A Case-Based Reasoning System for Classifying Malay Traditional Herbs

By

Ismaily bin Johari

WEK000342

Department of Artificial Intelligence

Under the supervision of

Dr. Syed Malek Fakar Duani

and moderated by

Encik Md Nor Ridzuan bin Daud

Submitted to the

Faculty of Computer Science and Information Technology

University of Malaya

In Partial Fulfillment of the Requirements for the Degree of

Bachelor of Computer Science

Submission Date (5 February 2003)

Abstract

The purpose of this project is to develop a system using case-based reasoning to classify the various types of Malay Traditional herbs. The task of classifying Malay Traditional herbs had always been done by botanist, Malay Traditional herbs practitioner, herbalists and people interested on the herbs. The classification were usually done by identifying the physical attributes of the herbs, based on the features of its leaf, root, flower, fruit and stem. Each of them owns its own unique physical features making the process of classifying essential to determine the type of the herbs.

This project focuses on the methods of indexing the herbs into a case and it into the case library using the case based reasoning techniques. Also being emphasized in this project is the method of matching and retrieval of the cases in order to retrieve the best solution from the case library. Besides that, the other purpose of this project is to evaluate the effectiveness of implementing case based reasoning to classify Malay Traditional herbs.

Acknowledgement

First of all, I would like to thank Mom and Dad for their love, support and wisdom. You both will always be the source of inspiration to me.

I would also like to express my gratitude to my supervisor, Dr. Syed Malek Fakar Duani in providing the guidance, support and advice throughout the development of this project.

Special tanks to Encik Md Nor Ridzuan bin Daud as my project moderator, for his suggestions and comments.

Special thanks also to all my friends in UPM, for their help in providing me with the required information upon completing this project.

Thanks also to all my fellow course mates for sharing their knowledge with me throughout this system development and implementation stage.

Last but not least, I would like to thank Diana who treated my project as if it is her own. Also for being helpful and generous to contribute her idea and shared the same amount of enthusiasm like I do about this whole project.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

List of Figures

List of Tables

1.0 INTRODUCTION

The title of this thesis is "*A Case-Based Reasoning (CBR) system on classifying Malay Traditional Herb* ".The process of classifying are believed to have health value were usually done by botanists, herbalists, biologists and even people interested in Malay traditional herbs. The herbs which have similarities may have the same quality for uses as alternative medicines.

The purpose of this project is to apply Case-Based Reasoning (CBR) techniques on classifying Malay Traditional herbs as the title presents. The system would be able to identify numerous types of traditional herbs, display the herbs unique characteristics, their potential as alternative medicines as well as its origins. The herbs would be categorized to its unique attributes in order to differentiate an individual herb with another. The herbs would be store in the case library and each herbs would be categorized as a single case.

The system would be as reliable as real expert on the subject. The system is able to reason for solutions and is able to justify its findings. The case library is designed to be expandable and can be modified.

1.1 Definition

1.1.1 Case Based Reasoning

Case-based reasoning (CBR) can mean adapting old solutions to meet new demands, using old cases to explain new situations, and using old cases to critique new solutions, or reasoning from precedents to interpret a new situation or create an equitable solution to a new problem. Human are much better at recognizing solutions than generating them. They usually attempt to solve new problems by reusing solutions to old problems.

1.1.2 Malay traditional herbs

Herbs are not drugs, vitamins, minerals or enzymes. Herbs are plants, food and other nutrients. Herbs can also be defined as a green plant without a woody stem that dies down to the ground after flowering or any part of a plant that is being used for food, medicine, scent or flavoring.

Malay traditional herbs are medicinal systems based on herbs which can be found in many parts of the Malay World (including Indonesia, Thailand, Philippines, etc) and is still being practiced as primary medicine by the numerous aboriginal races.

In proper words, the system will apply CBR techniques to classify different types of Malay traditional herbs. The system will store the herbs as cases in the case library and will reason using CBR techniques which we will discuss further later.

1.2 The aims of the system

1. To provide a system which is able classify the numerous types of herbs in Malaysia.
2. To help botanist, biologist or those who are interested identify unknown herbs based on case-based reasoning.
3. To have the possibility to match new herbs which are newly found with existing herbs in the case library in order to find the similarities and the differences.
4. To store the information on various types of herbs which are classified as Malay traditional herbs.
5. To promote the usage of CBR methods on solving everyday problems.

1.3 Objectives of the system

1. To apply case-based reasoning techniques on classifying Malay traditional herbs.
2. To provide a CBR system with a friendly user interface and easy to use.

3. To imitate experts in Malay herbs in ways they reason to a problem justify their solutions in determining Malay traditional herbs.
4. To reduce the dependency on domain experts by developing a system which provides the knowledge of the domain expert and able to provide solutions as would the real domain expert.

1.4 Scope of the system

The scope of the system is important in order to determine the boundary of the underlying subjects to avoid the contents falling out of scope and hence, loss its focus and objectives.

The scopes of the system are;

1. To classify Malay traditional herbs using CBR techniques by storing each herbs as a case in the case library.
2. To determine the characteristics or attributes of each herbs in order to classify and index the herbs as cases.
3. To make the system as reliable as the real domain expert.
4. To make the system expandable and customizable in order to add new data or to correct an existing data.

1.5 Significance of the system

The system is built specially for classifying Malay traditional herbs. The system will be important to classify each herb they found or discovered. A conventional information system will usually try to retrieve the answer by doing blind searching through the database without being able to justify its answer. A CBR system will do reasoning and try to compare the inputs to each case in the case library and try to extract the differences between the inputs and the outputs, and then tries to suggest the best solution. The system

would reduce dependencies on real experts on the subject and reduce paper works and old data retrieval methods.

The system would also be able to store the information of the herbs and this will encourage further studies on traditional Malay herbs as alternative medicines. The herbs were used as medicine for many centuries ago until now and we would not allow the herbs to be forgotten as the synthetic and modern medicines are constantly being introduced. The herbs are believed to have great potential to be explored as great source of medicine.

The system will be used by botanists, homoeopathists, Malay herbs practitioner, doctors, pharmacists and even the public.

1.6 Limitation of the system

The system is constrained to be on classifying Malay herbs only. The purpose is to limit the scope of the system. For future enhancement, we might consider a CBR system which will classify all types of plant, including trees, flowers, mushrooms and others.

Besides that, the system also cannot accommodate the use of natural language in its processing. User will need to input specified keywords which had been defined.

The system will be build as a standalone application. The case library would not be able to be accessed on other computer as it is not being shared publicly. The system may be designed to imitate a real expert on the area but there might be flaws on the expert knowledge. That is why the system is designed to be customizable and expandable.

1.7 Expected Outcome

On the completion of the project, the following are the expected as the outcome of the project;

1. A standalone application for classifying Malay Traditional Herbs.
2. A system which implements Case-Based reasoning in determining numerous types of Malay Traditional Herbs.
3. A system which is able to store past solutions for future problem solving.
4. A system which can choose the best solution among the found solution.
5. To analyze the effectiveness of Case-Based Reasoning in classification of Malay Traditional Herbs.

1.8 Future Enhancement

For future enhancement, we might consider building a web-based CBR application so that the case library may be retrieved and shared within the World Wide Web. A web-based system is flexible in terms that it may be shared publicly and accessed anywhere at any time.

Beside that, we would consider apply this system to a larger domain such as classifying trees, flowers, microorganisms, and even animals. The evaluation of this system will determine the effectiveness of applying Case-Based Reasoning for this kind of problem.

1.9 Project Schedule

This project schedule is important to ensure all the development phases are implemented in an appropriate period of time and the system can complete on schedule. A Gantt chart is a bar chart that shows each task activity, as a horizontal bar whose length is proportional to its time of completion. Gantt chart provides a method for determining the sequence and particular actions, which need to be taken to a given objective.

Activities	Jun	Jul	Aug	Sept	Oct	Nov	Dec	Jan	Feb
Introduction									
Literature Review									
System Analysis									
System Design									
System Coding									
System Testing									
System Documentation									

Table 1.1: Project Schedule

1.10 Overview of Chapters

1.10.1 Chapter 1: Introduction

An overview of the system including the system objectives and system aims. A planning of project schedules well as software and hardware requirements are also included.

1.10.2 Chapter 2: Literature Review

A research on existing CBR systems which are related and the underlying concepts and principles of CBR. This chapter also discusses the nature of the Malay traditional herbs, the uses and the characteristics. Included in this chapter are discussions on the tools that will be used in development of the system.

1.10.3 Chapter 3: System Analysis

An explanation on how to gain the appropriate information and knowledge on building the system and also an analysis of functional and non-functional requirements of the system.

1.10.4 Chapter 4: System Design

Discuss the process of system design, database design, program design and the user interface design.

1.10.5 Chapter 5: System Implementation

Describes the coding and development of the system using the selected tools to make the system as needed.

1.10.6 Chapter 6: System Testing

A description of the testing procedures that are needed on this system. Also the result of each testing procedure.

1.10.7 Chapter 7: System Evaluation

Details the strength of the system that have been developed and explore some possible ideas for future enhancement in the future.

2.0 LITERATURE REVIEW

2.1 Artificial Intelligence

Artificial Intelligence (AI) is a subdivision of computer science devoted to creating computer software and hardware that imitates the human mind. The main goal of AI is to make computer smarter by creating software that will allow a computer to mimic some of the functions of human brain in selected applications or to make computer do tasks which at the moment, people do better. The idea is not to replace human beings, but to provide us more powerful tools to assist the human kind.

There are many problem solving approaches in artificial intelligence and some of the popular approaches are heuristics, fuzzy logic, case-based reasoning, genetic algorithm, artificial neural networks, etc.

2.2 Introduction to Case-Based Reasoning

Case-based reasoning (CBR) can mean adapting old solutions to meet new demands, using old cases to explain new situations, and using old cases to critique new solutions, or reasoning from precedents to interpret a new situation or create an equitable solution to a new problem. Human are much better at recognizing solutions than generating them. They usually attempt to solve new problems by reusing solutions to old problems.

This works as a computational model by creating a library of past cases relevant to a given problem, for example a chef's repertoire of recipes, or a judge's knowledge of past decisions, or an architect's blueprint for different styles of building. However a library must be indexed in some way, so that we can recognize problems similar to the one at hand, and there must be some machinery for adapting the old solutions to the new problem.

A case base reasoner's reasoning depends on five things;

1. The experiences it had
2. Its ability to understand new situation in terms of those old experiences
3. Its adeptness to adaptation
4. Its adeptness at evaluation and repair
5. Its ability to integrate new experiences into its memory appropriately

The less experienced reasoner will always have fewer experiences to work with than the more experienced one. But, the answers given by a less experienced reasoner won't necessarily be worse than those given by the experienced one if the reasoner is creative in its understanding and adaptation and if it has had at least some relevant experience.

The ability to understand a new problem in terms of old experiences has two parts: **recalling** old experiences and **interpreting** the new situation in terms of the recalled experiences. The first we call **indexing problem** which also means finding in memory the experience closest to a new situation. **Interpretation** is the process of comparing the new situation to recalled experiences. When problem situations are interpreted, they are compared and contrasted to old problem situations. The result is an interpretation of the new situation, the addition of inferred knowledge about the new situation, or a classification of the situation.

Adaptation is the process of fixing up an old solution to meet the demands of the new situations. There are several methods of adaptation which had been identified and they can be used to insert something new into old solutions, to delete something or to make a substitution. Creative answer results from applying adaptation strategies in novel ways.

Evaluation and consequent **repair** are important contributors to the expertise of a case-based reasoner. In order to learn from experience, a reasoner would require feedback so that it can interpret what was right and what was wrong with its solutions. Evaluation can

be done in the context of the outcomes of other similar cases, can be based on feedback, or can be based on simulation.

Since the appearance of CBR, it has been applied in a wide range of domains such as [2];

- **Diagnosis**: case-based diagnosis systems try to retrieve past cases whose symptom lists are similar in nature to that of the new case and suggest diagnoses based on the best matching retrieved cases. The majority of installed systems are of this type and there are many medical CBR diagnostic systems.
- **Help Desk**: case based diagnostic systems are used in the customer service area dealing with handling problems with a product or service.
- **Classification**: A case based classifier asks whether the new instance is enough like another one to be assigned the same classification. Rather than classifying new cases using necessary and sufficient conditions, it does classification by trying to find the closest matching case in its case base to the new situation.
- **Decision Support**: in decision making, when faced with a complex problem, people often look for analogous problems for possible solutions. CBR systems have been developed to support this problem retrieval process (often at the level of document retrieval) to find relevant similar problems. CBR is particularly good at querying structured, modular and non-homogenous documents.
- **Design**: Systems to support human designers in architectural and industrial design have been developed. These systems assist the user in only one part of the design process, that of retrieving past cases, and would need to be combined with other forms of reasoning to support the full design process.

2.3 The CBR Cycle

At the highest level of generality, a general CBR cycle may be described by four tasks;

1. Retrieve the most similar case or cases
2. Reuse the information and knowledge in that case to solve the problems

3. Revise the proposed solution
4. Retain the parts of this experience

Each of the four CBR tasks involves a number of more specific sub tasks. An initial description of a problem (top of the CBR cycle) defines a new case. in the retrieve task this new case is used to find a matching case from the collection of previous cases. The retrieved case is combined with the input case- in the reuse task- into a solved case, i.e. a proposed solution to the initial problem. The revise task tests this solution for success, e.g. by applying it to the real life environment or has it evaluated by a teacher, and repaired if failed. This task is important for learning, since the system needs a feedback of how successful its proposed solution actually was. Retain is the main learning task, where useful experience is retained for future reuse, by updating the case base and possibly also the general domain knowledge.

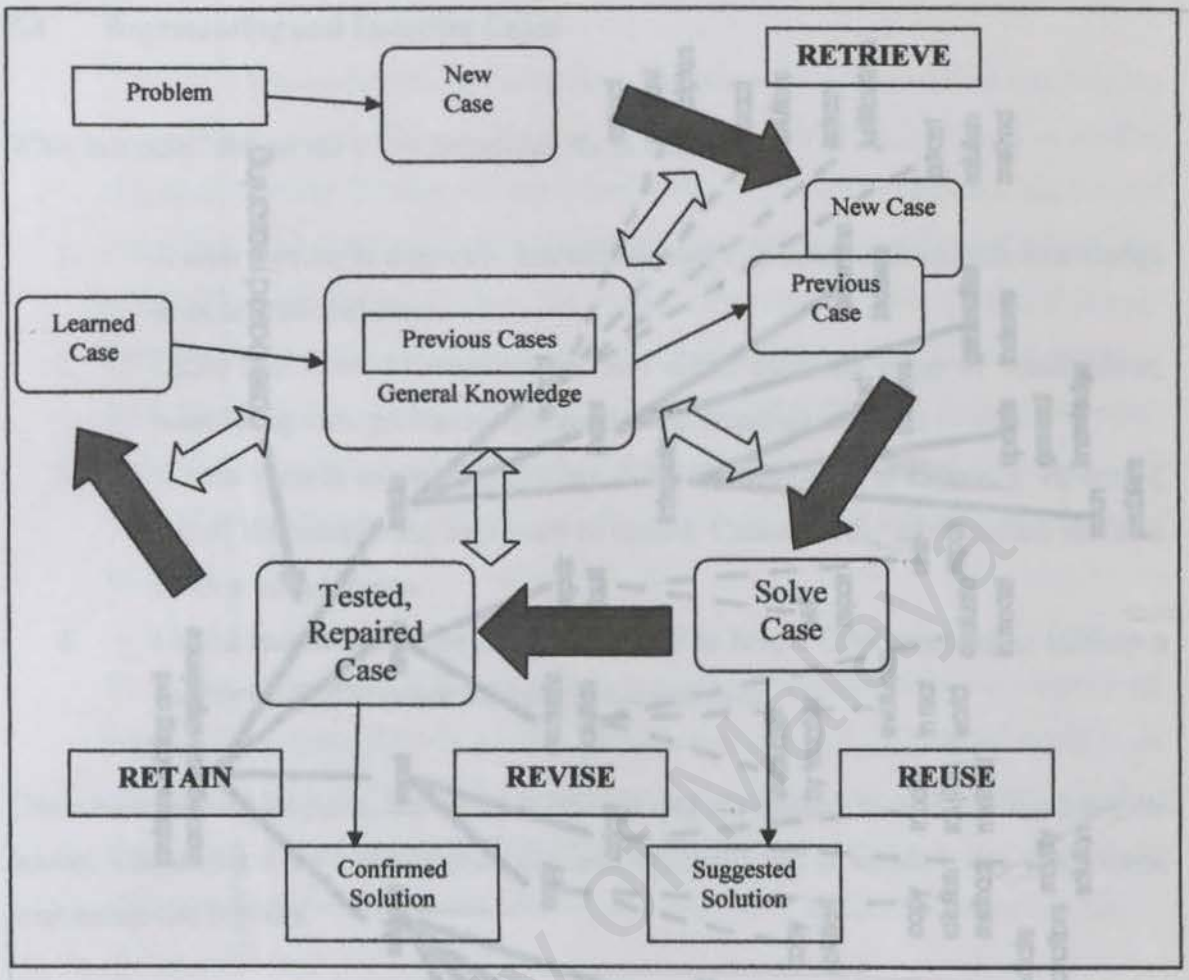


Figure 2.1: The CBR Cycle

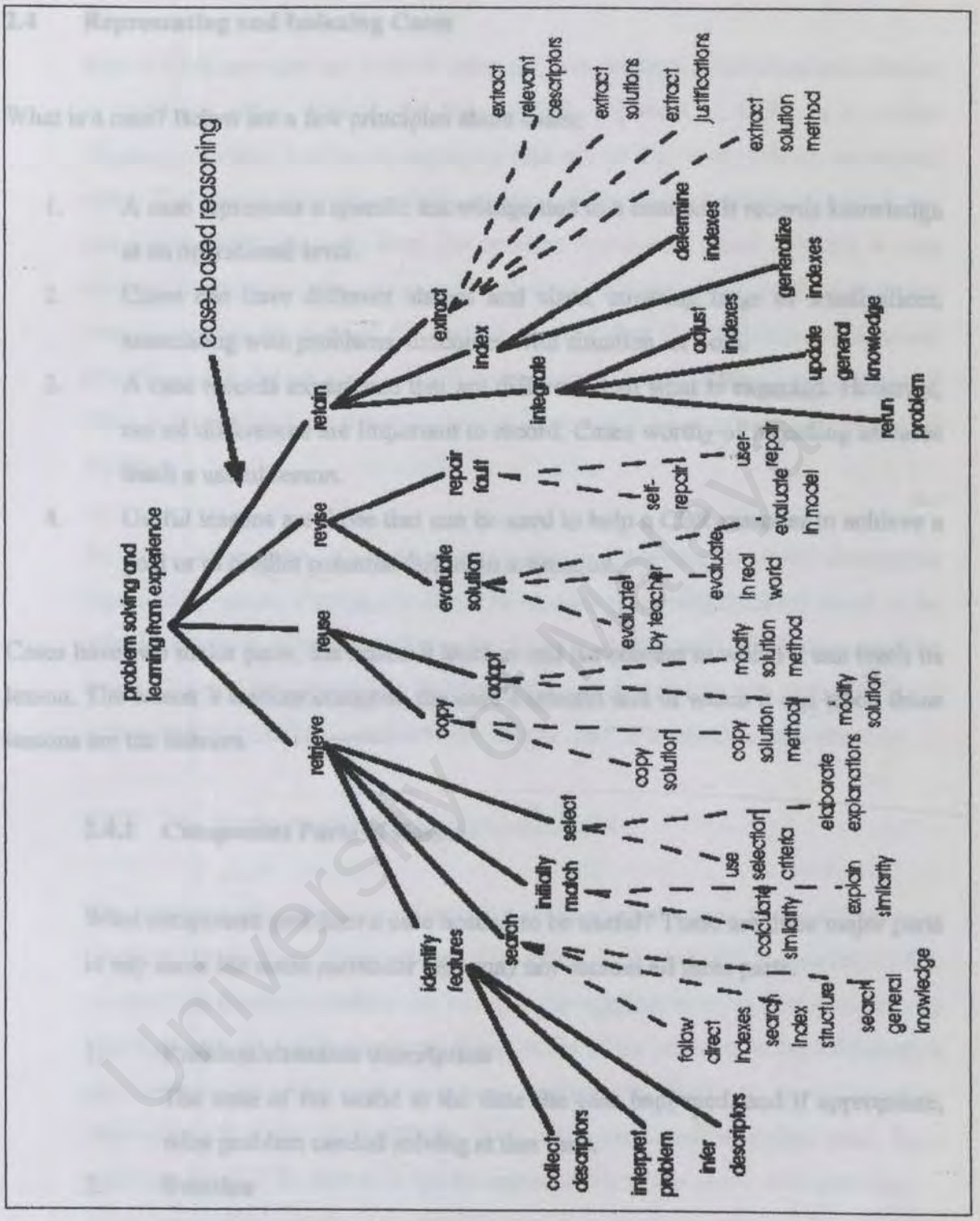


Figure 2.2: Task Method Decomposition of CBR

2.4 Representing and Indexing Cases

What is a case? Below are a few principles about cases;

1. A case represents a specific knowledge tied to a context. It records knowledge at an operational level.
2. Cases can have different shapes and sizes, covering large or small slices, associating with problems, outcomes with situation, or both.
3. A case records experience that are different than what is expected. However, not all differences are important to record. Cases worthy of recording as cases teach a useful lesson.
4. Useful lessons are those that can be used to help a CBR reasoner to achieve a goal or to predict potential failure in a situation.

Cases have two major parts; the lesson it teaches and the context in which it can teach its lesson. The lesson it teaches comprise the case's context and in which it can teach those lessons are the indexes.

2.4.1 Component Parts of Cases

What component part does a case needed to be useful? There are three major parts in any cases but some particular case may not contain all three parts.

1. **Problem/situation description**
   The state of the world at the time the case happened, and if appropriate, what problem needed solving at that time.
2. **Solution**
   The state or derived solution to the problem specified in the problem description, or the reaction to its situation.
3. **Outcome**
   The result after a solution was applied to a particular problem.

Simple CBR reasoner can be built using only the problem description and solution to build the cases. This is the way CASEY [1] work. CASEY is a cardiac diagnosis system; it takes as input descriptions of the new problem to the old ones. The found solution will be adapted in the new problem. CASEY can provide accurate diagnosis from the problem description given. CASEY is very accurate because it only stores good solutions, which is why CASEY does not need an outcome to evaluate the derived solution. But if a CBR reasoner does not contain a well define cases, it will make mistakes during the diagnosis, and if there is no feedback on the mistake, the system will keep on repeating the same mistake.

By using these three kinds of case components provides a useful framework representing cases. It gives out the idea in what type of knowledge needs to be used in a case in order for tasks to be carried out. Cases that include a problem description and solution can be used in deriving solution to a new problem. Cases with situation description and outcome can be used in evaluation new situation.

2.4.2 The Content of Problem Representations

The problem representation might be a problem that needs to be solved or a situation that needs to be interpreted, classified, or understood. In general, a CBR reasoner determines whether an old case is applicable to a new problem by examining the similarities between descriptions of the problem in the old situation and the new one. If a new situation is sufficiently similar to an old problem description, that case is selected. Thus, a problem representation must have sufficient detail to be able to judge the applicability of the case in new problem.

There are three major components of a problem representation;

1. Goals to be achieved in solving the problem.

2. Constraints of these goals.
3. Features of the problem situation and relationships between its parts.

The goals in a problem description describe the aims of the actor in the situation. In general, a CBR reasoner will try to achieve its goals or fulfill its aim successfully. Goals can be abstract or concrete, overarching or narrow. Which goal belongs in a problem situation depends on what lesson the case is aiming to teach. In the problemsolving situation overarching goals include 'diagnose', 'create', and 'plan' and often they further specify what should be diagnosed, created or planned. In the interpretive situation overarching goals include 'understand', 'explain', and 'evaluate' and those too might specify what in the situation is to be understood, explained or evaluated.

Constraints are conditions put on goals. As an example, JULIA [1] is a meal planning system. When JULIA creates meals, there are a few things that she has to keep in mid; the like and dislike of people who will eat the meal, caloric content, cost and so on. All these are constraints.

Features of the problem situation are the catchall that holds any other descriptive information about the situation relevant to achieving the situation's goal. In medical diagnosis program, for example, a patient's symptoms and test results are necessary descriptors of the situation that must be attended to in constructing a solution; other features of the situation include anything else that might be taken into account in fulfilling the situation's goal.

2.4.3 The content of Solutions

There are many types of solutions. The solution to design a problem is the artifact that was designed. The solution to a planning problem is the derived plan. The solution to an interpretive problem is the interpretation or classification finally assigned to the case. The solution to an explanation is the explanation. The

solution can be concluded as the concepts or objects that achieve the goals set forth in the problem description, taking into account the specified constraints and other specified contextual features.

With a solution in place, a CBR reasoner that receives a case can use its solution to derive a new solution. There are other components of a solution that is needed in the aid of adaptation and critiquing. Below are the common components of a solution.

- The solution itself.
- The set of reasoning steps used to solve the problem.
- The set of justifications for the decisions that were made in solving the problem.
- Acceptable solutions that were not chosen.
- Unacceptable solutions that were ruled out.
- Expectations of what will result upon deployment of the solution.

Reasoning step

Some of the CBR reasoner, like JULIA, record the set of reasoning step used to derive a solution in their case representation. This allows those reasoning steps to be repeated in later problem solving.

Justifications

Justifications provide a way of guiding adaptation of an old solution. It has two purposes; first, justifications provide guidance during evaluation of a solution, second, justifications guide index solution.

Alternative solutions

Alternative solutions that are available but not chosen can be useful in deriving new solution. It provides an alternative means of solving a problem if other mans are ruled out.

Expectations about outcome

Expectation is a piece of knowledge that can be included in a solution representation. When a case is selected to solve a problem, evaluation can be done by matching the outcome of the result and the expected outcome.

2.4.4 The Content of the Case Outcomes

The outcomes of a case specify what happened as a result of carrying out the solution or how the solution performed. Outcomes include both feedback from the world and the interpretation of the feedback.

The minimal outcome information includes feedback from the world detailing what happened as the result of carrying out a solution, along with a determination of its success and whether expectations were met. With this information, the CBR reasoner can anticipate potential problems and predict the outcome of a proposed solution. Outcome can include more, such as; if expectations are not met or a solution failed, the needed processes that used to solve the problem can be included in the outcome too.

Here are several parts of the outcome;

- The outcome itself.
- Whether the outcome was a success or violated expectations.
- Whether the outcome was a success or failure.
- Repair strategy.
- What would have been done to avoid the problem?
- Pointer to next attempt a solution.

Feedback is the most obvious component of the outcome, that is, the things that actually happened. The outcome of a case is used to tell how a solution performed in real world.

2.5 Methods for Index selection

Retrieving the appropriate cases in case based reasoning is an important task. The more appropriate a case is selected for a problem, the better the resulting of the outcome. This means that a computer has to remember the right ones at the right times. This is indexing. Indexing is used for retrieving the appropriate case. According to Janet Kolodner [1], there are three types of indexation can be used in a CBR reasoner; checklist based indexing, difference based indexing and explanation based indexing.

2.5.1 Checklist Based Indexing

Checklist based indexing indexes all cases on a fix and well specified set of dimensions. Given a checklist, the process of index selection is easy. For each dimension on the checklist, find or compute the value along that dimension that describe the case, and choose it as an index. Though this method of indexing make the computer's job easy, the method is only good as the checklist created by the system developer. An incomplete checklist will result in insufficient indexing; a checklist that doesn't discriminate between important and unimportant dimensions will result in over indexing and retrieval of too many cases. Below are the issues that must be considered in setting up these checklists;

- ❖ Guidelines and procedures for creating a checklist.
- ❖ Making sure the checklist is contextually sensitive.
- ❖ Selecting consistent indexing vocabulary.

2.5.2 Difference Based Indexing

By using difference based indexing, a CBR reasoner can differentiate cases from one another. During retrieval time, retrieval algorithms can choose best matching cases from the library. When a memory keeps track of what is common across similar cases from the library. When a memory keeps track of what is common across similar case, it has a means of discovering which features of the case

differentiate it from other similar cases, and it can choose as indexes those feature that differentiate cases. But to optimize the power of difference based indexing and make sure that only the predictive features are chosen, it is good to combine difference based indexing with some other method of choosing predictive features, such as checklist based indexing.

2.5.3 Explanation Based Indexing

Both of the indexing methods above provide a simple means of computing predictive features to use as indexes. But they have major drawback. They choose indexes based on a model of the kinds of features that are usually predictive but don't analyze cases individually for their predictive features. Explanation based indexing methods are aimed at choosing indexes appropriately for individual cases. The CBR reasoner tries to explain why the solution worked or didn't work, and then use explanation based generalizations methods to generalize the explanation. Indexes are chosen from the content of the generalized explanation. In explanation based indexing, domain knowledge is used to determine which facts of the case are the most relevant and which case can be safely ignored.

2.6 Case Retrieval

The retrieve task starts with a problem description, and ends when a best matching previous case has been found. Its subtasks are referred to as Identify Features, initially Match, Search, and Select, executed in that order. The identification task basically comes up with a set of relevant problem descriptors, the goal of the matching task is to return a set of cases that are sufficiently similar to the new case – given a similarity threshold of some kind, and the selection task works on this set of cases and chooses the best match.

While some case based approaches retrieve a previous case largely based on superficial, syntactical similarities among problem descriptors (e.g. the CYRUS system [1]), some approaches attempt to retrieve cases based on features that have deeper, semantically

similarities (e.g. the PROTOS systems, CASEY systems [1]). In order to match cases based on semantic similarities and relative importance of features, an extensive body of general domain knowledge is needed to produce an explanation of why two cases match and how strong the match is. Syntactic similarity assessment – sometimes referred to as a "knowledge poor" approach - has its advantage in domains where general domain knowledge is very difficult or impossible to acquire. On the other hand, semantically oriented approaches – referred to as "knowledge-intensive"- are able to use the contextual meaning of a problem description in its matching, for domains where general domain knowledge is available.

2.6.1 Identify Feature

Input descriptors can be used to identify a problem, but often and particularly for knowledge-intensive methods, a more elaborate approach is taken to 'understand' the problem. Unclear descriptors may be disregarded or further explanation needed from the user. To understand a problem involves filtering out noisy problem descriptors, to infer other relevant problem features, to check whether the feature values make sense within the context, to generate expectations of other features. Other descriptors than those given as input, may be inferred by using a general knowledge model, or by retrieving a similar problem description from the case base and use features of that case as expected features. Checking of expectations may be done within the knowledge model (cases and general knowledge), or by asking the user.

2.6.2 Initially Match

There are two task involved in finding a good match; an initial matching process which retrieves a set of possible solutions, and a more elaborate process of selecting the best among the retrieved solutions. The later is select, the Select task will be discussed below. The input descriptors are used as an index to the case

library to find a set of matching cases in a direct or indirect way. There are three ways in retrieving case from a set of cases.

1 By following direct index pointers from problem features.
2 By searching an index structure.
3 By searching in a model of general domain knowledge.

Cases may be retrieved based on the input. Cases that match all input features are good for matching, but sometimes, cases that are given only part of the problem may also be retrieved. Some of the CBR systems use a global similarity metric to retrieve a set of similar cases.

2.6.3 Select

From the set of similar cases, a best match is chosen. This may have been done during the initial match process, but more often a set of cases are returned from that task. The best matching case is usually determined by evaluating the degree of initial match more closely. This is done by an attempt to generate explanations to justify non-identical features, based on the knowledge in the semantic network. If a match turns out not to be strong enough, an attempt to find a better match by following difference links to closely related cases is made. This subtask is usually a more elaborate one than the retrieval task, although the distinction between retrieval and elaborate matching is not distinct in all systems. The selection process typically generates consequences and expectations from each retrieved case, and attempts to evaluate consequences and justify expectations. This may be done by using the system's own model of general domain knowledge, or by asking the user for confirmation and additional information. The cases are eventually ranked according to some metric or ranking criteria. Knowledge intensive generation methods typically generate explanations that support this ranking process, and the case that has the strongest explanation for being similar to the new problem is chosen. Other properties of a case that are considered in

some CBR systems include relative importance and discriminatory strengths of features, prototypically of a case within its assigned class, and difference links to related cases.

2.7 Case Reuse

The reuse of the retrieved case solution in the context of the new case focuses on two aspects;

1. The differences between the past and the current case and
2. What part of a retrieved case can be transferred to the new case.

2.7.1 Copy

Copy is the simplest way in reusing a case in Case based Reasoning system. The differences of a new problem and a stored case are abstracted away as they are considered not relevant, and the similarities are classified as relevant. The match case is applied in the new problem as the solution. This is an important type of reuse in CBR system. However, if the differences between the past and the current case are ignored, a case cannot be directly transferred to the new case but requires an adaptation process that takes into account those differences.

2.7.2 Adapt

There are three main ways to reuse past cases.

1. Reuse the past case solution (transformational reuse).
2. Reuse the past method that constructed the solution (derivational reuse).
3. Reuse the past case that substituted part of other case (substitutional reuse).

Below are the detail explanations of each type of reuse.

2.7.2.1 Transformational reuse

In transformational reuse, the past case solution is not directly a solution for the new case but it needed a transformational operator to transform the old solution to be used in a new case.

2.7.2.2 Derivational reuse

Derivational reuse looks at how the problem was solved in the retrieved case. The retrieved case holds information about the method used for solving the retrieved problem including a justification of the operators used, sub goals considered, alternatives generated, failed search paths, etc. Derivational reuse then replace the retrieved method to the new case and "replays" the old plan into the new context (usually general problem solving systems can be seen here as planning systems). During he replay successful alternatives, operators, and paths will be explored first while filed paths can be avoided; new sub goals are pursued based on the old ones and old sub plans can be recursively retrieved for them.

2.7.2.3 Substitutional reuse

In substitutional reuse, when a found solution is only capable to solve part of the problem, the reasoner will try to substitute the useless part of the solution with a useable part from other case. There a re a few types of substitution methods and each of the methods has their differences.

2.8 Case Revision

When a case solution generated by the reuse phase is not correct, an opportunity for learning from failure arises. This phase is called case revision and consists of two tasks;

1. Evaluate the case solution generated by reuse. If successful, learning from the success, or
2. Repair the case solution using domain specific knowledge.

2.8.1 Evaluate Solution

The evaluation task takes the result from applying the solution in the real environment (asking a teacher of performing the task in the real world). This is usually a step outside the CBR system, since it – at least for a system in normal operation – involves the application of a suggested solution to the real problem. The results from applying the solution may take some time to appear, depending on the type of application. In a medical decision support system, the success of failure of a treatment may take from a few hours up to several months. The case may still be learned, and be available in the case base in the intermediate period, but it has to be marked as a non-evaluated case. A solution may also be applied to a simulation program that is able to generate a correct solution.

2.8.2 Repair fault

Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them. This is included into a failure memory that is used in the reuse phase to predict possible shortcomings of plans. This form of learning moves detection of errors in a post hoc fashion to the elaboration plan phase were errors can be predicted, handled and avoided. A second task of the revision phase is the solution repair task. This task uses the failure explanations to modify the solution in such a way that failures do not occur. The repair module possesses general causal knowledge and domain knowledge about how to disable or compensate causes of errors in the domain. The revised plan can then be retained directly (if the revision phase assures its correctness) or it can be evaluated and repaired again.

2.9 Case Retainment

This is the process of incorporating what is useful to retain from the new problem solving episode into the existing knowledge. The learning from success or failure of the proposed solution is triggered by the outcome of the evaluation and possible repair. It involves selecting which information from the case to retain, in what form to retain it, how to index the case for later retrieval from similar problems, and how to integrate the new case in the memory structure.

2.9.1 Extract

In CBR the case base is updated no matter how the problem was solved. If it was solved by use of a previous case, a new case may be built or the old case may be generalized to subsume the present case as well. If the problem was solved by other methods, including asking the user, an entirely new case will have to be constructed. In any case, a decision needs to be made about what to use as the source of learning. Relevant problem descriptors and problem solution are obvious candidates. But an explanation or another form of justification of why a solution is a solution to the problem may also be marked for inclusion in anew case. The last type of structure that may be extracted for learning is the problem solving method, i.e. the strategic reasoning path, making the system suitable for derivational use.

Failures, i.e. information from the Revise task, may also be extracted and retained, either as separate failure cases or within total-problem cases. When a failure is encountered, the system can then get a reminding to a previous similar failure, and use the failure case to improve its understanding of – and correct – the present failure.

2.9.2 Index

The 'indexing problem' is a central and much focused problem in case based reasoning. It amounts to deciding what type of indexes to use for future retrieval, and how to structure the search space for indexes. Direct indexes, as previously mentioned, skip the latter step, but there is still the problem of identifying what type of indexes to use. This is actually a knowledge acquisition problem, and should be analyzed as part of the domain knowledge analysis and modeling step. A trivial solution to the problem is of course to use all input features as indices. This is the approach of syntax-based methods within instance-based and memory-based reasoning.

2.9.3 Integrate

This is the final step of updating the knowledge base with new case knowledge. If no new case and index set has been constructed, it is the main step of Retain. By modifying the indexing of existing cases, CBR systems learn to become better similarity assessors. The tuning of existing indexes is an important part of CBR learning. Index strengths or importance for a particular case or solution are adjusted due to the success or failure of using the case to solve the input problem. For features that have been judged relevant for retrieving a successful case, the association with the case is strengthen, while it is weakened for features that lead to unsuccessful cases being retrieved. In this way, the index structure has a role of tuning and adapting the case memory to its use.

2.10 K-Nearest neighbour Matching Algorithm for CBR

K-Nearest Neighbour (KNN) are a well known and intensively studied class of techniques for the solution of Classification and pattern Recognition problems. Nowadays KNN are widely exploited in the retrieval phase of case based reasoning systems In CBR, even if the solution In CBR, even if cases are not explicitly classified in

a set of finite groups (classes), often the solution space can be clustered in a collection of sets each of them containing similar solutions.

When a set of similar solution is labeled with a class tag, it is natural to match the retrieval step in a CBR system with the nearest neighbour search in NN classifier. In this framework, for example Bellaze et al. [11] have shown that the performance of a CBR system can be improved by driving the retrieval with the information of same relevant classification in the case space. i.e. reducing the retrieval problem to a classification task. In this perspective, improving the classification accuracy for NN algorithms becomes important for CBR.

The NN classification procedure is straightforward: given a set of classified examples, which are described as points in an input space, a new unclassified example is assigned to the known class of the nearest example. The nearest relation is computed using a similarity metric defined on the input space.

2.11 Malay Traditional Herbs

2.11.1 Malay Traditional Herbs

Worldwide demand for herbal medicine is increasing. Many people today are returning to herbal medicine as a result of the greater awareness of health issues, promoting by new government health policies and a growing distrust in conventional synthetic medicines [8].

Modern medical science is also taking advantage of this vast herbal tradition in its search for new drug discovery. This research is contributing enormous scientific literature on herbal actions and is attributing to the synthesis of herbal wisdom with modern scientific principles. In particular the world's rain forests are viewed as some of the last uncharted territory for new drug discovery.

As one of the most evolved and diverse ecosystems on earth, the Malaysian rain forest is a rich source of medicinal herbs. It is believed that forest dwellers in Southeast Asia use 6,500 different plants to treat illness. These same rain forests have supplied the western world with herbs and spices for centuries. Malaysia in particular, with its many indigenous groups, offers an invaluable source of knowledge in medicinal plants from the rain forest.

The main indigenous medicinal systems of Malaysia are Traditional Malay Medicine, with influences from Java, India and Arabia, and that practiced by the numerous aboriginal races. Of these, many still live according to traditional ways in the jungle, with a few still preferring to avoid influences of civilization altogether and continue to lead a semi-nomadic existence deep in the rain forests.

All these races offer invaluable ethnobotanical information. The herbal knowledge they provide comes from centuries of human experience of trial and error in herbal action, safety and toxicity. Knowledge that helps us in our search for cures for disease and food for our growing world population. We are faced with a race against time, however, as the effects of rapid development of the rain forests is leading to both the loss of medicinal species diversity, as well as of the ethnobotanical knowledge of the indigenous peoples found there. With the passing of each of the elders, the medicine men and women, a mine of valuable knowledge is lost forever.

2.11.2 Characteristics of Herbs

The parts of every typical plant are root, stem, leaf, flower, and fruit [8]. Discussed below are each of the parts in turn.

2.11.2.1 The Root

Roots are underground parts of plants (*but not all underground parts are roots*). They have two main functions: 1) they anchor the plant in the

ground; 2) they absorb water and minerals from the soil. Many roots, like the carrot, also serve as food storage organs for their plants.

A *taproot* is a single main root with distinctly smaller branch roots. *Fibrous roots* are thin and all more or less the same size. The development of the root system depends both on the type of plant and on soil conditions, varying from the use of only a few inches of soil to 50-foot-deep forays in search of water. A single plant with a highly branching root system not un-typically develops millions of roots totaling hundreds of miles in length and thousands of square feet in absorbent surface area.

2.11.2.2 The Stem

Some parts of herbaceous perennial plants that many people consider roots are actually underground portions of the plant's stem. These are classified as rootstocks (*or rhizomes*), stolons, corms, and bulbs.

A *rootstock* grows horizontally in the ground, sending down roots from its lower side and one or more erect stems (*or sometimes leaves*) from its upper side. One feature that distinguishes it from a true root is the presence of scaly leaves at regular intervals along its length. The rootstock lives from year to year, sending up new growth each season. Some rootstocks are thick and fleshy; others are long and thin. Some thin rootstocks develop locally thick parts for food storage; these are called *tubers* (*the potato being the best-know example*).

A *stolon* is much like a rootstock, but it grows along the surface of the ground, sending roots down and stems up at intervals. A *runner*, like that of the strawberry plant, is a type of stolon.

A *corm* is a short, thick, vertical underground stem that survives from one season to the next in a dormant state. The second season it produces one or

more aerial stems and also one or more new corms. The new corms store food produced by the growing plant and

then go through the next dormant period to produce their own plants and corms the following season. The "bulbs' of gladiolus are actually corms.

Bulbs are different from corms, although the latter are often called bulbs. A bulb consists of a short, erect stem enclosed by fleshy leaves (*as in onions*) or leaf bases (*as in daffodils*) that serve to store foods between growing seasons. Some bulbs survive for several years; others are replaced by new bulbs each year.

The portion of the plant that everyone recognizes as the stem is more precisely called the *aerial stem*. Its main function is to bear leaves, the stem with its leaves being called the *shoot. Herbaceous stems* are those that contain no woody tissue; these usually die down at the end of the growing season, unlike their woody counterparts in trees and shrubs. *Erect stems* are those that grow more or less upward without special support; vines have stems that trail on the ground or climb by attaching themselves to other plants or objects. In addition to bearing leaves, the aerial stem performs the vital functions of transporting water and minerals up from the roots to the leaves and transporting manufactured food substances as they are distributed to all parts of the plant for use or storage.

2.11.2.3 The Leaf

Leaves come in all sizes and shapes (*including some that look more like stems or like flowers*), but the typical leaf has a flat *blade* and a stalk, or *petiole*, by which it is attached to the stem. Some leaves manage nicely without a petiole; these are called *sessile*. Leaves tend to grow in regular patterns on the stem; *opposite leaves* grow in pairs from opposite sides at the same point along the stem; *alternate leaves* grow on opposite sides but at different points on the stem; *whorled leaves* grow in groups of three or

more around the stem at one point. *Radical leaves* grow directly from a non-aerial stem. *Simple leaves* have a one-piece blade; *compound* leaves consist of individual leaflets which grow either from a single point (*palmate leaf*) or oppositely along the leaf stalk (*pinnate leaf*).

The primary function of the average green leaf is to carry on photosynthesis - the process by which plants use the energy of sunlight to combine simple substances absorbed from the soil and the air into complex food substances. In the process, plants use up carbon dioxide from the air and produce oxygen. At night the balance reverses, and plants use up oxygen just as we do; but overall the amount of oxygen produced is greater than that consumed (*fortunately for us*). The critical agent in photosynthesis is the green pigment chlorophyll: only green plant parts are photosynthetic. Green leaves contain various other pigments as well, but these show up only when the leaf dies and its chlorophyll breaks down. The yellow and red autumn colors of many trees are due to leaf pigments which are present but are masked by the chlorophyll while the leaves are alive.

2.11.2.4 The Flower

Flowers are merely specialized shoots -- specialized for reproduction. The typical flower consists of several whorls (*circular ranks*) of parts set on a *receptacle*, the somewhat enlarged end of a stem or flower stalk. The outermost whorl is the *calyx*, a set of leaf-like parts (*sepals*) that protect the flower before it opens. The next whorl in is the *corolla*, consisting of modified, usually white or brightly colored leaves called *petals*. One or more whorls of club-shaped *stamens* come next; these are the male organs that provide the fertilizing pollen. The center of all this attention is the female organ, the *pistil*, consisting of one or more *carpels*. A carpel is made up of a bulbous *ovary* which contains the seeds-to-be (*ovules*), and a

stalk (the *style*), part of which (the *stigma*) is rough or sticky to capture pollen for fertilization.

Flowers that have the complete set of parts - sepals, petals, stamens, and one or more pistils - are complete; those that are missing one or more parts are incomplete. Perfect flowers have both stamens and pistils; imperfect flowers have only one or the other (*a few have neither*), being staminate (*male*) if they have stamens, pistillate (*female*) if they have pistils. Some plants bear both staminate and pistillate flowers on the same plant; others have the two kinds on different plants. The transfer of pollen from stamen to pistil - the pollination necessary for seed to form - is accomplished in various ways, depending on the plant and physical circumstances. The usual ways are by direct contact between stamen and stigma, by insects, or by wind.

Flowers can occur alone or in various kinds of clusters (*inflorescences*).

2.11.2.5 The Fruit

In botany, *fruit* has a much broader and yet more definite meaning than in popular usage: it is the ripened ovary or ovaries - sometimes with associated other parts - of a flower or flower cluster. *True* or *simple fruits* develop from ovaries only; *accessory fruits* (*like strawberries or rose hips*) develop from ovaries and one or more other parts of the flower. With few exceptions - seedless grapes and pineapples, for example - a fruit forms only after pollination.

Botanically, nuts, beans, corn grains, tomatoes, and dandelion "seeds" are just as much fruits as are blueberries, oranges, cherries, and peaches. You may be surprised to find, though, that tomatoes, cucumbers, and oranges are berries; walnuts and almonds are drupes like cherries and peaches; and peanuts are legumes like peas and beans. Fascinating though it is, a full explanation of the various types of fruit would require greater detail than

is possible here; the glossary of botanical terms contains basic definitions for most of them.

The basic function of fruit is to provide for the dispersal of seed at the proper time, but the fruit also serves to protect the seeds as they mature. Considering that seeds range in size from barely visible (*orchid*) to over a foot in diameter (*double coconut*), you should not be surprised to find considerable variety in the dispersal mechanisms that different plants have developed. Some fruits split open spontaneously while still on the plat to scatter seeds onto the ground or into the wind; others drop from the plant intact but have wings or feathered tufts attached to help them ride the wind. And some maverick plants - collectively called the tumble weeds - abandon themselves to the wind entirely and scatter seeds as they roll along the ground. Some seeds are spread mainly by birds and other animals, which eat the fruit but excrete the undigested seeds. Prickly fruits often hitch a ride on passing animals or people who carry them elsewhere; others can float on water until they are washed up on a new shoreline. There are still other ways, but these are enough to suggest the boundless ingenuity of Nature in providing for the propagation of each species.

2.12 Existing Websites on Herbs

Viable Herbal Solutions Home Page
http://www.viable-herbal.com/homepage.htm

Viable Herbal Solutions is a developer, manufacturer, and primary supplier of the most remarkable alternative and complementary medicine herbal health products available is the United States. This websites is an e-Commerce site providing guest interested in buying their products or their services. This website also provide in information on herbs, their uses and how to identify them. However, the site only emphasize on herbs which are available in the United States and not on Malay Traditional herbs.

Pusat Pembangunan Perniagaan

Jabatan Pertanian Semenanjung Malaysia's Home page

http://agrolink.moa.my/pusat_sumber/

This website is developed by the Ministry of Agriculture of Malaysia. It contains information on many kind of crops including herbs, vegetables, rice and others. This website provides information on Traditional Malay herbs, their characteristics, their uses and how to harvest them. This website also links to the available experts on Traditional Malay herbs.

3.0 SYSTEM ANALYSIS

In this chapter, analysis on Case Based Reasoning will be done as well as the implementations of CBR techniques on classifying Malay Traditional herbs will also be discussed here too.

3.1 Analysis of Implementation of CBR in the Problem Domain

A case-based classifier asks whether the new instance is enough like another one to be assigned the same classification. PROTOS [1], which diagnoses hearing disorders, works by doing classification by trying to find the closest matching case in its case base to the new situation. To do this, PROTOS keeps track on how protypical each of its cases is and what differentiates cases within one classification from each other.

Classifying Malay Traditional herbs will be most likely done in the same way. The system will first choose a most likely classification, and then chooses the most likely matching cases in that class. Based on differences between the case it is attempting to match and the new situation, it eventually zeros in on a case that matches its new one well and assigns the new case to the same category.

Searching will be done for the closest matching case using two major steps. First, it narrows its search the most likely candidates. Then, based on qualities of the match between its most likely candidate and the new item, it follows pointers around the case library in search of better matches. It repeats this second step until either finds an acceptable match or fails.

Advantages of CBR in classifying Malay Traditional herbs;

- Even though the system cannot suggest or adopt solutions in cases where the herbs which are being classified is not stored in the case base, but the system can

find the closest matching herbs which assure solutions where no perfect solution is available.

- The system can also cater incomplete information whenever a complete attributes of the herbs are almost impossible to define. This occurs when the user is only provided with incomplete parts of the herbs, which disable them to give all the inputs accordingly.

Disadvantages of CBR in classifying Malay Traditional herbs;

- Using the k-Nearest Neighbour searching algorithm, the system might be doing searching exhaustively and this requires high computation power and memory capacity.
- Case Based classifier might be tempted to use old cases blindly, relying on the cases in the case library and generate unsuitable solution for a new situation.

3.2 Case Representations and Indexing

A case is usually composed of three parts; problem/situation description, solution, and outcome. A well design case will improve the reliability of a retrieved case to be applied in a new problem.

3.2.1 The Content of Problem Representations

There are three major components of a problem representation:

1. Goals to be achieved in solving the problem
2. Constraints on those goals
3. Features of the problem situation and relationships between its parts.

In case based classification, the goal is to classify the herbs into its own respective class. As part of that process, the system will try to find the closest matching case with the case which is being queried.

Constraints of those goals will be the distance between the new case and the cases in the case library. Each time the system will have to calculate the shortest distance between the new case and the cases in the case library to find the most appropriate solution.

Features of the problem situation are the catchall that holds any other descriptive information about the situation relevant to achieving the situation's goals. When a matching process is being done between a query case and the case library, the features of the problem situation will hold information on the current situation during the matching process.

3.2.2 The Content of Solutions

There are three components in the solution of the problem:

1. The solution, which match perfectly with the query case,
2. The acceptable solutions, which closely match the problem description, and
3. The degree of distance between the query case and the cases which are selected as solution.

Using the k-Nearest Neighbour matching algorithm, computation on the highest degree of matching can be described as finding the closest matching cases with the query case, the case which has the best scores will be identified as the closest matching case with the query case.

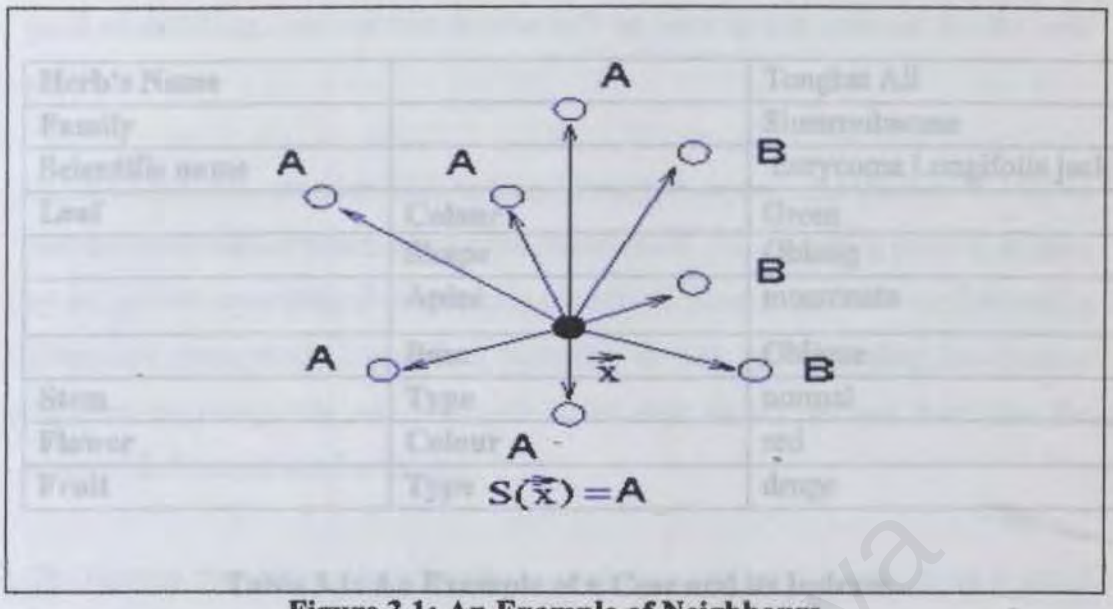


Figure 3.1: An Example of Neighbours

3.2.3 Method for Index Selection

There are three types of indexing in Case-Based reasoning, checklist based indexing, difference based indexing and explanation based indexing. Checklist based indexing will be applied in classifying Malay Traditional herbs. Checklist based indexing indexes all cases on a fix and well-specified set of dimensions. Given a checklist, the process of index selection is easy.

For classifying Malay Traditional herbs, the checklist will contain the name of the herbs, and its family in the botanical terms and the features of the herbs (the root, leaf, stem, flower and fruit).

Herb's Name		Tongkat Ali
Family		Simaroubaceae
Scientific name		Eurycoma Longifolia jack
Leaf	**Colour**	Green
	Shape	Oblong
	Apice	mucronate
	Base	Oblique
Stem	**Type**	normal
Flower	**Colour**	red
Fruit	**Type**	drupe

Table 3.1: An Example of a Case and its Indexes

3.3 Case Retrieval

In Classifying Malay traditional herbs, the retrieve task starts with a problem description: the input of the user, and end when a best matching case has been found. Three tasks are involved in the process of retrieval. Its subtasks are referred to as Identify Features, Initially Match and Select in that order.

3.3.1 Identify Features

In case of Malay Traditional herbs classifier, the input descriptors will be the the name of the herbs, and its family in the botanical terms and the features of the herbs (the root, leaf, stem, flower and fruit). The input descriptors will be used to identify the problem.

3.3.2 Initially Match

The input descriptors are used as an index to the case library to find a set of matching cases according to the inputs. The input descriptors are used as direct index pointers for cases in the case library. Cases that match all input features are

good at matching, and the match case will be used as the solution for the new problem.

But if no perfectly matched case is found during the search, the planner will try to find the most similar match among the stored cases. The matching process is done by using K-Nearest Neighbor matching algorithm. Cases that have similar feature values are conceptually closer than those that do not. By calculating the distance between the query case and the cases in the case library we can determine the cases with the most similar feature.

The Nearest Neighbour classification procedure is straightforward: given a set of classified examples, which are described as points in an input space, a new unclassified example is assigned to the known class of the nearest example. The nearest relation is computed using a similarity metric defined on the input space.

3.3.3 Select

If only one case is found in the process of initially match, then there are no need of the process of select. The found case will be retrieved as in this domain, there might be more solutions which matched the queried case. The select process will try to choose the best match cases for a problem among the retrieved cases. These selections are done by comparing the scores among the retrieved cases. The case with the highest score is the best match case for the problem.

3.4 Case Reuse

In this problem domain, when a case is retrieved, if there are no difference between the past and the current case, the process of copy will be applied to solve the current case. Copy is the process which a retrieved case will be applied wholly in solving the current problem.

But in cases where there is no perfectly match case found, the most similar or match case will be retrieved. Before that the retrieved case needs to undergo a process of adaptation before it can be used in the new problem. For this CBR classifier, there are two process of adaptation which it needs to used, reinstantiation and parameter adjustment.

Reinstantiation involves reinstantiating variables in an existing case with new values. For example, matching the color indigo with blue instead for the colour of the flower which has almost the same value.

Some cases may contain numerical values, such as the length of the stem. This value can vary inconsistently within a range. In order to match these values, parameter adjustment is essential to make sure the retrieved case correctly matched the new case.

3.5 Case Retainment

Case retainment in classifying Malay traditional herbs is to store newly derived case, or newly queried case, after solving a problem. The stored case can be used in future problem solving. However, for totally different cases, which are not stored in the case library before, the system is unable to suggest the name of the herbs as the system does not have the knowledge about those herbs. Before a newly derived case is stored, the user will need to verify the name of the herbs as well as other attributes which are not available before it is stored in the case library. This process will improve the accuracy of case retrieval in the future.

Integrate is the final step of updating the knowledge base with new case knowledge in the system. By modifying the indexing of existing case, whether by adaptation or manual editing, the system learns to become better similarity assessors. Index strengths or importance for a particular case or solution are adjusted to improve the reliability of future problem solving. In this way, the index structure has a role of tuning and adapting the case memory to its use.

3.6 System Methodology

The software process model that will be used for this project is the evolutionary prototyping method. In the evolutionary approach the prototype is used to learn more about the requirements or the solution. Once additional knowledge has been gained the prototype satisfies all the needs and can be used as the final system [6]. The prototyping and the production process are merged. This means, the the prototype gradually mevolves to become the final product.

This model is useful in situations when it is extremely difficult (or impossible) to establish detailed user requirements such as in user interface design and AI applications.

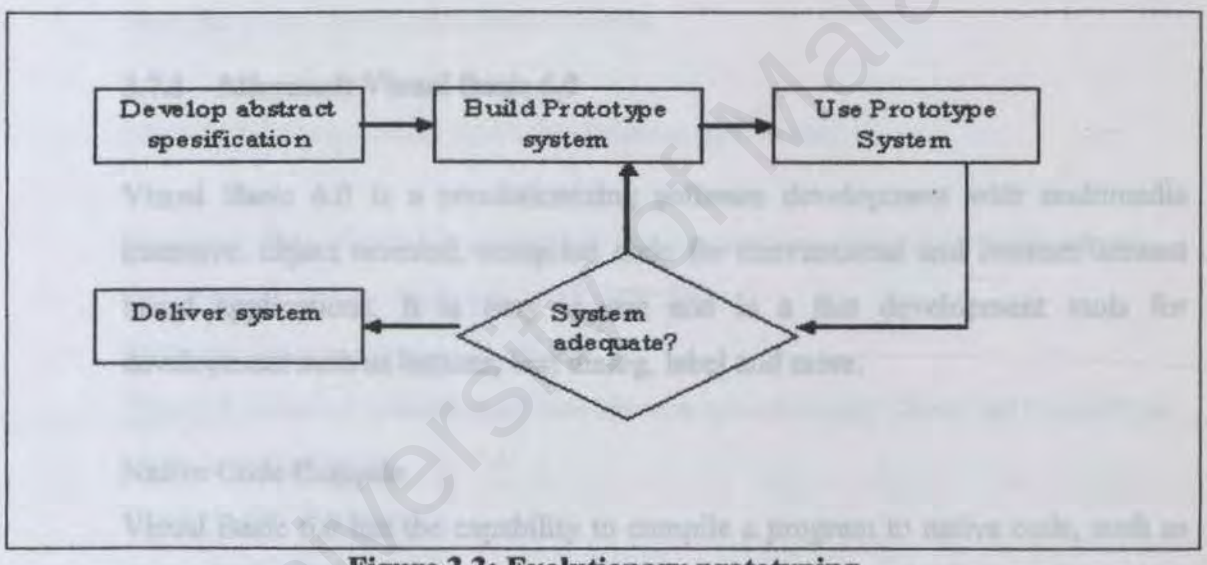


Figure 3.3: Evolutionary prototyping

The advantages of evolutionary prototyping include [4];

- Systems are developed and delivered rapidly.
- System development costs are reduced.
- As users are involved in the development, the system is likely to be appropriate for their real needs (leading to high user satisfaction).

The disadvantages are;

- ❖ The development process is not visible to managers. If systems are developed quickly, it is not cost effective to produce a great deal of system documentation.
- ❖ Systems are usually poorly structured. Continual change tends to corrupt the software structure. Maintenance is therefore likely to be difficult and costly.

3.7 Development Software

The development tools that will be used to develop the system are;

1) Microsoft Visual Basic 6.0
2) Microsoft Access 2000

3.7.1 Microsoft Visual Basic 6.0

Visual Basic 6.0 is a revolutionizing software development with multimedia intensive, object oriented, compiled code for conventional and Internet/Intranet based applications. It is easy to use and is a fast development tools for development such as buttons, text dialog, label and more.

Native Code Compile

Visual Basic 6.0 has the capability to compile a program to native code, such as C++. Therefore this will give a faster program. However, Visual Basic runtime library file is still needed to provide a fully functional program.

New Database Features

Visual Data Manager is another feature in Visual Basic 6.0. It eases the maintenance of database structure, as well as to input and edit the actual data. It also helps to create, test and save SQL statements in program.

Internet Features

Visual Basic 6.0 includes ActiveX Controls and Web Browsers Control to help Internet developers or programmers.

Others

Others features such as enhancement to code Editor and Development Environment.

3.7.2 Microsoft Access

Microsoft Access is a powerful program to create and manage databases. It has many built in features to assist you in constructing and viewing your information. Access is much more involved and is a more genuine database application than other programs such as Microsoft Works.

Microsoft Access breaks down the database as follows;

Database File: This is your main file that encompasses the entire database and that is saved to your hard-drive or floppy disk.

Example) StudentDatabase.mdb

Table:A table is a collection of data about a specific topic. There can be multiple tables in a database.

Example #1) Students

Example #2) Teachers

Field: Fields are the different categories within a Table. Tables usually contain multiple fields.

Example #1) Student LastName

Example #2) Student FirstName

Datatypes: Datatypes are the properties of each field. A field only has 1 datatype.

FieldName) Student LastName

Datatype) Text

4.0 SYSTEM DESIGN

System design is a process to convert the conceptual ideas from requirement specification in System Analysis into a more technical specification]. Each module will be discussed in this chapter.

4.1 System Overview

The CBR classifier is divided into seven modules. Discussed below is some brief information of each module.

- Identify Module
- Matching Module
- Similarity Module
- Adaptation Module
- Display module
- Evaluation Module
- Retain Module

Identify Module

This module is used to identify the problem based on the inputs. When the system receives a problem, the system will try to index the new problem. Identify Module will pass the index to the Matching Module. The index will be used as an identifier to search for a match case in the case library.

Matching Module

The Matching Module is responsible for retrieving a match case from the case library based on the index given by the Identify Module. This module can access the case library to search for the match case. If a match case is found, the module will pass the result to the Display Module to display the result. If no match case is found, the index will be passed to the similarity module.

Similarity module

If the Matching Module cannot find a perfectly match case from the case library, then the similarity module will try to get a usable or the closest match case for the problem. More then one case might be retrieved from this module. These cases will be passed to the adaptation module.

Adaptation Module

Cases found by the similarity module are not the perfect solution for the current solution. These cases need to be adapted before the case can be used to solve the problem. After the adaptation is done, the cases will be passed to the evaluation module.

Evaluation Module

After Adaptation Module has passed in all the adapted cases, the evaluation will try to rate each case based on the features such as the length of the herbs, to find the best case in order to find the best solution for the problem. The best rated cases will be sent to two next modules, the display module and the retain module.

Display Module

The display module is the interface of the CBR classifier. It is used to receive user's input to be passed to the matching module and display the result.

Retain Module

When a new solution is produced by the evaluation module to solve a new case, the new case will be retained by the retain module. This module will index the new case before storing the case into the case library.

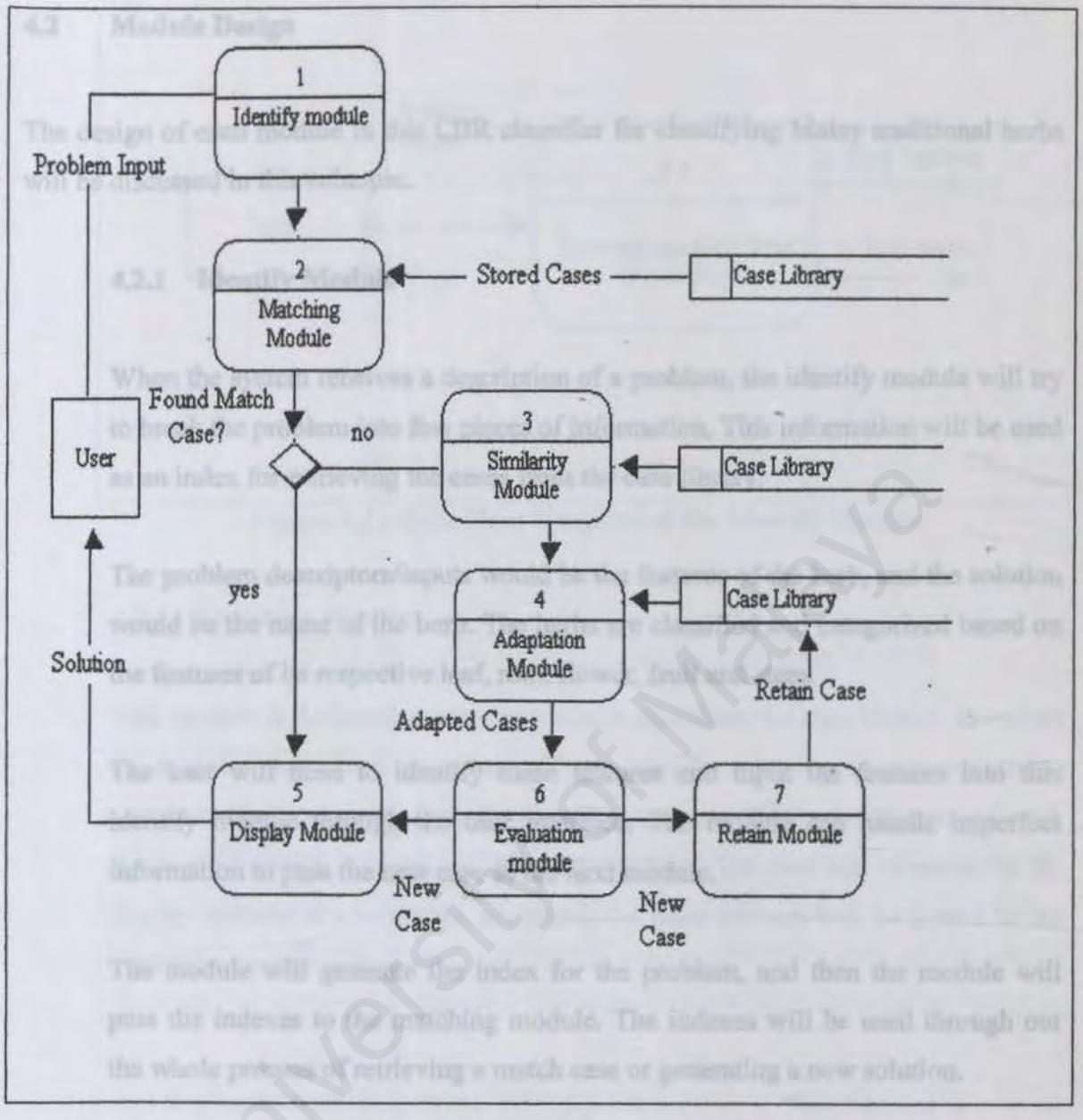


Figure 4.1 : Data Flow Diagram of the Main Module

4.2 Module Design

The design of each module in this CBR classifier for classifying Malay traditional herbs will be discussed in this subtopic.

4.2.1 Identify Module

When the system receives a description of a problem, the identify module will try to break the problem into few pieces of information. This information will be used as an index for retrieving the cases from the case library.

The problem descriptors/inputs would be the features of the herb, and the solution would be the name of the herb. The herbs are classified and categorized based on the features of its respective leaf, root, flower, fruit and stem.

The user will need to identify these features and input the features into this identify module through the user interface. The module can handle imperfect information to pass the new case to the next module.

The module will generate the index for the problem, and then the module will pass the indexes to the matching module. The indexes will be used through out the whole process of retrieving a match case or generating a new solution.

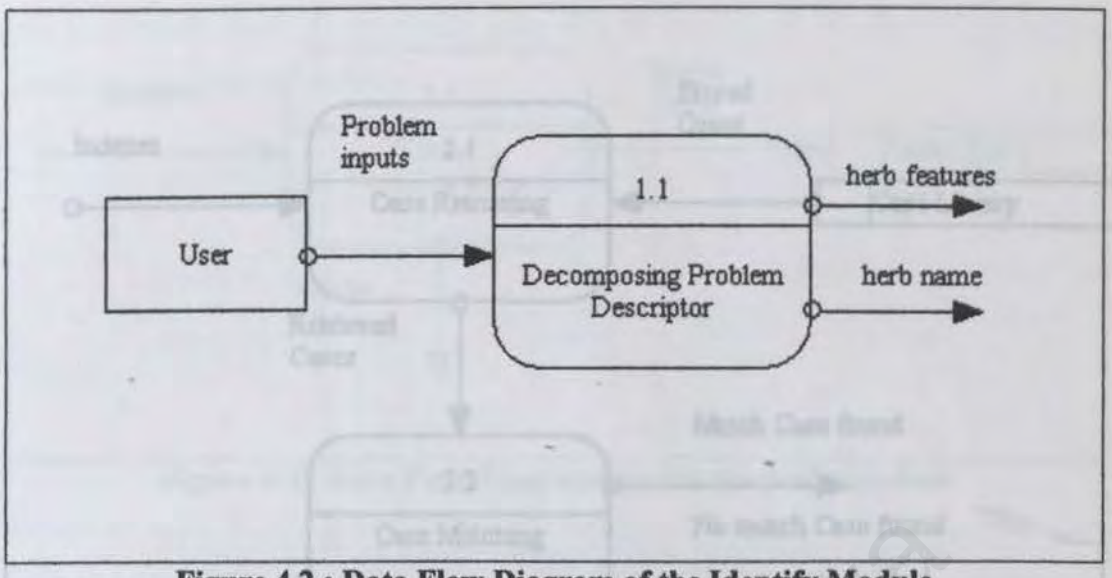


Figure 4.2 : Data Flow Diagram of the Identify Module

4.2.2 Matching Module

This module is designed to retrieve a match case from the case library. Based on the indexes passed over by the identify module, the matching module will try to search an identical match from the case library. If a perfectly match case is found, the found case will be used in solving the problem. The case will be passed to the display module. If no matches are found, the same indexes will be passed to the similarity module.

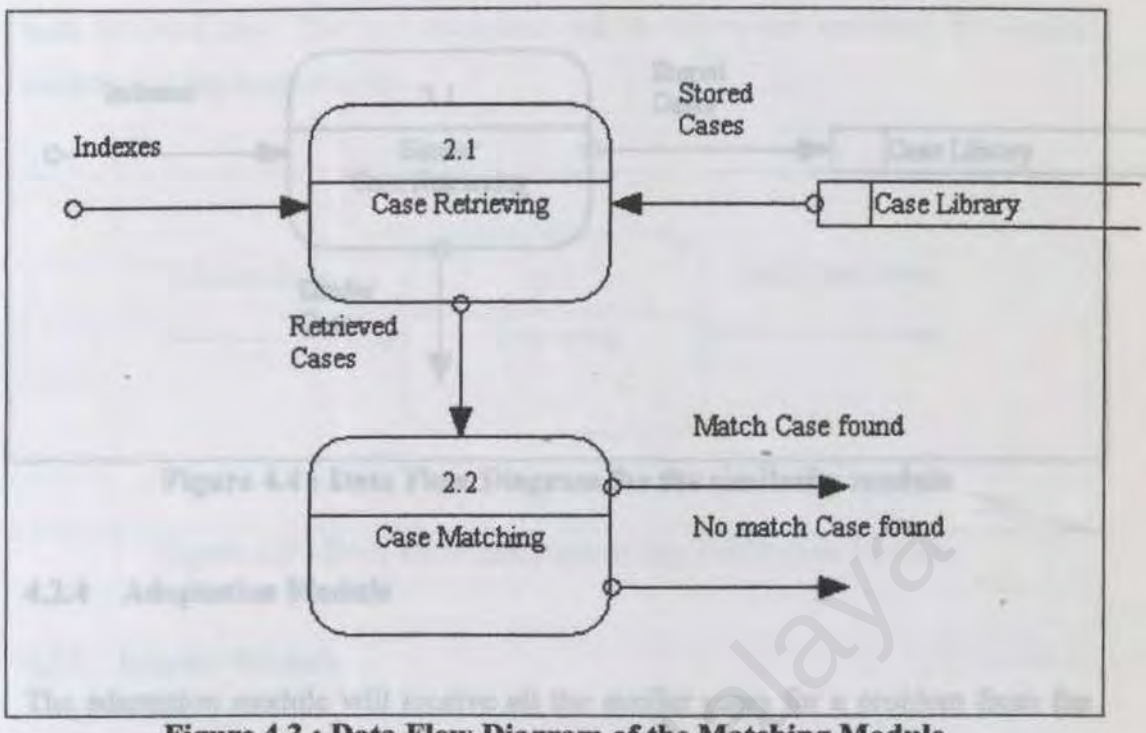


Figure 4.3 : Data Flow Diagram of the Matching Module

4.2.3 Similarity Module

This module is used to find the most similar cases which hopefully can be applied in the process of solving a problem when the matching module fails to find a solution from the case library. During this process, more than one case might be retrieved. All of the retrieved cases will be handed over to the adaptation module.

The similarity module searches for the similar case for a problem using the k-Nearest Neighbour Matching Algorithm. The module will try to find out which is the best solution by comparing the scores of each similar matches and the case with highest score is the decided the best solution for the problem.

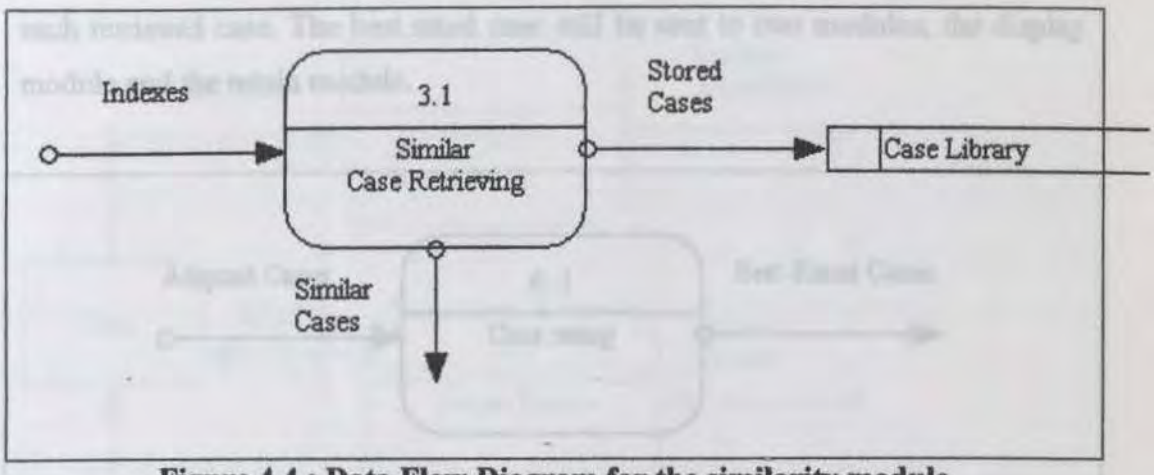


Figure 4.4 : Data Flow Diagram for the similarity module

4.2.4 Adaptation Module

The adaptation module will receive all the similar cases for a problem from the similarity module. These cases are solutions for other similar problem. To apply the solution to a new problem, the cases need to be adapted to the new problem situation.

The two techniques of adaptation used in this CBR classifier are instantiation and parameter adjustment (Refer to chapter 2).

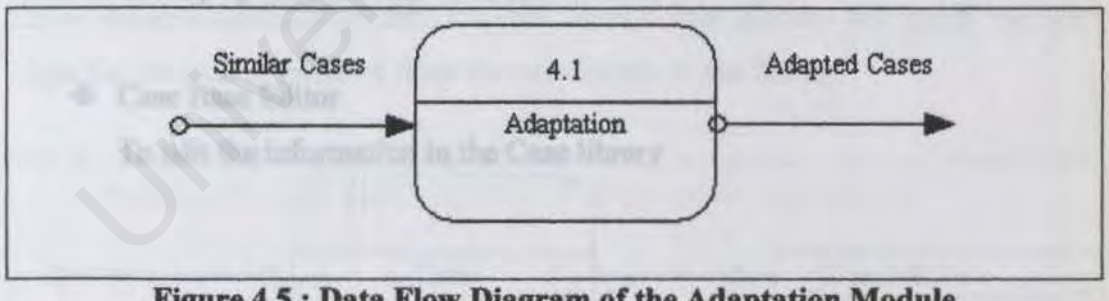


Figure 4.5 : Data Flow Diagram of the Adaptation Module

4.2.5 Evaluation Module

The evaluation module is used to rate all the newly derived case from the earlier modules. This module will evaluate each solution by comparing the scores of

each retrieved case. The best rated case will be sent to two modules; the display module and the retain module.

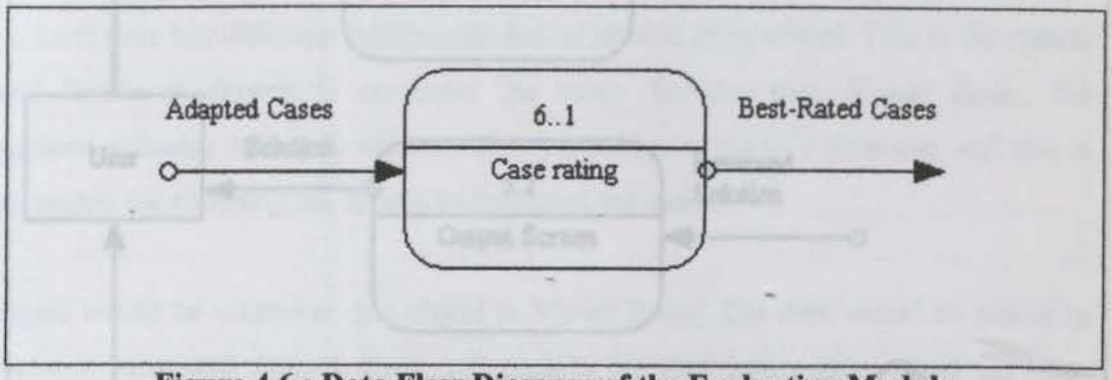


Figure 4.6 : Data Flow Diagram of the Evaluation Module

4.2.6 Display Module

The design of the user interface will consists of three main components;

- User input form
  User will input the description of the problem

- Output Screen
  To display the information of the solution

- Case Base Editor
  To edit the information in the Case library

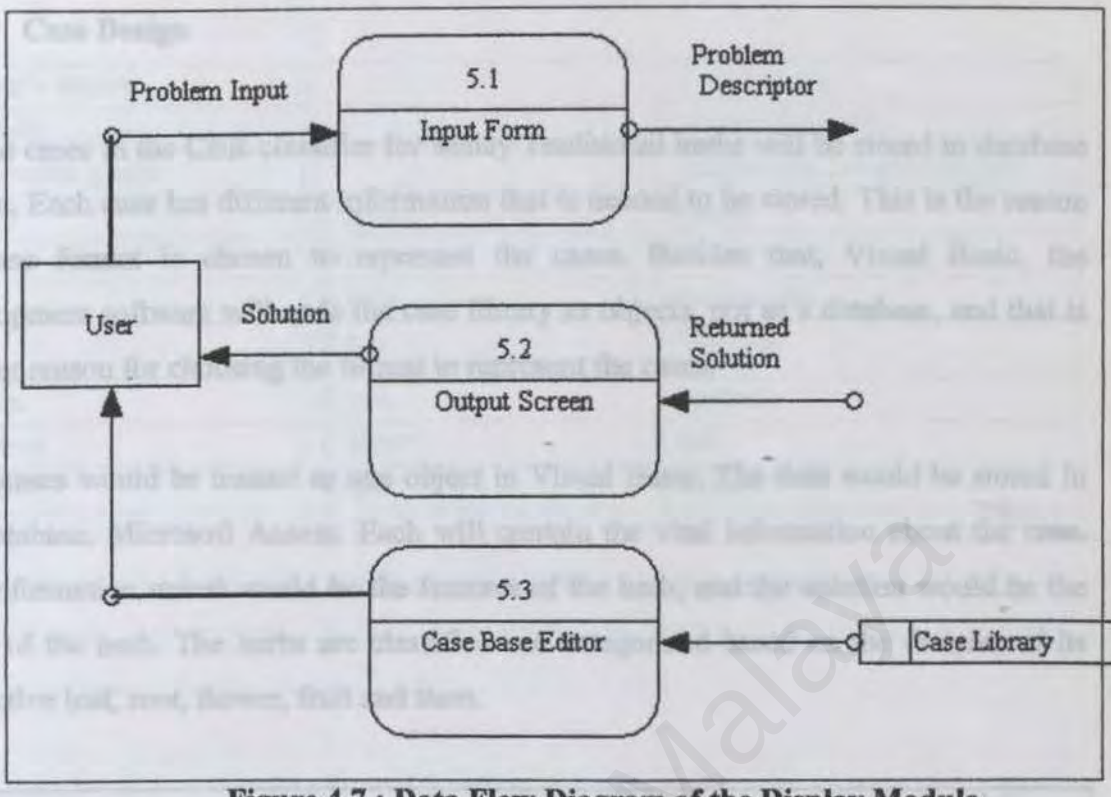


Figure 4.7 : Data Flow Diagram of the Display Module

4.2.7 Retain Module

When a new solution or case is generated or retrieved by the earlier modules, the case or solution will be passed to the retain module. This module will store the latest dericed case/solution into the case library. This module will index the new case for the ease of retrieval from the case library in the future.

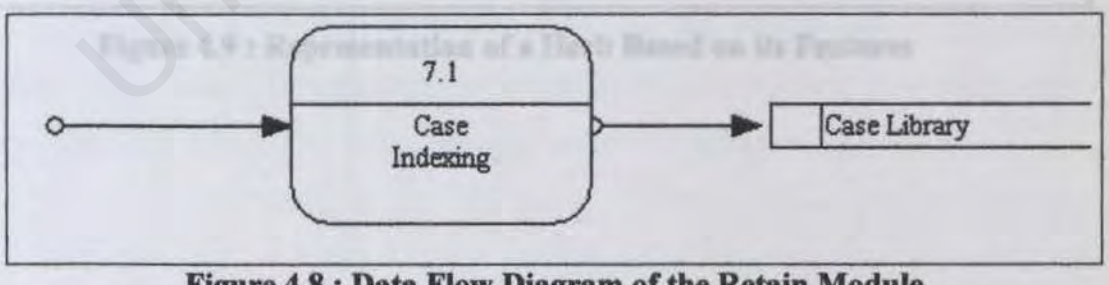


Figure 4.8 : Data Flow Diagram of the Retain Module

4.3 Case Design

All the cases in the CBR classifier for Malay Traditional herbs will be stored in database format. Each case has different information that is needed to be stored. This is the reason database format is chosen to represent the cases. Besides that, Visual Basic, the development software will code the case library as objects, not as a database, and that is another reason for choosing the format to represent the cases.

Each cases would be treated as one object in Visual Basic. The data would be stored in the database, Microsoft Access. Each will contain the vital information about the case. The information stored would be the features of the herb, and the solution would be the name of the herb. The herbs are classified and categorized based on the features of its respective leaf, root, flower, fruit and stem.

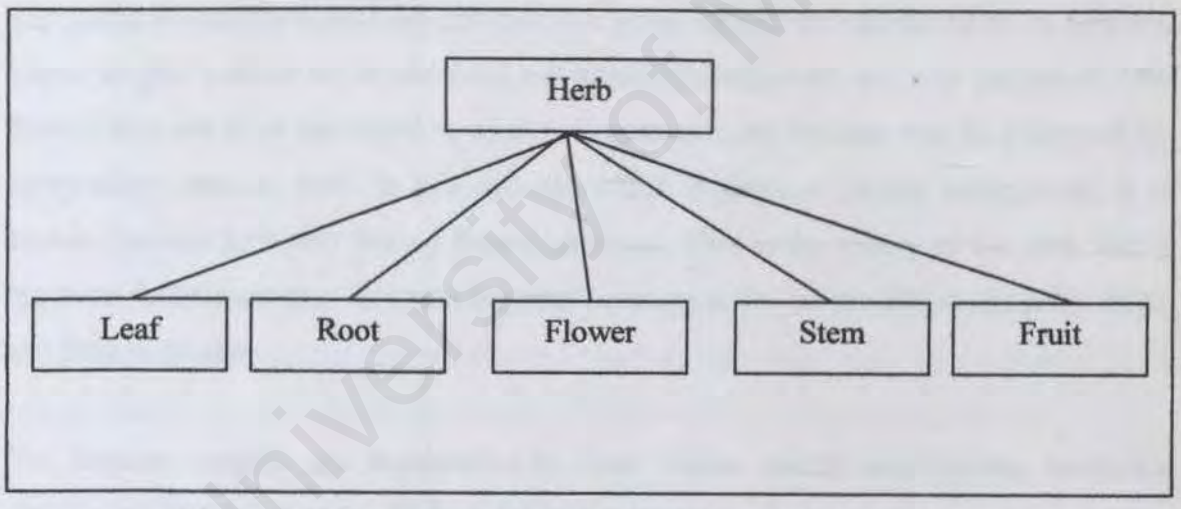


Figure 4.9 : Representation of a Herb Based on its Features

Herb's Name		Tongkat Ali
Family		Simaroubaceae
Scientific name		Eurycoma Longifolia jack
Leaf	**Colour**	Green
	Shape	Oblong
	Apice	mucronate
	Base	Oblique
Stem	**Type**	normal
Flower	**Colour**	red
Fruit	**Type**	drupe

Table 4.1 : An Example of a Case and its Indexes

5.0 IMPLEMENTATION

This chapter discuss the theory applied for the CBR system for classifying Malay Traditional Herbs, how the cases are represented and indexed, and how case retrieval are done. It also explains the coding approach taken for this stage, and the tools used.

5.1 Case Representation and Indexing

Two major issues in building the case base of a CBR system are the determination of the case representational formalism and the indexing mechanism. In this CBR system for classifying Malay Traditional Herbs, each case is represented based on its features. The domain is analyzed and the important dimensions are calculated and placed in a checklist. Cases are indexed by their values along these dimensions.

The features which are used for indexing, are given certain weights based on its priority. Three weights scheme are determined, not important, important and very important. One feature may not be as important as another feature as some features may be possessed by many other cases as well. In this domain, which is more of botany background, it is known that one herb may have different attributes. This is the variety of the herb. But it has been determined that the most important feature is the leaves, which uniquely differ one herb to another.

The features weights, are represented by three values, match contribution, mismatch penalty and absence penalty. In the implemented system, these values are not fixed. User can define which features they are think more important and otherwise. The weights also can be refined. The table below on the next page shows the relationship.

Priority	Match Contribution	Mismatch Penalty	Absence Penalty
Not Important	15	10	0
Important	30	20	10
Very Important	60	60	30

Table 5.1 : Features Weights

Each case consists of the same features with different values. After the knowledge elicitation task were done, these features have been determined for case indexing.

CASE->leaf_color, leaf_shape, leaf_apice, leaf_base, stem_type, flower_color, fruit_type

Each feature would carry different weight based on priority. Retrieval will match the query case with the cases within the case library using the weights assigned to the features.

5.2 Case Retrieval

Retrieval of appropriate cases are done using two separate algorithms. First is to repartition or to reduce the case base into related cases only, and then to find a set of closest matching case using the k-Nearest Neighbor(kNN) matching algorithm.

5.2.1 Reduced Case base

The idea of a reduced case base is like this, from all the cases of the case library, and a query case to find the closest matching case in the case library, before the cases are computed to find the most similar match, the most similar cases are clustered together to form a reduced set of cases, to perform exhaustive kNN on them.

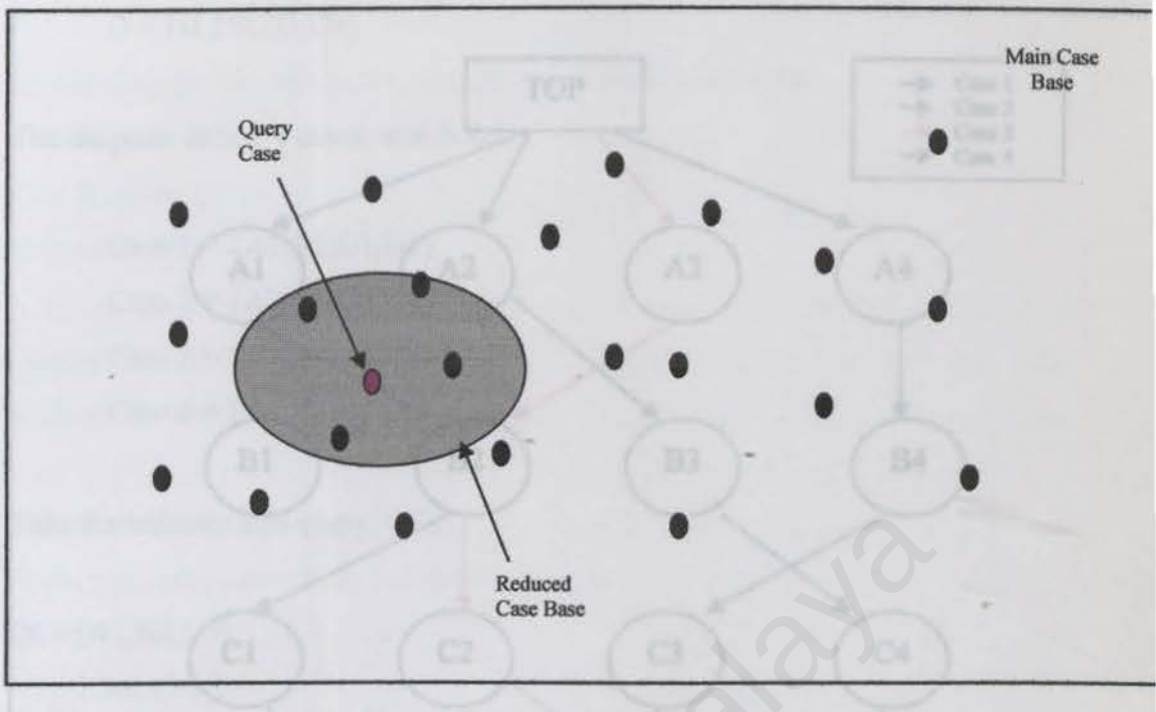


Figure 5.1 : Reduced Case Base

The cases are reduced by grouping them into a group which similar to the query case. The case base is refined for only a set of similar reduced cases. The case base can be visualized using the Hasse diagram on the next page.

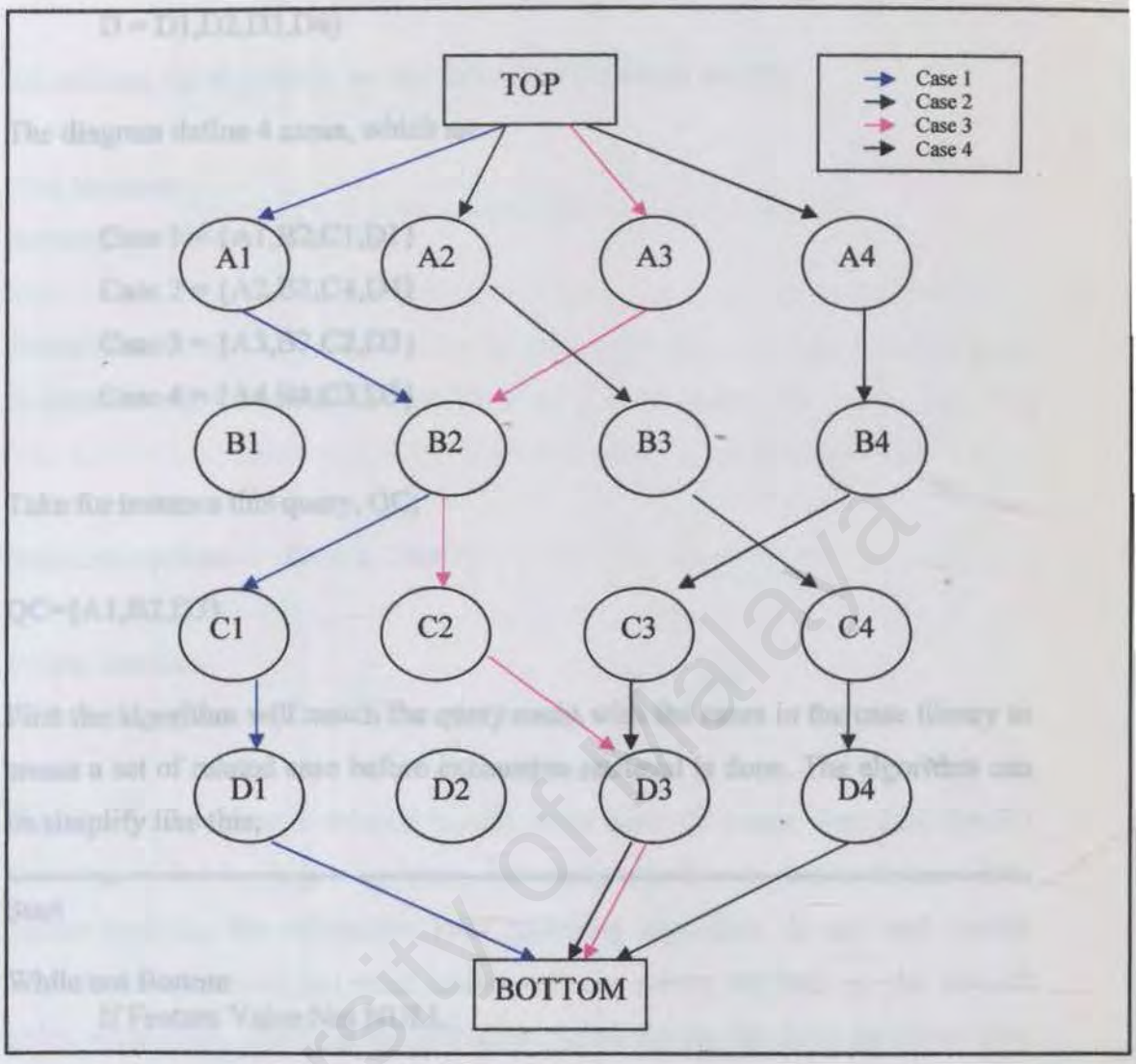


Figure 5.2 : Hasse Diagram For Case Base Representation

The figure on top represents how the cases are organized in the case library. The nodes are arranged in different levels. Each level represent a feature. The nodes in the same level are values of the features.

A = {A1,A2,A3,A4}

B = {B1,B2,B3,B4}

C = {C1,C2,C3,C4}

D = D1,D2,D3,D4}

The diagram define 4 cases, which are

Case 1 = {A1,B2,C1,D1}

Case 2 = {A2,B3,C4,D4}

Case 3 = {A3,B2,C2,D3}

Case 4 = {A4,B4,C3,D3}

Take for instance this query, QC;

QC={A1,B2,D3}

First the algorithm will match the query cases with the cases in the case library to create a set of related case before exhaustive retrieval is done. The algorithm can be simplify like this;

```
Start
While not Bottom
    If Feature Value Not NULL
        If FeatureValue = QueryValue Then
            Find related Cases
            Insert Case into ReducedCaseBase
            Move Next Feature
        Else
            Move Next Value
        End if
    Else
        Move Next Feature
    End if
Loop
Stop
```

By utilizing the algorithm, we can follow the iterations for QC,

First Iteration

ReducedCaseBase = {Case 1}

Second Iteration

ReducedCaseBase = {Case 1, Case 3}

Third Iteration

ReducedCaseBase = {Case 1, Case 3}

Fourth Iteration

ReducedCaseBase = {Case 1, Case 3, Case 4}

The case library is now reduced to only three case. Of course four case doesn't show much, but for large case bases, it is really effective to reduce the case base before applying the exhaustive kNN matching algorithm. In the real system implementation, there are more nodes and the values depends on the defined index vocabulary. There are seven features which means that there are seven level of nodes for iterations.

5.2.3 K-Nearest Neighbor Matching Algorithm

Nearest-neighbor retrieval is a simple approach that computes the similarity between stored cases and new input case based on weight features. A typical evaluation function is used to compute nearest-neighbor matching [Kolodner, 1993] as shown in Figure 2-2:

$$similarity(Case_I, Case_R) = \frac{\sum_{i=1}^{n} w_i \times sim(f_i^I, f_i^R)}{\sum_{i=1}^{n} w_i}$$

Figure 5.3 : A nearest-neighbor evaluation function

Where w_i is the importance weight of a feature, *sim* is the similarity function of features, and f_i^I and f_i^R are the values for feature *i* in the input and retrieved cases respectively. Generally, what the algorithm does is to find the closest matching case within the specified region, in this case, within the reduced case base.

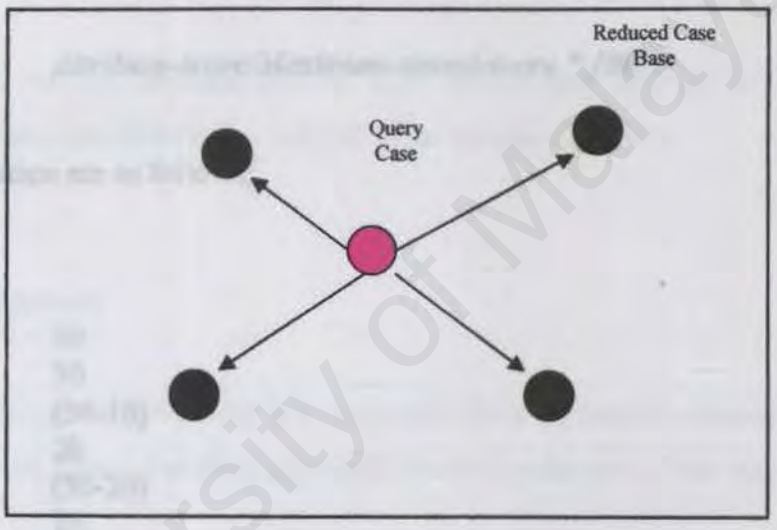


Figure 5.4 : Finding the Closest Matching Case in The Reduced Case Base

In the implemented system, the similarity are calculated using a modified version of nearest neighbor matching algorithm. There are three values which are included, match-contribution, mismatch-penalty and absence-penalty are utilized for case scoring. (Refer to Case Representation and Indexing).

Let say for example, QC={A1,B2,D3}

Lets assume that all the features priority are important. The weights are as follows;

Match-contribution = 30
Mismatch-penalty = 20
Absence-penalty = 10

The cases in the reduced case base are {Case 1, Case 3, Case 4}

The score is calculated by;

*Score = Attribute-score/Maximum-stored-score * 100*

The calculation are as follows;

Case 1

For A = 30
For B = 30
For C = (30-10)
= 20
For D = (30-20)
= 10
Total = 90
Score = 90/120*100
= 75 %

Case 3

For A = (30-20)
= 10
For B = 30
For C = (30-10)
= 20
For D = (30-20)
= 10
Total = 70
Score = 70/120*100
= 58 %

Case 4

For A = (30-20)
= 10
For B = (30-20)
= 10
For C = (30-10)
= 20
For D = 30
Total = 70
Score = 70/120*100
= 58 %

The results are Case 1=75%, Case 2=58%, Case 4=58%

As a result, we can conclude that the most similar case with QC is Case 1. The other two cases are also similar but not of the closest similarity.

5.3 Coding Approach

Coding is an iterative process whereby it is done until the programmer obtains the desired results. There are two types of coding approach, one is top down and the other is bottom up.

The top down approach slows the higher level modules to be coded first before the lower level modules. The codes in the lower level modules contains only an entry and an exit. A module with such characteristics is called a shell. The higher level modules will reference the lower ones if they are coded and available. This approach will ensure that the most important modules will be developed and tested first. It also gives a preliminary version of the system sooner.

The bottom up coding is based on coding some complete lower level modules and leaving the high level modules merely as skeletons that are used to call the lower modules, whereas the top down approach is reverse.

For this system, coding is done with the bottom up approach. The advantages of this approach are testing can be carried out on some of the functions as soon as it is completed, and critical functions can be coded first to test their efficiency.

5.4 Coding Tools

The final system was implemented in Visual Basic, and using Microsoft Access to store the data. The component used for Visual Basic to connect to the database is the Microsoft ActiveX Data Object Library 2.5. The system was built and compiled on a Windows 98 platform.

6.0 SYSTEM TESTING

6.1 Introduction

Testing is the process of executing the applications program with the intent of finding error. Before going live, the newly developed system should be thoroughly tested. This is achieved using well planned testing strategies and realistic data so that the entire testing process is methodically and rigorously carried out. In fact, testing cannot show the absences of faults, it can only show that software fault.

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, designs and code generation. In this chapter, software testing strategies and software debugging methods will be presented.

6.2 Testing Strategies

A strategy used to test this CBR system is actually a series of steps that are implemented sequentially. There are three types of testing, namely, unit testing, module testing and integration testing. After a program is completely coded, it will be tested under unit testing. Unit testing makes a heavy use of white-box testing technique, exercise specific paths in the module's control structure to ensure complete average and maximum error detection. Module testing will start when all the programs under a particular module have been completely coded and tested under unit testing. The integration testing is to recover errors associated with interfacing when integrating all the modules.

The objective of testing is to find error and fault. Fault identification is the process of determining what fault or faults caused the failure, and fault correction or fault removal is the process of making changes to the system so that the faults are removed.

6.3 Unit Testing

Unit testing tries to look for all the possible errors that will occur in a program. A complete test procedure should test all of the following categories of test data;

a) Normal data – to test a given correct data will produce the expected results
b) Erroneous data - for a given erroneous data, like invalid date format, does the system detect it or not?
c) Boundaries value analysis – data that are out of range specified will be used to test the system because errors may occur at the extreme point.
d) Condition testing data – some functions may be active under certain condition, therefore a set of data are tested on all possible condition.

For this case based reasoner, unit testing involves testing each program on its own, isolated from the other programs in the system. The following steps specify how unit testing is carried out for this system;

a) The codes of the program are examined by reading through it to spot for algorithmic faults and syntax faults.
b) All command buttons, text boxes and other control objects are tested to check its functionality.
c) Different types of test data are used like number, character, date and etc. to test all the control objects.
d) Test cases are developed to ensure that the input is properly converted to the desired output.

6.3.1 Examining the Code

The codes of the program are read and walked through with documentation to identify faults. This method is useful to identify faults that have been left out by the programmer.

6.3.2 Control Objects Testing

Command buttons are clicked to test their functionality and text boxes are tested with different data types and also null value to make sure invalid data will not cause any fault.

6.3.3 Different Data Type Testing

Different data types like numbers, characters or date is used to test certain function because some control objects will only accept certain data type, invalid data type can be traced by the system without causing any error.

6.4 Module Testing

Module testing is to test the MDI form of the system. All the programs under a sub module are grouped into one form and all the related forms are grouped into a module. This testing will make sure menu bar choices will make the correct form active and the control will pass back the specific form when the current form is closed. In this CBR system, testing of the individual class module is merely compiling the individual module. If error is found, debugging of the codes will be carried out immediately. If the compilation of the module is completed successfully, another module will be coded.

6.5 Integration testing

When the individual program worked properly, integration testing is started. If there is a fault during the testing, the fault does not lie within the unit of the system. Sandwich integration testing approach is used for the system. This approach combines top down strategy with bottom up strategy. The testing starts from the simplest form of the system and down to the lowest level of the form function back to the MDI form. This testing is repeated several times to make sure that all the control objects work properly.

6.6 Summary

From the testing that had been carried out, the errors are detected and debugged. The expected errors, for example to handle erroneous data, error handler are coded to handle expected error. All the changes are recorded for future references.

7.0 SYSTEM EVALUATION

This chapter will be divided into two section. The first section discuss the evaluation of the CBR system, and the second section will discuss about the application evaluation.

7.1 CBR Evaluation

In a CBR system, the performance of a case based reasoner can be measured according to three criteria;

1) Efficiency – the average problem solving time
2) Competence – the range of target problems that can be successfully solved
3) Quality – the average quality of a proposed solution

We are going to test two algorithm for this CBR system. The k-Nearest Neighbor on a reduced case base and the exhaustive k-Nearest Neighbor. The three criteria are compared and evaluated.

7.1.1 Efficiency

The first strategy is concerned with evaluating the efficiency of the retrieval algorithms over a range of case-base sizes. Efficiency is measured as the inverse of the number of cases examined during retrieval. This is a fair measure as the two algorithms perform simple search through a set of cases using the same similarity operator.

To evaluate this criteria, the method is to compare the processing time for both of the retrieval algorithm as the size of the case base increases. A function for calculating the elapsed time for each retrieval was defined.

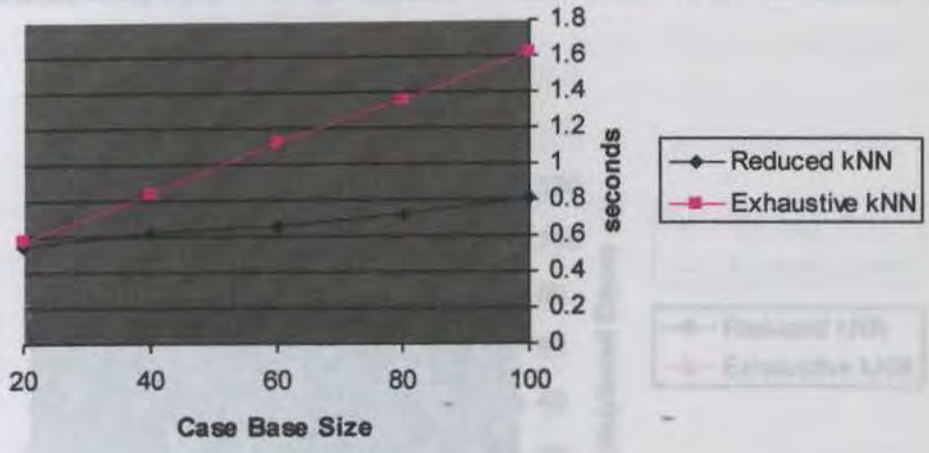


Figure 7.1 : Comparisons between Reduced kNN and Exhaustive kNN Vs Time

The figure compares reduced kNN and the exhaustive kNN. It compares the processing time between the two retrieval approach as the case base size increases. The figure also shows that the processing time for exhaustive kNN is more than the reduced case base approach. It can be concluded that reduced kNN is more effective than the exhaustive kNN approach.

For reduced kNN, the cases which are to be computed are only the cases within the case base. Meanwhile, for exhaustive kNN, it will calculate the whole cases in the case base. This explains why reduced kNN is more effective than exhaustive kNN.

7.1.2 Competence

There is typically a tradeoff between the efficiency and competence of a retrieval technique. In particular, since the reduced kNN approach do not examine every case in the case base, it is possible that important case are missed during retrieval thereby limiting the overall problem solving competence. In this experiment, we look at how each retrieval method performs in terms of competence, where

competence is defined to be the number of cases retrieved for calculation of the kNN.

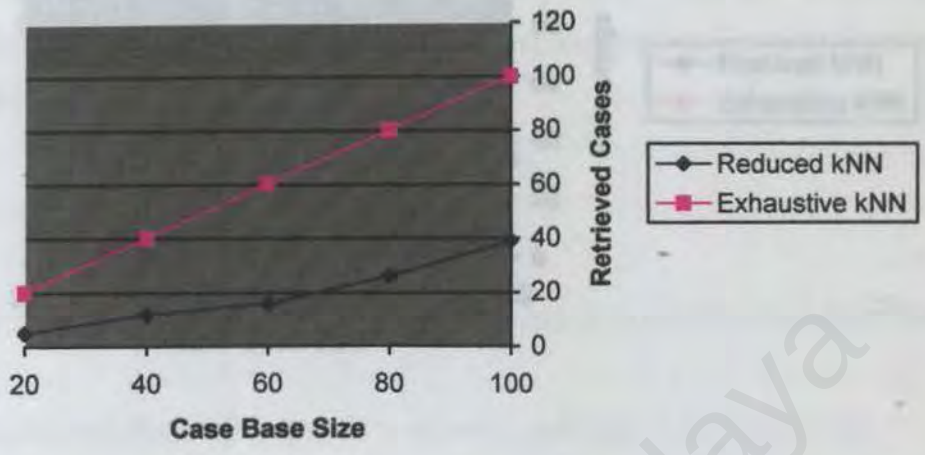


Figure 7.2 : Comparisons between Reduced kNN and Exhaustive kNN Vs Retrieved Cases

This figure indicates that the exhaustive kNN retrieve more cases than the reduced kNN. As the case base increase, the retrieved cases using the exhaustive method also increase.

For exhaustive kNN, all the cases in the case base are retrieved for computation. Meanwhile, the reduced kNN only retrieve the most similar cases only. Therefore, the exhaustive kNN is more competence than the reduced kNN approach.

7.1.3 Quality

To evaluate the quality of the proposed solution, the proposed solution of both algorithm are used. The percentage of solution proposed divided by the cases retrieved are used to gauge the quality of the case based reasoner. The higher the percentage of available solution indicates that the system is of better quality.

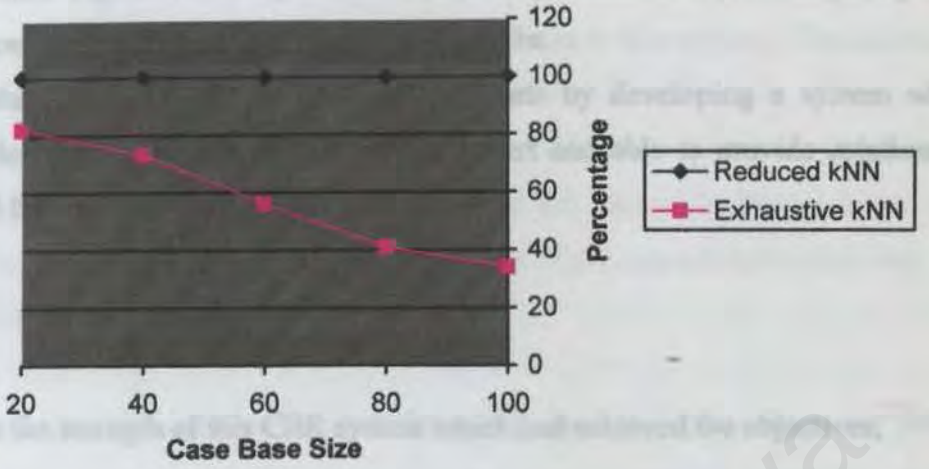


Figure 7.3 : Comparisons between Reduced kNN and Exhaustive kNN Vs Percentage of Proposed Solution

The figure indicate that the reduced kNN approach produced more consistent and all the cases retrieved is a solution. However, the exhaustive kNN, not all the retrieved cases are appropriate to become a solution. The reduced kNN reduced the case base only for the most similar cases before the scores are calculated, that explains why all the cases retrieved using the reduced kNN approach are more suitable to become a solution.

As a conclusion, the reduced kNN are of better quality than the exhaustive kNN.

7.2 System Evaluation

Some of the objectives stated in introduction had been successfully achieved. However, due to project boundaries, there are limitations in this CBR system.

This system is developed to achieve the objectives as below;

1. To apply case-based reasoning techniques on classifying Malay traditional herbs.

2. To provide a CBR system with a friendly user interface and easy to use.
3. To imitate experts in Malay herbs in ways they reason to a problem justify their solutions in determining Malay traditional herbs.
4. To reduce the dependency on domain experts by developing a system which provides the knowledge of the domain expert and able to provide solutions as would the real domain expert.

7.2.1 System Strength

Below are the strength of this CBR system which had achieved the objectives;

1) Case Based Reasoning applied to classifying Malay Traditional Herbs
The system utilize the CBR method to identify and classify the numerous type of Malay Traditional Herbs. Therefore it possess the quality of how an intelligent system should be.

2) Simple and user friendly interface
User interface in this CBR classifier is easy to understand and user friendly. In addition, the user interfaces are designed to suit a wide spectrum of user. User manual is also included to help user handle the system effectively.

3) Provide justifications to proposed solutions
The proposed solution are justified to inform the user how the scoring was done, and how the justification resolves the proposed solution.

4) Manageable and easy to access Case Base
The developed case base are easy to manage and the cases are easy to access and modify using the developed user interface.

7.2.2 System Limitations

Due to project boundaries, there are some limitations in this system. The limitations are stated as below;

1) The domain of the system is restricted to the Malay Traditional herbs only, therefore, the user interface is designed and the cases are indexed in way that will accommodate this domain.

2) The index vocabulary are predefined in the case base. Uses of natural language, to check wrong spelling or to accommodate synonyms are not supported.

3) The system is a stand alone system which means that the case library is not accessible via the network or the internet. It is an independent system which does not require any network support.

7.3 Future Enhancement

For future enhancement, we might consider building a web-based CBR application so that the case library may be retrieved and shared within the World Wide Web. A web-based system is flexible in terms that it may be shared publicly and accessed anywhere at any time.

Beside that, we would consider apply this system to a larger domain such as classifying trees, flowers, microorganisms, and even animals. The evaluation of this system will determine the effectiveness of applying Case-Based Reasoning for this kind of problem.

7.4 Conclusion

As a CBR system, using the reduced case base approach, this system are more effective and of better quality than the exhaustive kNN approach. However when competence is being concern, the exhaustive kNN approach is more competence than the reduced case base approach.

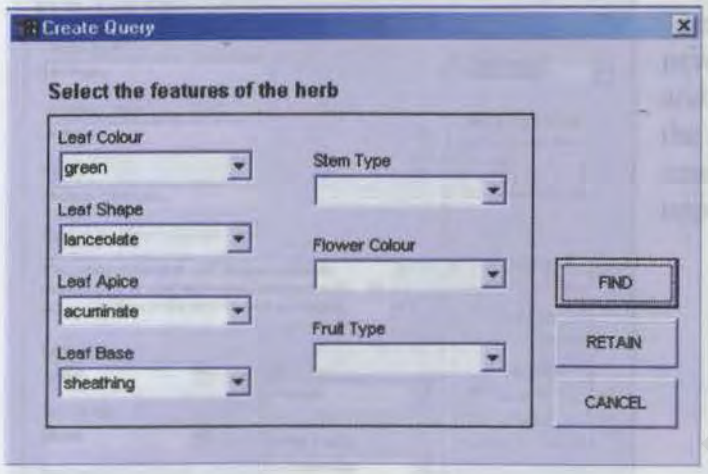
Overall, the system has achieved most of the main objectives and was able to be delivered on time. Most of the objectives and the requirements defined during the analysis phase are fulfilled and implemented in this system.

USER MANUAL

CBR System for Classifying Malay Traditional Herbs
Author : Ismaily bin Johari

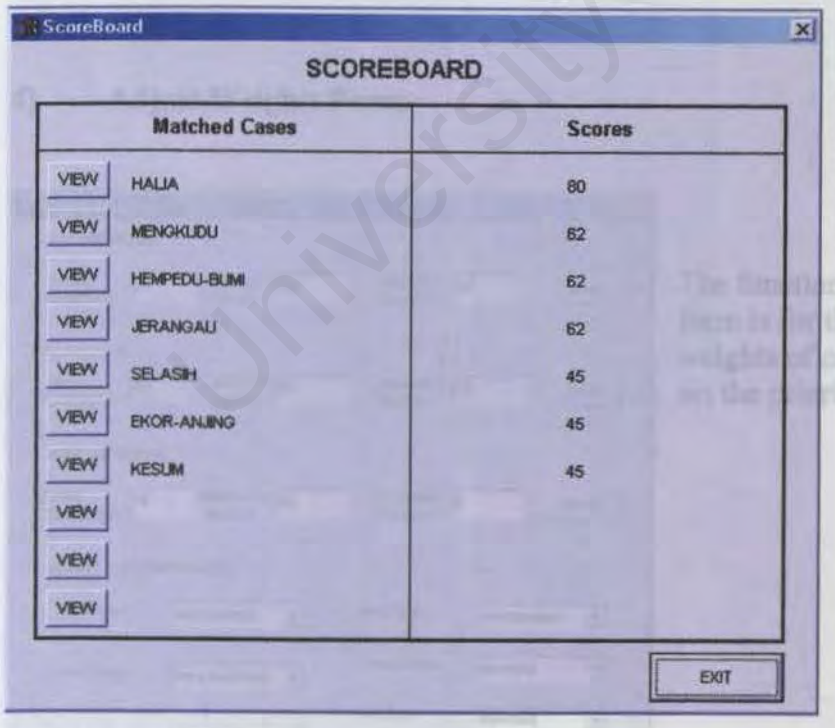
Description of Forms

1) Query Form



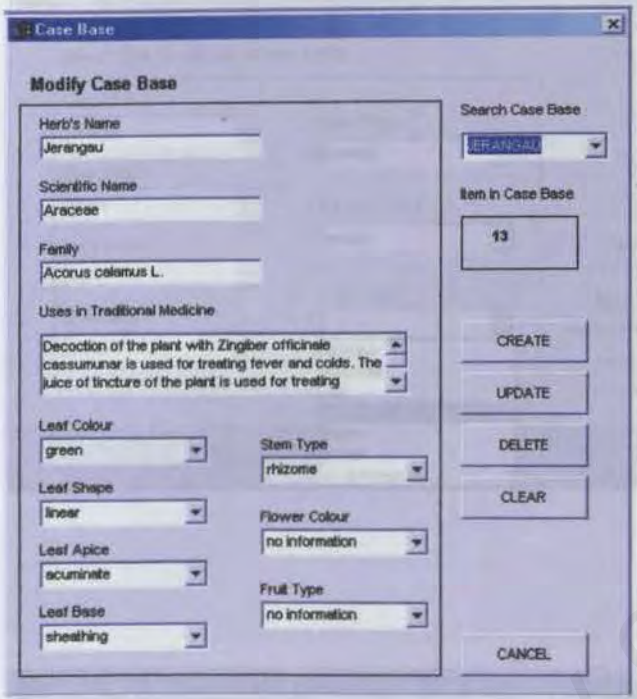
The function of this form is to make queries and to retain or save the queries into the case base.

2) Score Board Form



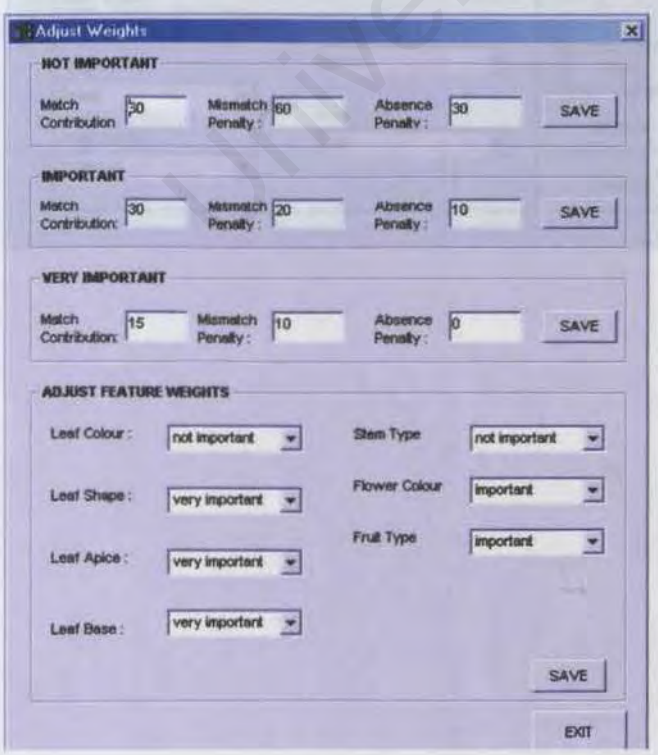
This form display the result of the query done. User may click on View button next to the index name of the herb to view the details of the herb.

3) Case Base Form



The function of this form is to create new cases, to update existing cases, and to delete the existing cases from the case base. User may also search the case base using the combo box on the upper right.

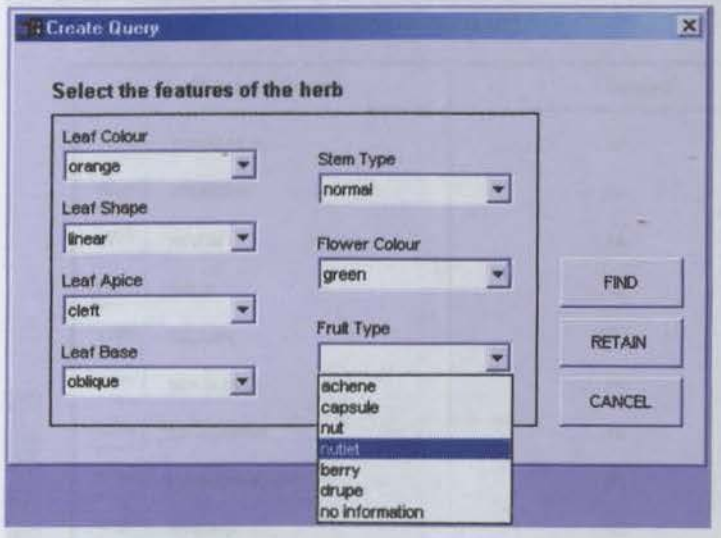
4) Adjust Weights Form



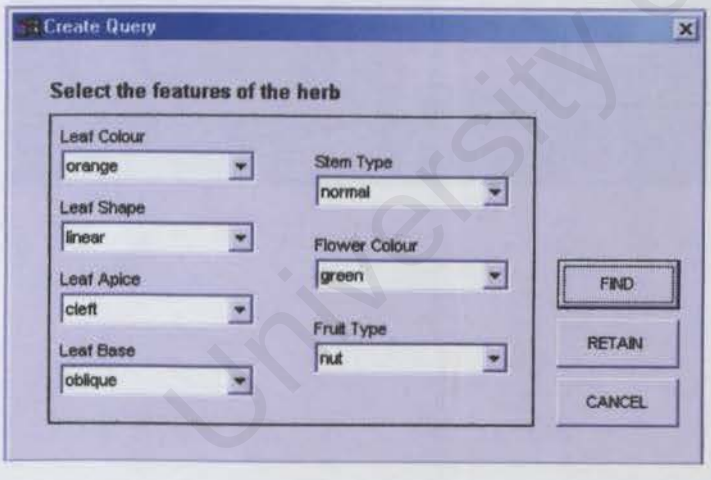
The function of the Adjust Weights form is for the user to adjust the weights of each feature as well as to set the priority of each feature.

Using the System

A) To Make a Query

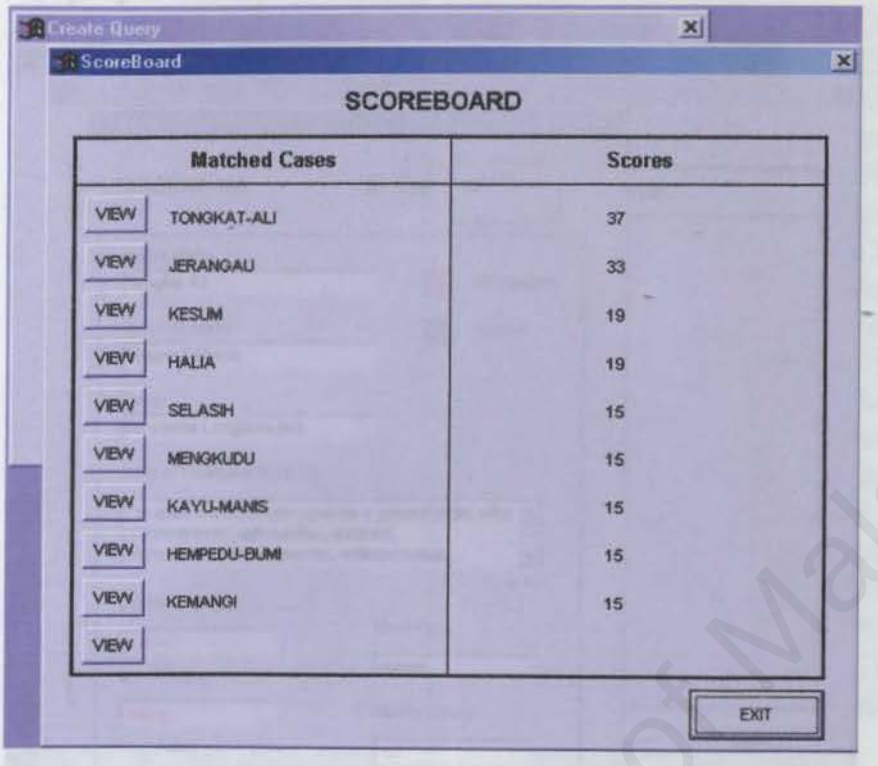


1) Select the features of the herbs. For unknown feature, user may just leave the combo box blank.

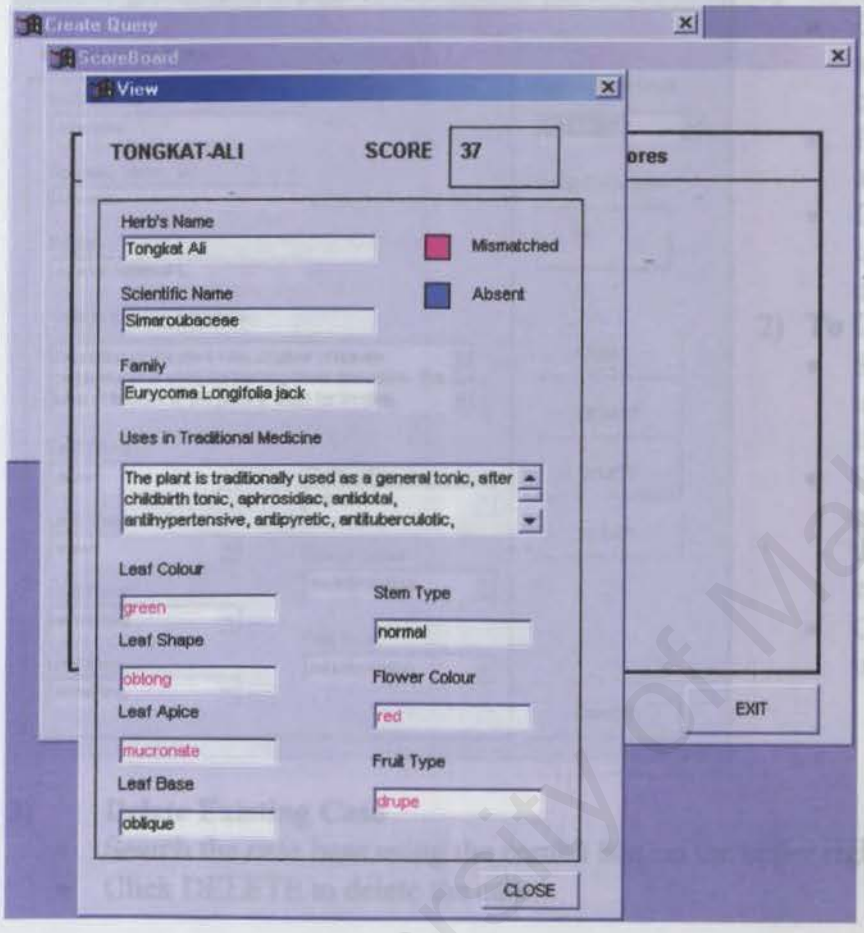


2) Click FIND to begin search

3) The Score Board will appear to display the results of the search. Click VIEW on the left of the index name of herbs to view the details of the herbs.

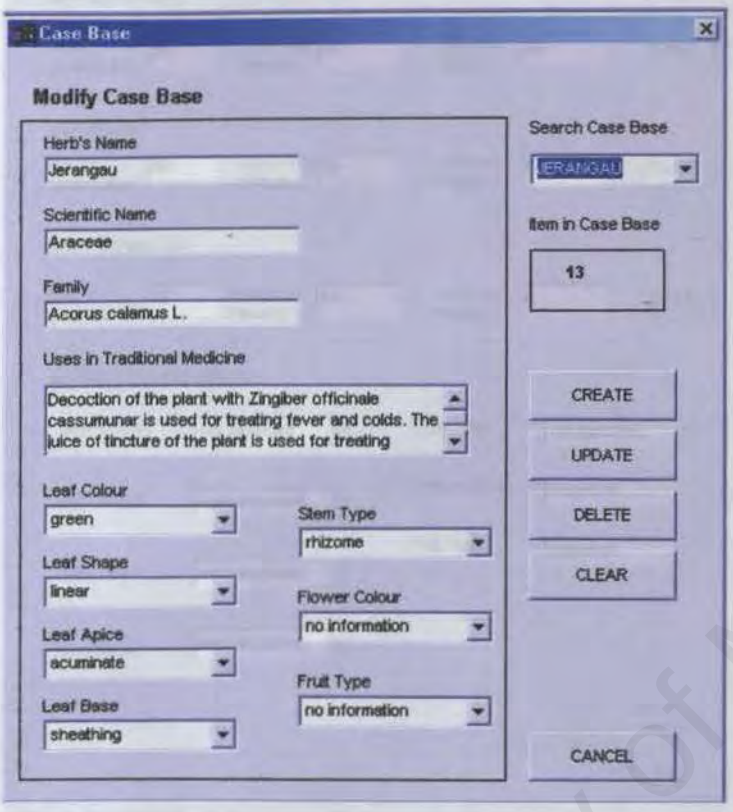


4) The View Window will appear to display the details of the herbs. Normal text indicate that the values matched with the query values, Red indicates that the values are mismatched with the query values while blue text indicates that the values are absent in the query window.



B) To Modify Case Base

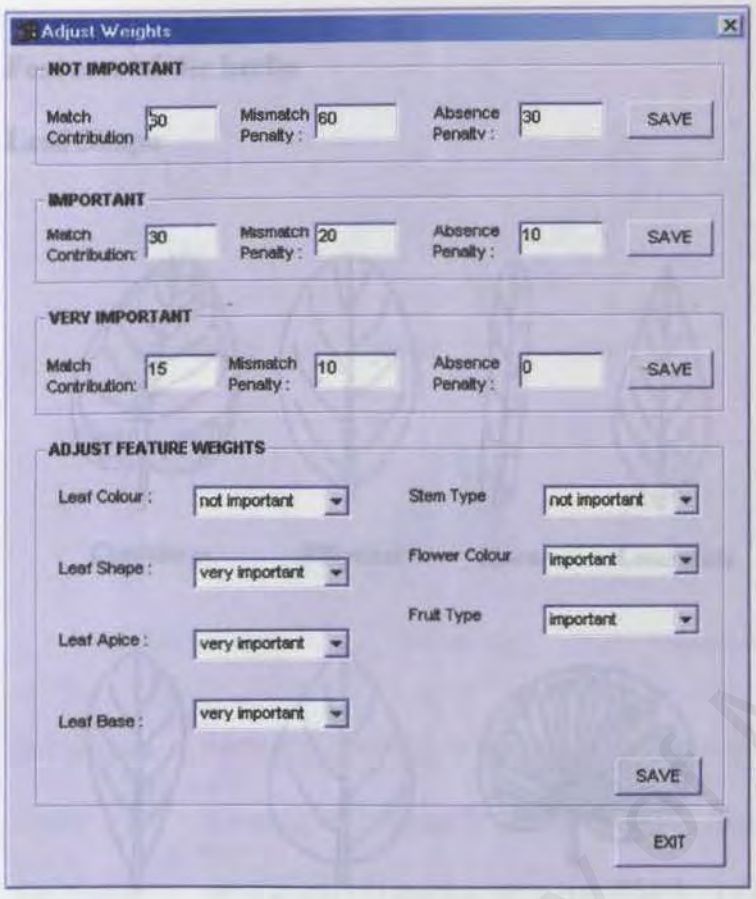
The Case Base window is used to modify the cases in the case base.



1) **To Create New Case**
   - Click CLEAR to clear all the textbox and combo box.
   - Fill in the text boxes and combo boxes.
   - Click CREATE to insert the new case into the case base.
2) **To Update Existing Case**
   - Search the case base using the combo box on the upper right.
   - Edit the case by adjusting the values in the text boxes and combo boxes.
   - Click UPDATE to update case base.

3) **Delete Existing Case**
   - Search the case base using the combo box on the upper right.
   - Click DELETE to delete the case.

C) Adjust Weights



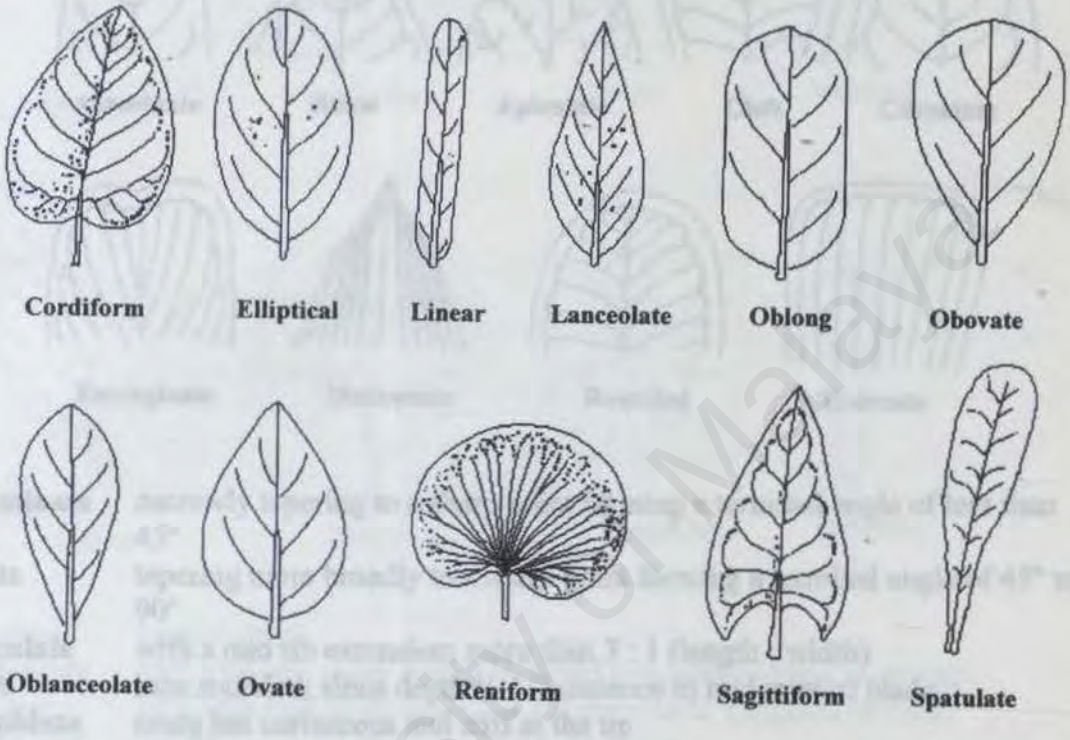
1) For each frame in the form (NOT IMPORTANT,IMPORTANT,VERY IMPORTANT and ADJUST FEATURE WEIGHTS), user will has to click SAVE each time they change the values in order for the changes to be applied.

2) User may set the priority of the features by adjusting the weights of the features in the ADJUST FEATURE WEIGHTS frame.

GLOSSARY

Features of the herbs

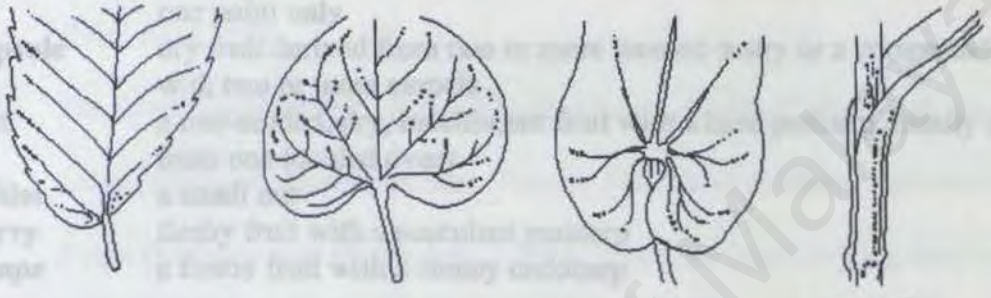
Leaf Shape



cordiform	heart-shaped
eliptical	shape of a flattened circle, usually twice as long as broad
linear	long and narrow with nearly parallel sides
lanceolate	lance-shaped, tapering from a broad base; much longer than wide
oblong	twice as long as broad and with sides parallel most of their length
oblanceolate	lanceolate, but with the broadest part near the apex
obovate	ovate, but the broadest part near the apex
ovate	egg-shaped with the broadest part towards the base
reniform	kidney-shaped
sagittiform	triangular-ovate with straight or slightly curved basal lobes
spatulate	oblong or abovate apically

Leaf Base

Attenuate Auriculate Cordate Decurrent



Oblique Reniform Saggittate Sheathing

Attenuate	long tapering
Auriculate	lobe rounded; outer margin concave and inner convex or straight
Cordate	lobe rounded; sinus depth 1/8-1/4 distance to midpoint of blade
Decurrent	extending along downward from leaf base
Oblique	having an asymmetrical base
Reniform	lobe rounded; outer margin convex or straight, inner concave
Saggitate	basal lobes drawn into points on either side of the petiole like the base of an arrowhead
Sheathing	having tubular structure enclosing the stem below apparent insertion of blade or petiole

Stem Type

normal	normal stem
stem tendrils	slender twining branches used to support climbers
thorns/spines	sharp and stunted branches
stolons/runners	stems that trail above the ground, often rooting at the nodes
rhizome/rootstock	thickened stem creeping horizontally underground and producing new shoots at their tips
tuber	an underground swollen branch that normally serves for food storage and vegetative growth

Fruit Type

achene	a one-seeded, dry, indehiscent fruit with seed attached to the fruit wall at one point only
capsule	dry fruit derived from two or more loculed ovary or a compound ovary with two or more carpels
nut	a one-seeded, dry, indehiscent fruit with a hard pericarp,usually derived from one loculed ovary
nutlet	a small nut
berry	fleshy fruit with a succulent pericarp
drupe	a fleshy fruit with a stoney endocarp

REFERENCES

[1] Janet Kolodner, ***Case-Based Reasoning***, Morgan Kaufmann Publisher, 1993

[2] Case-Based Reasoning, http://www.aiai.ed.ac.uk/links/cbr.html#intro

[3] Peter Jackson, ***Introduction to Expert Systems third edition***, Addison Wesley Publisher 1999

[4] P. Sellapan, ***Software Engineering, Management and Methods***, Sejana Publishing, 2000

[5] Dr. Abdiul Rahman Md. Derus, ***Herba Ubatan***, Multi Triple Vision Sdn. Bhd

[6] Davis, ***Software Requirements: Objects, Functions & States***, Prentice-Hall, 1993

[7] Pusat Pembangunan Perniagaan, Jabatan Pertanian Semenanjung Malaysia, http://agrolink.moa.my/pusat_sumber/

[8] Viable Herbal Solutions Home Page, http://www.viable-herbal.com/homepage.htm

[9] Whitten, Jeffrey L. & Bentley, Lonnie D., ***System Analysis and Design Methods***, McGraw-Hill International Editions, 1998

[10] Ralph Bergmann and Klaus-Dieter Althoff, ***Methodology for Building CBR Applications***

[11] R. Bellazi, S. Montani, and L. Portinale. ***Retrieval in a prototype based case library: A Case Study in diabetes therapy revision***. In European Workshop on Case based reasoning, 1998

[12] S.Barry & E.McKenna, ***Footprint Based Retrieval***, University College Dublin, 2001

[13] D.Wettschereck & D.W.Aha, ***Weighting Features,*** German National Research Center for Computer Science

[14] G.Finnie & Z.Sun, ***R5 Model for Case Based Reasoning,*** Elvesier Science's Knowledge based System journal, Issue 16 (59-65), 2003

[15] B.D.Agudo & P.A.Gonzalez-Calero, ***Formal Concept Analysis as a support technique for CBR,*** Elvesier Science's Knowledge based System journal, Issue 14 (163-171), 2001

[16] D. McSherry, ***The inseparability problem in interactive case-based reasoning, ,*** Elvesier Science's Knowledge based System journal, Issue 15 (293-300), 2002

[17] F.Ricci & paolo Avesani, ***Learning a Local Similarity Metric for Case Based Reasoning,*** Instituto per la Ricerca Scientifica e Technologica, 2002

[18] Indu Bala Jaganath & Ng Lean Teik, ***Herbs: The Green Pharmacy of Malaysia,*** Malaysia Agricultural Research and Development Institute (MARDI), Kuala Lumpur, 2002