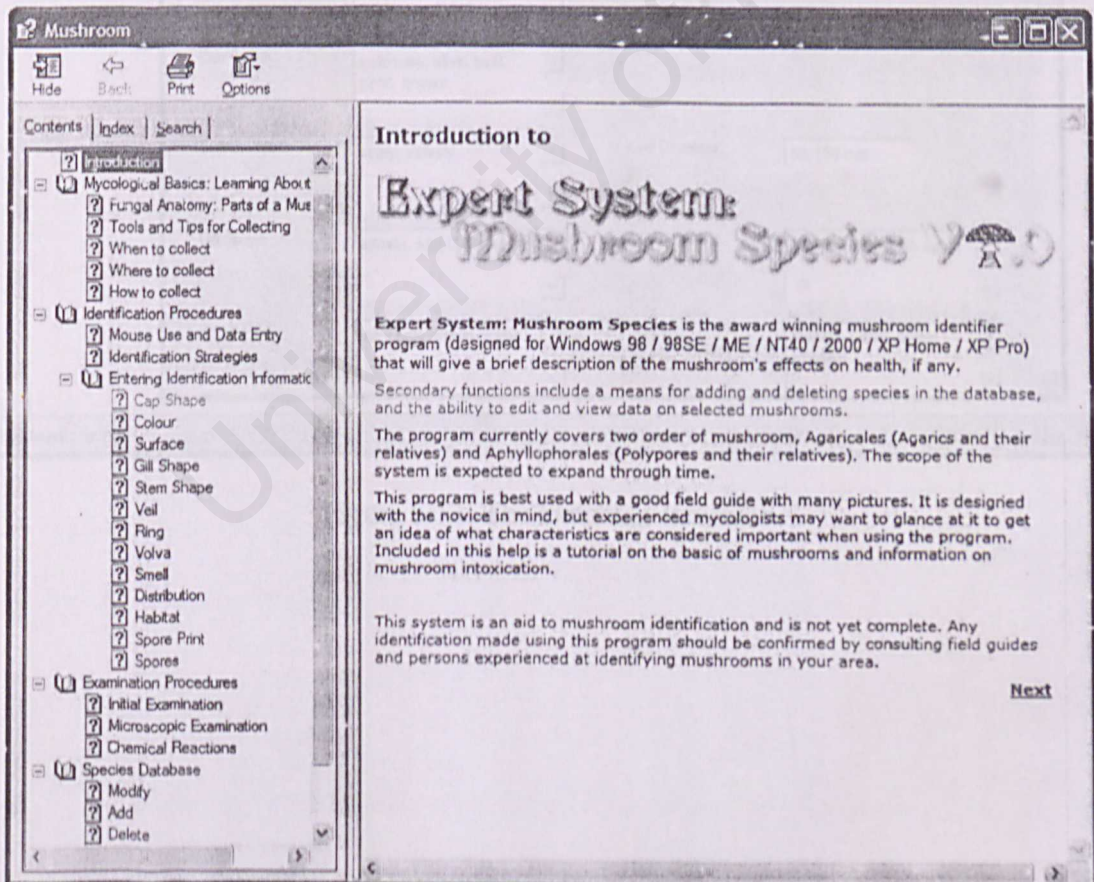


Figure 5.12: The Help Dialog

Figure 5.13 shows the database interface design. Users are accompanied with basic menus like 'Find', 'Print', 'Add', 'Edit', 'Delete' and 'Exit' to organized the data in the database.

Mushroom

File Edit Tools Favorites Go Window Help

AGARICALES

Agarics and their relatives

Scientific Name:	Armillaria mellea		
Common Name:	Honey mushroom		
Cap Shape:	convex, flattened, slightly depressed		
Cap Colour:	yellowish, olive, buff, send, brown		
Cap Surface:	scaly, velvety	Cap Diameter:	50-150 mm
Gill Shape:	adnate, slightly decurrent	Gill Colour:	whitish, brownish spots
Spore Print:	very pale cream colour	Spores:	medium-sized, hyaline, ellipsoid

Skip to next record

Figure 5.13: The Database Interface

6. System Implementations

6.1 Overview

When designing a frame based system, everything is tough of as an object. Following the first meeting with the expert, the major objects involved in the problem were listed. After identifying the objects, the next thing to do is to look for a way of organizing them. This step involves collecting similar objects together in a class-instance relationship, and defining various ways that object communicates with each other.

6.2 Define the classes and instances

Classes and instances are defined within the database. Firstly, the field is determined based on the facts of the mushrooms features. The fields are then organized into tables using the Table Designer. The mushrooms' scientific name is defined as the primary key to prevent duplication of the same species.

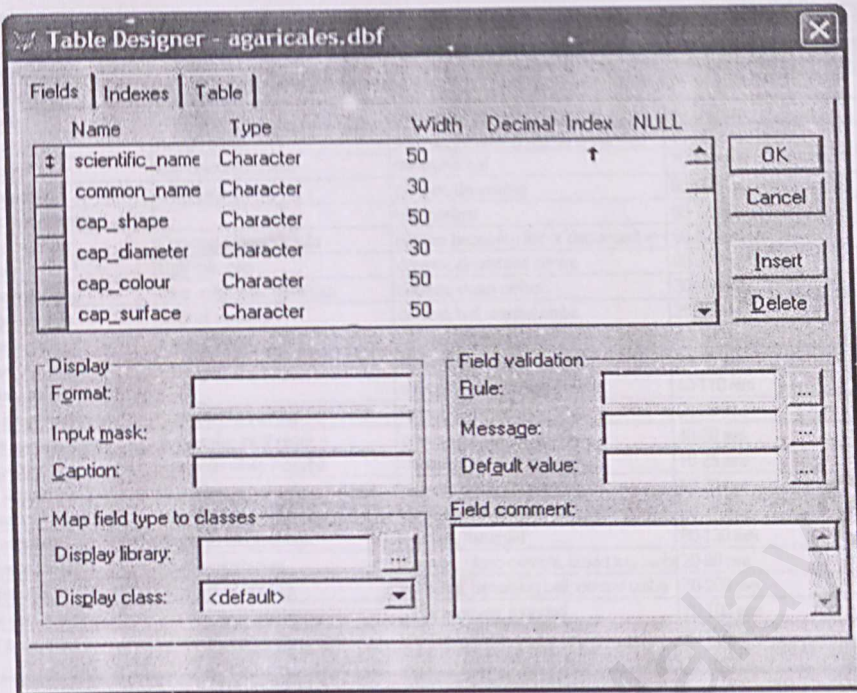


Figure 6.1: Table Designer

Relationships between tables were then defined, if any using the Database Designer

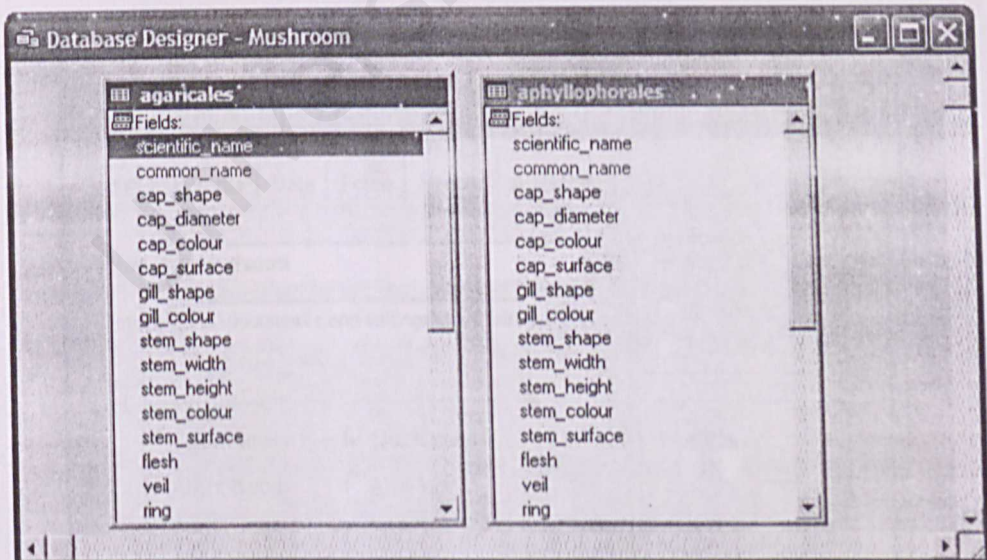


Figure 6.2: Database Designer

After refining the design, the existing data is added to the tables.

Agaricales				
Scientific name	Common name	Cap shape	Cap diameter	Cap color
Laccaria velutina	Weeping widow	convex, expanded with obtuse central	45-90 mm	dull clay
Lepista nuda	Wood blewits	depressed, flattened, rounded	70-100 mm	blue like
Lecanum scabrum	Rough stalks	convex, becoming slightly expanded	45-15 mm	pale br
Boletus badius	Bay-coloured bolete	hemispherical	70-130 mm	red bro
Paxillus involutus	Brown roll-rim	convex, depressed	50-120mm	ochre,
Cortinarius pseudosalar		bell, conical	60-125mm	brown,
Russula ochroleuca	Common yellow russula	convex becoming flat or depressed at	50-100mm	yellow i
Lactarius turpis	Ugly milk-cap	convex, depressed centre	60-200mm	dark ol
Chroogomphus rutilus	Pine spike-cap, wine-cap	convex, sharp umbo	30-150 mm	wine-c
Mycena galericulata	Bonnet mycena	conical, bell, central umbo	25-50 mm	greyish
Pluteus cervinus	Fawn pluteus	conical, plano-convex, flattened, persis	40-100 mm	dark br
Gymnopilus penetrans		convex, flattened	20-50 mm	golden
Melanoleuca melalucica		convex, flattened, umbonate	40-110 mm	dark br
Clitocybe infundibuliformis	Common funnel-cap	funnel	20-60 mm	yellowis
Hebeloma crustuliniforme	Fairy-cake mushroom	convex, hardly expanding	40-80 mm	pale ye
Inocybe geophylla	Common white inocybe	conical, bell, distinct umbo	10-25 mm	silvery
Laccaria laccata	Deceiver	convex, flattened, depressed	12-28 mm	reddish
Mycena sanguinolenta	Small bleeding mycena	conical, bell-shaped, umbonate	10-17 mm	reddish
Collybia maculata	Spotted tough-shank	convex, flattened	80-130 mm	white, s
Hygrocybe pratensis	Butter mushroom	convex, plano-convex, broad low umbo	20-80 mm	tan, pa
Lepiota procera	Parasol mushroom	rounded, becoming bell, central umbo	70-200 mm	dull bro
Calocybe gambosum	St George's mushroom	plano-convex, rounded	70-100 mm	creamy
Agaricus campestris	Field/Meadow mushroom	convex, plane	30-80 mm	white,

Figure 6.3: Table

Forms and reports were later built using their respective designers. After all of the components are organized, an executable version is built using the Application Builder.

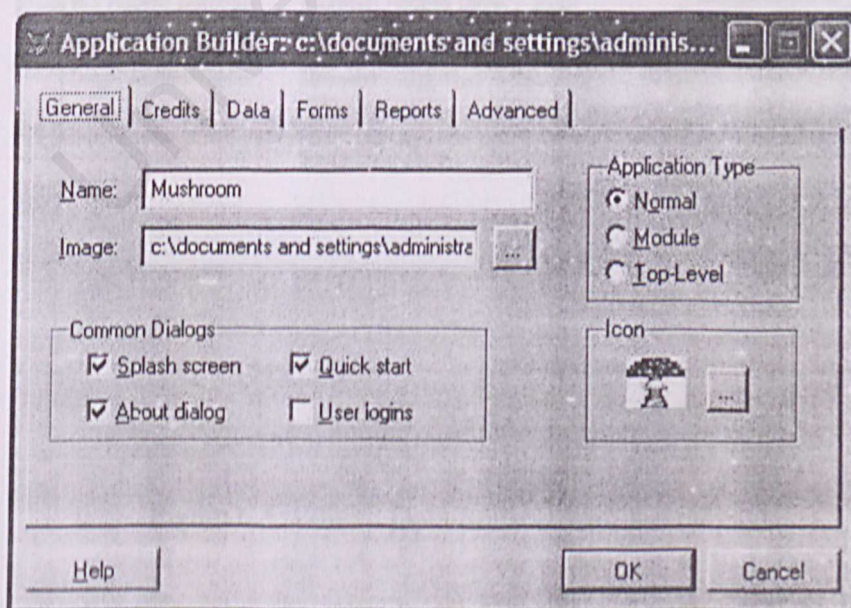


Figure 6.4: Application Builder

6.3 Define the rules and object communication

At this point, the author has classes and instances, each with slots that describe the various objects. The next step is to develop a way of working with this information to satisfy the problem specification, which is to identify the mushrooms using pattern matching rule.

Firstly, the data source will need to be configured using the ODBC Data Source Administrator. The ODBC Data Source Administrator is reachable via the Windows Control Panel.

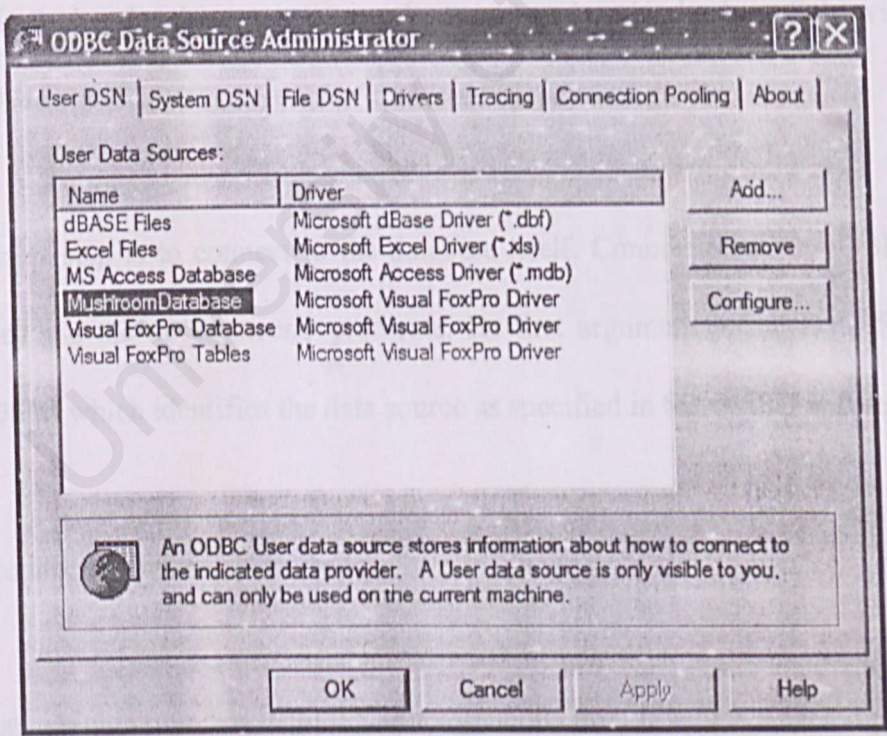


Figure 6.5: ODBC Data Source Administrator

The next step is to load the Prodata Interface using the following code:

```
ensure_loaded( system(dblink) ).
```

Prodata allows database tables to be accessed from Prolog as though they existed within Prolog's environment as unit ground clauses (facts). This facilitates the use of Prolog rules over the contents of the database, with no need to download any part of the database, as all database accesses are done 'on-the-fly'. Backtracking, cut, call, not and all other standard Prolog mechanisms work identically over the table accesses and the internal database, thus achieving the highest level of transparency possible. This software architecture allows a completely different style of database programming that leaves fourth generation languages way behind. All the benefits inherent in Prolog development can now be transferred to the database field without any disadvantages.

The next step is to connect to the database itself. Connection to the DBMS is established via the *db_connect/1* predicate. Its first argument is the Data Source Name (DSN) which identifies the data source as specified in the ODBC before.

```
db_connect( 'MushroomDatabase' ).
```

After that, a single rule that include variables can be used to match selected property values of each instances of each class, as shown below.

`db_tuple(Type, [A, B, op(C-CShape), D, op(E-CColour), op(F-CSurface), op(G-GShape), op(H-GColour), ... , op(U-SPrint), op(V-Spore), W, X, Y, Z]) .`

Following the coding, the system's interface was developed. Creating a new

The *db_tuple* predicate will take the table name and return a tuple as a list comprised of its fields, and will backtrack to retrieve the next record upon failure.

This rule generates an SQL query that retrieves selected information from the database, as shown below.

```
SELECT scientific_name, common_name, .....,  
main_uses, preparations, dosage, side_effects, additional_information, picture  
FROM agaricales  
WHERE (cap_shape LIKE '%convex%') AND (cap_colour LIKE '%white%') AND  
(cap_surface LIKE '%dry%') AND (gill_shape LIKE '%adnate%') AND .... AND  
(spore_print LIKE '%white%') AND (spores LIKE '%ellipsoid%')
```

Figure 6.1: Dialog Editor

A sample of the interface created and exported from the Dialog Editor toolkit is shown through the codes below:

6.4 Design the interface

Following the coding, the system's interface was developed. Creating a new graphical object within the interface actually means creating a new instance of one of these classes. The task starts by selecting from the Dialog Editor Toolkit, with a mouse using the click-and-drag techniques, the type of object desired.

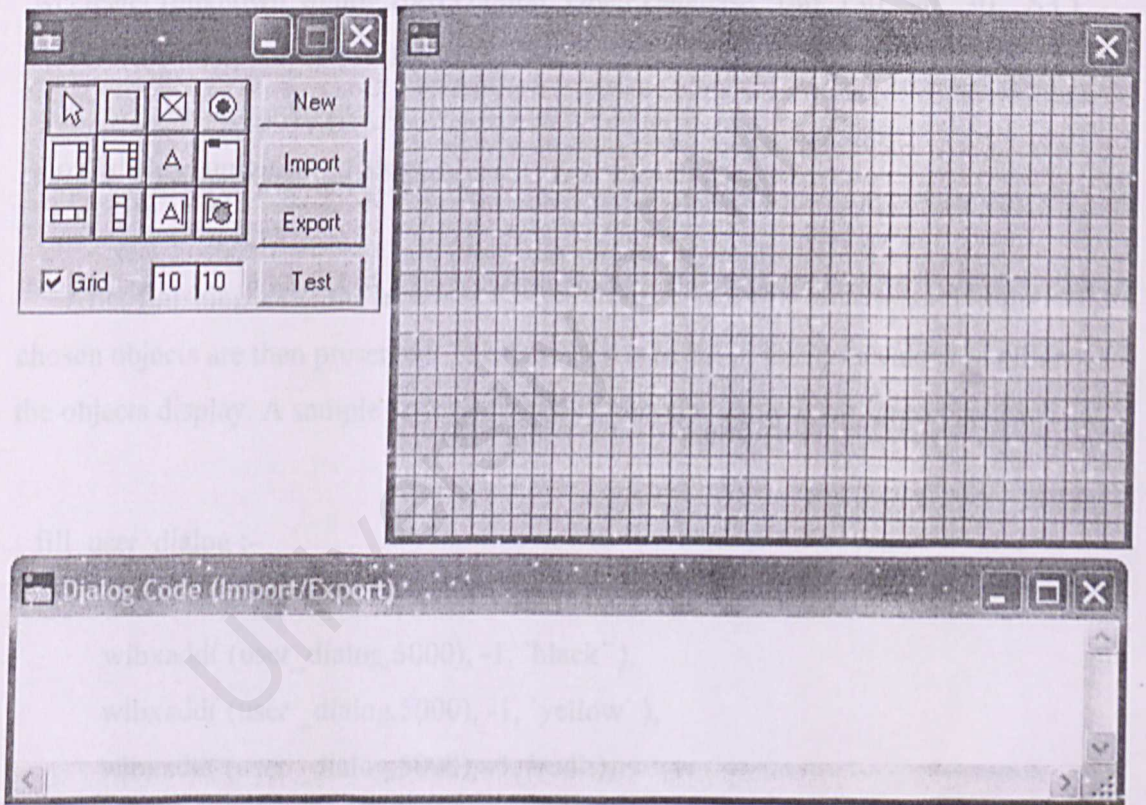


Figure 6.6: Dialog Editor

A sample of the interface created and exported from the Dialog Editor toolkit is shown through the codes below:

sample_user_interface :-

```
_S1=[dlg_ownedbyprolog,ws_sysmenu,ws_caption,ws_border,dlg_modalframe],
_S2=[ws_child,ws_visible,ss_center],
_S3=[ws_child,ws_visible,ws_tabstop,bs_pushbutton],
wdcreate( unknown_dialog, 'Unknown Mushroom', 333, 289, 378, 224, _S1 ),
wccreate( (unknown_dialog,10000), static,
'The mushroom you search for does not exist in the database..!!
What do you want to do? ', 30, 40, 310, 30, _S2 ),
wccreate( (unknown_dialog,1000),button, 'Restart', 30, 130, 70, 30, _S3 ),
wccreate( (unknown_dialog,1001),button,'Open Database',140, 130, 90, 30, _S3 ),
wccreate( (unknown_dialog,1002), button, 'Quit', 270, 130, 70, 30, _S3),
window_handler( unknown_dialog, unknown_handler ),
show_dialog( unknown_dialog ).
```

After finishing all of the interface design, forms containing slots related to the chosen objects are then presented. The next step is to fill in the slots values to tailor the objects display. A sample code to fill the colour slot value is presented below:

fill_user_dialog :-

```
wlbxadd( (user_dialog,5000), -1, 'white' ),
wlbxadd( (user_dialog,5000), -1, 'black' ),
wlbxadd( (user_dialog,5000), -1, 'yellow' ),
wlbxadd( (user_dialog,5000), -1, 'red' ),
wlbxadd( (user_dialog,5000), -1, 'blue' ),
wlbxadd( (user_dialog,5000), -1, 'green' ),
wlbxsel( (user_dialog,5000), 0, 1 ),
wfocus( (user_dialog,5000) ).
```

The ways the values of the slots are chosen are controlled by the following codes:

% given a combobox/listbox returns its selected item

get_selection(Lbx, Selection) :-

 wlbxsel(Lbx, 0, Sel),

 (Sel = 1

 -> wlbxget(Lbx, 0, ItemStr),

 atom_string(Selection, ItemStr)

 ; wlbxfnd(Lbx, 0, '', NextItem),

 get_selection(Lbx, NextItem, Selection)

).

% find the current selection

get_selection(Lbx, 0, Selection) :-

 !,

 fail.

get_selection(Lbx, Item, Selection) :-

 wlbxsel(Lbx, Item, Sel),

 (Sel = 1

 -> wlbxget(Lbx, Item, ItemStr),

 atom_string(Selection, ItemStr)

 ; wlbxfnd(Lbx, Item, '', NextItem),

 get_selection(Lbx, NextItem, Selection)

).

Some graphical objects like buttons are used to link to predefined function like opening the database, opening help files or popping message windows.

% on a database button open database

```
display_handler( ( display_dialog, 1002 ), msg_button, _, _ ) :-  
    exec( 'database\mushroom.exe', "", _ ).
```

% on a help button display help file

```
display_handler( ( display_dialog, 1004 ), msg_button, _, _ ) :-  
    exec( 'help\hh.exe', 'help\mushroom.chm', _ ).
```

% on a exit button close the program

```
display_handler( (display_dialog, 1003), msg_button, _, _ ) :- quit.
```

%exit the system

quit :-

```
( wait( 0 ),  
  wdict( Windows ),  
  \+ member( windows, Windows ),  
  msgbox( 'Expert System', 'Do you want to try again?', 36, Yes ),  
  ( Yes = 6  
    ; halt  
  )  
  ; halt(1) ).
```

How the buttons operate are controlled by the window handler

% pass all other messages to the default window handler

```
display_handler( Window, Message, Data, Result ) :-  
    window_handler( Window, Message, Data, Result ).
```

7. Testing and Evaluation

7.1 Overview

As the project proceeds, the system will need to be periodically tested and evaluated to ensure that its performance is converging towards established goals. The task of testing expert systems is unlike that found for conventional programs where the verification of the software is of primary concern. Verification studies attempts to determine whether the program completely satisfies initial requirements. Conventional programs usually have well defined specifications that can be measured according to some objective standard. Expert systems on the other hand are designed for problems that do not have a clear right or wrong answer. Therefore, a “gold standard” does not exist that can be compared with the system’s results. Because the lack of gold standard, the evaluation process is more concerned with system validation and user acceptance. Validation efforts determine if the system satisfactorily performs the intended task – a relaxation of the stricter verification process. User acceptance efforts are concerned with issues that impact how well the system addresses the needs of the user. The testing and evaluation stages used are based on MYCIN evaluation techniques.

7.2 System Validation

If the expert system is correctly designed, it should derive the same results as an expert reason in a manner similar to that of the expert. Therefore, validation efforts address the following:

- Validating the system's results
- Validating the system's reasoning process

7.2.1 Validate Results

Formal testing usually involves the use of a test case. There major considerations used to validate the results of an expert system are:

- Test criterion

In order to judge whether the project has successfully met its goal, a criterion is usually established against which the project is assessed. A different approach relies on comparing the system's performance relative to that of the domain expert. Important points of this evaluation are:

1. Relative comparison – Given the same test cases, the expert system was able to do as well or faster then the human expert.
2. Evaluation requires judgment - In general, when evaluating the results provided by an expert system, users need to make a judgment call on

the correctness of the result. Users were given a set of possible evaluation responses to fill out. The results were then used to judge the performance of the system and to decide whether further development is necessary.

- Test cases

The expert system was first tested with typical problems from the domain, for example common everyday mushrooms, and then the ones that are unusual. The system is proven to be worth-while in both problems areas.

- Evaluators

Evaluators are end users who are not associated with the projects. Other experts are not included because it was difficult to obtain the cooperation of so many individuals. Some evaluators were biased against a computer program that is designed to model human decision making. They do not think that the computer really knows what the mushroom looks like and may produce inaccurate matches. Inspired by the Turing Test (Turing 1950), the author presented the evaluators responses of a computer along with those from a human. Later, it is proven that the computer can be as intelligent as a human, as the evaluators cannot distinguish the computer output from that of a human.

7.2.2 Validate Reasoning

Besides evaluating the results, evaluators wanted to know if the system is getting the right answers for the right reason. The main reason for this is the limited number of test cases that might actually be used during evaluation studies. Two approaches are used to validate the system's reasoning.

- On the macro level, the evaluators studied the results of various subissues that led to the final result. Credibility of the system was established as the evaluators are convinced that correct performance is a product of intelligent reasoning.

On the micro level, the evaluators were able to trace back through all of the rules used for the case that led to the result and verify their correctness. This approach is similar to that used by system designer during debugging of the knowledge base.

7.3 User Acceptance

Perhaps the ultimate test of an expert system is will it be used? Therefore, a major part of any expert system evaluation study must address the needs of the user.

Important issues considered to evaluate the degree of user acceptance are:

- Ease of use
- Clarity of question
- Clarity of explanation
- Presentation of result

7.3.1 Ease of Use

Since the users are all experienced computer users, the interface design does not seem to give them any trouble and was well understood. However, it was redesign several times to make sure that even inexperienced computer users are able to benefit from the system.

7.3.2 Clarity of Question

The performance of an expert system is strongly dependent on the information it receives from the user. Though there were some hesitations in providing the answers, the users were able to finalize the process after several explanations.

7.3.3 Clarity of Explanation

The system does not include explanations of “why” some question is being asked and “how” some conclusions were reached. However, the users were able to trust the final result of the search based on the mushrooms’ facts itself.

7.3.4 Presentation of Results

The system’s presentation of the final results is just a single and sometimes multiple recommendations. Though it does not provide pictures, the result was proven to be clear and meet the user’s needs.

8. Discussion

8.1 Problems and Solutions

8.1.1 Problems

During the knowledge acquisition phase, two problems were faced. First, because of the vast experience and knowledge, the expert had given too many information. To use all this information would result in a really complex system that would take longer time to develop. The author had to extract the relevant knowledge that is suitable with the skills and time available. To do this, references like books and the internet were used, and this is where the second problem occurs. Books available are mostly old publications and from overseas, which results in inconsistent knowledge that sometimes contradicts the existing Malaysian information. Information from the internet sometimes provides incomplete or even incorrect knowledge.

Other problems were faced during the design phase. Too little information is available for the frame-based knowledge representation technique, as many existing expert system uses rule-based approach. This makes it hard to design the system and takes longer time to build it. Interface design is also a major problem because the author had no experience building graphical user interface with

Prolog before. Designing it takes more time than planned and this leaves insufficient time for other task to be fulfilled.

8.1.2 Solutions

Although there are several problems mentioned, the author had successfully found a way out by applying these simple steps:

For knowledge acquisition phase:

1. Books and other reference used must be proven to be reliable by verifying the author and the publisher. References that are recently published are preferred as the source of information is updated.
2. Any confusing knowledge is referred back to the expert for verification.

In some case where the expert cannot remember everything, step 1 is applied again.

For design phase:

1. More references are used and more research time is put on to it.

8.2 Advantages and Disadvantages

8.2.1 Advantages

Since the system uses object communication rather than rules, it has an efficient way of encoding procedural knowledge. This permits the use of variables within the rules in the form of pattern matching statements that do the work of many standard rules. A single pattern matching rule can scan all the instances of a class. Working with only one rule enhances the maintenance and debugging of the system. Instances can be freely add or delete from the problem without touching the rule. The rules can also be modified without changing the frames.

The system also provides a well defined help file that includes all the relevant information of mycology. Additional information provided will benefit those who wish to understand more about the mushroom identifications itself.

8.2.2 Disadvantages

Due to some misidentified information, the system cannot be turn into an executable version because the existing Prolog version available at the faculty does not have the application builder function. Therefore, users who wish to evaluate the system must have Win-Prolog to run it.

Further more, the system does not have picture integrated within it, including the final display which is suppose to give a picture of the mushroom that the system has identified.

Also, as the knowledge base gets bigger, it can be difficult to debug a system where actions flow through a large number of objects. When something goes wrong, you will need to trace back through all of the actions to locate the problems. It will also take a longer time for the system to search and identify the mushroom.

8.3 Future Plans

The purpose of this project was to show the utility of the idea of developing an expert system for mushroom identification. Having made the point, the next step is to enhance the program for a real-life use. This involves updating the key with more discerning features, so that not only it identifies the existing mushroom (currently within the capability of the program) more accurately, but also identifies a larger set of mushroom, which would make the program much more acceptable. Such improvement of the program could be done by either re-building the key from the scratch, or by letting real experts (mycologist) test the system and asking them to see if some mushroom is misidentified or if some mushroom are not being identified by it. When such situations occur, update the key and then update the program accordingly. The incremental updating would benefit the system because there will probably never be a really 'exhaustive' key for all possible mushroom. However, the author expects the system to keep on improving over time.

The presentation of results can also be enhanced. For example, it can be possibly rank ordered or accompanied with a numeric reflecting the level of believe in the results. This is essential to provide correct identification, especially when there are several slightly identical mushrooms found to meet the criteria.

Any useful system nowadays is also expected to have a higher level graphical user interface with graphical/image-based explanations. The author plans to enhance the program toward this direction for the ease of use.

Appendix A

Glossary of Expert System Terms

Abduction	An uncertain inference
Algorithm	A set of step-by-step instruction for accomplishing task
Attribute	A property of an object
Best-first search	Search technique that uses knowledge about the problem to guide the search
Branch	Connection between nodes in a tree
Breadth-first search	Search technique that looks for a solution along all of the nodes on one level of a problem space
Class	A collection of objects that share common properties
Clause	A conditional statement held in the premise part of rule
Data driven	Inference method where data is obtained and the system determines what it can conclude from this information. Also called forward chaining
Declarative knowledge	Descriptive or factual knowledge
Deduction	Coming to a conclusion by the process of reasoning deductively
Depth-first search	Search technique that explores each branch of a search space to its full vertical length, and then proceed using chosen rule of search
Domain	The problem area
Domain expert	A person who possesses the skill and knowledge to solve a specific problem in a manner superior than others
Fact	A declarative assertion or statement that has the property of being either true or false
Fire	To activate the conclusion of a rule if the premises are true
Frame	Knowledge representation method that associate an object with a collection of features

Goals	A hypothesis to prove or node in search space containing solution
Goal driven	Inference technique that begins with a goal and works backwards through the rules in an attempt to prove the goal. Also called backward chaining
Heuristic	Knowledge, often expressed as a rule of thumb
Induction	Inducing rules from knowledge contained in a set of examples
Inference	The process of deriving new information from known information
Inference engine	Processor in an expert system that matches facts contained in the working memory and the domain knowledge contained in the knowledge base, to draw conclusion about the problem
Instance	A specific object from a class of objects
Knowledge	A collection of facts, rules and concept used to reason with
Knowledge acquisition	Process of acquiring, organising and studying knowledge
Knowledge base	Part of an expert system that contains the domain knowledge
Knowledge engineering	The process of building an expert system
Knowledge representation	The method used to encode knowledge
Predicate	A statement about the subject of a proposition
Rule	A method of representing knowledge consisting of premises and a conclusion
Rule-based system	A computer program that processes problem-specific information contained in the working memory with a set of rules contained in the knowledge base
Semantic network	A method of knowledge representation using a graph made up of nodes and arcs
Shell	Development package that has all the facilities for building an expert system
State space	A graphic representation of all the potential problem states
Working memory	Part of an expert system that contains the known facts of a given session with an expert system

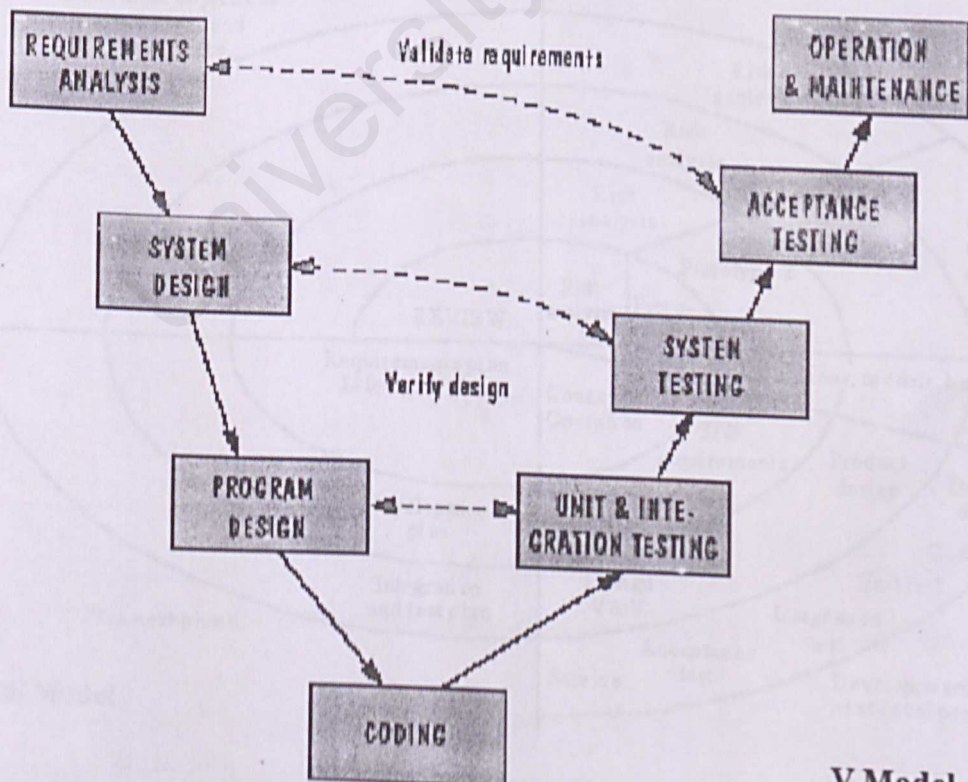
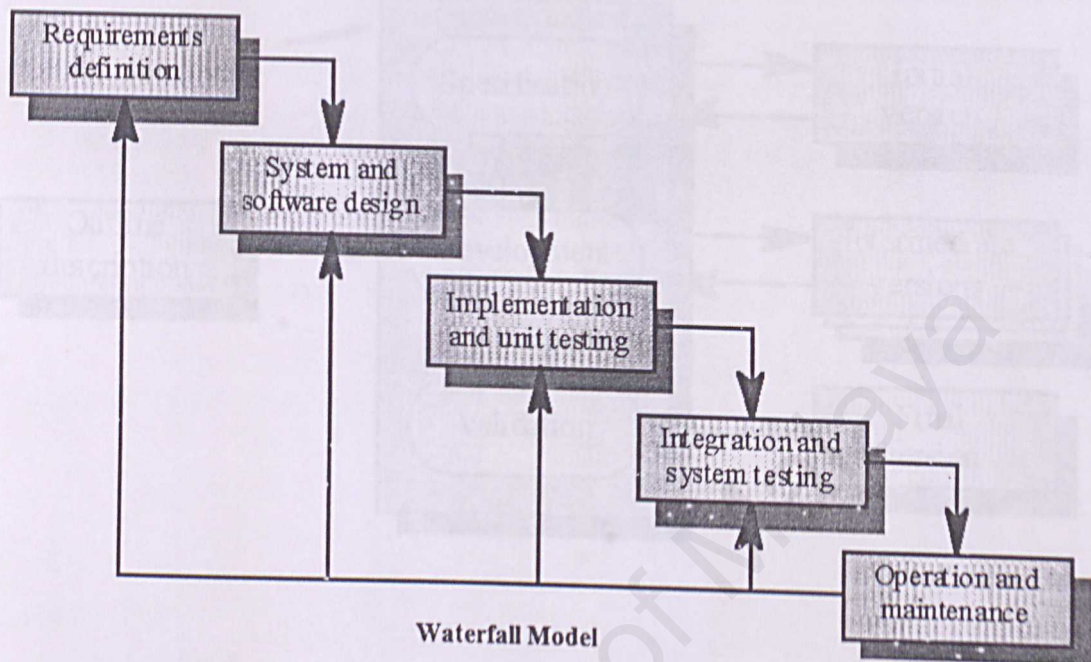
Appendix B

Glossary of Common Mycological Terms

Acomycotina	Asci, containing ascospores, result from sexual process
Ascospores	Sexual spores of ascomycetes
Basidioma	Synonym for basidocarp (plural, basidiomata)
Basidiomycotina	Fungi in which sexual process involves the production of haploid basidiospores born on a basidium
Basidiomycetes	Fungi that have fruit bodies (basidiocarps), plus a few yeast
Basidiospores	Sexual spores of basidiomycetes
Basidium	The enlarged terminal cell of a hypha which bears basidiospores, in basidiomycetes
Deuteromycotina	Fungi imperfection, classified on the basis of asexual morphology
Fruit(ing) body	The large spore bearing structures in Ascomycetes and Basidiomycetes
Gasteromycetes	Fungi that have basidiospores that are not actively launched
Hymenium	A surface of a fruit body, which sexually-produced spores is born
Hymenomycetes	Fungi that have a hymenium
Hypa	The tubular cell growing at one end which is the development unit of mycelium
Mastigomycotina	Fungi that have motile spores (zoopores) or gametes
Mycelium	The mass of hyphae, not in the form of large structures like mushroom, of which the fungi are mainly composed
Teliomycetes	The rust, so called because the rust coloured masses of spores
Ustomycetes	The smuts, important plant pathogens
Zoospore	Spores which can swim in water using one or two flagella
Zygomycotina	Fungi that have large resting spores, results from sexual process

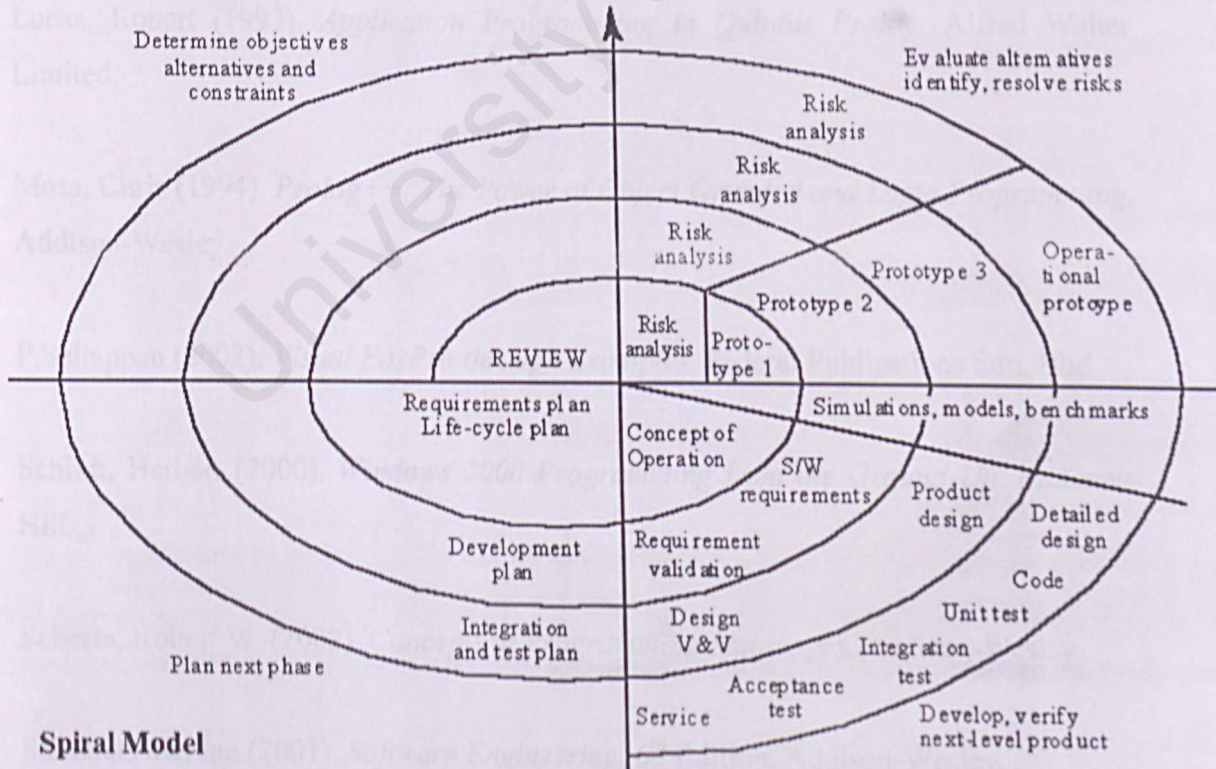
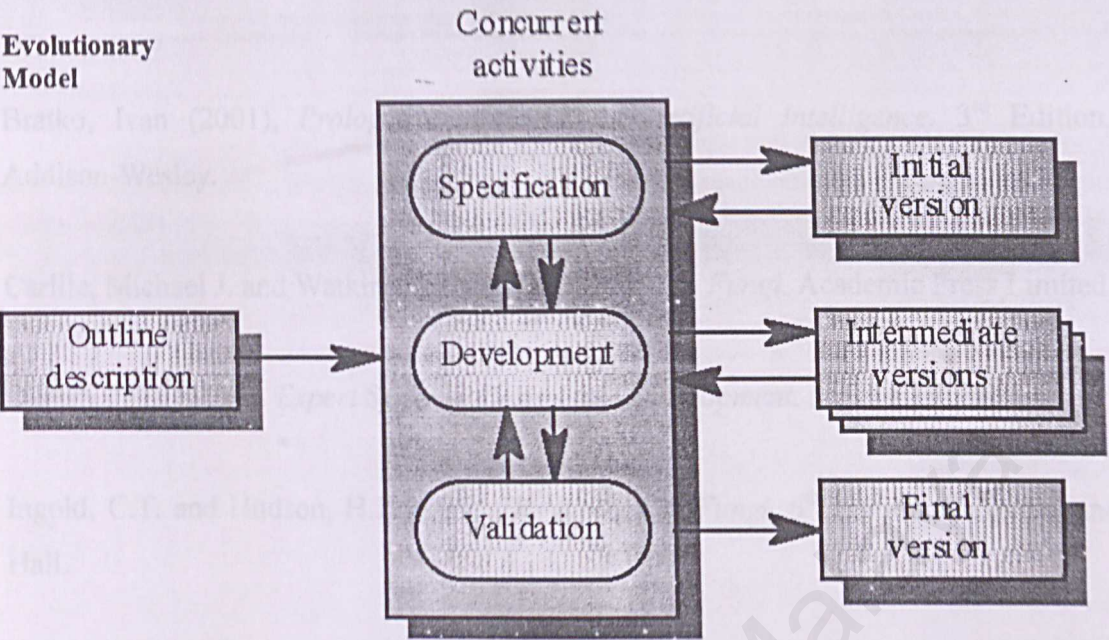
Appendix C

Illustrated Figures of the Development Models



V Model

Evolutionary Model



Spiral Model

Reference

- Bratko, Ivan (2001), *Prolog Programming for Artificial Intelligence*. 3rd Edition. Addison-Wesley.
- Carlile, Michael J. and Watkinson, Sarah C. (1996). *The Fungi*. Academic Press Limited
- Durkin, John (1994). *Expert Systems: Design and Development*. Maxwell Publishing
- Ingold, C.T. and Hudson, H.J (1995). *The Biology of Fungi*. 6th Edition. Chapman and Hall.
- Karahan, Esen O. (1990). *Database Management: Concepts, Design and Practice*. Prentice Hall.
- Lucas, Robert (1993). *Application Programming in Quintus Prolog*. Alfred Walter Limited.
- Moss, Chris (1994). *Prolog++: The Power of Object Oriented and Logic Programming*. Addison-Wesley.
- P.Sellappan (2002). *Visual FoxPro through Examples*. Federal Publications Sdn. Bhd
- Schildt, Herbert (2000). *Windows 2000 Programming from the Ground Up*. McGraw-Hill.
- Sebesta, Robert W. (2002). *Concepts of Programming Languages*. Addison-Wesley
- Sommerville, Ian (2001). *Software Engineering*. 6th Edition. Addison-Wesley.

Watling, Roy (1973). *Identification of the Larger Fungi*. Hulton Educational Publications Ltd.

Whitten, Jeffrey L., Bently, Lonnie D. and Dittman, Kevin C. (2002). *System Analysis and Design Methods*. 5th Edition. McGraw-Hill.

LPA Win-Prolog at <http://www.lpa.co.uk/>, accesses July 2003

Automated Fungi Identification at <http://www.agarics.org/Index.jsp>, accessed July 2003

An Illustrated Key for the Identification of Fungi, Wild Mushrooms and Toadstools at <http://mycosoft.co.uk/home.htm>, accessed July 2003

Of Mushrooms and Machine Learning: Identifying Algorithms in a PDP Network at <http://www.cnbc.cmu.edu/~medler/papers/mushroom.html>, accessed July 2003

An Interactive Guide to Massachusetts Snakes. at http://www.umass.edu/umext/nrec/snake_pit/, accessed July 2003.

Common Conifers of the Pacific Northwest. at <http://fmc.cof.orst.edu/1110a.html>, accessed July 2003.

Douglas-fir Cone and Seed Insects at <http://www.for.gov.bc.ca/hti/IID/index.htm>, accessed July 2003

Whale Watcher at <http://www.aiinc.ca/demos/whale.html>, accessed July 2003

Mushroom Expert.com at <http://www.bluewillowpages.com/mushroomexpert/index.html>, accessed August 2003