



University of Malaya

Perpustakaan SKTM



**Nutritional Advisory System
Using Case Based Approach**

By
Chew Wei Liang
WEK 000 112

Under the supervision of
Assoc. Prof. Dr. Syed Malek

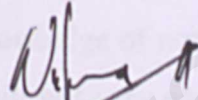
And moderated by
Mr. Woo Chaw Seng

Submitted to the
Faculty Computer Science and Information Technology
University of Malaya

In partial Fulfillment of the Requirement For
The Degree of
Bachelor of Computer Science
Session 2002/2003
Submission Date: 7 February 2003

DECLARATION

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university



Chew Wei Liang

February 2002

DISCLAIMER

The ownership of this report is reserved by University of Malaya and no part of this assignment should be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording or otherwise without prior written consent from University of Malaya.

ACKNOWLEDGEMENT

To finish an academic exercise, the effort that the people supporting is no doubt a enormous task. Finally I would like to express my almost gratitude to my supervisor, Assoc. Prof. Dr. Syed Malik for his guidance, advice and encouragement. I salute him with a small of applause.

ABSTRACT

In today world, people are getting more and more aware of the important toward a healthy living. There is a need to cater this people with the knowledge of nutrition they want. As today, the number of nutritionists and dieticians are still not very encouraging. So to fill in the vacancies, a nutritionist advisory system should be developed to improve the current lacking. To accomplish this task, case-based reasoning seems to be a better approach.

Rule-based systems need a good definition of the system and are not able to learn very well from the user choices. What is more, it is not able to handle missing information or unexpected values. On the other hand, a nutritional advisory system build with a case-based reasoning system should be able to solve these problems.

In order to show the benefits of a case-based reasoning system for a nutritional advisory system, the system will follow all the guidelines and recommendation from World Health Organization (WHO) and US RDA. The system involves a full case-based reasoning cycle and will focus very much in the adaptation part of the cycle. It will then be used to demonstrate the flexibility and adaptability of this kind of system.

ACKNOWLEDGEMENT

To finish an academic exercise, the effort that the people supporting is no doubt a enormous task. Firstly I would like to express my utmost gratitude to my supervisor, Assoc. Prof. Dr. Syed Malek for his guidance, advice and encouragement. I salute him with a round of applause.

Special thank to Mr. Woo Chaw Seng, the project moderator for his valuable suggestion and comments. And a big thanks to my expert domain, Mr. F.E Chong for his time and suggestion and also to Dr. Lau Wah Keong for his patient in explaining the important of nutrition to me.

Thanks to all my fellow course-mates for sharing their knowledge with me throughout the whole project. And also to all my fellow seniors who had done a great jobs in advising me on what to do during the system development stage.

Last but not least, I would like to thanks my family for their continued encouragement and support. I also like to thank my mum again for allowing me to borrow all her health books to me.

CHAPTER 3: METHODOLOGY & SYSTEM ANALYSIS

3.1 Methodology

27

3.1.1 Prototyping solves the problems of Traditional Waterfall Model

28

TABLE OF CONTENTS

LIST OF FIGURES LIST OF TABLES

I
II

CHAPTER **1** : INTRODUCTION

1.1 PROJECT DEFINITION	1
1.2 OBJECTIVES	2
1.3 PROJECT SCOPE	3
1.4 SYSTEM LIMITATIONS	4
1.5 REPORT LAYOUT	4
1.6 PROJECT SCHEDULE	6

CHAPTER **2** : LITERATURE REVIEW

2.1 INTRODUCTION TO CASE BASED REASONING	7
2.1.1 Case-based problem solving	9
2.1.2 Learning in Case-based Reasoning	9
2.1.3 Combining cases with other knowledge	10
2.2 Case-Based Reasoning Techniques	10
2.3 Problems in CBR	24
2.4 CBR and Other Reasoning Methods	24
2.4.1 Advantages of CBR	25
2.4.2 Disadvantages of CBR	26

CHAPTER **3** : METHODOLOGY & SYSTEM ANALYSIS

3.1 Methodology	27
3.1.1 Prototyping solves the problems of Traditional Waterfall Model	28

3.2 Cased Based versus Rule Based Reasoning	31
3.2.1 Advantages of Rule Based Reasoning	31
3.2.2 Disadvantages of Rule Based Reasoning	32
3.2.3 Advantages of Case Based Reasoning	32
3.2.4 Disadvantages of Case Based Reasoning	32
3.2.5 Conclusion	33
3.3 Analysis of the implementation of CBR in Nutritional Advisory System	34
3.3.1 Case Representation and indexing in Nutritional Advisory System	36
3.3.1.1 The Contents of Problem Representation	36
3.3.1.2 The Content of Solutions	37
3.3.1.3 Methods for Index Selection in Nutritional Advisory System	37
3.3.2 Case Retrieval in Nutritional Advisory System	38
3.3.2.1 Identify Features	38
3.3.2.2 Initially Match	38
3.3.2.3 Select	38
3.3.3 Case Reuse in Nutritional Advisory System	39
3.3.4 Case Retainment in Nutritional Advisory System	39
3.4 Development Environment	40
3.4.1 Operating System	40
3.4.2 Case Based Reasoning Shell and Programming Language	43
3.4.2.1 CBR engine	44

CHAPTER 4: SYSTEM DESIGN

4.1 System and User Requirement	45
4.1.1 Functional Requirements	45
4.1.2 Non-Functional Requirements	46
4.2 System Overview	48
4.3 Module Design	51
4.3.1 Identify Module	51
4.3.2 Matching Module	52
4.3.3 Similarity Module	53
4.3.4 Adaptation Module	55
4.3.5 Evaluation Module	56
4.3.6 Display Module	57

4.3.7 Retain Module	58
4.4 Case Design	59
4.5 System User Interface	59

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 Introduction	60
5.2 Development Environment	60
5.2.1 Hardware Requirements	60
5.2.2 Software Tools / CBR Shell Requirements	61
5.2.2.1 1 Description of Development Application / Tools	62
5.3 System Implementation	63
5.4 Interface	63
5.5 Nutritional Advisory System Framework Implementation	64
5.5.1 Attributes Weighting Implementation	65
5.5.2 Matching Implementation	69
5.5.3 Adaptation Implementation	70
5.5.4 Display Module Implementation	71
5.5.5 Function Implementation	73

CHAPTER 6: SYSTEM TESTING

6.1 Introduction	74
6.2 Algorithm Testing	76
6.2.1 Testing the Accuracy of the Retrieval	77
6.3 Weighting Test	77
6.4 Testing the Causal Model	78

CHAPTER 7: SYSTEM EVALUATION & CONCLUSION

7.1 Introduction 80

7.2 Problem Encountered and Solutions 81

7.3 System Strengths 83

7.4 System Limitations 84

7.5 Future Enhancement 85

7.6 Conclusion 86

BIBLIOGRAPHY 89

APPENDICES

Source Code 91

Journal 104

LIST OF FIGURES

PAGE	FIGURE
14	Figure 2.1: CBR Cycle
21	Figure 2.2: A Task method Decomposition of CBR
27	Figure 3.1: TheProject Process Model
41	Figure 3-3: Windows 2000 mean time to failure greatly exceeds that of Windows NT Workstation 4.0 and Windows 98.
42	Figure 3-4: Windows 2000 Professional simply did not fail during the 90 days it was tested by ZD Labs.
50	Figure 4.1: Data Flow Diagram for Nutritional Advisory System
51	Figure 4.2: Data Flow Diagram of Identity Module
52	Figure 4.3: Data Flow Diagram of the matching module
54	Figure 4.4: Data Flow Diagram of the similarity module
55	Figure 4.5: Data Flow Diagram of the adaptation module.
56	Figure 4.6: Data Flow Diagram of the evaluation module.
57	Figure 4.7: Data Flow Diagram of the display module
58	Figure 4.8: Data Flow Diagram of the retain module
59	Figure 5.1: Nutritional Advisory System Framework
60	Figure 5.2: Causal Model
61	Figure 5.3: Command Interpreter
62	Figure 5.4: Command Interpreter

CHAPTER 1

LIST OF TABLES

INTRODUCTION

PAGE	TABLE
6	Project Schedule
61	Workstation
65	Weighting –Match Contribution
67	The Hierarchical Pyramid
68	Structure of Case Retrieval Net

1.1 Project Definition

Not many people know about the truth about health – especially about the roles of diet and nutrition in our personality, behavior and state of well-being. Nutrients provide the foundation for a global look at the social sciences that shape the dietary patterns of people throughout the world. Most of the people will just treat the symptoms of their illness but not the reasons for the symptoms [1]. Of course, we still need a practitioner and his diagnostic skills. His therapy usually is limited to a certain extend of relieving the pain, which is short of the mark. Until the medical profession has all the answers, the patient still need to take some responsibility in his or her daily food intake. But for the time being, the technology we have now is adequate to produce a system that enable the patient to access to the information the patient needs to recoverate from a particular illness using alternative medicine. This includes the warning and suggestions the patient needs to recover from the illness after treatment from medical practitioner.

CHAPTER 1

INTRODUCTION

This introductory chapter gives a brief description or purpose of the project and problems to be solved. The significance and rationale of the project will be discussed here. Furthermore, the system functions, limitations and its assumptions will also be dwelt into later in this chapter.

1.1 Project Definition

Not many people know about the truth about health –specifically about the roles of diet and nutrition in our personality, behavior and sense of well-being. Nutrients provide the foundation for a global look at the social sciences that shape the dietary patterns of people throughout the world. Most of the people will just treat the symptoms of their illness but not the reasons for the symptoms [1]. Of course, we still need a practitioner and his diagnostic skills, but his therapy usually is limited to a certain extend of relieving the pain and falls short of the mark. Until the medical profession has all the answers, the patients still need to take some responsibility in his or her daily food intake. But for the time being, the technology we have now is adequate to produce a system that enable the patient to access to the information the patient needs to recuperate from a particular illness using alternative medicine. This includes the nutrition and supplements the patient needs to recover from the illness after treatment from medical practitioner.

This project will focus mainly on giving nutritional advises to patient but also have the ability to give advises to healthy people. The whole project is mainly concentrated on case-based reasoning techniques. During the process the system will try to simulate as close as possible to a nutritionist in giving advises to their patients.

1.2 Objective

There are some reasons or objectives which this system has been proposed

1) *As an alternative way to get advises in term of nutrition and diet*

Most of the people in Malaysia is still uncomfortable of getting advises from nutritionist or dietician after they get treatment from medical practitioner. In the recuperating period, most of the people will just buy the vitamins of the shelves in supermarkets or pharmacies without considering the side-effect of the supplements with the medicine they were given by the doctor. To some extend this may cost

2) *Applying AI techniques and Utilizing Case-Based Reasoning in Medical sector*

In this project the main objective is to develop a system that is intelligent enough to output result that is reliable to the medical world. Not only the system is intelligent in giving out result but also must be able to reason .This means that whenever the user input any illness and their basic particular, the systems will able to generate more questions to acquire knowledge about the illness from the user. This is important to generate a reliable result that is nutritional advice in this matter. In the process of generating the result the system will rely mainly on case-based reasoning.

3) *Preventive measurement of major illness*

As for example, an inflamed appendix is a symptom of a low-roughage diet. It is too late at this stage for a nutritional approach, but this must be initiated postsurgically to avoid future difficulties with hemorrhoids, fissures, diverticulitis and possible bowel cancer, all related to nutrition and diet.

4) *To make nutritional advices accessible to all*

Not many people in Malaysia know where to find a nutrition consultant and dietician. It is very troublesome to seek advice from a nutritionist as in the country on have about 500 of them. This statistic only includes those who register under the Nutritional Association of Malaysia. Through this system, people can access to nutritional advice more easily.

1.3 Project Scope

For the time being the system will only be a template for the real system. And this system will function mainly on a set on past cases and a database that act as a dictionary for all the supplements and all the food that are related to the supplement. The scope that been covered are those people with a specific illness and those with multiple illness. There will be a sub function to monitor the result and to store the new problem as a case in the database. The cases will be input by the medical expert who uses this system and the medical expert will initialize all the requirements. This system will also generate precaution for the user regarding the medicine the user took and the food that is recommended by the system. Example some medicine like Calcitriol cannot be taken

with vitamin D or magnesium and must avoid high phosphorous foods. If taken it will cause anorexia.

1.4 System Limitation

The main idea is to have a system that can generate any result if the procedure is been followed properly. But some major illness still cannot be prevented or cure using proper food and supplement because some of these illnesses are cause by bacteria infection and virus attack e.g. dengue, meningitis, encephalitis and etc. This kind of diseases is mainly non-related to nutritional and diet.

This system also is not build for medical diagnosis. The system can not figure out what kind of illness does the patient suffering by analyzing the symptoms. It is just for the user to enter the name of illness and the system will generate a list of recommended nutrition and supplement list.

1.5 Report Layout

The purpose of the report is to document essential information gathered during this system development and implementation stage. This report is divided into 4 respective chapters.

Chapter 1: Introduction

This chapter introduces the project in a general view and the rationale of the project. It made an overview on the, definition, project objectives, project scopes, project limitation and the project schedule.

with vitamin D or magnesium and must avoid high phosphorous foods. If taken it will cause anorexia.

1.4 System Limitation

The main idea is to have a system that can generate any result if the procedure is been followed properly. But some major illness still cannot be prevented or cure using proper food and supplement because some of these illnesses are cause by bacteria infection and virus attack e.g. dengue, meningitis, encephalitis and etc. This kind of diseases is mainly non-related to nutritional and diet.

This system also is not build for medical diagnosis. The system can not figure out what kind of illness does the patient suffering by analyzing the symptoms. It is just for the user to enter the name of illness and the system will generate a list of recommended nutrition and supplement list.

1.5 Report Layout

The purpose of the report is to document essential information gathered during this system development and implementation stage. This report is divided into 4 respective chapters.

Chapter 1: Introduction

This chapter introduces the project in a general view and the rationale of the project. It made an overview on the, definition, project objectives, project scopes, project limitation and the project schedule.

Chapter 2: Literature Review

For this chapter, it discusses the researches that are carried out during the analysis and design phase at the project. This includes techniques and ideas used in the building of the system.

Chapter 3: Methodology

For this chapter is mostly about the information and techniques that will be used in the process of making this system. Types of software and prototyping will also be discussed in details here.

Chapter 4: System Design

This chapter describes the design considerations including processing design, the user interface design and also the database design of this project.

1.6 PROJECT SCHEDULE

	1	2	3	4	5	6	7	8	9	10++
PHASE I										
Getting briefing from lecturer and propose the topic.										
Identifying objective of the proposed topic.										
Estimate scope of the system										
Searching information from various sources about case-based reasoning from the Internet and books.										
Analyzing and research the information.										
Determine system module.										
Design the recommended system.										
Presentation of the proposal.										
Documentation										
PHASE II										
Develop system.										
Documentation.										
Testing and maintaining the designed system.										
Implementation of the system.										
Presentation.										

Prepared by,

Chew Wei Liang

CHAPTER 2

LITERATURE REVIEW

Most of my researches were done during this period of time. My major source of information comes from the internet and the Faculty of Medicine's Library. Other sources like books and journals also have been used in my research. Discussion with my supervisor and my expert domain also helped me to gain more knowledge in this area. This part of the project is the most demanding and must be done with caution –not to cause unwanted errors and failure in the coming phase of the project. To reduce this unwanted errors and failure, I take in consideration of what the user of the system wants and the normal practice of a nutritionist in giving advice as one of my criteria in designing the system.

2.1 Introduction to Case Based Reasoning(CBR)

Over the last few years, case-based reasoning (CBR) has grown from a rather specific and isolated research area to a field of widespread interest. Activities are rapidly growing - as seen by the increased rate of research papers, availability commercial products, and also reports on applications in regular use.

Case-based reasoning is a problem solving paradigm that in many respects is fundamentally different from other major AI approaches. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the *specific* knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in the new problem situation. A second important difference is that CBR also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems.[2]

So what is case-based reasoning anyway? Basically: To solve a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation. Let me illustrate this by looking at some typical problem solving situations:

- A physician - after having examined a particular patient in his office - gets a reminding to a patient that he treated two weeks ago. Assuming that the reminding was caused by a similarity of important symptoms (and not the patient's hair-color, say), the physician uses the diagnosis and treatment of the previous patient to determine the disease and treatment for the patient in front of him.
- A drilling engineer, who have experienced two dramatic blow out situations, is quickly reminded of one of these situations (or both) when the combination of critical measurements matches those of a blow out case. In particular, he may get a reminding to a mistake he made during a previous blow-out, and use this to avoid repeating the error once again.
- A financial consultant working on a difficult credit decision task, uses a reminding to a previous case, which involved a company in similar trouble as the current one, to recommend that the loan application should be refused.

According to David B. Leake in [4], the CBR approach is based on two tenets about the nature of the world. The first tenet is that the world is regular; similar prior problems are useful starting point for a new problem solving. The second tenet is the types of problems an agent encounters tend to recur. Consequently, future problems are likely to be similar to current problem. When the two tenets hold, it is worthwhile to remember and reuse current reasoning: case based reasoning is effective reasoning strategy.

Since the appearance of CBR, it has been applied in a wide range of domains such as [4]:

- 1) Diagnosis
- 2) Help Desk
- 3) Assessment
- 4) Decision Support
- 5) Design

2.1.1 Case-based problem solving.

As the above examples indicate, reasoning by reusing past cases is a powerful and frequently applied way to solve problems for humans. This claim is also supported by results from cognitive psychological research. Part of the foundation for the case-based approach, is its psychological plausibility. Several studies have given empirical evidence for the dominating role of specific, previously experienced situations (what we call cases) in human problem solving. Schank [2] developed a theory of learning and reminding based on retaining of experience in a dynamic, evolving memory structure. Case-based reasoning and analogy are sometimes used as synonyms. Case-based reasoning can be considered a form of intra-domain analogy. In CBR terminology, a *case* usually denotes a problem situation. A previously experienced situation, which has been captured and learned in such way that it can be reused in the solving of future problems, is referred to as a past case, previous case, stored case, or retained case. Correspondingly, a new case or unsolved case is the description of a new problem to be solved. Case-based reasoning is - in effect - a cyclic and integrated process of solving a problem, learning from this experience, solving a new problem, etc. Note that the term problem solving is used here in a wide sense, coherent with common practice within the area of knowledge-based systems in general. This means that problem solving is not necessarily the finding of a concrete solution to an application problem, it may be any problem put forth by the user. For example, to justify or criticize a solution proposed by the user, to interpret a problem situation, to generate a set of possible solutions, or generate expectations in observable data are also problem solving situations.

2.1.2 Learning in Case-based Reasoning.

A very important feature of case-based reasoning is its coupling to learning. The driving force behind case-based methods has to a large extent come from the machine learning community, and case-based reasoning is also regarded as a subfield of machine learning. Thus, the notion of case-based reasoning does not only denote a particular reasoning method, irrespective of how the cases are acquired, it also denotes a machine learning paradigm that enables sustained learning by updating the case base after a problem has been solved. Learning in CBR occurs as a natural by-product of problem solving. When a

problem is successfully solved, the experience is retained in order to solve similar problems in the future. When an attempt to solve a problem fails, the reason for the failure is identified and remembered in order to avoid the same mistake in the future.

Case-based reasoning favors learning from experience, since it is usually easier to learn by retaining a concrete problem solving experience than to generalize from it. Still, effective learning in CBR requires a well worked out set of methods in order to extract relevant knowledge from the experience, integrate a case into an existing knowledge structure, and index the case for later matching with similar cases.

2.1.3 Combining cases with other knowledge.

By examining theoretical and experimental results from cognitive psychology, it seems clear that human problem solving and learning in general are processes that involve the representation and utilization of several types of knowledge, and the combination of several reasoning methods. If cognitive plausibility is a guiding principle, architecture for intelligence where the reuse of cases is at the centre, should also incorporate other and more general types of knowledge in one form or another.

2.2 Case-Based Reasoning Techniques

Central tasks that all case-based reasoning methods have to deal with are to identify the current problem situation, find a past case similar to the new one, and use that case to suggest a solution to the current problem, evaluate the proposed solution, and update the system by learning from this experience. How this is done, what part of the process is focused, what type of problems drives the methods, etc. varies considerably, however. Below is an attempt to classify CBR methods into types with roughly similar properties in this respect.

2.2.1 Main types of CBR methods

The CBR paradigm covers a range of different methods for organizing, retrieving, utilizing and indexing the knowledge retained in past cases. Cases may be kept as concrete experiences, or a set of similar cases may form a generalized case. Cases may be stored as separate knowledge units or split up into subunits and distributed within the knowledge structure. Cases may be indexed by a prefixed or open vocabulary, and within a flat or hierarchical index structure. The solution from a previous case may be directly applied to the present problem, or modified according to differences between the two cases. The matching of cases, adaptation of solutions, and learning from an experience may be guided and supported by a deep model of general domain knowledge, by more shallow and compiled knowledge, or be based on an apparent, syntactic similarity only. CBR methods may be purely self-contained and automatic, or they may interact heavily with the user for support and guidance of its choices. Some CBR method assume a rather large amount of widely distributed cases in its case base, while others are based on a more limited set of typical ones. Past cases may be retrieved and evaluated sequentially or in parallel.

Actually, "case-based reasoning" is just one of a set of terms used to refer to systems of this kind. This has lead to some confusions, particularly since case-based reasoning is a term used both as a generic term for several types of more specific approaches, as well as for one such approach. To some extent, this can also be said for analogy reasoning. An attempt of a clarification, although not resolving the confusions, of the terms related to case-based reasoning are given below.

* *Exemplar-based reasoning.*

The term is derived from a classification of different views to concept definition into "the classical view", "the probabilistic view", and "the exemplar view". In the exemplar view, a concept is defined *extensionally*, as the set of its exemplars. CBR methods that address the learning of concept definitions (i.e. the problem addressed by most of the research in machine learning), are sometimes referred to as exemplar-based. Examples are early papers by Kibler and Aha, and Bareiss and Porter. In this approach, solving a problem is

a *classification task*, i.e. finding the right class for the unclassified exemplar. The class of the most similar past case becomes the solution to the classification problem. The set of classes constitutes the set of possible solutions. Modification of a solution found is therefore outside the scope of this method.

* *Instance-based reasoning.*

This is a specialization of exemplar-based reasoning into a highly *syntactic* CBR-approach. To compensate for lack of guidance from general background knowledge, a relatively large number of instances are needed in order to close in on a concept definition. The representation of the instances are usually simple (e.g. feature vectors), since a major focus is to study *automated learning* with no user in the loop. Instance-based reasoning labels recent work by Kibler and Aha and colleagues, and serves to distinguish their methods from more knowledge-intensive exemplar-based approaches (e.g. Protos' methods). Basically, this is a non-generalization approach to the concept learning problem addressed by classical, inductive machine learning methods.

* *Memory-based reasoning.*

This approach emphasizes a collection of cases as a *large memory*, and reasoning as a process of accessing and searching in this memory. Memory organization and access is a focus of the case-based methods. The utilization of *parallel processing* techniques is a characteristic of these methods, and distinguishes this approach from the others. The access and storage methods may rely on purely syntactic criteria, as in the MBR-Talk system, or they may attempt to utilize general domain knowledge, as in PARADYME and the work done in Japan on massive parallel memories.

* *Case-based reasoning.*

Although case-based reasoning is used as a generic term in this paper, the *typical* case-based reasoning methods have some characteristics that distinguish them from the other approaches listed here. First, a typical case is usually assumed to have a certain degree of richness of information contained in it, and a certain *complexity* with respect to its

internal organization. That is, a feature vector holding some values and a corresponding class is not what we would call a typical case description. What we refer to as typical case-based methods also has another characteristic property: They are able to *modify*, or adapt, a retrieved solution when applied in a different problem solving context. Paradigmatic case-based methods also utilizes *general background knowledge* - although its richness, degree of explicit representation, and role within the CBR processes varies. Core methods of typical CBR systems borrow a lot from cognitive psychology theories.

* *Analogy-based reasoning.*

This term is sometimes used, as a synonym to case-based reasoning, to describe the typical case-based approach just described. However, it is also often used to characterize methods that solve new problems based on past cases from a *different domain*, while typical case-based methods focus on indexing and matching strategies for single-domain cases. Research on analogy reasoning is therefore a subfield concerned with mechanisms for identification and utilization of cross-domain analogies. The major focus of study has been on the *reuse* of a past case, what is called the mapping problem: Finding a way to transfer, or map, the solution of an identified analogue (called source or base) to the present problem (called target).

Throughout the paper we will continue to use the term case-based reasoning in the generic sense, although our examples, elaborations, and discussions will lean towards CBR in the more typical sense. The fact that a system is described as an example of some other approach, does not exclude it from being a typical CBR system as well. To the degree that more special examples of, e.g. instance-based, memory-based or analogy-based methods will be discussed, this will be stated explicitly.

2.2.2 The CBR Cycle

The processes involved in CBR can be represented by a schematic cycle [2]. CBR typically described as a cyclical process comprising *the four REs*:

- RETRIEVE the most similar case(s);
- REUSE the case(s) to attempt to solve the problem;
- REVISE the proposed solution if necessary, and
- RETAIN the new solution as a part of a new case.

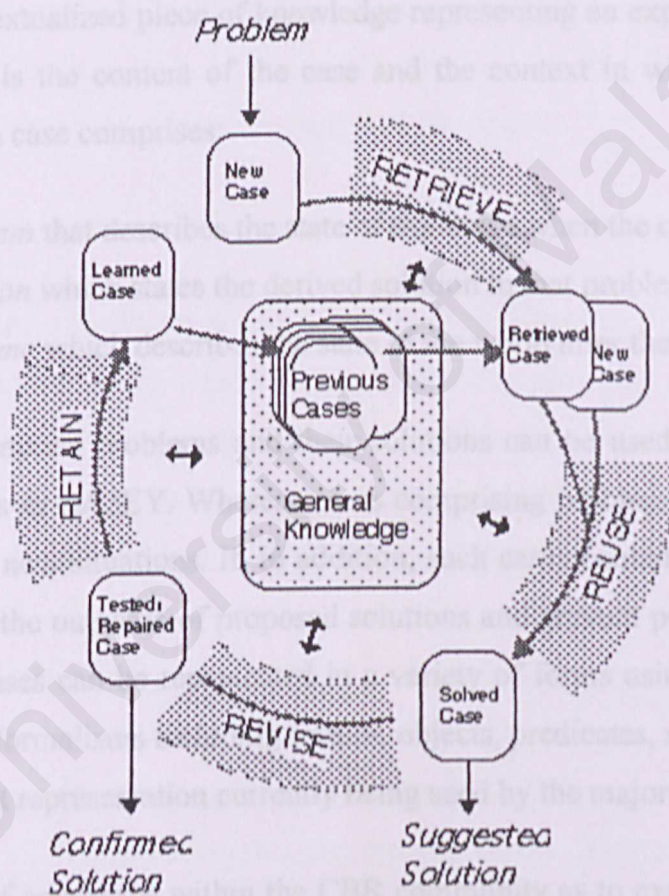


Figure 2.1: CBR Cycle

A new problem is matched against cases in the case base and one or more similar cases are *retrieved*. A solution suggested by the matching cases is then *reused* and tested for success. Unless the retrieved case is a close match the solution will probably have to be *revised* producing a new case that can be *retained*.

This cycle currently rarely occurs without human intervention. For example many CBR tools act primarily as case retrieval and reuse systems. Case revision (i.e., adaptation) often being undertaken by managers of the case base. However, it should not be viewed as weakness of CBR that it encourages human collaboration in decision support. The following sections will outline how each process in the cycle can be handled.

Case Representation

A case is a contextualized piece of knowledge representing an experience. It contains the past lesson that is the content of the case and the context in which the lesson can be used. Typically a case comprises:

- the *problem* that describes the state of the world when the case occurred,
- the *solution* which states the derived solution to that problem, and/or
- the *outcome* which describes the state of the world after the case occurred.

Cases which comprise problems and their solutions can be used to derive solutions to new problems, as in CASEY. Whereas cases comprising problems and outcomes can be used to evaluate new situations. If, in addition, such cases contain solutions they can be used to evaluate the outcome of proposed solutions and prevent potential problems as in MEDIATOR. Cases can be represented in a variety of forms using the full range of AI representational formalisms including frames, objects, predicates, semantic nets and rules - the frame/object representation currently being used by the majority of CBR software.

There is a lack of consensus within the CBR community as to exactly what information should be in a case. However, two pragmatic measures can be taken into account in deciding what should be represented in cases: the functionality and the ease of acquisition of the information represented in the case.

Indexing

Case indexing involves assigning indices to cases to facilitate their retrieval. Several guidelines on indexing have been proposed by CBR researchers. Indices should:

- be predictive,
- address the purposes the case will be used for,
- be abstract enough to allow for widening the future use of the case-base, and
- be concrete enough to be recognized in future

Both manual and automated methods have been used to select indices. Choosing indices manually involves deciding a case's purpose with respect to the aims of the reasoner and deciding under what circumstances the case will be useful.

There are an ever increasing number of automated indexing methods including:

- Indexing cases by features and dimensions that tend to be predictive across the entire domain i.e., descriptors of the case which are responsible for solving it or which influence its outcome. In this method the domain is analyzed and the dimensions that tend to be important are computed. These are put in a checklist and all cases are indexed by their values along these dimensions. For example, MEDIATOR uses this method by indexing on type and function of disputed objects and relationship of disputants, whilst CHEF indexes on texture and taste. This kind of technique is called *checklist-based* indexing.
- Difference-based indexing selects indices that differentiate a case from other cases as in CYRUS. During this process the system discovers which features of a case differentiate it from other similar cases, choosing as indices those features that differentiate cases best.
- Similarity and explanation-based generalization methods, which produce an appropriate set of indices for abstract cases created from cases that share some common set of features, whilst the unshared features are used as indices to the original cases.

- Inductive learning methods, which identify predictive features that are then used as indices. These techniques are widely used (e.g., in Cognitive system's ReMind) and commonly use variants of the ID3 algorithm used for rule induction.
- Explanation-based techniques, which determine relevant features for each case. This method analyses each case to find which of their features predictive ones are. Cases are then indexed by those features.
- However, despite the success of many automated methods, Janet Kolodner believes that people tend to do better at choosing indices than algorithms, and therefore for practical applications indices should be chosen by hand.

Storage

Case storage is an important aspect in designing efficient CBR systems in that, it should reflect the conceptual view of what is represented in the case and take into account the indices that characterize the case. The case-base should be organized into a manageable structure that supports efficient search and retrieval methods. A balance has to be found between storing methods that preserve the semantic richness of cases and their indices and methods that simplify the access and retrieval of relevant cases. These methods are usually referred to as *case memory models*. The two most influential case memory models are the *dynamic memory model* of Schank and Kolodner, and the *category-exemplar model* of Porter and Bareiss.

The dynamic memory model

The case memory model in this method is comprised of *memory organization packets* or MOPs. MOPs are a form of frame and are the basic unit in dynamic memory. They can be used to represent knowledge about classes of events using two kinds of MOPs:

- *instances* representing cases, events or objects, and
- *abstractions* representing generalized versions of instances or of other abstractions.

The case memory, in a dynamic memory model, is a hierarchical structure of *episodic memory organization packets* (E-MOPs), also referred to as *generalized episode* (GEs) developed from Shank's more general MOP theory

The basic idea is to organize specific cases which share similar properties under a more general structure (i.e., a generalized episode). A GE contains three different types of objects: *norms*, *cases* and *indices*. Norms are features common to all cases indexed under a GE. Indices are features which discriminate between a GE's cases. An index may point to a more specific generalized episode or to a case, and is composed of an index name and an index value.

The case-memory is a discrimination network where nodes are either a GE, an index name, index value or a case. Index name-value pairs point from a GE to another GE or case. The primary role of a GE is as an indexing structure for storing, matching and retrieval of cases. During case storage when a feature (i.e., index name and index value) of a new case matches a feature of an existing case a new GE is created. The two cases are then discriminated by indexing them under different indices below the new GE (assuming the cases are not identical). Thus, the memory is *dynamic* in that similar parts of two cases are dynamically generalized into a new GE, the cases being indexed under the GE by their differences.

However, this process can lead to an explosive growth in the number of indices as case numbers increase. So for practical purposes most CBR systems using this method limit the number of permissible indices to a limited vocabulary.

The category-exemplar model

This model organizes cases based on the view that the real world should be defined extensionally with cases being referred to as *exemplars*. The case memory is a network structure of categories, semantic relations, cases and index pointers. Each case is associated with a category. Different case features are assigned different importance in describing a case's membership to a category. Three types of indices are provided, which may point to a case or a category:

- *feature links* that point from problem descriptors (features) to a case or category,
- *case links* that point from categories to its associated cases, and
- *difference links* pointing from categories to the neighboring cases that only differ in a small number of features.

A feature is described by a name-value pair. A category's exemplars are stored according to their degree of prototypically to the category. Within this memory organization, the categories are inter-linked within a semantic network containing the features and intermediate states referred to by other terms. This network represents a background of general domain knowledge that enables explanatory support to some CBR tasks. A new case is stored by searching for a matching case and by establishing the relevant feature indices. If a case is found with only minor differences to the new case, the new case may not be retained, or the two cases may be merged.

Retrieval

Given a description of a problem, a retrieval algorithm, using the indices in the case-memory, should retrieve the most similar cases to the current problem or situation. The retrieval algorithm relies on the indices and the organization of the memory to direct the search to potentially useful cases.

The issue of choosing the *best* matching case has been addressed by research into analogy. This approach involves using heuristics to constrain and direct the search. Several algorithms have been implemented to retrieve appropriate cases, for example: serial search, hierarchical search and simulated parallel search.

Case-based reasoning will be ready for large scale problems only when retrieval algorithms are efficient at handling thousands of cases. Unlike database searches that target a specific value in a record, retrieval of cases from the case-base must be equipped with heuristics that perform partial matches, since in general there is no existing case that exactly matches the new case.

Among well known methods for case retrieval are: nearest neighbor, induction, knowledge guided induction and template retrieval. These methods can be used alone or combined into hybrid retrieval strategies.

Nearest neighbor

This approach involves the assessment of similarity between stored cases and the new input case, based on matching a weighted sum of features. The biggest problem here is to determine the weights of the features. The limitation of this approach includes problems in converging on the correct solution and retrieval times. In general the use of this method leads to the retrieval time increasing linearly with the number of cases. Therefore this approach is more effective when the case base is relatively small. Several CBR implementations have used this method to retrieve matching cases, for example: BROADWAY for selection of car models, the Compaq SMART System for a customer product support help desk, and ANON for situation assessment in plan failure.

A typical algorithm for calculating nearest neighbor matching is the one used by Cognitive Systems ReMind software reported in Kolodner where w is the importance weighting of a feature (or slot), sim is the similarity function, and fI and fR are the values for feature i in the input and retrieved cases respectively.

Induction

Induction algorithms (e.g. ID3) determine which features do the best job in discriminating cases, and generate a decision tree type structure to organize the cases in memory. This approach is useful when a single case feature is required as a solution, and where that case feature is dependent upon others.

Knowledge guided induction

This method applies knowledge to the induction process by manually identifying case features that are known or thought to affect the primary case feature. This approach is frequently used in conjunction with other techniques, because the explanatory knowledge is not always readily available for large case bases.

Template retrieval

Similar to SQL-like queries, template retrieval returns all cases that fit within certain parameters. This technique is often used before other techniques, such as nearest neighbor, to limit the search space to a relevant section of the case-base.

Adaptation

Once a matching case is retrieved a CBR system should adapt the solution stored in the retrieved case to the needs of the current case. Adaptation looks for prominent differences between the retrieved case and the current case and then applies formulae or rules that take those differences into account when suggesting a solution. In general, there are two kinds of adaptation in CBR:

- *Structural adaptation*, in which adaptation rules are applied directly to the solution stored in cases. This kind of adaptation is used in JUDGE and CHEF.
- *Derivational adaptation* that reuses the algorithms, methods or rules that generated the original solution to produce a new solution to the current problem. In this method the planning sequence that constructed that original solution must be stored in memory along with the solution as in MEDIATOR. Derivational

adaptation, sometimes referred to a *reinstantiation*, can only be used for cases that are well understood.

An ideal set of adaptation rules must be strong enough to generate complete solutions from scratch, and an efficient CBR system may need both structural adaptation rules to adapt poorly understood solutions and derivational mechanisms to adapt solutions of cases that are well understood.

Several techniques, ranging from simple to complex, have been used in CBR for adaptation. These include:

- *Null adaptation*, a direct simple technique that applies whatever solution is retrieved to the current problem without adapting it. Null adaptation is useful for problems involving complex reasoning but with a simple solution. For example, when someone applies for a bank loan, after answering numerous questions the final answer is very simple: grant the loan, reject the loan, or refer the application.
- *Parameter adjustment*, a structural adaptation technique that compares specified parameters of the retrieved and current case to modify the solution in an appropriate direction. This technique is used in JUDGE, which recommends a shorter sentence for a criminal where the crime was less violent.
- *Abstraction and respecialisation*, a general structural adaptation technique that is used in a basic way to achieve simple adaptations and in a complex way to generate novel, creative solutions. The PLEXUS planning system uses this technique.
- *Critic-based adaptation*, in which a critic looks for combinations of features that can cause a problem in a solution. Importantly, the critic is aware of repairs for these problems. PERSUADER is a system which uses all the techniques of adaptation discussed above.
- *Reinstantiation* is used to instantiate features of an old solution with new features. For example, CHEF can reinstantiate *chicken* and *snow peas* in a Chinese recipe with *beef* and *broccoli* thereby creating a new recipe.

- *Derivational replay* is the process of using the method of deriving an old solution or solution piece to derive a solution in the new situation. For example, BOGART, which replays stored design, plans to solve problems.
- *Model-guided repair*, uses a causal model to guide adaptation as in CELIA, which is used for diagnosis and learning in auto mechanics, and KRITIK used in the design of physical devices.
- *Case-based substitution*, uses cases to suggest solution adaptation as in ACBARR a system for robot navigation

2.3 Problems in CBR

- 1) How cases should be represented?
- 2) How indices should be chosen for organizing memory efficiency?
- 3) How to structure relationship between cases and of different cases?
- 4) How to handle large case-bases?
- 5) How to develop general adaptation heuristic of analyzing past cases or the solutions to fit the new cases.
- 6) How many criteria should be use to match a case with a past cases and can the system handle variation of difference numbers of criteria.

2.4 CBR and Other Reasoning Methods

In the traditional perspective of reasoning, both in Artificial Intelligent and cognitive psychology, is largely a process of remembering abstract operators and composing them with each other. However, CBR views reasoning in a much different way. It views reasoning as a process of remembering one or small set of concrete instances or cases and generate new solution based on the comparison of past cases with the current case. The CBR views imply 2 things:-

- 1) CBR emphasizes in the use of concrete instances over abstract operators. It regards large chunks of composed knowledge as the starting point for reasoning. The reason is that concrete instances provide operator knowledge that guide

problem solving than abstract operator. Smaller and more abstract chunks of knowledge in memory are however secondary to CBR.

- 2) CBR emphasizes on the manipulation of cases over composition, decomposition and recomposition process

	MBR	CBR
Domain Applicability	Applicable when a causal model exists. That is when a domain is well understood.	Applicable under the same conditions as MBR but also domain that is not well understood. The set of cases plays the role of the generalized model when a domain is not well understood.
Task Applicability	Provide means to verify solutions but solutions generated is unguided	Provide effective solution generation and evolution that is based on the best case available.

These methodologies are rather complementary because MBR validations and evaluation of solution is not provided in CBR as a full.

2.4.1 Advantages of CBR

Every Artificial Intelligence has its advantages and disadvantages. These approaches usually provide acceptable solutions but not the optimal solutions. Domains that require

optimal answers are probably not appropriate for heuristic methods at all- case based or other.[5]

- 1) CBR allows reasoners to provide solution to problem quickly, avoiding the time needed to verify these problems from scratch.
- 2) CBR allows reasoners to propose solution that are not completely understood by the reasoner.
- 3) CBR provides the means of evaluation solutions when no algorithmic method is available for a evaluation
- 4) CBR is useful in warning potential problem that have occurred in the past
- 5) Cases in CBR help reasoners to focus on important parts of problems by pointing out what function of the problem are important.

2.4.2 Disadvantages of CBR

- 1) CBR might be tempted to use old cases blindly relying previous experience without validating it in the new problem situation.
- 2) CBR might allow case bias to the expert domain in problem solving.
- 3) Often people, especially novices are not reminded of the most appropriate sets of cases when they are reasoning.
- 4) CBR does not fully explore its solution space, resulting some optimal solutions might not be found.

CHAPTER 3

METHODOLOGY & SYSTEM ANALYSIS

The ideas and technologies of developing a nutritional system had been discussed in chapter 2. In this chapter, analysis on Case Based Reasoning will be done and how CBR can be implemented in Nutritional Advisory System.

3.1 Methodology

The software process that will be used to develop the system is waterfall model with prototyping. In term of a case base reasoning approach, there are not clear methodologies in implementing it. So in this application development, waterfall model with prototyping is chosen. Below is the figure showing the flow of waterfall model with prototyping [6].

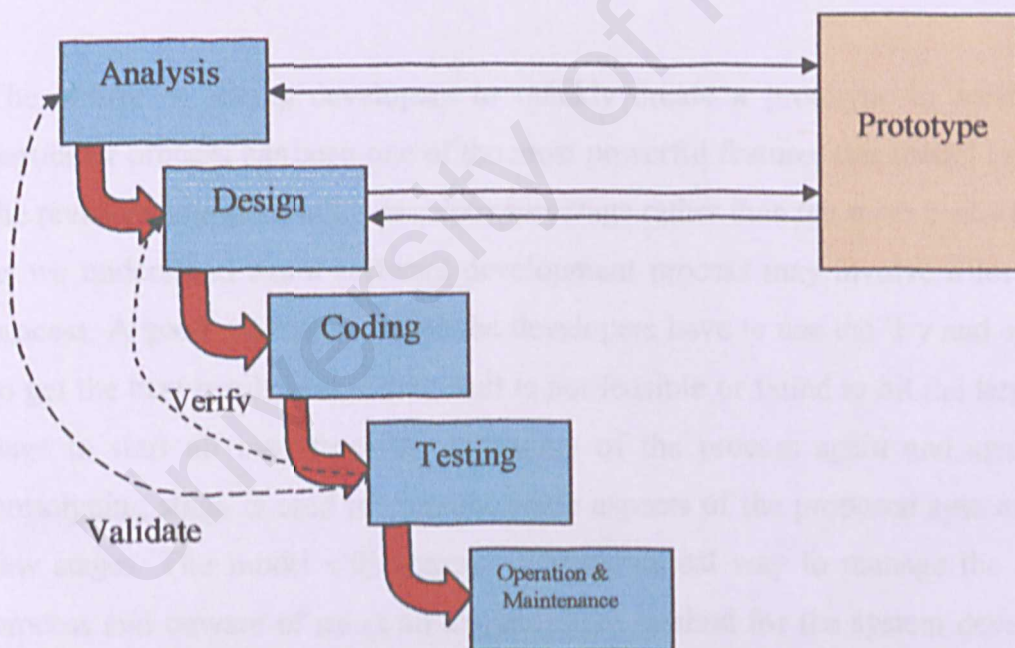


Figure 3.1: The Project Process Model

Conventional development and prototyping may be combined. The Waterfall Model with prototyping is chosen because the strengths of each can be achieved on a single project. This model is originated from the Waterfall model but some modification had been implemented to improve the flow of the software development process. This model

also sometimes combined with iteration so that any improvement and verification that wanted to change can move back one phase up at a time to make adjustment. Sometimes this type of model is called Waterfall Model with Iteration and Prototyping. Prototyping is sub process; prototype is a partially developed product that enables customers and developers to examine some aspect of the proposed system and decide if it suitable or appropriate for the finished product. Often the user interface is built and tested as a prototype, so the user understands what the new system will be like, and the designers get a better sense of how the users like to interact with the system. Thus, major kinks in the requirements are addressed and fixed well before the requirements are officially validated during the system testing; validating ensures that the systems have implemented all of the requirements in the specification. System testing also verifies the requirement; verification ensures that each function work correctly.[7]

3.1.1 Prototyping solves the problems of traditional Waterfall Model

The ability of letting developers to quickly create a prototype to verify the needs particular process has been one of the most powerful features this model has. Therefore, the revisions are made at the requirements stage rather than the more costly testing stage, as we understand that a software development process may involve a lot of iterations process. A good example is when the developers have to use the 'try and error' method to get the best result; and if the result is not feasible or failed to hit the target, they will have to start all over from the beginning of the process again and again. Thus the prototyping stage is used to examine some aspects of the proposed system for the first few stages. The model will organize a systematical way to manage the development process and beware of using an inappropriate method for the system development. By validation of these stages, it will effectively reduce the possibility of repeating any process caused by the using of any inappropriate methods. Therefore, when the software development process come to the system testing stage, it will automatically validate the requirements of the system and also verify the system design as planning in the earlier stages.

The waterfall model with prototyping approach that will be adopted in the proposed project encompasses the activities at system analysis, system design, coding, testing and implementation. Each of the stage is discussed below.

	Prototyping Model	Waterfall Model
Advantages	<ul style="list-style-type: none"> ▪ Reduce risk and uncertainty in development 	<ul style="list-style-type: none"> ▪ It is easier for the explanation to a customer who is not familiar with software development. This model will increase the confidence of a software developer during the developing process. ▪ Most of the latest models are built or modified according to this model.
Disadvantages	<ul style="list-style-type: none"> ▪ Produce the product in limited time will ignore the quality for the software. Thus, more time is needed to maintain the system. ▪ Developers may develop a system within unsuitable platforms or programs. 	<ul style="list-style-type: none"> ▪ Not showing how the basic coding is designed or created. ▪ Do not have any reference to refer to when any disaster or changes is happening to the product or activities ▪ Failed to face the software as the problems solving process, which we notice that the waterfall model is actually modified from hardware development process.

The waterfall model with prototyping approach that will be adapted in the proposed project encompasses the activities at system analysis, system design, coding, testing and implementation. Each of the stage is discussed below.

Analysis

The goal to the system analysis is to understand the proposed system and to establish system requirements. The system analysis phase is concerned with data gathering and data analysis. Data will be gathered from the sources like written materials, internet, as well as observation and examination of others existing systems available. The iterative process of prototyping-revision will be Data Flow Diagram (DFD) is chosen to analyze the collected data because it enables the information domain and functional domain to be modeled at the same time. It is to be used to graphically show the flow of the data through the system. The most important outcome from this phase will be an accurate system requirement specification.

Design

The system design phase is the phase in which requirements produced in the previous phase are translated into a representation of the system. This phase will be concerned with user interface design, database design and system design. The interactive process of prototyping-revision will be used to revise the design of the user interface. Entity-relational (E-R) modeling will be involved in the logical design of the Ms. Access database. In system design, structure chart will be involved in structuring the system's modules and flow chart might be used to depict the design of procedural details.

Coding

This stage translates and implements the details design representation of the system into programming realization. System coding will be done in this phase to develop working modules and classes. Java is used to develop the whole +system; Microsoft Access will be used to develop the database system.

Testing

Testing will be critical step in assuring the quality of the developed system and will represent the ultimate review of specification, design, and coding. First, unit testing will be performed to verify each program module. Next, integration testing is performed. It is to integrate unit-tested program modules and conduct tests that uncover errors associated with the interfacing of those modules. Validating test succeeds when the system functions in the manner that is reasonably expected.

Operation and Maintenance

Maintenance process should be an ongoing activity in real development. Monitoring a necessary adjustment continue so that the system produces the expected results. However, system enhancements and maintenance will be carried out in the proposed project if time constraint allowed.

3.2 Cased Based versus Rule Based Reasoning

3.2.1 Advantages of Rule Based Reasoning

- The ability to use directly experimental knowledge acquired from human experts. It is very easy to transform some knowledge into a set of rules
- The modularity of rules cases construction and maintenance. In rule-based reasoning, the link between the rules is weak. It is easy to change, add or delete a rule.
- Good performances in a limited domain. If the domain is limited and well defined, it is easy to cover all the possibilities and foresee all the possible situations. So, that is why a rule-based system can be very accurate and fast in a limited domain.
- Good explanation facilities. As it is possible in a rule-based system to obtain the information that has led to a conclusion
- Rules chaining are easy to trace and debug

3.2.2 Disadvantages of Rule Based Reasoning

- Often, rules from human expert are heuristic and lack a deeper knowledge of the domain. If the knowledge of the domain is not extremely precise there is always a chance that an unexpected case occurs and breaks the rule-based system.
- Heuristic rules cannot handle missing information or unexpected data values. If a value is missing or a value is not expected, the rule-based system is not able to fire its rules. So, it can't give any result.
- Rules are not able to adapt when there is a new problem
- The knowledge is very task dependent. Often a set of rule is adapted to a particular problem and it is not possible to adapt it to another problem.

3.2.3 Advantages of case-based reasoning

- Extensive analysis of domain knowledge is not required. In case-based reasoning, the most important thing is to know how to compare two cases. This task is not directly dependant of the domain
- It allows shortcuts in reasoning, If an appropriate case is found, a solution can be found very fast(faster than it would be to generate a solution from scratch)
- It enables systems to avoid past errors and exploit past success. In case-based reasoning, the system keeps a record of each situation that occurred and uses it for each new problem. That is why it 'learns'.

3.2.4 Disadvantages of Case-based reasoning

- There are no explanatory facilities. In case-based reasoning, there is no way to explain why a solution is taken because it is just because it is just based on its similarity with other cases. A solution is not a logical decision as in a rule-based system.

- A large case can suffer problems of efficiency due to the storing and computing of the past cases. In case-based reasoning, a lot of cases need to be retrieved in order to compare them. That can be major drawback if there is a large number of cases in the database
- It is difficult to get good criteria for indexing and matching cases. The vocabulary for the retrieving and similarity matching algorithm must be carefully hand crafted. This can offset many of the advantages case-based reasoning offers for knowledge acquisition.

3.2.5 Conclusion

Human experts are not systems of rules, they are libraries of experiences

Riesbeck and Schank (1989)

Ruled-based systems are often brittle because they can't learn from experience and they collapse dramatically when faced with a problem beyond their plateau of expertise. On the other hand, case-based reasoning can deal with such situations by acquiring new cases. The Nutritional Advisory System needs to adapt itself to the user's behavior but, because we don't know his behavior in advance, it is not possible to create a set of rules adapted each case. A case-based reasoning system does not need to know as much about the domain as does a rule-based reasoning one.

Rule-based reasoning has some explanatory facilities that case-based reasoning does not have. Nevertheless in the Nutritional Advisory System context, this is counter react by having another method of explanatory through past case. The main characteristic it should have is its adaptation to the user. And this can only be provided by a case-based reasoning.

It seems that the advantages of rule-based reasoning are not very useful for an advisory system like Nutritional Advisory System. What is more, its drawbacks are too important to be overcome in this situation. Finally, the best solution for the Nutritional Advisory System is a case-based reasoning.

3.3 Analysis of the implementation of CBR in Nutritional Advisory System

Case Based Reasoning is based on two tenets about the nature of the world; similar problems have similar solutions and same problems tend to occur. The problems are then stored so that it can be used in future of the same situation or similar situation occur. CBR tasks are divided into two main categories that are interpretive CBR and problem solving CBR. And for Nutritional Advisory System, it is a problem solving CBR. This is due to the nature of the system as a solution provider. Problem solving CBR involves situation assessment, case retrieval, and similarity assessment/evaluation.

To get to the solution for a problem involved two spaces, the problem description space and the solution space. The problem descriptions space contains the entire problem's description. As for Nutritional Advisory System, the user's personal data, sickness symptoms and type of sickness are in the problem descriptions space. The solution space for Nutritional Advisory System will be type of meals, supplement, do and don't in daily meals and medicine precaution. When a new case is inputted, the system will compare the case with the previous case that had been solved. For the matched features, the solution will be retrieved from the storage. The new solution will be generated and will be stored in the storage.

Learning is important in Case-Based Reasoning. There are two type of learning process in Case Based Reasoning: success-driven learning and failure-driven learning. The Nutritional Advisory System will be using the both system as a hybrid. All the successful case and solution will be retained in the system database for future use.

Each approach implemented must have its advantages and disadvantages on the system. Here are some of the advantages and disadvantages found during the analysis phase.

Advantages of CBR in Nutritional Advisory System

Most of the time a diagnostic system will search through the whole database using breadth first search or depth first search to enable the system to get to the solution. This method is very time consuming and the question and answer session will be very long

winded and sometimes irrelevant. But for Case Based Reasoning, derivation of solution does not really start from scratch and this will actually save some computation time and memory compare with the expert system ways.

1. Even though the Nutritional Advisory System cannot completely elicit information from user, it can still generate a solution to the problem using past cases as reference
2. Cases is used to focus its reasoning on important parts of the problem by pointing out what features of a problem are the important ones. In this case, only the symptoms, type of sickness and personal particular.

Disadvantages of CBR in Nutritional Advisory System

1. Case-Based reasoning in Nutritional Advisory System might be tempted to use old case blindly, relying on previous experience and generate unsuitable solution for a problem.
2. Case-Based reasoning might allow cases to bias a specific person too much in solving a new problem.
3. Some optimal solution might not be found as Case-Based Reasoning in nutritional advisory does not fully explore its solution space.

The CBR can be dividing in to five types: exemplar-based reasoning, instance-based reasoning, memory-based reasoning, case-based reasoning and analogy-based reasoning. All CBR methods include identify the current problem situation and find a past case similar to the new one for solution. But still each method has different approach in solving a problem.

The case-based reasoning method among the five methods will be use in the development of the Nutritional Advisory System. A typical case in this method is usually assumed to have a certain degree of richness of information contained in it, and a certain complexity with the respect to its internal organization. This method is able to modify, adapt, a retrieved solution when applied in a different problem solving context. In the Nutritional advisory, a typical case will contain the basic information for a disease and personal particular; these two components are the basic component for a case. Beside that symptoms and medicine prescribed by medical practitioner will also be sub-

features in the case. If a match is found for a new problem, the solution will be applied to the new problem, but most of the time it will not be identical then modification will be done to generate the nearest solution to the problem.

3.3.1 Case Representation and Indexing in Nutritional Advisory System

A case is usually composed of three parts; problem/situation description, solution and outcome. A well design case will definitely improve the quality and the outcome of the system.

3.3.1.1 The Contents of Problem Representation

There are three major components of a problem representation:

1. Goals to be achieved in solving a problem
2. Constraints on the goals
3. Features of the problem situation and relationship between its part

The goals of this system are simple; generate the most appropriate nutrition for the user. For example, the user is down with diabetic and is on the process of recuperating so the system will generate the appropriate meals, supplements and do & don't of foods during the recuperating period.

Constraints are conditions put on goals. For this system, the constraint will be the type of disease the user have and the personal particular. While generating the solution the system will take into consideration these two constraints. This is to narrow down the scope of the diseases and for the personal particular, this is for calculating the age and the health status of the user.

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 this system, a user symptoms and diseases must be tested by a medical practitioner first. For example a user with high blood pressure must be examined by a medical practitioner so that the user are qualitative proven to have high blood pressure.

3.3.2 Case Retrieval in Nutritional Advisory System

In the Nutritional Advisory System, retrieving tasks start with a problem description. The

3.3.1.2 The Content of Solutions

There are two components in the solution of a case; the complete result and the nearest result. The complete result will contains all the meals and supplements and do and don't in food and medicine precaution. The nearest result will be an adaptation from the previous result and it is acceptable in the sense it is still providing the appropriate result. Example when a diabetic user is using the system and when the system examines whether the result of an old solution will hold in the new situation. If the past cases is a user with diabetic and high blood pressure than the system will try to serve meal that are low in salt and low in sugar in the meal but in the new situation the user just have diabetic. So the system will have to justify the situation on whether to serve the identical meal or try to looks for alternative meals.

3.3.1.3 Methods for Index Selection in Nutritional Advisory System

There are three type of indexing in Case Based Reasoning; checklist-based indexing, difference-based indexing and explanation-based indexing. The method of indexing the system use is checklist-based indexing. Checklist-based indexing is a well defined indexing method that is suitable for Nutritional Advisory System. This is due to the facts that checklist-based indexing is well defined and fix in its dimension. When using a checklist, process of indexing will be much easier.

For Nutritional Advisory System, user particular, diseases and medicine prescribed by medical practitioner. All of these components will be used in indexing a case. With these well defined checklists, retrieving and matching will be more accurate and definitely faster.

3.3.2 Case Retrieval in Nutritional Advisory System

In the Nutritional Advisory System, retrieving tasks start with a problem description: the input of the user and the best matching found in the previous cases. Three major tasks are involved in the process of retrieval. Its subtasks are referred to as identify Features, Initially match, and Select in that order.

3.3.2.1 Identify Features

As for Nutritional Advisory System, the inputs of the user are the features that used to identify a problem.

3.3.2.2 Initially Match

The identify features are use as an index to find a match in the Nutritional Advisory System's Case Library. Case that matches all input features is good for matching and the match case will be used as the solution for the new problem.

Usually in Nutritional Advisory System a 100% match is not always found. The system will try to find a similar match among the cases store in the library. How is this done? The system will look into a causal model for reference towards the identify features for a similar disease e.g. asthma; the system will search for it in the causal model and try to locate the nearest neighbor and something like breathing complication or respiratory disease will be use as a substitute to the unmatched feature. The third task in case retrieval will be use to select the best case.

3.3.2.1 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 used in solving the problem. But sometimes similar cases will be found but the system will take in consideration each features in the case as criteria in selection. These problem should not be a major problem to the system as the system only stored cases that have a variation in the identify features as a different case. And if no match was found then only the causal model came in view to replace the

unmatched feature and this will let the system continue to match the others features. The replacement is only for the disease feature and all the other features are not replaced. After the second matching, some adaptation will be made to the solution so that it will be in the optimum solution for the user.

3.3.3 Case Reuse in Nutritional Advisory System

The system will retrieve the case when there is a match. The case retrieved may not be a 100% match so the system will make adaptation and modification before it can be use in the new case. But if the case is a perfect match then there will not be any adaptation and modification process and the case will not be stored as a new case. But some sort of remarks will be made to the retrieved case.

3.3.4 Case Retainment in Nutritional Advisory System

Case Retainment in Nutritional Advisory System is to store the newly derived case in problem solving. The stored case can be used in future problem's solving. The new case will be indexed before it is stored in the case library. This process will ease the search for the case in the future. After the solution is being displayed to the user, the user can choose to accept/reject or make some surface adaptation to the case. If the solution is accepted by the user, the case will be retained in the case library. If the case is rejected totally by the user, there will not be any adaptation. For the surface adaptation, it is actually a kind of negotiable changes towards the solution provided. Example, when the system generated a meal for the user with milk as a drink, the user can suggest for changes to the meal as maybe the user alleged to milk and so the system will try to make some changes towards the meal like suggesting soya bean milk as a replacement.

Integrated is the final step of updating the knowledge base with new case knowledge in Nutritional Advisory System. By modifying the indexing of existing cases, the system learns to become a better similarity assessor. 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.

3.4 Development Environment

Development environment is very important in providing a stable environment for the system to be tested and run. Some of the criteria for selecting a development environment are stability, platform dependence and timeliness.

3.4.1 Operating System

Windows 2000 Professional

Windows 2000 Professional is the Windows operating system for business desktop and laptop systems. It is used to run software applications, connect to Internet and intranet sites, and access files, printers, and network resources.

Windows 2000 Professional is built on Windows NT technology and the easy-to-use, familiar Windows 98 user interface. It gives business users increased flexibility. Microsoft had take full advantage of the new technologies in Windows 2000, delivering products that are more manageable and reliable, thereby reducing the cost of computing. Working in partnership with hundreds of Independent Software Vendors, (ISVs) Microsoft has developed a directory of Windows 2000 compatible applications. This directory lists products that have been tested on the Windows 2000 platform and are ready for deployment [10].

Advantages of Windows 2000 Professional

a) *PCs stand up and running*

Windows 2000 includes a built-in safeguard called Windows File Protection. This feature helps prevent critical operating system files from being deleted or altered by users or applications.

If a system file should be changed or deleted, Windows File Protection can detect the change, retrieve a correct version of the file from a cache, and restore it to the system file folder. The end user never knows the repairs have been made because Windows 2000 just keeps running.

b) *Protects Against User Error*

Microsoft provides "self healing applications" to repair the mistake made by user by incorrectly installed or removed an application, or accidentally changed one of the application files, which could cause a system failure.

c) *Fewer Reboots*

Performing routine maintenance on your system requires significantly fewer reboots, therefore less downtime, with Windows 2000. In addition, with its support for Plug and Play, Windows 2000 automatically recognizes and adapts to hardware changes.

Windows 2000 Professional is up to 30 percent faster and, according to Microsoft and Independent Hardware and Software Testing (NTSL) test, 13 times more reliable than Windows 98. In short, Windows 2000 Professional is the most reliable Windows ever.

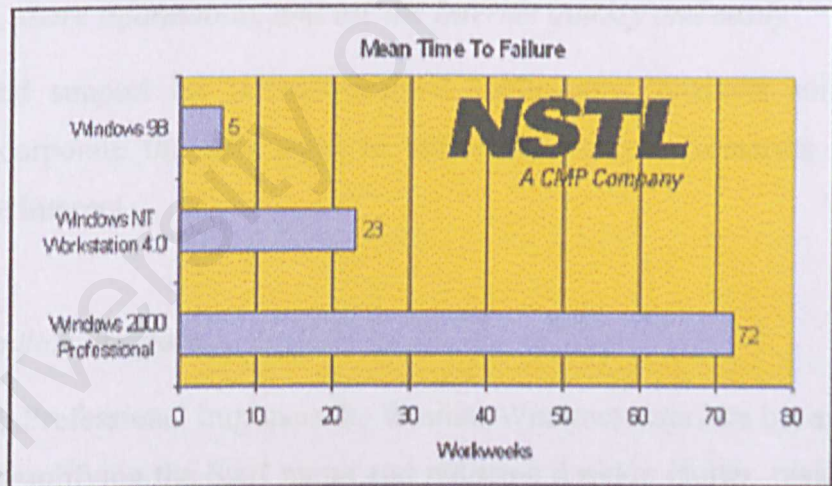


Figure 3-3: Windows 2000 mean time to failure greatly exceeds that of Windows NT Workstation 4.0 and Windows 98.

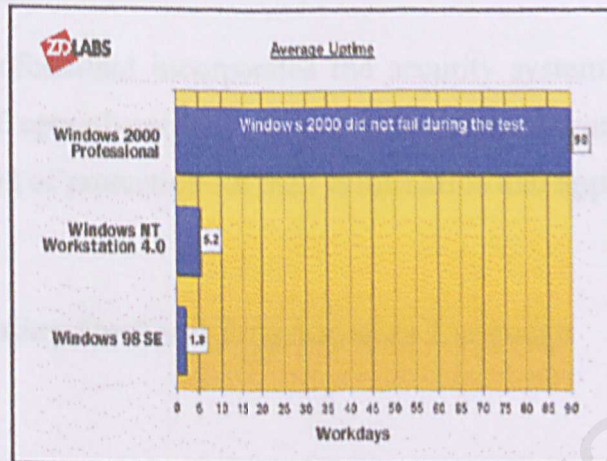


Figure 3-4: Windows 2000 Professional simply did not fail during the 90 days it was tested by ZD Labs.

d) *Communicate, share information, and use the Internet quickly and easily*

With integrated support for Internet-enabled applications, business software developers incorporate the new ways to create and share information made possible by the Internet.

e) *Improved, Familiar Interface*

Windows 2000 Professional improves the familiar Windows interface by, among other things, simplifying the Start menu and reducing desktop clutter, making it the easiest Windows to use, ever. Personalized Menus, a new "smart" feature that adapts the Start menu to the way you work, only displays the applications you use most often.

f) *Most Secure Windows*

Windows 2000 Professional incorporates the security system that is a part of every Windows NT operating system. It allows users and administrators to select the appropriate level of protection for their information and applications.

3.4.2 Case Based Reasoning Shell and Programming Language

*ART*Enterprise*

*ART*Enterprise* is the latest incarnation of Inference Corporation's flagship development product. In the 1980's ART and then ART-IM were marketed as AI development tools. Inference have dropped the AI label and now market *ART*Enterprise* as "an integrated, object-oriented applications development tool. Designed for MIS developers". *ART*Enterprise* offers a variety of representational paradigms including:

- objects supporting multiple inheritance, encapsulation and polymorphism;
- rules; and
- cases.

This is all packaged with a GUI builder, version control facilities, and the ability to link to data repositories in most proprietary DBMS formats for developing client-server applications. Moreover, *ART*Enterprise* offers cross-platform support for most operating systems, windowing systems and hardware platforms.

The CBR component in *ART*Enterprise* is essentially the same as that in CBR-Express (or rather vice-versa since CBR-Express uses code from ART to provide its CBR functionality). Thus, *ART*Enterprise* offers nearest neighbor matching and the impressive text handling mentioned above. However, the CBR functionality of *ART*Enterprise* is more controllable than in CBR-Express. Moreover, the integration with other knowledge representational paradigms means that this offers an excellent

environment to integrate CBR with other techniques and to use MBR techniques for case adaptation. However, although the package as a whole is very powerful the CBR functionality of ART*Enterprise is less powerful than a tool such as ReMind.

ART*Enterprise is currently undergoing advanced beta-testing with selected sites and will be available commercially shortly. At the time of publication the authors did not have first hand experience of this product. This review is based on information supplied by contacts within Inference, attendance at product seminars, and contacts with beta-testers.[9]

3.4.2.1 CBR engine

The CBR engine provided by Art*Enterprise are as follows:

- The CBR engines supports 32 bit addressing so that theoretically, it can support up to approximately 4 billion case
- The CBR engine provided by ART*Enterprise provides direct access to C++ and a plug-in to ART-Script
- In Art-Script, case can be define as object in ART*Enterprise or define directly in CBR engine
- Cases in the CBR engine are accessed by name rather than by an assigned key-number. Thus this make ART*Enterprise only contain the relevant information
- The types of matching provided by CBR engine is spell checker preprocessing. Beside this new kind of matching, this engine also provide two types of matching; Recursive Object Matching and Taxonomic or Object Proximity Matching

CHAPTER 4

SYSTEM DESIGN

This chapter will discuss about the design of the system. Each stage in the process has been elaborated into a more details sub process. Data Flow Diagram (DFD) will be used to describe the facets in the proposed system where each of the modules in the system has been drawn out. The details represented in this chapter will serve as a reference and important guidance for the system development phase as well as the system implementation and maintenance phase.

4.1 SYSTEM AND USER REQUIREMENTS

The systems requirements need to be drawn out before develop a system. A requirement is a feature of the system or a description of the system is capable of doing in order to fulfill the system purpose [9]. There are two types of requirement, which is as followed:

- Functional requirement
- Non-functional requirement

4.1.1 Functional Requirements

A functional requirement describes an interaction between the system and its environment. It also describes how the system should behave when given a certain stimuli. The functional requirements for this system are stated below:

- **Results generated by the system must be reliable and consistent-** this is due to the fact that each type of disease will have the same symptoms and cause some similar malnourish problem. And the medicine the user taking will also cause the same side-effect as other who took the medicine. The variation of the result will only be the amount of supplements and minerals that a user needs and the type of food the user prefer. As a whole, the component of the nutrition will still be the same.

- **Able to provide alternative nutritional advice-** the system must be able to provide advises even though the system do not have the exact matching. And the advice produce should not be mediocre to the exact matching case.
- **Able to provide explanation on the generated result-** the system should be intelligent enough to have explanation accompany the main result. The explanation should be easy enough for the user to understand.
- **Have to guide user through the whole eliciting process-** this is a type of help provided by the system to ease the user while the system elicit the user for information.
- **Nutritional Advise must follow the guideline provided by the World Health Organization (WHO) & United States Recommended Dietary Allowances (US RDA) -** this is important because the amount of supplement and minerals intake per person is limited. Over dose of vitamins and minerals also will cause health problem. The system follows the US RDA standard because the standard they provided is reliable and up to date. Whereas the Malaysian RDA are very similar to the Australian RDA and British RDA. But most of the RDA is the same. The health guidelines provided by the World Health Organization should be followed so that no misleading and controversial advice are dispense.

4.1.2 Non-functional Requirement

A non-functional requirement is a description of other features, characteristics and constraints that define a satisfactory system [9]. Below are the non-functional requirements of the system:

i. Maintainability

Maintainability is the degree to which the system can be cost-effectively made to perform its functions in a possibly changing operating environment. The system are easy to modify and test in updating process to meet the new request, correcting errors, or move to a different computer system. Reusability is also one of the main points for maintainability. This is due to the fact that function that is reusable can be

replaced easily as all the system just call the same place for the function. Example for output display template, whenever the system needs to display the result, it will call the template file to retrieve the template. If any of template need amendment, the administrator just needs to change the template file.

ii. Reliability

The system operates in a user-acceptable and does not produce dangerous or costly failure when it is applied in a reasonable manner.

iii. Response Time and Performance

The time needed for matching the case should not be more than 2 minutes but most of the time is dependent on the amount of cases in the database that needed to be retrieve and match.

iv. User friendliness

The graphic user interface should be simple and nice. The font size and color be proportioned to the size of the page. Pop-up list and drop-down list is added to ease user in filling the form. And additional help button is provided to facilitate the user.

v. Accurate and robust

Should be able to wind stand any type of errors caused by user or computer. If multiple case searching is execute by the system; it should not crash as a CBR engine could process more than 1 billion cases. For this, I assume the optimum numbers of cases for the system would be around 400 million cases.

vi. Security

Although it is not very important for this system but it still have some kind of security functions like only the administrator of the system can change the solution in the case library and the feature weight.

4.2 System Overview

Nutritional Advisory System basically divided into seven main modules. Below are the modules in brief:-

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

Identify Module

This module is basically the input module for the user. This place is where the selected features are extract from the data inputted by the user. Beside this, when there are information needed for further confirmation there will be an extra window to let the user to key in the relevant information. The identified features will be passed to the matching module. For the personal particular e.g. name, address will be passed to the output display for printing purpose. A proper indexing will be used to index a new case. The personal particular given by the user will have to check with the weight and height chart to calculate the status of the user as Overweight, Underweight and Normal.

Matching Module

The core of this module is to retrieve case from the case library and match the case in library with the new case. If a match is found then the module will pass the result to the display module. If no match was found then the new case will passed to the similarity module for further matching. The threshold for matching is also being declared here.

Similarity Module

When no perfect matching is found the system will go to the causal model to retrieve some similar feature that matches the new case's features. For example, when there is no match for asthma, the similarity module will search the causal model for similar category disease like bronchitis, tuberculosis, pneumonia, and legionnaire. After replacing the feature, the system will try to match the similar case with the case in the library. All the similar cases will pass to the adaptation module for further modification and adaptation.

Adaptation Module

Cases found by the similarity module are the solution for other problem but are quite near to the present case. These cases need some adaptation to the present case before it can be use in the new situation. After the adaptation is done, the new case will be passed to the evaluation module.

Evaluation Module

After all the case are adapted to the new problem. A threshold is set to evaluate which case have the highest value will be fired and the second highest value case will be the alternative case for the user. The best problem will be send to the user using the display module.

Display Module

The display module is the interface for the Nutritional Advisory System. It is use to output the result to the user and a decision point for whether the solution to the problem should retain in the case library

Retain Module

After the decision to retain the new case the retain module will save the case in the case library. Before the case is retained inside the case library, indexing will be made for the new case.

4.3 Module Design

Below is the explanation of the Data Flow Diagram for Nutritional Advisory System.

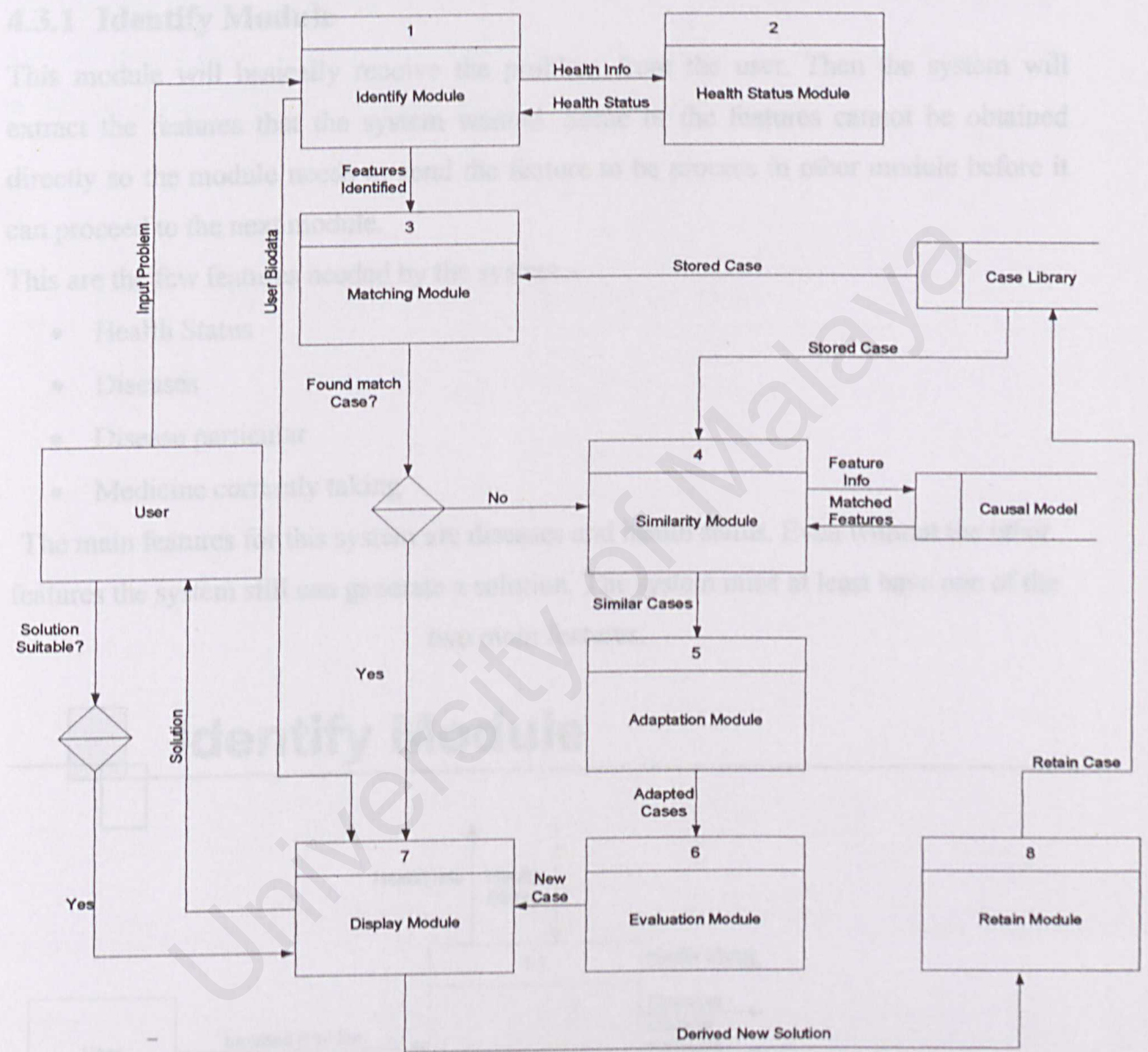


Figure 4.1 Data Flow Diagram of Nutritional Advisory System

4.3 Module Design

Below is the design for each module using Data Flow Diagram Level One as diagram for explanation.

4.3.1 Identify Module

This module will basically receive the problem from the user. Then the system will extract the features that the system wanted. Some of the features cannot be obtained directly so the module needs to send the feature to be process in other module before it can proceed to the next module.

This are the few features needed by the system:-

- Health Status
- Diseases
- Disease particular
- Medicine currently taking

The main features for this system are diseases and health status. Even without the other features the system still can generate a solution. The system must at least have one of the two main features.

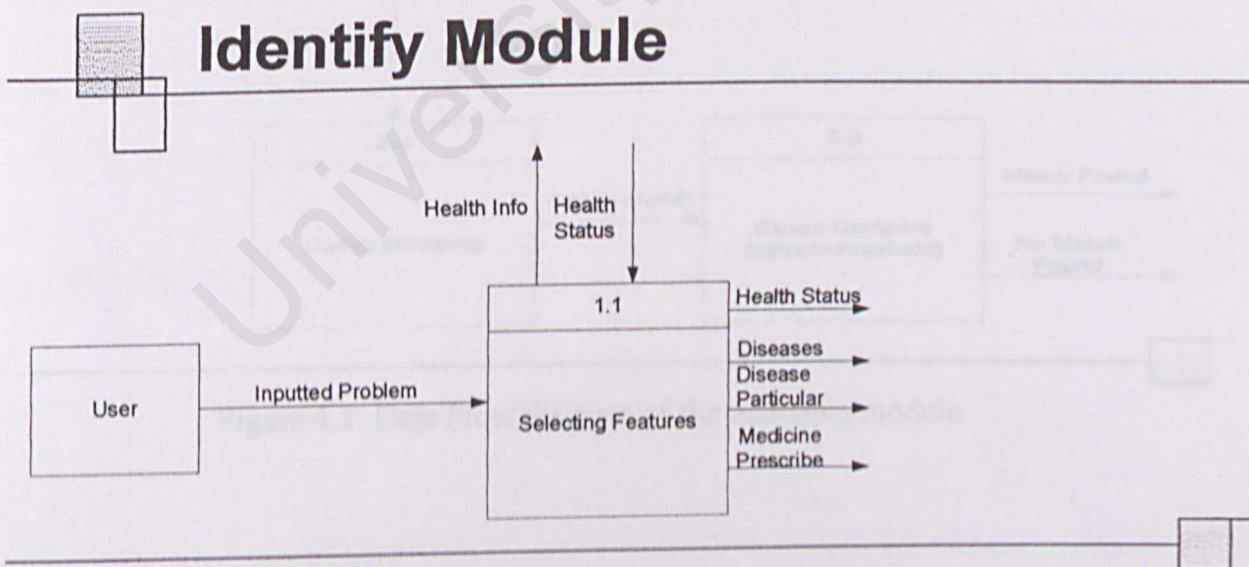


Figure 4.2: Data Flow Diagram of Identify Module

4.3.2 Matching Module

In this module, the matching process will be executed here. The previous cases will be retrieved from the case library to match with the new case. Based on the weight and indexes given by the identify module, the matching module will search an identical match from the case library. If the case found is more than the threshold set in this module the case will passed to the display module. If there is no match or all the matches are below the threshold value than the new case will pass to the similarity module.

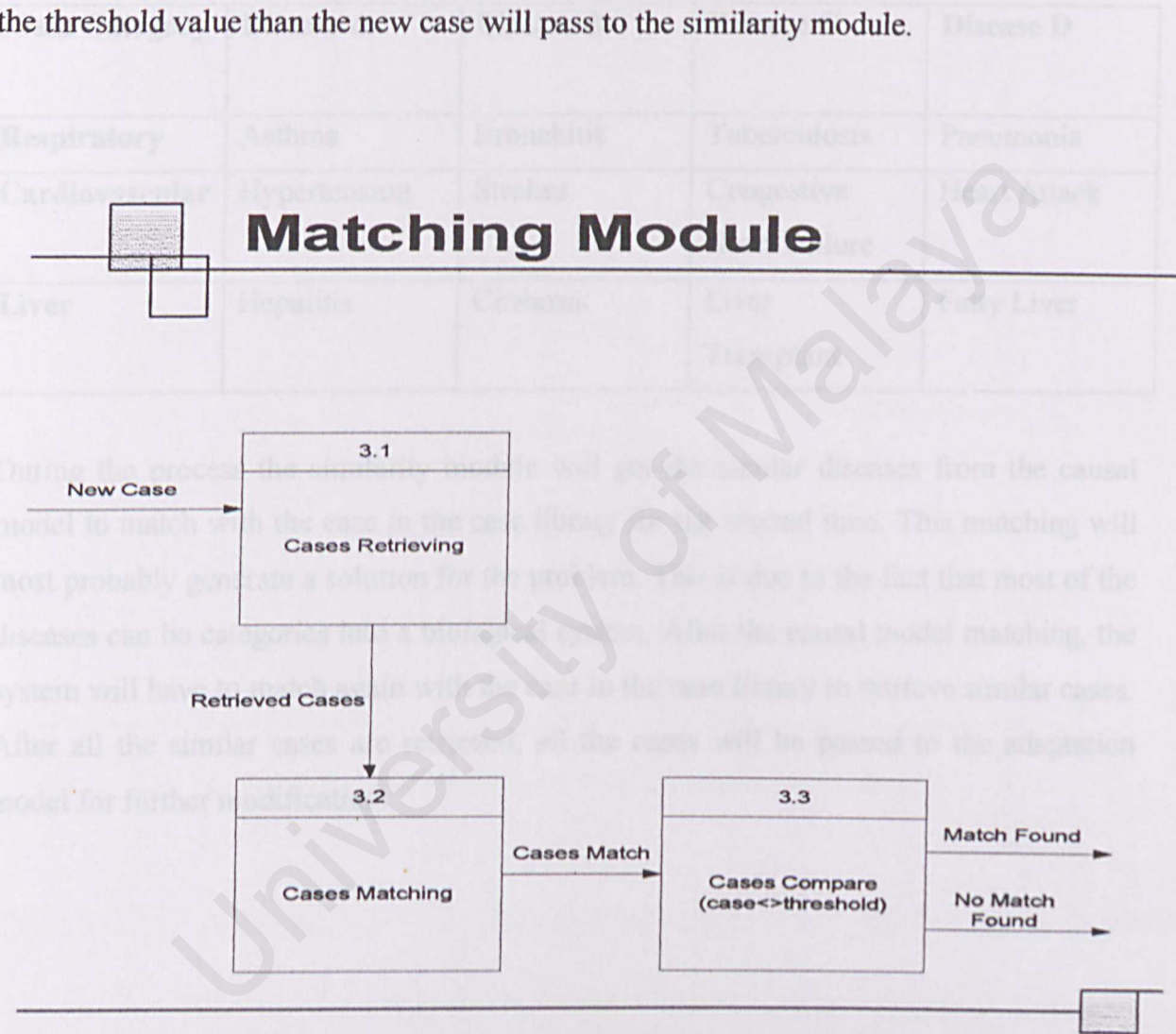


Figure 4.3: Data Flow diagram of the matching module

4.3.3 Similarity Module

In this model, the system uses a causal model to deepen the search. The causal model works as a table for the system to search for similar feature. For example, a example for the causal model

Main Category	Disease A	Disease B	Disease C	Disease D
Respiratory	Asthma	Bronchitis	Tuberculosis	Pneumonia
Cardiovascular	Hypertension	Strokes	Congestive Heart Failure	Heart Attack
Liver	Hepatitis	Cirrhosis	Liver Transplant	Fatty Liver

During the process the similarity module will get the similar diseases from the causal model to match with the case in the case library for the second time. This matching will most probably generate a solution for the problem. This is due to the fact that most of the diseases can be categories into a biological system. After the causal model matching, the system will have to match again with the case in the case library to retrieve similar cases. After all the similar cases are retrieved, all the cases will be passed to the adaptation model for further modification.

Similarity Module

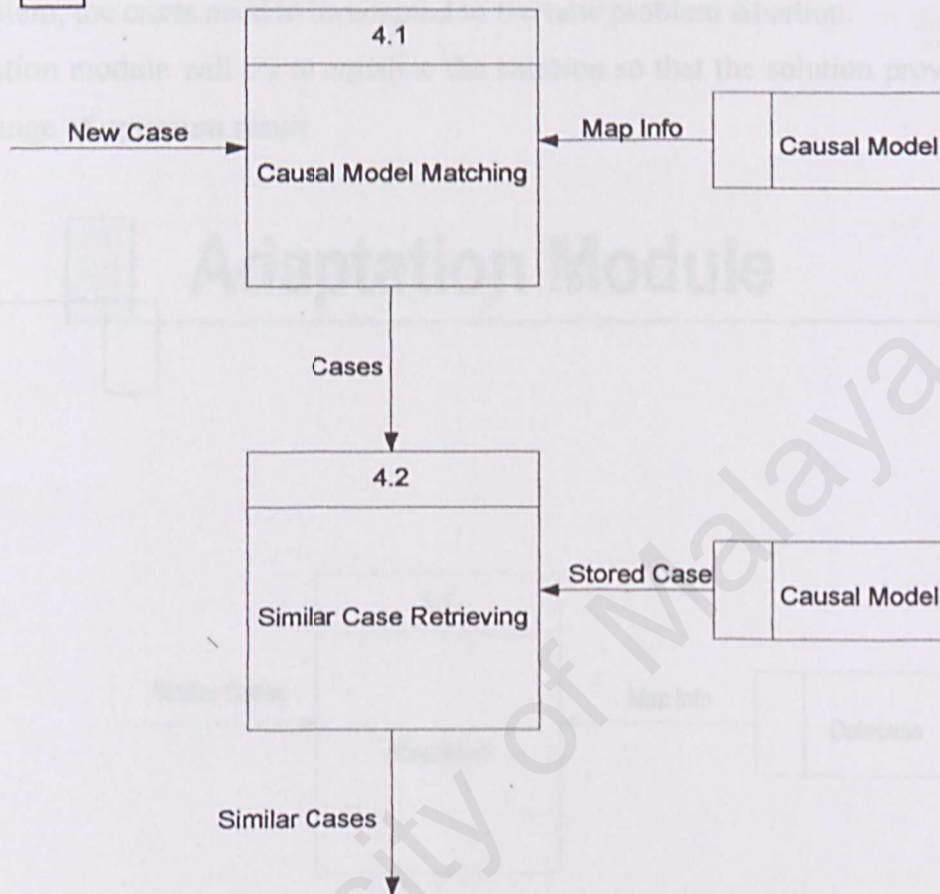


Figure 4.4: Data Flow diagram of the similarity module

4.3.4 Adaptation Module

The adaptation module will receive all the similar cases for a problem from the similarity module. These cases are the 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 adaptation module will try to equalize the solution so that the solution provided will be in the range of optimum result.

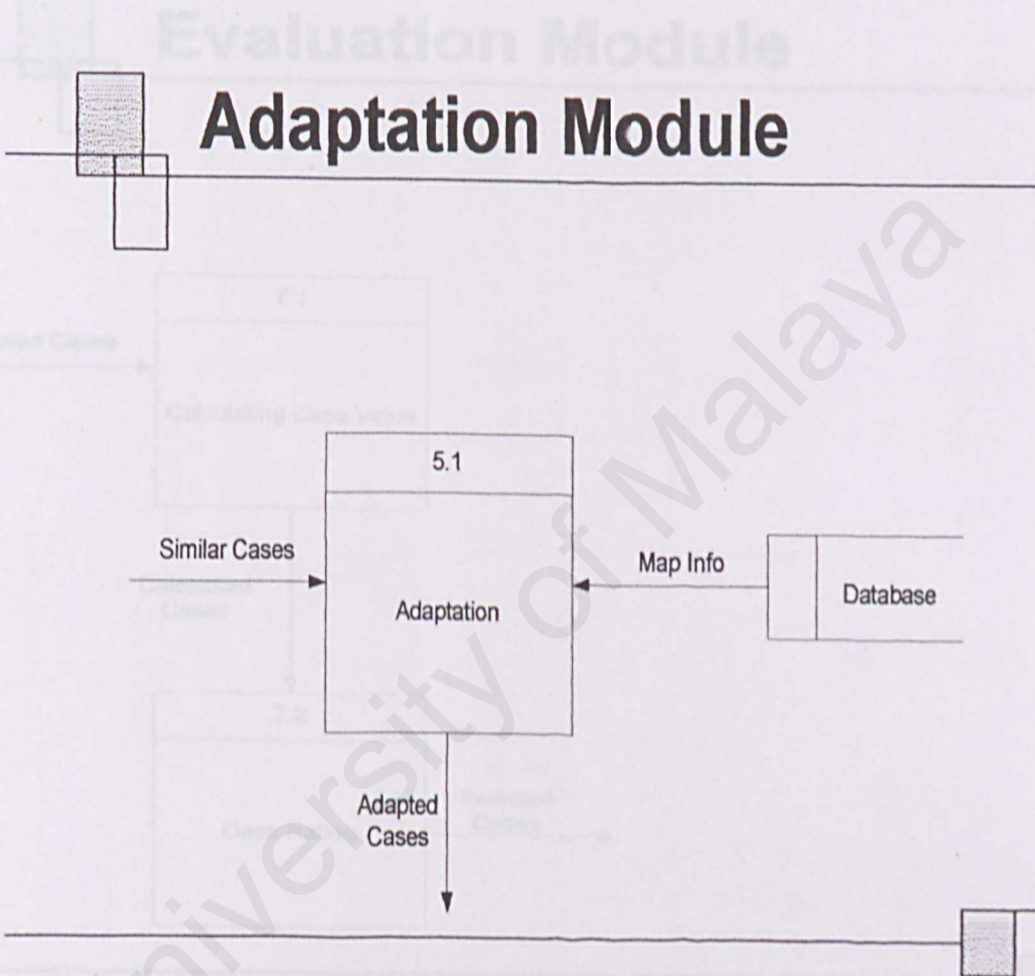


Figure 4.5: Data Flow Diagram of the adaptation module

4.3.5 Evaluation Module

The evaluation module is used to rank all the derived case from the adaptation module. The evaluation module will rate each case. The case with the highest global value will be rank first and so fourth. The first and second rank case will be send to the display module.

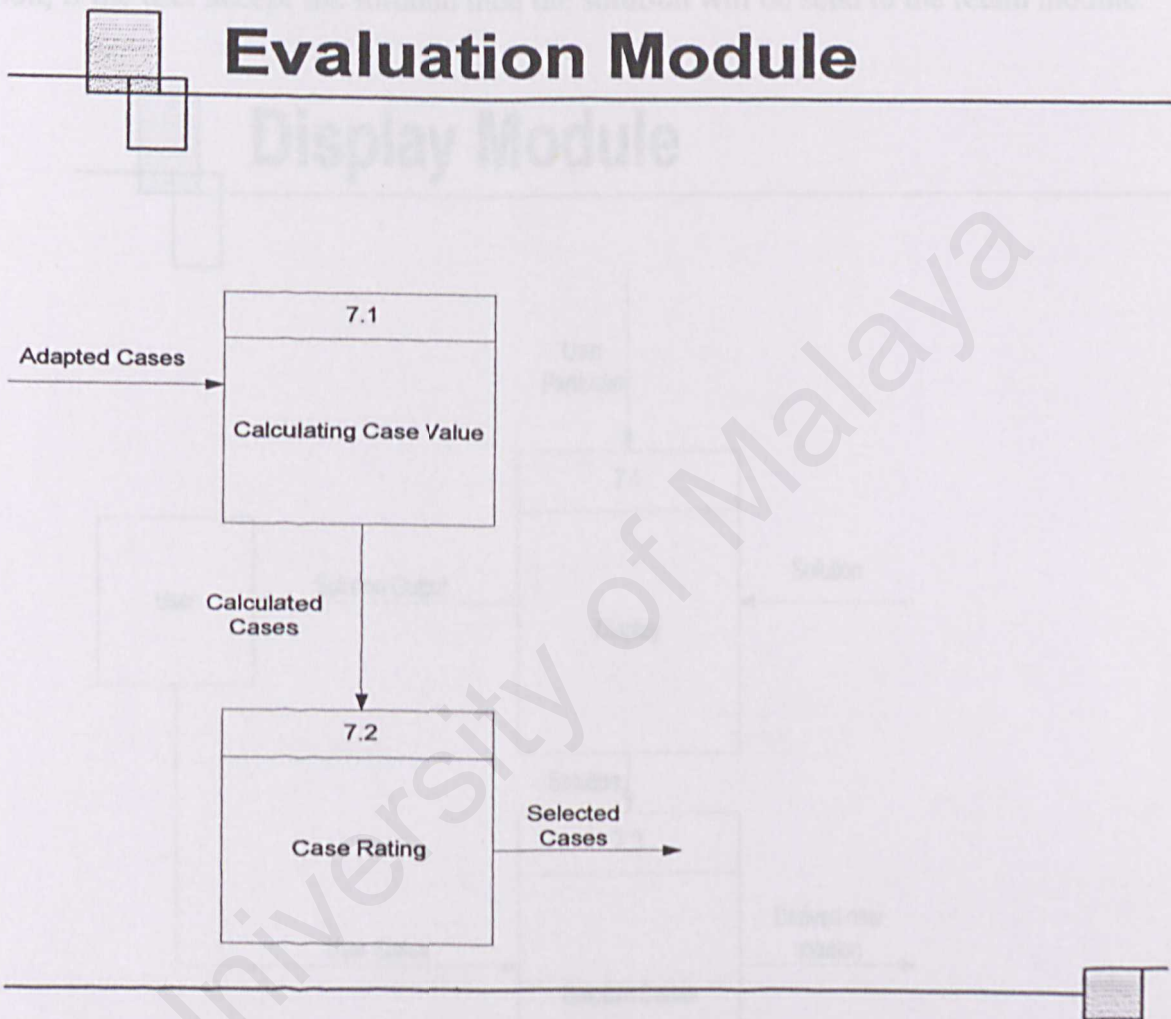


Figure 4.6: Data Flow diagram of the evaluation module

4.3.6 Display Module

This module is used to display results. This module will get the solution from the evaluation module. Before the solution is been display the module will get the user particular from the identify module. This is to let the user feel more secure about the result. After the solution being display, the module will ask the user whether to accept the solution, if the user accept the solution then the solution will be send to the retain module.

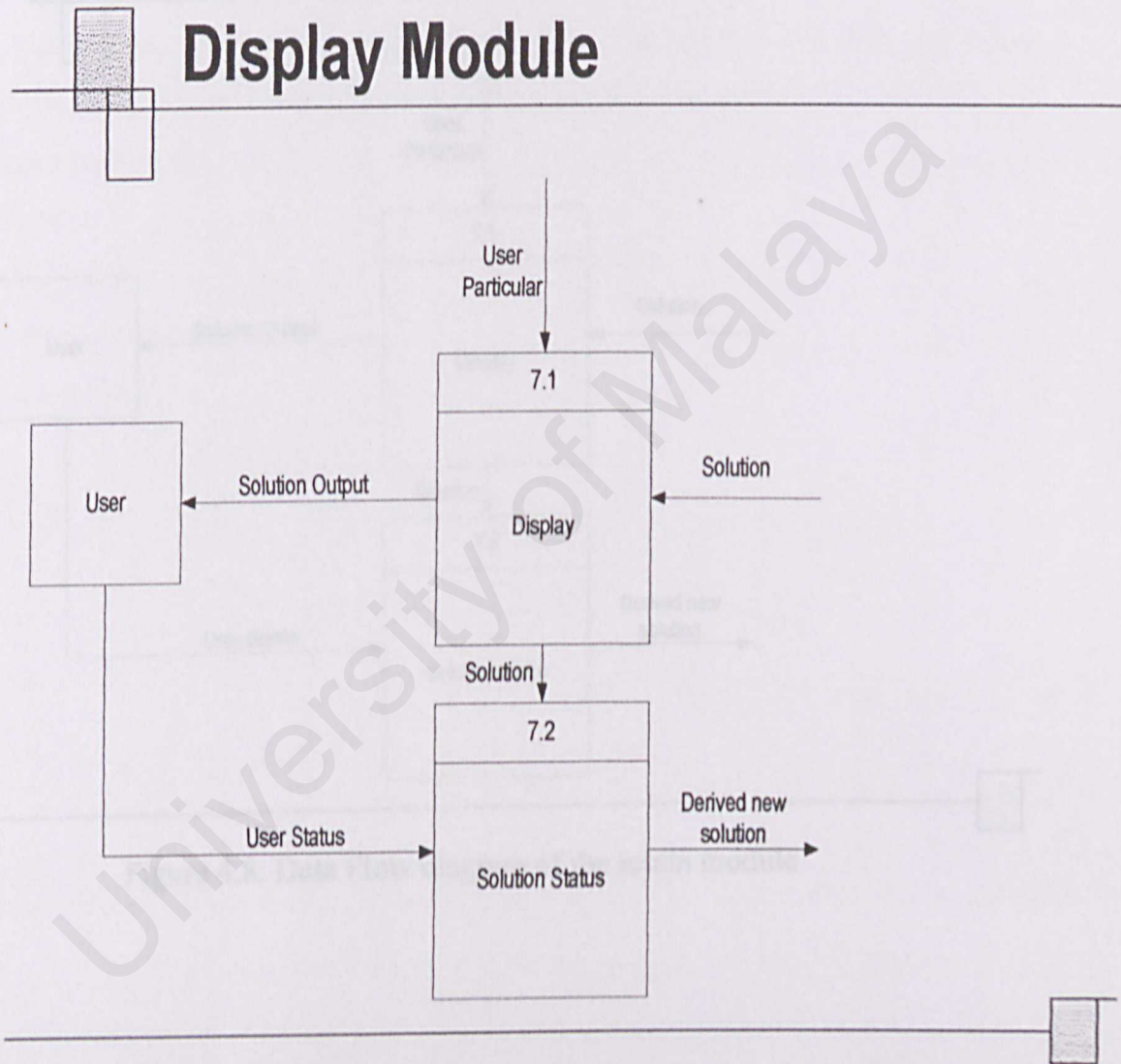


Figure 4.7: Data Flow diagram of the display module

4.3.7 Retain Module

When the user accepted the solution the new solution will be send to this module. This module will store the newly derived case in the case library. This module will index the new case for the ease of retrieval from the case library in the future.

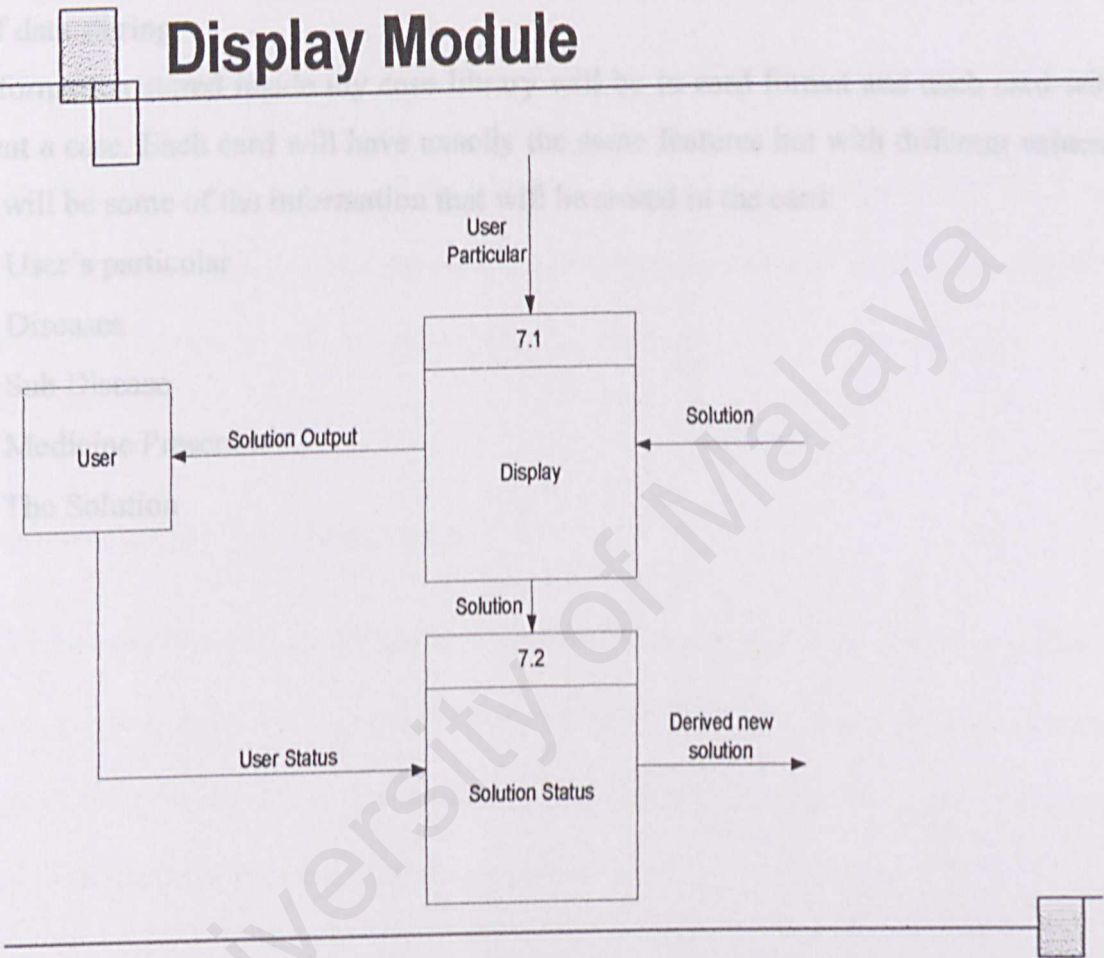


Figure 4.8: Data Flow diagram of the retain module

4.4 Case Design

All the cases in the system will be stored in a card format like the catalogue system. Each case will have their own card with the features wanted pre-stated. This is very different from database as these cards are stored inside the system whereas no DBMS is needed for storing cases. But database is needed for my causal model as database is more rigid in term of data storing.

The information stored inside my case library will be in card format and each card will represent a case. Each card will have exactly the same features but with different values.

Below will be some of the information that will be stored in the card:

- User's particular
- Diseases
- Sub Disease
- Medicine Prescribe
- The Solution

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Introduction

In this phase, the system requirements and design model of a system will be converted into a workable product. System implementation includes coding, testing and documenting the system as well as training the end users and system administrators.

5.2 Development Environment

Development environment has certain impact on the development of a system. Using the suitable hardware and software not only help to speed up the system development but also determine the success of the project. The hardware and software tools used to develop the entire system are as below:

5.2.1 Hardware Requirements

The hardware specification of the machine for development in this project is:

Workstations
<ul style="list-style-type: none"> ▪ Running on Windows 98 or Windows 2000 Professional or Windows XP ▪ Consist of 128 MB RAM ▪ Alongside a Pentium processor 500 MHZ ▪ Other standard PC component

Table 5.1: Workstation

1. Application coding tools

- ArtScript 2.0.1

• Create CBR Cases

- Define Type of Matching Score to use

In the development of Nutritional Advisory System, the software basically consisted of CBR Shell and tools. The shell included all the technology used to support the functionality of the system such as matching and adaptation. Whereas the tools applied are those development applications used to design and develop.

- Creates and refines all CBR cases and objects for the system

- Debug Application

- Monitor if any error occur during the programming process

- Commercial tool

- To help programmer

2. Graphics / Interface Modeling Tools

- Art*Enterprise (4.0)

- Easy the task of creating and editing the windows interface design

5.2.2.1 Description of Development Application / Tools

Below is a listing of application / tools categories used for the Nutritional Advisory System project development:

1. Application coding tools

- ***ArtScript 3.0.1***

- Create CBR Classes
- Define Type of Matching Score to use
- Create application function
- Create Object proximity class
- To generate result

- ***Art*Enterprise Studio***

- Creates and refines all CBR classes and objects for the system.
- Debug Application
- Monitor If any changes occur during the programming progress

- ***Command Interpreters***

- To run the application.

2. Graphics / Interface Modeling Tools

- ***Art*Enterprise 3.0.1***

- Ease the task of creating and editing the windows interface design.

5.3 System Implementation

Implementation comprises of the system design structure to a computer readable system. The system will be evolved from scratch design to a run able application. There are several implementations for this system.

5.4 Interface

Basically the system do not have a graphic interface because in ART*Enterprise the development of software comprises of three distinct levels that is the user interface level, the database integration level and application level. For my Nutritionist Advisory System it is in the application level. Due to this, all our output and input are pass though the Command Interpreter provided by Art*Enterprise.

The Command Interpreter is like a command based interface where the users have to key in the command as they are keying in a DOS based system. So to overcome the unfriendliness of the Command Interpreter, the NAS System was designed to accommodate the need of the user by providing guidelines and help along the process of using the system.

5.5 Nutritional Advisory System Framework Implementation

The system can be divided into a few major steps, first the system needs to be setup by the expert domain, and then the system can be use by the end user to input their case. The system will then match the presented case with the stored cases. The top ten cases will be retrieved according to the matched values. Then the threshold will only allowed the those cases that fulfill the system requirement to be retrieved by the user. After the user have the final result, the system will save the result to a temporary case-base for the treatment period advised by the expert domain. After the treatment period, the expert domain will make the decision to save or not to save the case. If the expert domain wanted to save the case, he will retrieve the case from the temporary case-base and move it to the permanent case base.

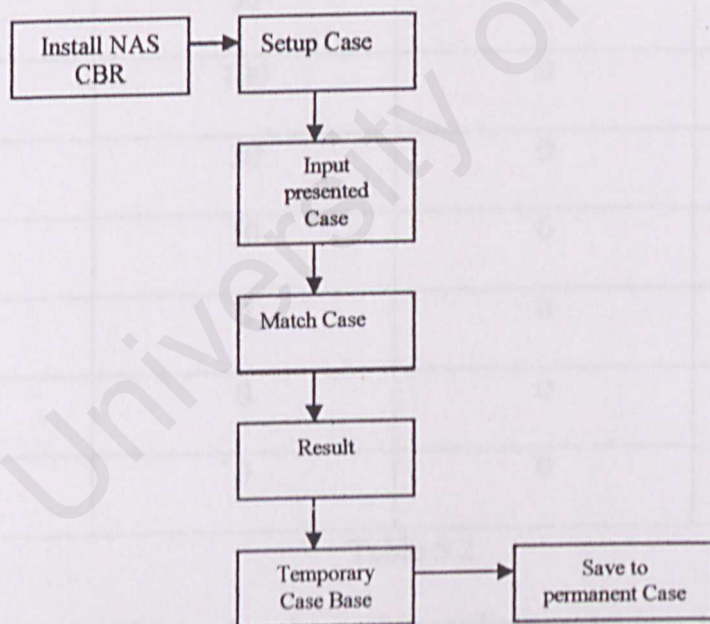


Figure 5.1

5.5.1 Attributes Weighting Implementation

The attribute score for a stored case is computed from a match contribution, mismatch penalty, or absence penalty specified for the attribute as follows:

- If the attribute values in the stored and presented cases match, the match contribution is added to the score
- If the attribute values in the stored and presented cases do not match, the mismatch penalty is subtracted from the score
- If the attribute exists in the presented case but not in the stored case, the absence penalty is subtracted from the score

The attributes in the system are defined as default by the system. The weight for each attributes is as follow:-

Attributes	Match Contribution	Mismatch Penalty	Absence Penalty
Category	30	30	5
Disease	100	0	40
Sub-disease	30	0	0
Medication	10	0	0
Meal	0	0	0
Precaution	0	0	0
Supplements	0	0	0

Table 5.2

In this system the weight can be changed according to the user requirement if the user does not change the values then the values as above will be used for matching. The default weights are tested with familiar cases to get the best weighting values for all the attributes. The weight can be set as follow:

(define-attribute category slot);;.....	1
(cbr:define-attribute nas-case-base category cbr:word ;;.....	2
:match-contribution 30 :mismatch-penalty 30 :absence-penalty 5);;....	3

The attribute name can be define as in line 1(shown in the box above) and 2nd line is to define the case base that the attribute is related to and what type of attribute type – for the example above it is word. The 3rd line is to define the weighting.

For the first attribute there is additional function build in it. This attribute have a pyramid kind of tree built in it to facilitate the searching process like shown in table 5.3. This is due to the fact that the system searching is based on index heap sorting and the case is store in an object oriented programming way that each attribute are being linked to the other attribute in the object and other objects. When the first attribute are matched against the stored attribute, the stored attribute that matched the presented value will be retrieved and eventually the whole object instance will also be retrieved together. To retrieve the second matching will also be easier as the next nearest case are next to the first matching because the case base records are stored in alphabetical order.

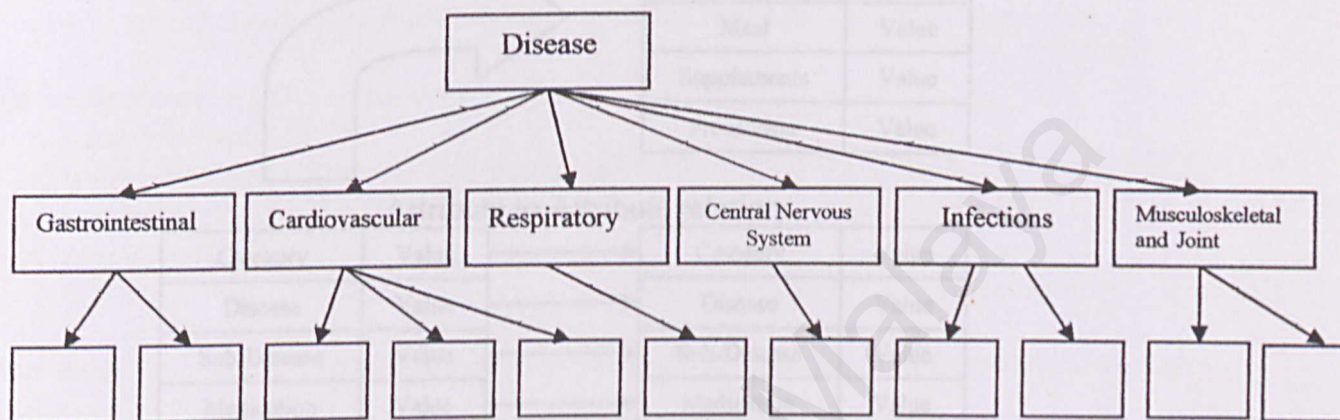


Table 5.3: The Hierarchical Pyramid

3.5.2 Matching Implementation

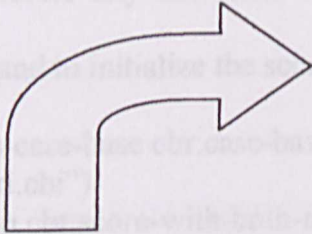
The matching algorithm for this system is score with stored-max and is a sub-matching

algorithm of K-Nearest Neighbors matching algorithm. It is used to define the matching method before any new case can be added to the system for matching.

Below is the command to initialize the system with the matching algorithm.

```
(define-instance new-case cbr case-base
(cbr-index-file "data/cbr"
(cbr->cmg-func (lambda (x) (cbr->cmg x))
(cbr-match-func (lambda (x) (cbr->match x))
(cbr-max-m
```

Object Relation
One Object to Other



Category	Value
Disease	Value
Sub-Disease	Value
Medication	Value
Meal	Value
Supplements	Value
Precaution	Value

Attribute to Attribute relation

Category	Value	→	Category	Value
Disease	Value	→	Disease	Value
Sub-Disease	Value	→	Sub-Disease	Value
Medication	Value	→	Medication	Value
Meal	Value	→	Meal	Value
Supplements	Value	→	Supplements	Value
Precaution	Value	→	Precaution	Value

Table 5.4 Structure of Case Retrieval Net

5.5.2 Matching Implementation

The matching algorithm for this system is score-with-stored-max and is a sub matching algorithm of K Nearest-neighbor matching and ranking. The system has to define the matching method before any new case can be presented to the system for matching.

Below is the command to initialize the score-with-stored-max matching algorithm.

```
(define-instance nas-case-base cbr:case-base
  (cbr:index-file "nutri.cbi")
  (cbr:scoring-function cbr:score-with-both-max)
  (cbr:match-threshold 40)
  (cbr:max-matches 5))
```

This algorithm can be changed to cbr:score-with-presented-max or cbr:score-with-stored-max to suite the situation. The matching threshold is set default to 40 /100 and the maximum numbers of matches are minimizing to 5. The threshold are set to 60% because the matching scores that dip below 60% are consider irrelevant to the presented case. The maximum numbers of matches are minimized to 5 was due to the fact that only the best and second best stored cases will be considered as finalize result so it is illogical to have redundant cases matched.

5.5.3 Adaptation Implementation

The adaptation module is implemented with a causal model. This model is activated when there is no match being found so the system will look for the similar attributes values from the causal model. The causal model is place in the nutrifolder folder. This causal model only implies to the second attribute and third attribute only. The causal model is like the figure below:-

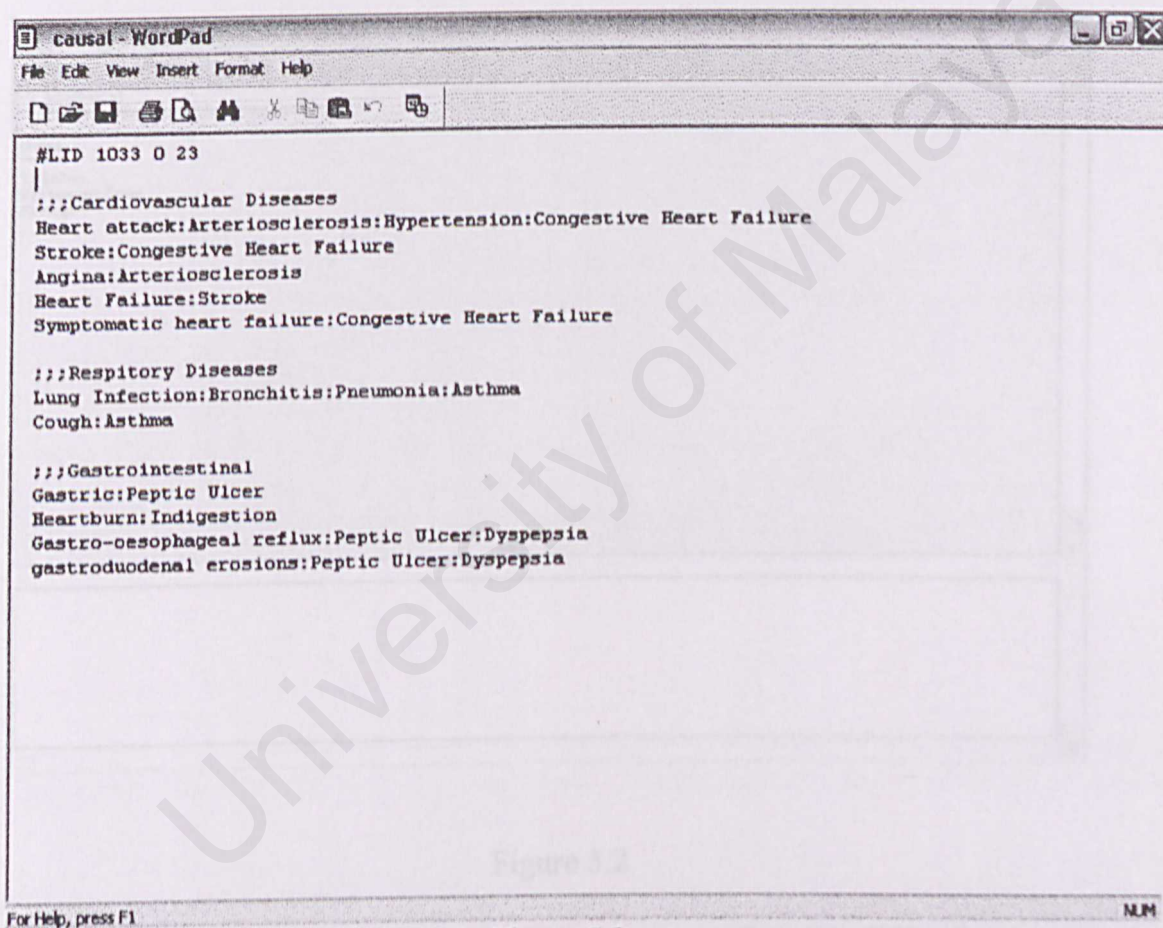


Figure 5.2

The presented case will not be altered at the end of the matching but during the matching it may replace the disease attribute with the value in the causal model to find the most suitable case.

5.5.4 Display Module Implementation

The display module is very simple because it is just a command interpreter to show the output and to input data. It is an interface for I/O stream between the user and the system. It can only display textual document and for graphic image a pop-up windows is needed. Below is the command interpreter example:-

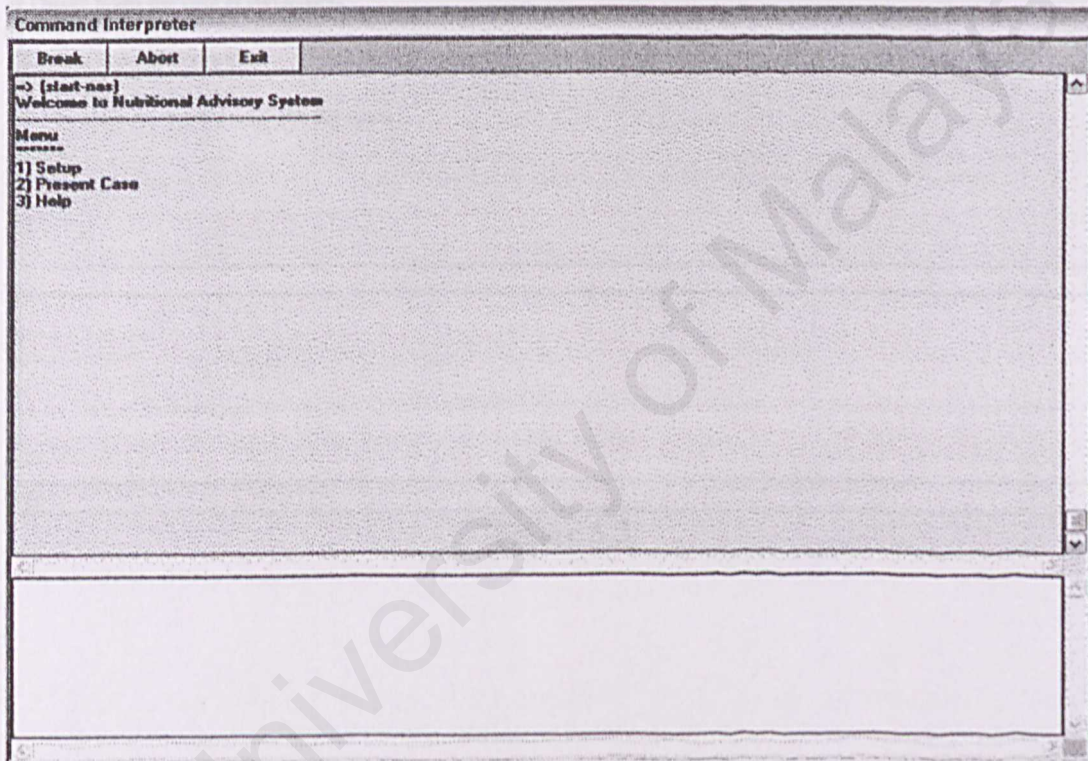


Figure 5.2

The Display module will get the relevant object instance from the CBR file stored in the folder directly. And the system allows user to key in directly into the object instance. The relevant case can be displayed in the command interpreter.

5.5.5 Function Implementation

In this system, there are a few main functions that are essential for the whole system to

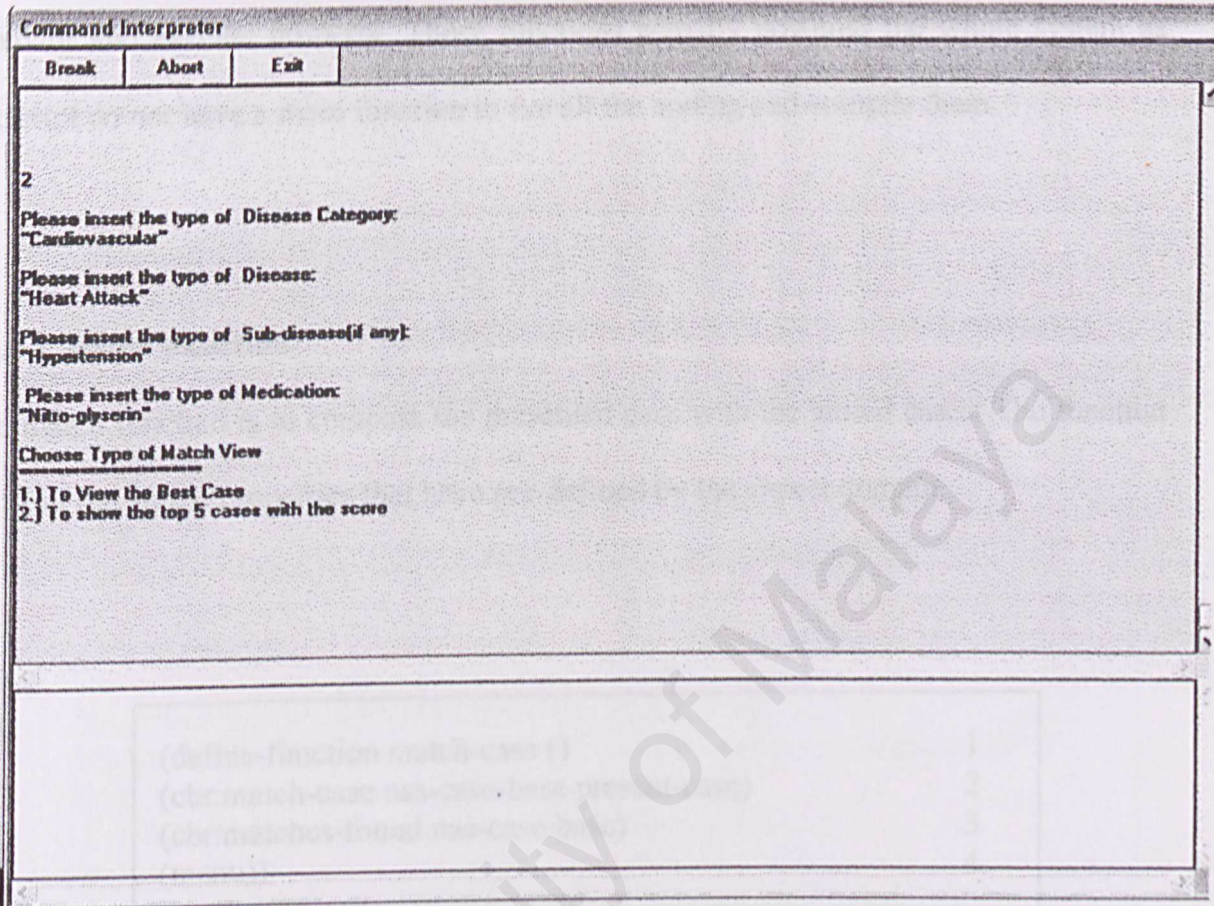


Figure 5.3

The 1st line in the above window is the function name. In the above case it is match-case. For the second line, third line is for calling the CRR matching logic. For the 3rd line, the function will use the base class that is CRR to match the generated case with the stored case. The 3rd line list out all the matches found that matches the generated case.

5.5.5 Function Implementation

In this system, there are a few main functions that are essential for the whole system to behavior and execute properly. These functions actually powered the whole system as Artscript do not have a main function to run all the coding and compile them.

5.5.5.1 Match Function

The match function is to compare the presented case with the stored cases. The function will call out all the properties that have pre-defined by the expert domain.

(define-function match-case ()	1
(cbr:match-case nas-case-base present-case)	2
(cbr:matches-found nas-case-base)	3
(menu))	4

The 1st line in the above function is the function name. In the above case is match-case. For the second line and third line is for calling the CBR matching code. For the 2nd line, the function will call the base class that is CBR to match the presented case with the stored case. The 3rd line list out all the matches found that matches the presented case.

5.5.5.2 Result Display Function

In this system, it has two kind of display function. The first one is to show the top cases that exit the system limit threshold and the second function is to display the system recommended case.

```
(define-function print-matches (?case-base)
  (for ?m from 1 to (cbr:matches-found ?case-base) do
    (printout t "Match " ?m ": Case " (cbr:get-match-case ?case-base ?m)
      " matched with a score of "
      (cbr:get-match-score ?case-base ?m) t)))
```

The above source code is for the top cases that match the presented case.

```
(define-function print-recommendation (?case-base)
  (printout t "This is the meal and supplements the system recommended" t)
  (bind ?case (cbr:get-match-case ?case-base 1))
  (bind ?attr (cbr:first-case-attribute ?case-base ?case))
  (while ?attr do
    (bind ?value (cbr:first-attribute-value ?case-base ?case ?attr))
    (while ?value do
      (printout t " " ?attr ": " ?value t)
      (bind ?value (cbr:next-attribute-value ?case-base ?case ?attr ?value))
      )
    (bind ?attr (cbr:next-case-attribute ?case-base ?case ?attr))
  ))
```

The above source code is for the system recommended case.

5.5.5.3 Menu Function

The menu function is to facilitate user to navigate through the whole process of presenting the case till retrieving the result. Without this function retrieving will be more to using command and more trivial in presenting and obtaining the result.

```
(define-function menu ()  
  (printout t "Choose Type of Match View" t)  
  (printout t "*****" t)  
  (printout t "1.) To View the Best Case" t)  
  (printout t "2.) To show the top 5 cases with the score" t)  
  (bind ?m (read))  
  (if (= ?m 1) then (print-recommendation nas-case-base)  
    else  
    (if (= ?m 2) then (print-matches nas-case-base)  
      else  
      (printout t "Invalid Insertion" t))))
```

The function above is to facilitate user in selecting which type of result display that the user wanted.

```
(define-function start-nas ()  
  (printout t "Welcome to Nutritional Advisory System " t)  
  (printout t "-----" t)  
  (printout t "Menu " t)  
  (printout t "*****" t)  
  (printout t "1) Setup" t)  
  (printout t "2) Present Case " t)  
  (printout t "3) Exit System" t)  
  (bind ?m (read))  
  (if (= ?m 1) then (printout t " Still Under Construction" t)  
    else  
    (if (= ?m 2) then (input-case))  
    else (exit)))
```

The above is the startup menu for the system. The user needs only to key in the selection number to get through to the user destination.

CHAPTER 6

SYSTEM TESTING

6.1 INTRODUCTION

Software testing is one of the main phases in the Waterfall Life Cycle model. In this phase, the process of testing and debugging are done to detect defects and bugs of a system. These processes are usually done incrementally with system development.

This phase is also often referred to as Verification and Validation (V & V). Verification refers to the set of activities that ensure the software correctly implements a specific function. Validation refers to a different set of activities that ensure the software has been built is traceable to user requirements. A successful test is one in which no errors are found.

The objectives to test this system are:

- a) To reveal inaccuracy and error by the matching algorithm of the CBR system.
- b) To compare the expected outcome with the actual outcome. Eventually, debug it to enhance it functionality and capability.
- c) To ensure stability and performance at the best.

The testing method used in this system is by [12].

CASE LIBRARY SUBSET TEST METHODOLOGY

This section describes the proposed validation technique for CaBR systems called the *Case Library Subset Test* (CLST) technique. The main concept underlying this validation method is the selection of a subset of cases from the case library and using this subset as a test set to evaluate the effectiveness of the system's retrieval and adaptation features. The comparison standards of the test set are considered to be correct because they are

part of the case library. But first, the validation criteria has to be selected, as it affects the final correctness of the systems. This process is described below.

Determination of Validation Criteria

The first task is to develop a validation criteria. This consists of determining two basic parameters, the *Result Acceptability Criteria (RAC)*, and the *System Validity Criteria (SVC)*. The RAC serves to determine whether an individual test case has been solved correctly by the CaBR system. It mandates that the distance between the system solution to a test case, and the benchmark standard to which it is compared be calculated. If the solution is provided in numerical terms, then the Relative Error (RE) can be the percent difference between the two quantities. If, on the other hand, the output of the CaBR system is symbolic or Boolean, then *optimal*, *acceptable* and *unacceptable* solutions may be defined as the benchmark standard may allow. The SVC serves to determine whether, in light of the executed and evaluated suite of test cases, the system can be considered valid. The SVC requires that upon completion of all testing, the percent of all acceptable test cases be greater than its value before the CaBR system can be considered valid. The RAC and the SVC are typically obtained from either experts or users, and it may be defined in the requirements specification. Upon selection of the above validation criteria, the CLST technique begins as described below.

Description of the Case Library Subset Test

These are described in more detail below.

- **CaBR Retrieval Test.**

Case indexing and case classification issues are intended to improve the effectiveness and efficiency of case retrieval and to reduce the complexity of similarity calculations. The correctness of the retrieval process is, therefore, one of major concerns in CaBR systems. The CaBR Retrieval Test is designed to evaluate the correctness of the retrieval function. The indexing system used, although not evaluated independently, is clearly part of the retrieval evaluation test, and deficiencies in indexing will show up as poor retrieval performance. The

comparison function is also likewise validated. Briefly, the Retrieval Test requires that each historical case in the case library "spawn" a test case identical to itself in all ways. A pointer to the historical case is maintained for the purpose of comparison later. This process generates a set of test cases, not only for the retrieval test, but also for the adaptation test as will be seen later. As part of the retrieval test, each test case is, in turn, presented to the CaBR system as the current case. The CaBR system goes through the comparison and retrieval processes, arriving at an internal list of library cases ranked in decreasing order of similarity. In order for any test case to be marked as successfully executed, the historical case which spawned the current test case should be found as the top-ranked historical case in this internal list, and the similarity distance should be the minimum allowed in the chosen measuring scheme (or very close to it).

- **CaBR Adaptation Test**

The Retrieval Test ensures that the comparison and retrieval functions are correctly carried out. It is the purpose of this test to ensure that adaptations are properly made from valid retrieved cases. Therefore, the Adaptation Test should only be done after a successful Retrieval Test. The test case set used here is the same as that of the retrieval test (e.g., spawned from each historical case in the case library). The significant difference is that in the Adaptation Test, the historical case corresponding to the test case being presented to the CaBR system is removed from the case library. Thus, if a case library has N cases in it, the modified case library will only contain $N-1$ cases at all times. The outputs of this test include retrieved cases, the final solution, and its RE. Although the test case is not longer in the case library, the CaBR system retrieves the most similar case from the case library and adjusts the closest matching case(s) with the adaptation strategies to obtain the final solution to this test case. Since the retrieval process has already been validated, this test isolates and evaluates the adaptation process of the CaBR system

6.2 Algorithm Testing

The algorithm in the system needs some testing before the real system can be used by the public. Basically the testing of algorithm is in the searching sector. The searching method

testing is based on [10]. The testing of the speed of retrieval of relevant case from the stored database is basically divided into two main types. The first one is based on the simple heap sort technique that come with the ART*Enterprise CBR shell. The second technique is still using heap sort technique retrieval but have some modification to the behaviors of the retrieving method. Firstly it has an added pyramid tree as mention in chapter 5. This tree helps the retrieval process by having the first attribute acts as a cue to what the system to know which stored cases to retrieve.

The testing is done to test the accuracy of the retrieving process and secondly the speed.

6.2.1 Testing the Accuracy of the Retrieval

The testing is based on the most familiar case and sees if the retrieval is correct or not. This is done based on the result that we already know and try to input the problem into the system then we try to determine if the answer is correct or not. If the answer is incorrect then try to find out that the error is from the retrieving algorithm or the weighting differences (weighting testing will be discussed in details after this).

There retrieving algorithm errors can be amended by correcting the path of the pyramid. And test the system again to see if the system behave as what we wanted it to be.

6.3 Weighting Test

The weighting test can be divided into two basic parts the testing of local weights and the testing of global weight. The global weight is generally derived from the sum of the local weights. Since Nutritional Advisory System is based on K Nearest Neighbor model, the weight is in numerical from. But for every attribute the weighting is different. To set the optimum weighting, we need to first define the local weight according to the ordering of important of each attribute. After assigning value for each attribute respectively, we test the weighting by presenting to the system a known case. If the system behaves as we wanted it to be then we can start fine tuning each weight. If the systems don't behave as what we wanted it to be then we have to review the weighting from the scratch starting

from testing each local attribute's weight to the sum of global value. As a matter of fact we can mathematically calculate the weighting for each attribute and then calculate the sum as the global value before going into real time testing.

6.4 Testing the Causal Model

As we all know the system have a causal model to assist the matching of stored cases against the causal. To test the validity of the matching assisted by the causal model we have to present a case that the system do not have the prototype case stored in the database. Then we check the result to see if it is correct or not. If the result is in the range we want then no more amendment should be made to the causal model. If the result is way out of our expectation then we have to rectify the problem and try to correct it.

6.5 Integration Test

After performing all the testing above, the modules are integrated or combined into a working system. During the integration, the testing was carried out in order to identify the fault and failure caused by the integration.

The integration testing includes structure tests and functional tests. Structure tests emphasis is on exercising all input and output parameters of each module, and exercising all modules and all calls, including calls to utility routines. For example, the code blocks for menu bar on each module are integrated into only one procedure that can be called by all the modules. This will eliminate redundant codes and make the coding simpler.

For functional tests, the goal is to demonstrate that all functions specified for the system in the requirements and specification documents are operational.

During the integration, all the modules were combined and tested in a testing environment. The testing environment was consistent for all the modules in terms of interface and function calling procedures. The program flow of the modules were reviewed and identified. Finally the program flow for the entire program were reviewed and tested with some test cases.

CHAPTER 7

SYSTEM EVALUATION & CONCLUSION

7.1 Introduction

System evaluation is implemented by more than simply comparing the information obtained with the information which is expected. It is also related to the user environment, attitudes, information principles and several other matters which must be given consideration before the actual efficacy can be concluded.

At all phases of the system approaches, evaluation is a process that occurs continuously, drawing on a variety of sources and information.

The role of this evaluation phase was to determine:

- The extent to which the expected outcomes have been realized,
- The prescriptive value of the process where extraneous factors were taken consideration.

7.2 Problem Encountered and Solutions

Although this project is completed on time, but within the development of the system, there are a few problems that had encountered and need to be solved to continue the progress. The problems and solutions for each problem are:

➤ ***Unfamiliar with the Development Environment***

This is the first time for me to create a system using ART*Enterprise Studio and using ArtScript 3.1 technology. Never encounter before this kind of programming that similar with LISP so familiarization and experiment with the ART*Enterprise Studio is time consuming.

➤ ***Lack experience In Programming Language***

I have not been using ARTScript technology before, so having a lot of glitches here and there throughout the entire development process. Referencing books is my main source of knowledge about the language.

➤ ***Unfamiliar with Case-Based Reasoning***

I never use case-based reasoning techniques before in my application development so it is very new to me. The problems come from what kind of matching technique to use and how to implement adaptation technique. Is the technique I chosen suitable with the CBR shell provided by Brightware. I manage to go around it by reading a lot of journals paper provided by my supervisor and the Kolodner Case-Based Reasoning's book.

➤ ***Limited Knowledge About Nutritional and Diseases***

The knowledge I have in the medical fields is very limited. I have to do a lot of reading and research in the medical field. I use the University of Malaya's medical library as my main resource of information. But from time to time I will talk to my expert domain, Mr F.E Chong to narrow down the scope of my searching and advised from him on the development of the system from the view of a nutritionist.

➤ **No Graphic User Interface**

This is the biggest headache I faced during the entire development of the system as the CBR system is not linked to the GUI layer via variables setting like those in Visual Basic or Java 2. It has a standard GTK commands to link each form to the application level but not to the internal matching function of the CBR. So the GUI only can link to the application level of the system but not the CBR level as I wanted it to be. I haven found the solution to the problem yet due to the lack of online support as the company- Brightware Inc had been bought over by Firepond Inc and the new company discontinued this product.

7.3 System Strengths

➤ **Fast Retrieval of Cases from the Stored Case Base**

The system has a very comprehensive retrieving algorithm based on the pyramid/hierarchical model. The retrieval is faster than normal heap sort retrieval process. So the system is able to manage large stored case base based on the retrieving ability.

➤ **Comprehensive Attributes based on Real World**

The matching attributes that I use is based on the real world situation and the weighting percentage is being tested by the expert domain. The attributes are taken from the standard medical form of WHO.

➤ **Realistic Matching**

All the matching is based on the AHP matching algorithm [11]. It is a more precise match than normal K nearest neighbor match. All the attributes have different weighted that contribute different percentage to the matching.

➤ **Causal Model Implementation**

This system has a causal model to refer to if no matches were found to be matches more than the score of 60. The system will automatically refer to the causal model for the relational of the disease.

7.4 System Limitations

➤ No Graphic User Interface

The system does not have a proper end-user interface. Cannot bring out the user-friendliness of the system.

➤ No temporary storage system

The system do not allowed a temporary storage of case that has not being certified by the general practitioner or nutritionist. This is because the integration of subsystem is not allowed in this development software.

7.5 Future Enhancement

➤ **Integrate with DBMS**

This system allowed to connect to most major database but is not feasible to do so now due to fact that the number of cases in the case-base storage is still manageable.

➤ **Graphic User Interface**

The ability to have GUI is a major influent of the application to the end-user market. Some research needs to be done to enable the system to integrate with web languages

➤ **Web-enable**

It can be web-enabled to a certain extent such as presenting cases through the web or setup the case through the web.

7.6 Conclusion

The project is considered a success in term of the implementation and research of the matching and searching algorithms but for a real application is still lacked in graphic user interface and help module.

The system was implemented using a rare programming language called ARTScript. This programming language supports object oriented, rule-base and case base programming. ArtScript is a multipurpose scripting language that based on LISP. The major set back of this programming language is that they do not have online support and any technical online communities to facilitate the usage of this software. Also the model of development of this programming language is totally different from the norm e.g. the GUI is totally separated from the repositories and deployment layer and don't support visual programming. Due to this, much of the time is spend on experimenting with the programming tools.

The expert domain plays an important part in development and growing of the system. The expert domain provides medical and nutrition information that are vital to the content of the software. They also played their part in testing and feedback of the system.

The implementation of Case Based Reasoning in nutritional advisory system can be considered a stepping stone of the nutritional world into the Artificial Intelligence community. The attractions of CBR to the nutritionist are warmly supported but much research still has to be done before major rollout of commercial CBR in nutritional advisory. Some of the research areas pinpointed are (my point of views):

1. Categorization of supplements
2. Reaction of drugs toward the effectiveness of nutrition absorption
3. The optimum type of matching algorithm in Nutritional Advisory.
4. Implementation of dynamic case base reasoning in Nutritional advisory.

To end my conclusion Nutritional Advisory System can be implemented in CBR and the prospect of the system is big.

- [1] E.N Whaley, C.B Catalda, S.R Roffex, "Understanding Normal and Clinical Nutrition, West Publishing Co. 1994
- [2] A. Amadi, E. Plaza (1994), AICom - Artificial Intelligence Communications, IOS Press, Vol. 7: 1, pp. 39-45.
- [3] Mair, F., & Watson, I.D. (1996), Case-Based Reasoning: A Categorized Bibliography. The Knowledge Engineering Review, Vol. 9 No. 4, pp.355-381.
- [4] D.B Leake (1996), Case-Based Reasoning: Experiences, Lessons and Future Directions. Menlo Park: AAAI Press/MIT Press, 1996.
- [5] J.Kolodner, Case-Based Reasoning, Morgan Kaufmann Publisher, 1993
- [6] Pressman, S. Roger, Software Engineering: A Practitioner's Approach, 4th Edition McGraw-Hill International Edition, 1997
- [7] S.L. Pflieger, Software Engineering: Theory and Practice, 2nd Edition Prentice Hall, 2001
- [8] <http://www.malaya.edu.my>, Date Retrieved: January 2002.
- [9] ART*Enterprise, ART*Script Programming Guide 2: Rules & Clauses, February 1999 Edition, Brightware Inc., 1999
- [10] Rainer Schmitt, Leake/Case-Based Reasoning for radiologists therapy advice: an investigation in artificial intelligence and medicine. Artificial Intelligence in Medicine 33 (2001) Pages 173-186

Bibliography

- [1] E.N Whitney, C.B Cataldo, S.R Rofles, "Understanding Normal and Clinical Nutrition, West Publishing Co. 1994
- [2] A. Aamodt, E. Plaza (1994); AICom - Artificial Intelligence Communications, IOS Press, Vol. 7: 1, pp. 39-59.
- [3] Marir, F., & Watson, I.D. (1994). Case-Based Reasoning: A Categorized Bibliography. The Knowledge Engineering Review, Vol. 9 No. 4: pp.355-381.
- [4] D.B Leake (1996). Case-Based Reasoning: Experiences, Lessons and Future Directions. Menlo Park: AAAI Press/MIT Press, 1996
- [5] J.Kolodner, Case-Based Reasoning, Morgan Kaufmann Publisher, 1993
- [6] Pressman, S. Roger, Software Engineering: A Practitioner's Approach, 4th Edition McGraw-Hill International Edition, 1997
- [7] S.L. Pfleeger, Software Engineering: Theory and Practice, 2nd Edition Prentice Hall, 2001
- [8] <http://www.microsoft.com>, Date Referred: January 2002.
- [9] ART*Enterprise: ARTScript Programming Guide 2: Rule & CBR; February 1999 Edition, Brightware Inc, 1999
- [10] Rainer Schmidt, Lothar Gierl. Case-based reasoning for antibiotics therapy advice: an investigation of retrieval algorithms and prototypes. Artificial Intelligence in Medicine 23 (2001) Page 171-186

Appendices

- [11] Cheoal-Soo Park, Ingoo Han(2002). A case-based reasoning with the feature weights derived by analytic hierarchy process for bankruptcy prediction. Pergamon Elsevier Science.
- [12] Avelino J. Gonzalez / Lingli Xu / Uma M. Gupta. Validation Techniques For Case-Based Reasoning Systems. IWK 1997 VCBR
- [13] Margaret T. Shannon, Billie Ann Wilson, Carolyn L. Stang, Drugs and Nursing Implications, 8th Edition Appleton & Lange Pub
- [14] Robert S. Goodhart, Maurice E. Shils, Modern Nutrition in Health and Disease Dietotherapy, 5th Edition Lea & Febiger Pub
- [15] <http://www.nhsllothian.scot.nhs.uk/lothianformulary/>
United Kingdom's Site for hospital based nutritional concoction advisory.

Appendices

Source Code

The following code is to load the case base reasoning class to the application. The second line is to add the NutriSys as a sub system to the application.

```
(rep:load "case-based-reasoning")  
(rep:add-subsystem "NutriSys" "case-based-reasoning")
```

The variables are as below. The variables are global and the values are default as some-value. These variables are for the user to insert values into the system.

```
(define-global ?*inputcategory* = some-value)  
(define-global ?*inputdisease* = some-value)  
(define-global ?*inputsubdisease* = some-value)  
(define-global ?*inputmedication* = some-value)
```

The follows are the definition of the CBR instances with the name nas-case-base. The scoring function is pre-define as score-with-both-max. The threshold is being defined as 40 upon 1000 and the maximum match is 5.

```
(define-instance nas-case-base cbr:case-base  
(cbr:scoring-function cbr:score-with-both-max)  
(cbr:match-threshold 40)  
(cbr:max-matches 5))
```

Below are the individual attribute that contained in an object. Most of the attributes are from the word type for easy manipulation.

```
(define-attribute category slot)  
(cbr:define-attribute nas-case-base category cbr:word  
:match-contribution 30 :mismatch-penalty 30 :absence-penalty 5)  
  
(define-attribute disease slot)  
(cbr:define-attribute nas-case-base disease cbr:word  
:match-contribution 100 :mismatch-penalty 0 :absence-penalty 40)  
  
(define-attribute subdisease slot)  
(cbr:define-attribute nas-case-base subdisease cbr:word  
:match-contribution 30 :mismatch-penalty 0 :absence-penalty 0)  
  
(define-attribute medication slot)
```

```
(cbr:define-attribute nas-case-base medication cbr:word
:match-contribution 10 :mismatch-penalty 0 :absence-penalty 5)
```

```
(define-attribute meal slot)
```

```
(cbr:define-attribute nas-case-base meal cbr:word
:match-contribution 0 :mismatch-penalty 0 :absence-penalty 0)
```

```
(define-attribute supplement slot)
```

```
(cbr:define-attribute nas-case-base supplement cbr:word
:match-contribution 0 :mismatch-penalty 0 :absence-penalty 0)
```

```
(define-attribute precaution slot)
```

```
(cbr:define-attribute nas-case-base precaution cbr:word
:match-contribution 0 :mismatch-penalty 0 :absence-penalty 0)
```

The attributes above are put together to form an object or class. The class/object is a children node of bc:core.

```
(define-class nutricase bc:core
```

```
(category)
```

```
(disease)
```

```
(subdisease)
```

```
(medication)
```

```
(meal)
```

```
(supplement)
```

```
(precaution))
```

The below coding is to add new cases to the case base system.

```
(for ?c in-instances-of nutricase do
```

```
(cbr:add-case nas-case-base ?c :ignore-undefined-attributes? t))
```

The below coding is use to let user to present case to the system. The global variables are use here to get data from the user for matching.

```
(define-instance present-case nutricase
```

```
(category ?*inputcategory*)
```

```
(disease ?*inputdisease*)
```

```
(subdisease ?*inputsubdisease*)
```

```
(medication ?*inputmedication*) )
```


The function below is to assist user to input the presented case. And the set-attribute-values is to set the user's input values to the attribute defined in the presented case.

```
(define-function input-case()
  (printout t "Please insert the type of Disease Category:" t)
  (bind ?*inputcategory* (read))
  (printout t "Please insert the type of Disease:" t)
  (bind ?*inputdisease* (read))
  (printout t "Please insert the type of Sub-disease(if any):" t)
  (bind ?*inputsubdisease* (read))
  (printout t "Please insert the type of Medication:" t)
  (bind ?*inputmedication* (read))
  (set-attribute-values present-case
    category ?*inputcategory*
    disease ?*inputdisease*
    subdisease ?*inputsubdisease*
    medication ?*inputmedication*)
  (match-case)
)
```

The case below is to test the system basic function.

```
(define-instance presented-case-2 nutricase
  (category "Renal")
  (disease "Acute Renal Failure")
  (subdisease "no")
  (medication "magnesium hydroxide"))
```

Below functions are used to assist user in terms of the navigation of the application.

```
(define-function start-nas ()
  (printout t "Welcome to Nutritional Advisory System " t)
  (printout t "-----" t)
  (printout t "Menu " t)
  (printout t "*****" t)
  (printout t "1) Setup" t)
  (printout t "2) Present Case " t)
  (printout t "3) Exit System" t)
  (bind ?m (read))
  (if (= ?m 1) then (printout t "Still Under Construction" t)
    else
    (if (= ?m 2) then (input-case)
      else (exit))))
```

Match Case Function

```
(define-function match-case ()  
  (cbr:match-case nas-case-base present-case)  
  (cbr:matches-found nas-case-base)  
  (menu))  
  
(define-function print-matches (?case-base)  
  (for ?m from 1 to (cbr:matches-found ?case-base) do  
    (printout t "Match " ?m ": Case " (cbr:get-match-case ?case-base ?m)  
      " matched with a score of "  
      (cbr:get-match-score ?case-base ?m) t))  
  (query-recommendation))  
  
(print-matches nas-case-base)
```

Display the best case

```
(define-function print-recommendation (?case-base)  
  (printout t "This is the meal and supplements the system recommended" t)  
  (bind ?case (cbr:get-match-case ?case-base 1))  
  (bind ?attr (cbr:first-case-attribute ?case-base ?case))  
  (while ?attr do  
    (bind ?value (cbr:first-attribute-value ?case-base ?case ?attr))  
    (while ?value do  
      (printout t " " ?attr ": " ?value t)  
      (bind ?value (cbr:next-attribute-value ?case-base ?case ?attr ?value))  
    )  
    (bind ?attr (cbr:next-case-attribute ?case-base ?case ?attr))  
  )  
  (query-matches))
```

```
(define-function print-recommendation2 (?case-base)  
  (printout t "This is the meal and supplements the system recommended" t)  
  (bind ?case (cbr:get-match-case ?case-base 2))  
  (bind ?attr (cbr:first-case-attribute ?case-base ?case))  
  (while ?attr do  
    (bind ?value (cbr:first-attribute-value ?case-base ?case ?attr))  
    (while ?value do  
      (printout t " " ?attr ": " ?value t)  
      (bind ?value (cbr:next-attribute-value ?case-base ?case ?attr ?value))  
    )  
    (bind ?attr (cbr:next-case-attribute ?case-base ?case ?attr))  
  )  
  (query-recommendation2))
```



```

(define-function menu ()
  (printout t "Choose Type of Match View" t)
  (printout t "*****" t)
  (printout t "1.) To View the Best Case" t)
  (printout t "2.) To show the top cases with the score" t)
  (printout t "3.) To View the Second System Recommendation" t)
  (bind ?m (read))
  (if (= ?m 1) then (print-recommendation nas-case-base)
    else
    (if (= ?m 2) then (print-matches nas-case-base)
      else
      (print-recommendation2 nas-case-base))))

```

```

(define-function menu1 ()
  (printout t "Choose Type of Match View" t)
  (printout t "*****" t)
  (printout t "1.) To View the Best Case" t)
  (printout t "2.) To show the top cases with the score" t)
  (printout t "3.) To View the Second System Recommendation" t)
  (printout t "4.) Return to Main Menu" t)
  (bind ?m (read))
  (if (= ?m 1) then (print-recommendation nas-case-base)
    else
    (if (= ?m 2) then (print-matches nas-case-base)
      else
      (if(= ?m 3) then (print-recommendation2 nas-case-base)
        else
        (start-nas))))))

```

```

(define-function Query-recommendation ()
  (printout t "=====")
  (printout t "1.) Display the System Recommendation Case" t)
  (printout t "2.) Display the System Second Recommendation" t)
  (printout t "3.) Return to Main Menu" t)
  (bind ?m (read))
  (if (= ?m 1) then (print-recommendation nas-case-base)
    else
    (if(= ?m 2) then (print-recommendation2 nas-case-base)
      else
      (start-nas))))

```

```

(define-function Query-matches ()
(printout t "=====")
(printout t "1.) Display the top matches with the score" t)
(printout t "2.) Display the Second System Recommendation" t)
(printout t "3.) Return to Main Menu" t)
(bind ?m (read))
(if (= ?m 1) then (print-matches nas-case-base)
else
(if (= ?m 2) then (print-recommendation2 nas-case-base)
else
(start-nas))))

```

```

(define-function Query-recommendation2 ()
(printout t "=====")
(printout t "1.) Display the System Recommendation Case" t)
(printout t "2.) Display top matches with the score" t)
(printout t "3.) Return to Main Menu" t)
(bind ?m (read))
(if (= ?m 1) then (print-recommendation nas-case-base)
else
(if (= ?m 2) then (print-matches nas-case-base)
else
(start-nas))))

```


The below are the source code for the hierarchical model.

```
(define-class main-disease bc:core)

(define-class cardiovascular main-disease
  (cbr:common-parent cardiovascular))

(define-class respiratory main-disease
  (cbr:common-parent respiratory))

(define-class gastrointestinal main-disease
  (cbr:common-parent gastrointestinal))

(define-class renal main-disease
  (cbr:common-parent renal))

(define-class infection main-disease
  (cbr:common-parent infection))

(define-class gynaecology main-disease
  (cbr:common-parent gynaecology))

(define-class malignant main-disease
  (cbr:common-parent malignant))

(define-class ENT main-disease
  (cbr:common-parent ENT))

(define-class endocrine main-disease
  (cbr:common-parent endocrine))

(define-class chr :  
  (chr:name  
    (chr:value-type  
      (chr:storage-type  
        (chr:row-attribute  
          (chr:method  
            (call-next-method  
              (if (not (get-attribute-value 'chr:storage-type)  
                  (set-attribute-value 'chr:storage-type  
                    (get-attribute-value 'chr:storage-type  
                      (chr:define-attribute-type 'chr:storage-type
```

The below source code are some of the part that being modified to suit my application.

```
(define-instance cbr:word cbr:word-type  
  (cbr:name "word"))
```

```
(define-instance cbr:direct-word cbr:word-type  
  (cbr:name "direct_word")  
  (cbr:generated-lexicon nil))
```

```
(define-class cbr:number-type cbr:attribute-type  
  (cbr:name "range")  
  (cbr:value-type :float)  
  (cbr:storage-type :integer)  
  (cbr:raw-attribute-value-saved nil))
```

```
(define-instance cbr:range cbr:number-type)
```

```
(define-class cbr:unscored-type cbr:attribute-type  
  (cbr:raw-attribute-value-saved t))
```

```
(define-class cbr:unscored-float cbr:unscored-type  
  (cbr:name "unscored_float")  
  (cbr:value-type :float)  
  (cbr:storage-type :integer))
```

```
(define-class cbr:unscored-string cbr:unscored-type  
  (cbr:name "unscored_string")  
  (cbr:value-type :string)  
  (cbr:storage-type :string))
```

```
(define-class cbr:unscored-symbol cbr:unscored-type  
  (cbr:name "unscored_symbol")  
  (cbr:value-type :string)  
  (cbr:storage-type :string))
```

```
(define-class cbr:user-attribute-type cbr:attribute-type  
  (cbr:name)  
  (cbr:value-type :string) ; Set to determine type.  
  (cbr:storage-type :string) ; Set according to input. Must match.  
  (cbr:raw-attribute-value-saved t)) ; Currently must be t! -da.
```

```
(define-method initialize-instance ((cbr:user-attribute-type ?self) (?pairs :keylist))  
  (call-next-method ?self $?pairs)  
  (if (not (get-attribute-value ?self cbr:name)) then  
    (set-attribute-value ?self cbr:name  
      (symbol-to-string (gentemp (symbol-to-string ?self)))))  
  (cbr::define-attribute-type ?self))
```



```

(add-attribute-value cbr:case-base cbr:attribute-types ?self)
?self)

;; Default method will not work for floats!
(define-method cbr:condition-feature ((cbr:user-attribute-type ?type) ?case-
base ?case ?attribute ?value)
?value)

(define-method cbr:score-value ((cbr:user-attribute-type ?type) ?case-
base ?case ?attribute
?value ?presented-value ?match ?mismatch)
(if (eq ?value ?presented-value)
then ?match
else ?mismatch))

(define-class cbr:case-base bc:core
(cbr:index nil)
(cbr:index-file)
(cbr:scoring-function cbr:score-with-presented-max)
(cbr:naming-attribute :object-name)
(cbr::invert-name nil)
(cbr:match-threshold 0)
(cbr:max-matches 10)
(cbr:attribute-types cbr:string cbr:word cbr:range)
(cbr:raw-attribute-values-saved t)
(cbr:ignored-attributes (is-a instance-of)))

(define-method initialize-instance ((cbr:case-base ?self) (?pairs :keylist))
(call-next-method ?self $?pairs)
;; Ideally this should cause make-instance to fail.
(cbr::open-index ?self)
?self)

(define-method destroy-instance ((cbr:case-base ?self))
(cbr::close-index ?self)
(call-next-method ?self))

(define-method cbr:is-dirty ((cbr:case-base ?self))
(cbr::is-dirty (get-attribute-value ?self cbr:index)))

(define-method cbr:save ((cbr:case-base ?case-base) (?file-name :key))
(if ?file-name then
(modify-schema-value ?case-base cbr:index-file ?file-name
(and (bind ?index (get-schema-value ?case-base cbr:index))
(bc:is-reset-state-value ?case-base cbr:index ?index))))
(cbr::save-index (get-attribute-value ?case-base cbr:index)

```



```

(get-attribute-value ?case-base cbr:index-file)))

(define-method cbr:set-raw-attribute-values-saved ((cbr:case-base ?case-base) ?value)
  (set-attribute-value ?case-base cbr:raw-attribute-values-saved ?value)
  (cbr::set-raw-attribute-values-saved (get-attribute-value ?case-base cbr:index) ?value))

(define-method cbr:set-final-score-smoothed ((cbr:case-base ?case-base) ?value)
  (cbr::set-final-score-smoothed (get-attribute-value ?case-base cbr:index) ?value))

(define-method cbr:get-final-score-smoothed ((cbr:case-base ?case-base))
  (cbr::get-final-score-smoothed (get-attribute-value ?case-base cbr:index)))

(define-method cbr:define-attribute ((cbr:case-base ?case-base)
  ?name ?type
  (?match-contribution
   :key (get-attribute-value ?type cbr:default-match-contribution))
  (?mismatch-penalty
   :key (get-attribute-value ?type cbr:default-mismatch-penalty))
  (?absence-penalty
   :key (get-attribute-value ?type cbr:default-absence-penalty))
  (?lowest-attribute-value :key 0)
  (?highest-attribute-value :key 0)
  (?match-interval :key 0))
  (if (eq ?match-contribution :perfect) then
    (bind ?match-contribution ?cbr:*perfect*))
  (if (eq ?mismatch-penalty :perfect) then
    (bind ?mismatch-penalty ?cbr:*perfect*))
  (cbr::define-attribute (get-attribute-value ?case-base cbr:index)
    ?name (get-attribute-value ?type cbr:name)
    ?match-contribution ?mismatch-penalty
    ?absence-penalty ?lowest-attribute-value ?highest-attribute-value
    ?match-interval))

(define-method cbr:get-attribute-type ((cbr:case-base ?case-base) ?name)
  (cbr::get-attribute-type (get-attribute-value ?case-base cbr:index) ?name))

(define-method cbr:get-attribute-match-contribution ((cbr:case-base ?case-base) ?name)
  (cbr::get-attribute-match-contribution (get-attribute-value ?case-base
cbr:index) ?name))
(define-method cbr:get-attribute-mismatch-penalty ((cbr:case-base ?case-base) ?name)
  (cbr::get-attribute-mismatch-penalty (get-attribute-value ?case-base cbr:index) ?name))
(define-method cbr:get-attribute-absence-penalty ((cbr:case-base ?case-base) ?name)
  (cbr::get-attribute-absence-penalty (get-attribute-value ?case-base cbr:index) ?name))
(define-method cbr:get-attribute-lowest-value ((cbr:case-base ?case-base) ?name)
  (cbr::get-attribute-lowest-value (get-attribute-value ?case-base cbr:index) ?name))
(define-method cbr:get-attribute-highest-value ((cbr:case-base ?case-base) ?name)

```



```

(cbr::get-attribute-highest-value (get-attribute-value ?case-base cbr:index) ?name))
(define-method cbr:get-attribute-match-interval ((cbr:case-base ?case-base) ?name)
  (cbr::get-attribute-match-interval (get-attribute-value ?case-base cbr:index) ?name))

(define-method cbr:undefine-attribute ((cbr:case-base ?case-base) ?name)
  (cbr::undefine-attribute (get-attribute-value ?case-base cbr:index) ?name))

;;; I am not sure how to this correctly.
;;; I need to be able to invert the case-id for all of the accessors.
;;; This is linear with the number of case objects, assuming they all exist
;;; in memory, which is not required.
(define-method cbr:get-case-id ((cbr:case-base ?case-base) ?case)
  (bind ?name-attribute (get-attribute-value ?case-base cbr:naming-attribute))
  (if (eq ?name-attribute :object-name) then
    ?case else
    (get-attribute-value ?case ?name-attribute)))

(define-method cbr:define-case ((cbr:case-base ?case-base) ?case-name)
  (cbr::define-case (get-attribute-value ?case-base cbr:index) ?case-name))

(define-method cbr:undefine-case ((cbr:case-base ?case-base) ?case-name)
  (cbr::undefine-case (get-attribute-value ?case-base cbr:index) ?case-name))

(define-method cbr:add-attribute-value ((cbr:case-base ?case-base)
  ?case ?attribute ?value
  (?match-contribution :key ?cbr:*default-weight*)
  (?mismatch-penalty :key ?cbr:*default-weight*))
  (if (eq ?match-contribution :perfect) then
    (bind ?match-contribution ?cbr:*perfect*))
  (if (eq ?mismatch-penalty :perfect) then
    (bind ?mismatch-penalty ?cbr:*perfect*))
  (cbr::add-attribute-value (get-attribute-value ?case-base cbr:index) ?case
    ?attribute ?value ?match-contribution ?mismatch-penalty))

(define-method cbr:remove-attribute-value ((cbr:case-base ?case-base)
  ?case ?attribute ?value)
  (cbr::remove-attribute-value (get-attribute-value ?case-base cbr:index) ?case
    ?attribute ?value))

(define-method cbr:remove-attribute-values ((cbr:case-base ?case-base)
  ?case ?attribute)
  (cbr::remove-attribute-values (get-attribute-value ?case-base cbr:index)
    ?case ?attribute))

(define-method cbr:modify-attribute-value ((cbr:case-base ?case-base)
  ?case ?attribute ?value

```



```

(?match-contribution :key ?cbr:*default-weight*)
(?mismatch-penalty :key ?cbr:*default-weight*)

(if (eq ?match-contribution :perfect) then
  (bind ?match-contribution ?cbr:*perfect*))
(if (eq ?mismatch-penalty :perfect) then
  (bind ?mismatch-penalty ?cbr:*perfect*))
(cbr::modify-attribute-value (get-attribute-value ?case-base cbr:index) ?case
  ?attribute ?value ?match-contribution ?mismatch-penalty))

(define-method cbr:add-case ((cbr:case-base ?case-base) ?object
  (?ignore-undefined-attributes? :key nil))
  (bind ?case (cbr:get-case-id ?case-base ?object))
  (cbr:define-case ?case-base ?case)
  (if ?ignore-undefined-attributes? then
    (for ?attr in-attributes-of ?object do
      (if (cbr:is-attribute ?case-base ?attr) then
        (for ?value in-attribute-values-of ?object ?attr do
          (cbr:add-attribute-value ?case-base ?case ?attr ?value))))
    else
    (bind ?ignored (get-attribute-value ?case-base cbr:ignored-attributes))
    (for ?attr in-attributes-of ?object do
      (if (not (member$ ?attr ?ignored)) then
        (for ?value in-attribute-values-of ?object ?attr do
          (cbr:add-attribute-value ?case-base ?case ?attr ?value))))))

(define-method cbr:add-presented-attribute-value ((cbr:case-base ?case-base
  ?attribute ?value
  (?match-contribution :key ?cbr:*default-weight*)
  (?mismatch-penalty :key ?cbr:*default-weight*))
  (cbr::add-presented-attribute-value (get-attribute-value ?case-base cbr:index)
    ?attribute ?value ?match-contribution ?mismatch-penalty))

(define-method cbr:remove-presented-attribute-value ((cbr:case-base ?case-base
  ?attribute ?value)
  (cbr::remove-presented-attribute-value (get-attribute-value ?case-base cbr:index)
    ?attribute ?value))

(define-method cbr:remove-presented-attribute-values ((cbr:case-base ?case-base
  ?attribute)
  (cbr::remove-presented-attribute-values (get-attribute-value ?case-base cbr:index)
    ?attribute))

(define-method cbr:modify-presented-attribute-value ((cbr:case-base ?case-base
  ?attribute ?value
  (?match-contribution :key ?cbr:*default-weight*)
  (?mismatch-penalty :key ?cbr:*default-weight*))

```



```

(cbr::modify-presented-attribute-value (get-attribute-value ?case-base cbr:index)
    ?attribute ?value ?match-contribution ?mismatch-penalty))

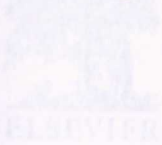
(define-method cbr:clear-presented-attribute-values ((cbr:case-base ?case-base))
  (cbr::clear-presented-attribute-values (get-attribute-value ?case-base cbr:index)))

(define-method cbr:expand-quanta ((cbr:case-base ?case-base) ?value ?expanded)
  (cbr::expand-quanta (get-attribute-value ?case-base cbr:index) ?value ?expanded))

(define-method cbr:match ((cbr:case-base ?case-base)
    (?threshold :key (get-attribute-value ?case-base cbr:match-threshold))
    (?max-matches :key (get-attribute-value ?case-base cbr:max-matches)))
  (cbr::match (get-attribute-value ?case-base cbr:index)
    ?threshold ?max-matches))

(define-method cbr:match-case ((cbr:case-base ?case-base) ?case
    (?keys :keylist)
    (?ignore-undefined-attributes? :key nil))
  (cbr:clear-presented-attribute-values ?case-base)
  (if ?ignore-undefined-attributes? then
    (for ?attr in-attributes-of ?case do
      (if (cbr:is-attribute ?case-base ?attr) then
        (for ?value in-attribute-values-of ?case ?attr do
          (cbr:add-presented-attribute-value ?case-base ?attr ?value))))
    else
      (bind ?ignored-attributes (get-attribute-value ?case-base cbr:ignored-attributes))
      (for ?attr in-attributes-of ?case do
        (if (not (member$ ?attr ?ignored-attributes)) then
          (for ?value in-attribute-values-of ?case ?attr do
            (cbr:add-presented-attribute-value ?case-base ?attr ?value))))))
  (cbr:match ?case-base $?keys))

```



Diabetic patients management exploiting case-based reasoning techniques

Stefania Montani^a, Riccardo Bellizzi^{a,*}, Luigi Portinale^b,
Giuseppe d'Amore^c, Stefano Floridi^c, Mario Stefanel^d

^a *Department of Information & Systems, University of Pavia, via Ferrata 1, I-27100 Pavia, Italy*

^b *Department of Informatics, University of Pavia, viale Beltrami 7, I-27100 Pavia, Italy*

^c *IRCCS Policlinico S. Matteo, P.A. Italy, I-27100 Pavia, Italy*

^d *Received 5 December 1996; received in revised form 3 May 1997; accepted 10 May 1997*

Abstract

In this paper we propose a case-based decision support system designed to help physicians in Insulin therapy revision through the intelligent retrieval of data related to past situations (or 'cases') similar to the current one. A case is defined as a set of variable values for a number of parameters and during a case, we defined taxonomy of prototypical patients' conditions, or classes, to which each case could belong. For each input case, the system allows the physician to find similar past cases, both from the same patient and from other cases. We have implemented a two-steps proceeding: (1) it finds the classes to which the input case could belong; (2) it lists the most similar cases from these classes, through a nearest neighbor algorithm. The system was evaluated using a decision model for decision taking. The performance of the system has been tested on a set of 147 real cases, collected at the Policlinico S. Matteo Hospital of Pavia. The test is fully integrated in a web-based system for the management of Insulin Dependent Diabetes Mellitus (IDDM) project, in 1996. Copyright © 1998 Elsevier B.V. All rights reserved.

Keywords: Case-based reasoning; Diabetes management; Decision support; Expertise

1. Introduction

After the publication of the DCCT study [1], intensive insulin therapy (IIT) has become mandatory for patients suffering from Insulin dependent diabetes mellitus (IDDM). IIT, consisting in three to four insulin injections per day or the use of insulin pumps, is a data and knowledge intensive process, since it requires frequent blood glucose

level (BGL) measurements, and the reporting of insulin injection amounts and other information on patient's diet and lifestyle. Moreover, the quantity and quality of the interactions between the patient and the health care providers is necessarily increased, in order to maintain a tight metabolic control and, at the same time, to improve the patient's self-management and education on the disease. A useful goal of information technology (IT) is, hence, to provide a valuable support to this disease management process [2].

* Corresponding author. Tel.: +390321/261111.
E-mail address: riccardo@isp.unipv.it (R. Bellizzi).

Diabetic patients management exploiting case-based reasoning techniques

Stefania Montani^a, Riccardo Bellazzi^{a,*}, Luigi Portinale^b,
Giuseppe d'Annunzio^c, Stefano Fiocchi^c, Mario Stefanelli^a

^a Dipartimento di Informatica e Sistemistica, Università di Pavia, via Ferrata 1, I-27100 Pavia, Italy

^b Dipartimento di Informatica, Università di Torino, corso Svizzera 185, I-10149 Torino, Italy

^c I.R.C.C.S. Policlinico S. Matteo, P.le. Golgi 2, I-27100 Pavia, Italy

Received 8 December 1998; received in revised form 5 May 1999; accepted 12 May 1999

Abstract

In this paper we propose a case-based decision support tool, designed to help physicians in 1st type diabetes therapy revision through the intelligent retrieval of data related to past situations (or 'cases') similar to the current one. A case is defined as a set of variable values (or features) collected during a visit. We defined taxonomy of prototypical patients' conditions, or classes, to which each case should belong. For each input case, the system allows the physician to find similar past cases, both from the same patient and from different ones. We have implemented a two-steps procedure: (1) it finds the classes to which the input case could belong; (2) it lists the most similar cases from these classes, through a nearest neighbor technique, and provides some statistics useful for decision taking. The performance of the system has been tested on a data-base of 147 real cases, collected at the Policlinico S. Matteo Hospital of Pavia. The tool is fully integrated in the web-based architecture of the EU funded Telematic management of Insulin Dependent Diabetes Mellitus (T-IDDM) project. © 2000 Elsevier Science Ireland Ltd. All rights reserved.

Keywords: Case-based reasoning; Diabetes management; Decision support; Telemedicine

1. Introduction

After the publication of the DCCT study ([1]), intensive insulin therapy (IIT) has become mandatory for patients suffering from 1st type diabetes mellitus (DM-1). IIT, consisting in three to four insulin injections per day or the use of insulin pumps, is a data and knowledge intensive process, since it requires frequent blood glucose

level (BGL) measurements, and the reporting of insulin injection amounts and other information on patients' diet and life-style. Moreover, the quantity and quality of the interactions between the patients and the health care providers is necessarily increased, in order to maintain a tight metabolic control and, at the same time, to improve the patient's self-consciousness and education on the disease. A natural goal of information technologies (IT) is, hence, to provide a valuable support to this disease management process [2].

* Corresponding author. Tel.: +39-0382-505511.

E-mail address: ric@aim.unipv.it (R. Bellazzi).

Nowadays, the data collection is highly facilitated by the capability of commercial reflectometers of data storage and downloading, as well as by the increasing use of telemedicine systems [3]. It can also help diabetologists by providing them with a collection of tools for improving the quality of patient's care [4], from data-bases to simulation and education packages, and finally to decision-support systems. Anyway, managing all the data piling in the physicians' electronic desk, and extracting from them reliable information about the patients' status, often remains a problem difficult to be satisfactorily solved.

For these reasons, it is of interest to study methods that enable physicians in performing an intelligent consultation of the available data-bases, either for supporting their decisions during therapy revision, or for extracting useful information from the accumulated experience. It is also interesting to keep track of the 'problem/solution' patterns that occurred in the past, in order to manage and disseminate the expertise of the diabetologists. In the context of chronic diseases (and hence in particular in diabetes monitoring), all the aspects related to the management and maintenance of knowledge play a crucial role, at least for two different reasons: first, it is necessary to maintain the knowledge about a specific patient over time, so that, even in presence of changes in the physicians staff, the quality of the patient's care is not decreasing due to the lack of information; second, it is useful to manage the 'operative' knowledge of experts, in order to bring in surface their expertise and to keep it in the institution even when they move or retire.

To cope with the above mentioned problems, we have designed a new tool for data-base retrieval and knowledge management, based on the case-based reasoning (CBR) technology [5–7]. CBR is a problem-solving paradigm that utilizes the specific knowledge of previously experienced situations, called cases. Each case is usually described by a set of variable values, called features, and it is associated to a solution (or decision) and to an outcome. CBR basically consists in retrieving past cases that are similar to the current one and in reusing (by, if necessary, adapting) past successful solutions; the current case can be re-

tained and put into the base of cases. CBR can, hence, be viewed as a methodology able to combine retrieval, reasoning and learning steps and to produce solutions to problems by taking into account past experience.

In current medical practice, CBR techniques can be limited to provide diabetologists with a tool to perform an intelligent retrieval of the data-base of past cases, in order to detect, during a periodical control visit, if the same metabolic behavior has already occurred to the same patient or to a similar one, and, in that case, in seeing what decision was taken in the past and what was the resulting outcome. Supporting physicians in this activity may be particularly interesting, considering that they may visit more than 100 patients every month. A crucial capability of the tool will be the possibility of analyzing the overall history related with the retrieved cases, showing the sequence of decision steps that precede and follow the retrieved situation. In addition, it will be possible to calculate from the selected sub-population some statistics that may be interesting for taking decisions, as well as for assessing the quality achieved in the treatment of the sub-population at hand [4].

The CBR tool, we have realized is fully integrated within the architecture of the EU funded Telematic management of Insulin Dependent Diabetes Mellitus (T-IDDM) project, devoted to the implementation of a web-based telemedicine system. In this project, physicians can exploit a medical workstation, that comprises a number of IT services useful for managing this kind of chronic patients, an information system for data handling, a data analysis and visualization system, a decision support system and a communication tool for data exchange with the patients' house; all of this sub-systems are integrated in a web-based service. Details on the T-IDDM project can be found in [8,9].

In this paper, we will describe the fundamental components of the CBR system, with examples and an evaluation of the retrieval tool performed on a set of 147 cases collected at the Department of Pediatrics of the Policlinico S. Matteo Hospital of Pavia, Italy.

2. The case-based reasoning paradigm

CBR is a reasoning paradigm that instead of relying on general rules or models, utilizes the specific knowledge contained into already solved instances of problems [6,7]. In the CBR model, problem-solving experience is explicitly taken into account by storing past solved problems and by suitably 'remembering' them when a new problem has to be tackled. A case is then a structured representation of past problems suitable for re-use.

Generally, a case consists of the following three basic information,

- the problem description, typically a set of <feature, value> pairs in terms of which the problem corresponding to the case may be characterized;
- the case solution representing the solution adopted for solving the corresponding problem;
- the case outcome representing the outcome of the applied solution.

For instance, in a medical domain the problem description may be the set of symptoms of the patient under examination, and of pathophysiological entities providing a compact description of the clinical time course; the solution may be the possible treatment, and the outcome may be the result of the treatment.

The use of CBR plays a significant role in many relevant tasks like diagnostic problem solving [10] or planning [11], since it can mimic (at some extent) the capability of human experts in solving a new case by retrieving similar cases solved in the past and by suitably adapting them to the situation at hand.

The suitability of CBR to solve complex problems has been widely discussed in the last few years and two basic possibilities emerged:

- Precedent CBR, where previous solutions to cases similar to the current one are used as a justification for the solution of the current case with almost no adaptation (e.g. legal reasoning).
- Case-based problem solving, where retrieved solutions to previous similar cases need to be adapted to fit the current situation (e.g. planning, design, diagnosis, etc.).

Case-based problem solving is, of course, the most general approach and can be summarized by the following four basic step known as the CBR cycle or the four 'Res' [7]:

1. RETRIEVE the most similar case(s) from the case library.
2. REUSE the case knowledge (typically the solution) to solve the new problem.
3. REVISE the proposed new solution.
4. RETAIN the relevant parts of this experience (typically the current case) for future problem solving.

The above CBR cycle puts emphasis on several aspects each CBR system has to deal with, namely how to represent cases, how to organize the case library, which kind of algorithm to use for retrieval, how to adapt a retrieved solution and when to add new cases or forget old ones.

In the whole cycle, some steps may be missing or they may be collapsed. For example, it is quite common to view the REUSE and REVISE steps as a single one or to avoid the RETAIN step if the current case is in some sense 'covered' by other cases already stored in the library.

Moreover, very often CBR systems are built essentially as tools for flexible retrieval, leaving to the user all the decisions concerning adaptation and re-use of retrieved solutions. Indeed, even if no capability for automated adaptation is provided by the CBR system, in many applications concerning decision support it is very useful to extract the knowledge concerning relevant past cases for further analysis. This is also the view taken by most commercial CBR tools [12]. As a consequence, devising an efficient retrieval process is fundamental for dealing with large case libraries, as can be the case in a medical setting.

3. Case-based retrieval for DM-I management

As previously noticed, in the context of a periodical visit, CBR may have an important role in decision making, the retrieval of the therapy schemes adopted in the past, and of some indicators of the outcomes obtained on those occasions, could provide a first guideline on how to cope with the current problems. Since, we are inter-

ested in providing useful information to physicians, the data-base of past cases (called case memory) was structured by resorting to a taxonomy of prototypical classes, that express typical problems that may occur to patients. The retrieval process implemented in our tool is, hence, composed by two steps, a classification step, that proposes to the physician the class of cases to which the current case could belong, and a retrieval step, that effectively retrieves the 'closer' past cases. In the following, we will describe in detail the case memory structure and each one of the retrieval steps.

3.1. The case memory

3.1.1. Features

As shown in Section 2, a case is generally defined as a collection of features summarizing the problem, together with a solution and with the outcome obtained by applying the solution itself. More formally, a case C can be viewed as a triple

$$C = \{ \langle F:f \rangle, \langle S:s \rangle, \langle O:o \rangle \}$$

where f being the vector of values for the set of descriptive features F , s the solution schemata selected from the solution space S and o the outcome of the solution selection in the space of the possible outcomes O .

In our application a case just coincides with a periodical visit, whose data, after having been collected and, when needed, discretized, represent the case features of set F . In more detail, we have defined 27 variables (see Table 1), of which 21 are nominal (discrete) and six are linear (continuous), extracted from three sources of information:

- General characterization, 11 features able to generically describe the patient, such as, sex, age, distance from diabetes onset.
- Mid-term information, 13 features actually collected during the visits, like weight and glycosylated hemoglobin (HbA1c) values.
- Short term (day-by-day) information, three features collected during home monitoring activity, i.e. the number of hypoglycemic episodes, the metabolic control and the physical activity. Some features may be automatically obtained as abstractions of the raw data collected during

the visit. For example, control trend and requirement trend of Table 1 are calculated as the variation of HbA1c and requirement, respectively, since the previous visit; metabolic control and hypoglycemias discretize and summarize the information coming from the day-by-day BGL data collection. By now, the day-by-day features of the cases stored in the data-base have been extracted from the patients' log-books, as we were working on retrospective data. A routinely use of the T-1DDM service, permitting the telematic transmission of the monitoring data from the patient's house to the medical workstation, will enable an automatic collection and elaboration of the metabolic control and life style information.

The solution s is represented by an array containing insulin types and doses, decided by the physician from an analysis of the data in the set F . The outcome o of the therapeutic decision is summarized by HbA1c and by the number of hypoglycemic events collected at the following visit.

3.1.2. Classes

As previously mentioned, the case memory was structured through the partitioning induced by a set of mutually exclusive classes. Such classes express the medical knowledge on the prototypical situation that may occur to DM-1 pediatric patients.

More precisely, we have defined a taxonomy, where the roots class (patient's problems) represents the most general class describing all the possible cases we may store into the case memory. Each remaining class in the hierarchy is a prototypical description of the set of situations it summarizes; the class/sub-class link represents the relation of further specialization (see Fig. 2). More precisely, leaf nodes represent the most detailed description of pathologies, or clinical course conditions taken into consideration; an inner node represents a class with certain properties that all the classes of its descending sub-tree have in common. For example, the inner node behavioral-puberal problems has four descendants: change life style; falsifier; no motivation; typical puberal problems.

The first three leaves are possible situations deriving from the patient's behavior, from problems related with a non-reported change in lifestyle to other psychological problems related with young patients, like data falsification or loss of

motivation in following ITT. The fourth class identifies the typical alterations experienced by diabetic patients in puberty.

The prior knowledge associated to the inner node indicates a normal weight and a metabolic

Table 1
The features defining a case

Feature name	Type	Values
<i>Characterization</i>		
Sex	Nominal	[Male, female]
Height	Linear (continuous)	
Age	Linear (continuous)	
Neuropathies	Nominal	[Yes, no]
Other chronic diseases	Nominal	[Unrelated, related to hyperglycemia, related to hypoglycemia, absent]
Puberal stage	Nominal	[Infant, beginning puberal, puberal, adult]
Job	Nominal	[Not-sedentary-worker, sedentary-worker, not-sedentary-student, sedentary-student]
Retinopathies	Nominal	[Yes, no]
Anti insulin antibodies	Nominal	[Yes, no]
Nephropathies	Nominal	[Yes, no]
Distance from onset	Nominal	[Short, long]
<i>Mid-term features</i>		
Weight	Linear (continuous)	
Weight excess	Nominal	[Overweight, underweight, normal]
HbA1c	Linear (continuous)	
Other hormonal disorders	Nominal	[Yes, no]
Requirement trend	Nominal	[Increase, decrease, stationarity]
Control trend	Nominal	[Increase, decrease, stationarity]
Regular insulin	Nominal	[Regular, actrapid]
NPH insulin	Nominal	[Monotard, protaphane, intermediate]
Premixed insulin	Nominal	[Isophace, actraphane]
Premixed ratio	Nominal	[90/10, 80/20, 73/30, 60/40, 50/50]
Number of injections	Linear (continuous)	
Diet	Nominal	[Free, prescribed, controlled]
Requirement	Linear (continuous)	
<i>Short-term features</i>		
Metabolic control	Nominal	[Good, hypoglycemias, hyperglycemias, instable]
Hypoglycemias	Nominal	[None, some, many]
Physical activity	Nominal	[None, intensive-continuous, medium-continuous, light-continuous, intensive-occasional, medium-occasional, light-occasional]

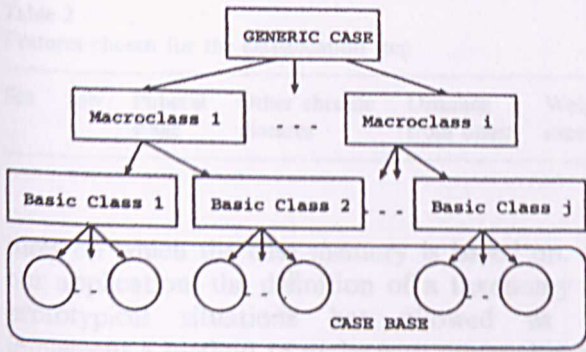


Fig. 1. Case memory organization.

control characterized by hyperglycemia. Each of the leaves is a specialization of the inner node, where additional feature values are specified. For example, for typical puberal problems, the feature

puberal stage has to be puberal or beginning puberal, while this condition is not mandatory for the other three classes.

Leaves in the taxonomy are called basic classes, each case in the case memory is an instance of a unique basic class and can be retrieved through such a class. Classes at the upper level are denoted as macroclasses (see Fig. 1).

Of course, a revision of the taxonomy would be required for representing adult patients' problems, note for example, the presence of the puberal stage class and the absence of cardiovascular complications, that are frequent in adults.

3.2. Classification

Situation assessment and case search are strongly influenced by the organizational struc-

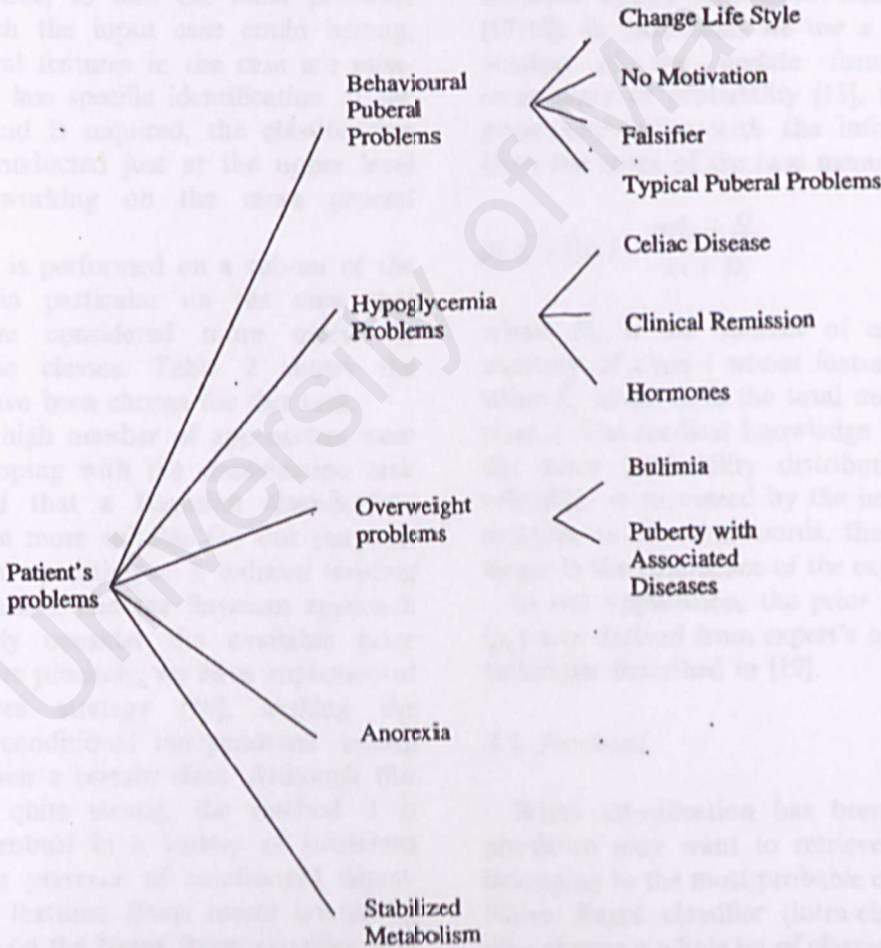


Fig. 2. Taxonomy of classes of prototypical situations that may happen during monitoring of pediatric DM-1 patients.

Table 2
Features chosen for the classification step

Sex	Job	Puberal stage	Other chronic diseases	Distance from onset	Weight excess	Diet	Control trend	Required trend	Metabolic control	Hypos	Physical activity
-----	-----	---------------	------------------------	---------------------	---------------	------	---------------	----------------	-------------------	-------	-------------------

tures on which the case memory is based on. In our application, the definition of a taxonomy of prototypical situations has allowed us to implement a method to make case retrieval more flexible. The first step of the method is classification, the search space for similar cases is limited by identifying what is the context in which the current case should be interpreted, i.e. by finding what are the classes in the hierarchical structure that better represent the case itself. Classification may be performed on the leaves of the taxonomy tree, to find the most probable classes to which the input case could belong; but when several features in the case are missing, or when a less specific identification of the situation at hand is required, the classification step may be conducted just at the upper level of the tree, working on the more general macroclasses.

Classification is performed on a sub-set of the features, and in particular on the ones that physicians have considered more useful to discriminate the classes. Table 2 shows the features that have been chosen for this task.

Although, a high number of approaches were available for coping with the classification task [13], we found that a Bayesian classification method was the more suitable for our purpose. In fact, we were operating on a reduced training data set (147 cases), and the Bayesian approach let us explicitly consider the available prior knowledge. More precisely, we have implemented a Naive Bayes strategy [14], making the hypothesis of conditional independence among the features given a certain class. Although this assumption is quite strong, the method it is known to be robust in a variety of situations [14,15], even in presence of conditional dependencies among features. Some recent interesting research results on the Naive Bayes classifier may be found in [16].

For applying Naive Bayes, we calculate the probability that a case belongs to class c_i , given that the set of its features $f = \{f_1, \dots, f_M\}$ is \hat{f} , through the following formula.

$$P(c_i|f=\hat{f}) \propto \prod_{j=1}^M p(c_i)p(f_j=\hat{f}_j|c_i)$$

The method classifies a case as belonging to the class that maximizes $P(c_i|f=\hat{f})$. The conditional probabilities $p(f_j=\hat{f}_j|c_i)$ are obtained through the Bayesian update formula for discrete distributions [17,18]; in particular, we use a re-parameterized version of the update formula known as m -estimate of probability [13], that modifies the prior knowledge with the information coming from the cases of the case memory as follows.

$$p(f_j=\hat{f}_j|c_i) = \frac{m\hat{p}_{ij} + \hat{N}_{ij}}{m + D_i}$$

where \hat{N}_{ij} is the number of cases in the case memory of class i whose feature f_j assumes the value \hat{f}_j , while D_i is the total number of cases in class i . The medical knowledge is synthesized by the prior probability distribution (\hat{p}_{ij}), whose reliability is expressed by the implicit number of samples m . In other words, the larger is m , the larger is the confidence of the expert on the prior.

In our application, the prior probability value (\hat{p}_{ij}) was derived from expert's opinion through a technique described in [19].

3.3. Retrieval

When classification has been completed, the physician may want to retrieve only past cases belonging to the most probable class found by the Naive Bayes classifier (intra-class retrieval), or may choose a whole set of classes to be considered (inter-class retrieval).

3.3.1. Intra-class retrieval

Looking into the portion of the case memory that store the most probable class, we can apply a nearest neighbor technique for retrieving cases, so that only the closest cases are shown to physicians. The distances of the cases from the current one is calculated by using a distance metric called Euclidean-overlap metric (HEOM) [20]

$$\text{HEOM} = \sqrt{\sum_f d_f(x, y)^2}$$

where $d_f(x, y) = 1$, if x or y are missing; $d_f(x, y) = \text{overlap}(x, y)$, if f is a symbolic feature, (i.e. 0 if $x = y$, 1 otherwise); $|x - y|/\text{range}_f$, if f is a numeric and continuous feature.

3.3.2. Inter-class retrieval

For inter-class retrieval, we have implemented another nearest neighbor technique, again able to cope with missing data, and to take into account both numeric and symbolic features. Such technique is based on the heterogeneous value difference metric (HVDM) [20].

$$\text{HVDM} = \sqrt{\sum_f d_f(x, y)^2}$$

where $d_f(x, y) = 1$, if x or y is missing; $d_f(x, y) = \text{norm}_f(x, y)$ if f is a symbolic variable; $|x - y|/4 \times \sigma_f$, if f is a numeric and continuous variable.

In more detail

$$\text{norm}_f(x, y) = \sum_c \left| \frac{N_{fxc}}{N_{fx}} - \frac{N_{fyc}}{N_{fy}} \right|$$

where N_{fxc} is the number of cases in which $f = x$ in class c , and N_{fx} is the number of cases in which $f = x$ in all the considered classes; the same applies to value y .

This nearest neighbor technique may be computationally inefficient when working with large data-bases. In fact, the complexity of HVDM is known to be $O(FnC)$ [20], where F is the number of features, n the number of cases and C is the number of classes (proportional to the number of cases).

For this reason, we have adapted also a non-exhaustive search procedure, implementing a pivoting strategy (see [21] for details).

The mechanism consists in:

- Finding the median case, i.e. the case with the minimum distance from all the other cases at hand, and computing the distance between the median case and all the other cases.
- Computing the distance between the median case and the input case
- Estimating the distance between the input case and all the remaining cases by using triangle inequality, thus finding a lower and an upper bound for the distance value.
- Applying an iterative procedure that progressively eliminates cases whose interval lower bound is higher than the minimum of all the upper bounds.

At the end of the retrieval step, all the cases belonging to the selected classes are ordered on the basis of their distance from the current case. The interface shows the first ten cases in the list, expected to be the most reliable; anyway, the user is allowed to inspect the remaining ones, and to have a general view of the information stored in the classes he is working on.

In the current implementation, it is also possible to retrieve all past patient's history, so verifying the outcomes of the therapeutic choice on the metabolic control, in both short and long periods. In fact, each case belonging to a certain patient is connected to the previous and to the following one by two chains of pointers. An intra-patient, upper-level retrieval can thus be performed, in order to learn strategies that, starting from a certain condition, led the patient to a target status (for example stabilized metabolism), through a series of class transitions (see Fig. 3). If a similar

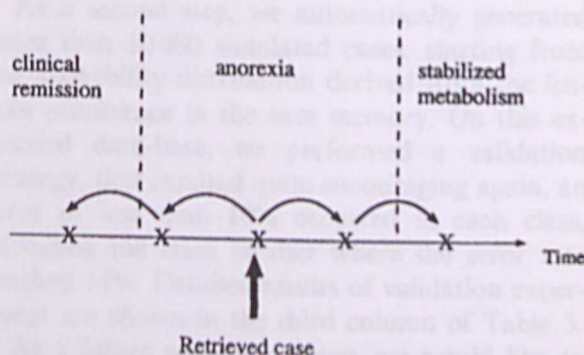


Fig. 3. An example of patient's history with class transitions.

background exists for the current patient, we may assume that therapeutic choices similar to the retrieved ones could lead to similar class transitions, and therefore to a similar conclusion after a certain time.

After the retrieval step, our tool makes some statistics on the basic decisional features of the retrieved cases; for example, it calculates the average variations in the injection number, and the proportion of cases in which the requirement trend has increased, limited to the cases with a positive outcome (i.e. with a reduced number of hypoglycemic episodes, and with a decreasing trend of HbA1c). This information is presented to the physician, who may then decide whether to rely on it, and to apply to the current protocol changes oriented in the average direction of the retrieved ones.

4. Results

Through the collaboration of the endocrinology unit of the Pediatric Clinic of Policlinico S. Matteo Hospital in Pavia, we collected 147 cases, coming from the clinical records of 29 pediatric

Table 3
Cross-validation and validation error rates expressed in terms of ratio of incorrect classified cases and related percentages

Class	Error rate cross-validation on real data	Error rate validation on generated data
Stabilized metabolism	2/20 (10%)	69/1014 (7%)
Insulin resistant	0	0
Clinical remission	0	63/1011 (6%)
No motivation	0	0
Falsifier	4/18 (22%)	147/1009 (14%)
Change life style	12/15 (8%)	89/1011 (9%)
Celiac disease	0	72/1014 (7%)
Hormones	2/9 (22%)	0
Bulimia	0	0
Anorexia	0	0
Puberty with associated diseases	0	29/1003 (3%)
Typical puberal problems	0	98/1005 (10%)

patients. On such data, we have tested the system performances. We are aware that these results are very preliminary, due to the relatively small number of data collected; nevertheless, we believe it is interesting to present this information, as a start point for a future validation on a larger data-base.

The classification step has been tested through a leave-one-patient out cross-validation technique, both on the basic classes and on the macroclasses.

To classify a case c_j belonging to a certain subject s_i , first all cases of patient s_i were removed from the case base; then the case c_j was reinserted and classified. The posterior distributions of the class were calculated with the m -estimate formula applied to the cases belonging to the other patients. In this way, the cross-validation was performed only on cases independent from c_j .

When working on basic classes, a correct classification was obtained for the 83% of the whole case base, while in the 98% the correct class was one of the two most probable classes. A poor outcome was obtained for class falsifier, this is not surprising, since falsifiers are usually young patients that report wrong data on their diaries, to avoid complaints by parents and physicians about their nutritional and life habits. It is therefore, quite difficult, even for physicians themselves, to identify a falsifier, or to list a set of peculiar traits sufficient to describe him or her. Detailed results obtained performing the classification step are presented in the second column of Table 3.

The outcome was slightly improved when working on macroclasses, the correct class was obtained in 84% of the cases, and in 100% of the cases it was in the list of the most probable ones.

As a second step, we automatically generated more than 10 000 simulated cases, starting from the probability distribution derived from the feature occurrence in the case memory. On this expanded data-base, we performed a validation strategy, that resulted quite encouraging again, an error of less than 10% occurred in each class, excluding the class falsifier where the error rate reached 14%. Detailed results of validation experiment are shown in the third column of Table 3.

As a future verification step, we would like to conduct a prospective validation on real patients'

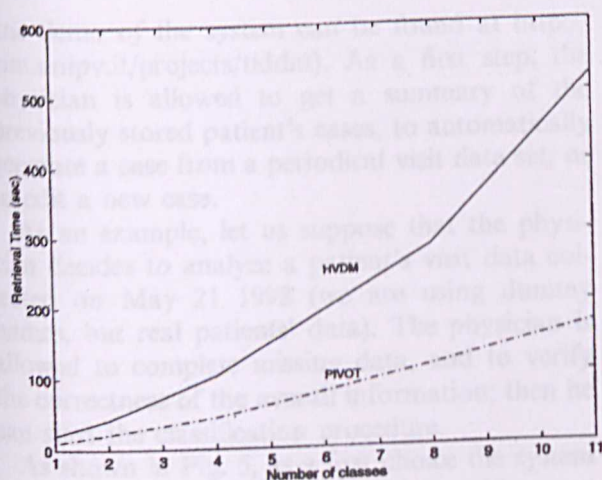


Fig. 4. The retrieval performances (in s) as a function of the number of classes.

cases, to compare the Bayesian classifier suggestions with a physician's opinion.

We tested the computational efficiency of the retrieval step on a Sun Sparc 10 machine; the classification time was of the order of milliseconds, while retrieval with the HEOM formula took 2 s in the 147 cases data-base, and up to 23 s in the 10 000 cases data-base.

Retrieval time with the HVDM algorithm ranged from 5 (on about 20 cases), to more than 500 s on the whole 10 000 cases data-base. The computation time with the pivoting algorithm grew linearly with the number of cases in the search space. In this situation retrieval time ranged from 3 (on about ten cases) to 170 s on the entire 10 000 cases data-base (see Fig. 4). Hence, by resorting to the most suitable algorithm, the system has a good performance (with respect to the application requirements) even in the presence of large data-bases.

5. Implementation

5.1. Details and integration in the T-IDDM architecture

The case-based retrieval tool described in the previous sections is one of the decision support tools of the medical workstation developed within

the T-IDDM project. From an implementation point of view, it is fully integrated in a distributed, web-based environment, managed by *lispweb*, an extended, special-purpose web server, written in common lisp, that makes it possible to create more 'intelligent' and 'secure' applications while remaining in the context of web-based systems [9].

The components of the medical workstation application are:

- a knowledge base, composed of a structured description of the ontology of the domain under consideration;
- a relational data-base that collects all data as instances of the concepts defined at the ontological level;
- a data analysis tool, able to extract the patients' status from the collected data;
- a set of decision support tools (among which the CBR tool), that work on the objects contained in the knowledge base and on the analysis made on the data residing in the data-base;
- a user interface, providing data visualization and knowledge acquisition functionality;
- a telecommunication system, able to connect the medical workstation to the patients' houses.

In particular, the interaction between the CBR reasoning tool and the user (definition of a new case, classification and retrieval) takes place through a set of HTML pages, containing dynamically generated information, such as multicolumn tables and forms. All the cases are stored in an Oracle™ data-base, whose table structure mirrors the classes taxonomy; each leaf of the taxonomy tree matches a table, whose columns corresponds to the case features, and whose rows are instances of the class at hand.

The CBR tool does not require any additional effort from the user. Then, when the tool is invoked, it exploits all the knowledge saved in the information system, and provides the physician with a set of retrieved cases and with some statistics on their main features.

5.2. The system at work: an example of decision support

The CBR tool described in this paper is accessible through the T-IDDM user interface (an on-

line demo of the system can be found at <http://aim.unipv.it/projects/tiddm>). As a first step, the physician is allowed to get a summary of the previously stored patient's cases, to automatically generate a case from a periodical visit data set, or to edit a new case.

As an example, let us suppose that the physician decides to analyze a patient's visit data collected on May 21 1998 (we are using dummy names, but real patients' data). The physician is allowed to complete missing data, and to verify the correctness of the overall information; then he can start the classification procedure.

As shown in Fig. 5, as a first choice the system suggests Jane Doe to be a patient with some hormonal disorders (hypothyroidism); other probable alternatives are puberty with additional related diseases and data falsification.

By performing inter-class retrieval on the three most probable classes (hormones, puberty with

associated diseases and falsifier), the system can rely on 30 cases. Nine of them are positive cases (due to the reduction of HbA1c at the following visit), and they are taken from the history of five different patients. Table 4 provides some statistical analysis on such cases.

If the physician concentrates only on the cases from the most probable class (i.e. hormones), he gets the list displayed in Fig. 6.

Table 5 shows the results of the statistical analysis performed on these nine retrieved cases, belonging to three different patients (patient 21, patient 40 and patient 44). Due to missing data about the outcome of cases number 6, 8 and 9, the system can just work on six cases, in three of which the selected therapy was not effective, because HbA1c did not decrease.

Cases number 2, 4 and 5 show a positive outcome (decreasing value of HbA1c after the application of the solution therapy, without an increase

Patient Case
Doe Jane

Suggested Classification (check the box to choose classes)
Choose one or more class from the table below:

Choose	Suggested Class	Normalized Score
<input type="checkbox"/>	HORMONES	1.0
<input type="checkbox"/>	PUBERTY WITH ASSOCIATED DISEASES	0.73193866
<input type="checkbox"/>	FALSIFIER	0.28974235
<input type="checkbox"/>	STABILIZED METABOLISM	0.096093564
<input type="checkbox"/>	TYPICAL TUBERAL PROBLEMS	0.011785691
<input type="checkbox"/>	CHANGE LIFE STYLE	4.01937e-4
<input type="checkbox"/>	NO MOTIVATION	1.128855e-4
<input type="checkbox"/>	CELIAC DISEASE	2.933261e-5

Accept Cancel Help

Fig. 5. Output of the classification step on the basic classes.

Table 4
Statistical analysis on the positive cases retrieved from classes hormones, puberty with associated diseases, and falsifier

Class	Positive case	Number of patients	Required trend	HbA1c	Hypos	Metabolic control
Hormones	3	1	2 Inc., 1 dec.	3 Inc.	3 Some	3 Instable
Puberty with A.D.	1	1	1 Inc.	1 Inc.	1 Lots	1 Instable
Falsifier	5	3	3 Inc., 1 stat., 1 dec.	1 Inc., 2 stat., 1 dec.	4 Some, 1 none	4 Instabel, 1 hyper

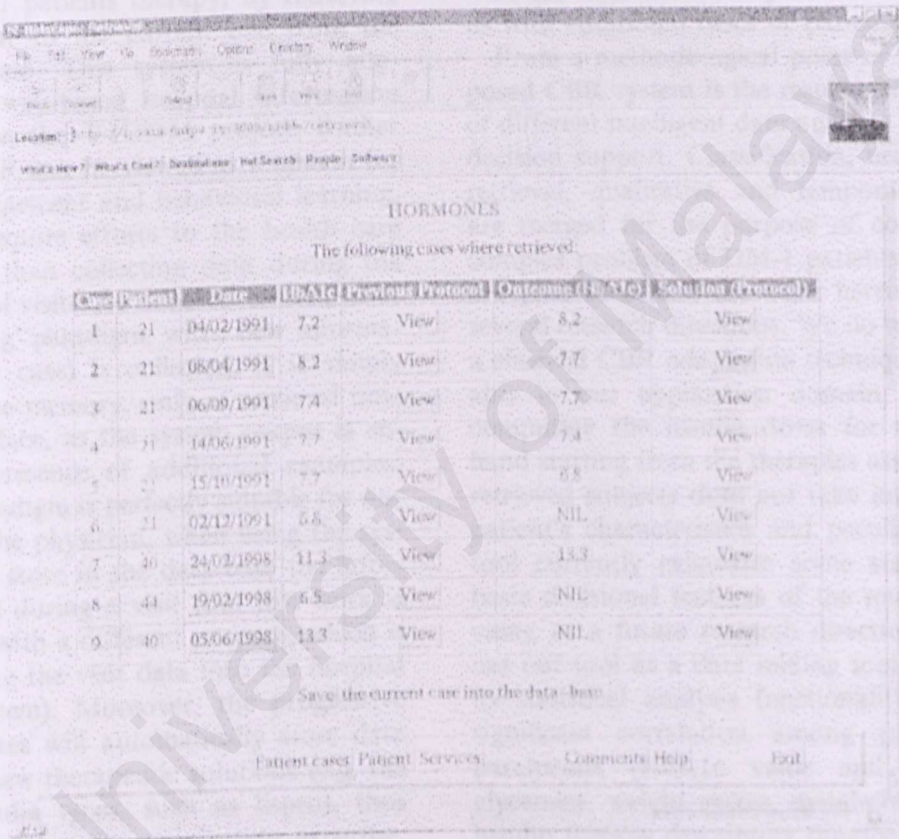


Fig. 6. Output of the intra-retrieval step.

Table 5
Detailed statistics on the cases retrieved from class hormones

Patient id	Sex	Age	Positive outcome cases	Negative outcome cases	Unreliable cases
21	Female	17	3	2	1
40	Female	16	0	1	1
44	Female	19	0	0	1

in the number of hypoglycemias). Table 6 summarizes some statistics on the three cases' features. From them the physician can observe a progressive increase in the value of insulin requirement, and may examine the current case to decide if a similar action could be suitable as well.

6. Conclusions and future research directions

In this paper, we proposed a case-based retrieval tool, able to support physicians during the revision of DM-1 patients therapy, by retrieving past cases similar to the current one from the available data-base. This system is fully integrated with the web-based hospital information system defined in the T-IDDM project. Rather interestingly, CBR can be viewed as a system for knowledge management and behavioral learning, that does not require efforts to the health care providers other than collecting data during the periodical control visits. As a matter of fact, CBR is a 'lazy learning' paradigm, when new information (i.e. a new case) is collected, it is simply stored in the case memory, only at retrieval time learning takes place, as the system output is enriched by the presence of additional examples. This kind of paradigm is perfectly suitable for our application, as the physician, when using the system, just has to store in the data-base the information collected during a visit (but this work is routinely done, with a different purpose, which is the one of saving the visit data into the hospital information system). Moreover, the progressive collection of cases will automatically store data and results on new therapeutic solutions (e.g. the use of new insulin types, such as lispro), thus enriching the health care organization expertise,

without requiring an explicit revision of the knowledge base.

We expect to obtain the first results about the clinical utility of the above presented tool through the evaluation studies that are currently carried on within the T-IDDM project. In particular the T-IDDM verification phase is taking place at the Policlinico S. Matteo, involving ten pediatric patients and three physicians, who will be using the system prototype during the next months. Patients and physicians from the other European medical centers of T-IDDM consortium will soon start with the demonstration phase as well, providing us with additional cases or patients.

From a methodological point of view, the proposed CBR system is the result of the integration of different intelligent data analysis techniques for decision support. Classification, nearest neighbor retrieval, qualitative and temporal abstractions are merged for the purpose of coping with the complex problem of DM-1 patients management. We plan to extend the work herein presented in several research directions. We do not believe that a classical CBR adaptation technique will be suitable in our application domain, automatically computing the insulin doses for the patient at hand starting from the therapies assigned to other retrieved subjects does not take into account the patient's characteristics and peculiar needs. Our tool currently calculates some statistics on the basic decisional features of the retrieved positive cases; as a future research direction, we plan to use our tool as a data mining tool, by extending its statistical analysis functionality, looking for significant correlation among patient's status parameters (HbA1c value and trend, hypoglycemias, weight excess, metabolic control) and insulin therapy descriptors (number of injections,

Table 6
Statistics on the reliable cases in class hormones^a

Patient	Date	HbA1c	HbA1c trend	Required	Required trend	Metabolic control	Hypos	Injection number
21	08/04/1991	8.2	Increase	1.09	Decrease	Instable	Some	3
21	14/06/1998	7.7	Increase	1.1	Increase	Instable	Some	2
21	15/10/1998	7.7	Increase	1.11	Increase	Instable	Some	2

^a Trends are calculated with respect to the previous visit.

requirement value and trend, single dose values and dose distributions over the day). The results will be analyzed to discover if it is possible to learn general strategies, typical of the different macro-classes or basic classes. In this case, classification would already provide an indication on the therapy adjustment directions to be applied in the current case. Moreover, additional strategies could be dynamically inferred from the retrieved cases, even if they do not belong to a single class. In this situation, instead of relying on precompiled indications, we would exploit the similarities between the current case and the retrieved ones; the presence in the case memory of additional cases would enhance the reliability of the results provided at this step. Finally, the most ambitious goal will be to integrate the CBR tool with other existing decision support tools, and in particular with a rule-based system, already implemented in the T-IDDM medical workstation. The search for general strategies provided by the CBR classification and retrieval would define a context in which the current patient's data should be interpreted, and would therefore reduce the search space of the rule-based system through the definition of a set of context-based meta-rules. The rule-based system would then provide a proper solution for the case at hand, by generating a personalized protocol, instead of just listing a series of different protocols, generically suitable for coping the patient's problems, as it currently does.

Acknowledgements

This paper is part of the EU TAP project T-IDDM HC 1047.

References

- [1] The Diabetes Control and Complication Trial Research Group, The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus, *New Engl. J. Med.* 329 (1993) 977–986.
- [2] E.D. Lehmann, Application of computers in clinical diabetes care, *Diab. Nutr. Metab.* 10 (1997) 45–59.
- [3] A.M. Albisser, R.I. Harris, S. Sakkal, E. Pemberton, A. Rieflin, J.L. Nealon, Diabetes intervention in the information age, *Med. Inform.* 21 (1996) 297–316.
- [4] K.R. Piwernetz, et al., Monitoring the targets of the Saint Vincent declaration and the implementation of quality management in diabetes care: the Diabcare initiative, *Diabetic Med.* 10 (1993) 371–377.
- [5] D.B. Leake, J.L. Kolodner, A tutorial introduction to CBR, in: *Case Based Reasoning: Experiences, Lessons and Future Directions*, AAAI Press, 1996, pp. 31–65.
- [6] J.L. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann, Los Altos, CA, 1993.
- [7] A. Aamodt, E. Plaza, Case-based reasoning: foundational issues, methodological variations and systems approaches, *AI. Commun.* 7 (1994) 39–59.
- [8] R. Bellazzi, C. Cobelli, E. Gomez, M. Stefanelli, The T-IDDM Project: Telematic management of insulin dependent diabetes mellitus, in: M. Bracale, F. Denoth (Eds.), *Health Telematics'95*, 1995, pp. 271–276.
- [9] A. Riva, R. Bellazzi, M. Stefanelli, A web-based system for the intelligent management of diabetic patients, *MD. Computing* 14 (1997) 360–364.
- [10] J. Kolodner, R. Kolodner, Using experience in clinical problem solving: introduction and framework, *IEEE Trans. Syst. Man Cybernetics* 17 (1987) 420–431.
- [11] K.J. Hammond, *Case-Based Planning: Viewing Planning as a Memory Task*, Academic Press, New York, 1989.
- [12] I. Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann, Los Altos, CA, 1997.
- [13] T. Mitchell, *Machine Learning*, Mc Graw Hill, New York, 1997.
- [14] I. Kononenko, Inductive and Bayesian learning in medical diagnosis, *Appl. Artif. Intell.* 7 (1993) 317–337.
- [15] I. Zelic, I. Kononenko, N. Lavrač, V. Vuga, Induction of decision trees and Bayesian classification applied to diagnosis of sport injuries, in: *Proceedings of IDAMAP 97 workshop*, IJCAI 97, Nagoya, Japan, 1997, pp. 61–67.
- [16] S. Monti, G. Cooper, The impact of modeling the dependencies among patient findings on classification accuracy and calibration, *JAMIA, Symp. Suppl.* 1998, 592–596.
- [17] D. Spiegelhalter, A. Dawid, S. Lauritzen, R. Cowell, Bayesian analysis in expert systems, *Stat. Sci.* 8 (1993) 219–283.
- [18] A. Riva, R. Bellazzi, Learning temporal probabilistic causal models from longitudinal data, *Artif. Intell. Med.* 8 (1996) 217–234.
- [19] S. Montani, R. Bellazzi, L. Portinale, S. Fiocchi, M. Stefanelli, A case-based retrieval system for diabetic patients therapy, in: *Proceedings of IDAMAP 98 workshop*, ECAI 98, Brighton, UK, 1998, pp. 64–70.
- [20] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, *J. Artif. Intell. Res.* 6 (1997) 1–34.
- [21] L. Portinale, P. Torasso, D. Magro, Selecting most adaptable diagnostic solutions through pivoting-based retrieval, in: *Lecture Notes in Artificial Intelligence*, vol. 1266, Springer, Berlin, 1997, pp. 277–288.

Risk Analysis for Electronic Commerce Using Case-Based Reasoning

Changduk Jung, Ingoo Han* and Bomil Suh

Korea Advanced Institute of Science and Technology, Seoul, Korea

ABSTRACT Electronic commerce (EC) appears to be essential for an organization's survival and growth. Then the security of the EC systems, which ensures authorized and correct transaction processing, becomes one of the most critical issues in implementing the systems. The analysis of risk that a system faces is the core part of security management since risk analysis can identify the principal assets, the threats and the vulnerabilities of those assets, and the risks confronting the assets. This study intends to develop a risk analysis system in an EC environment using the case-based reasoning (CBR) technique. The process of the proposed system is composed of four steps: initial data collection, asset evaluation, threat and vulnerability evaluation, and result generation of risk analysis. This process follows the traditional risk analysis process. This system employs the casebase of past analyses and security accidents. Although some studies introduced several case-based systems for risk analysis of traditional information system, none of them is under an EC environment. The proposed system is the first to apply the CBR technique for risk analysis of an EC system. Copyright © 1999 John Wiley & Sons, Ltd.

INTRODUCTION

Electronic commerce (EC) appears to be essential for an organization's survival and growth. In an EC environment, the internal systems and processes of an organization are no longer operated in isolation from one another. Linked together, the organization exchanges information and transactions in such ways as unanticipated in the traditional environment. As a result, the security of EC systems, which ensures that the systems do not allow unauthorized transactions and that trading partners

have received and processed transactions correctly, becomes one of the most critical issues in implementing the systems.

The traditional approach to information system security was based on the assumption that the security be applied to mainframe computers. However, the new approach to security is strongly required in the fast-changing environment with different needs for the connectivity to the network.

Risk analysis is the process to examine the threats facing the information technology (IT) assets and the vulnerabilities of these assets and to show the likelihood of the threat to be realized. Based on this examination, inspection of the risk related to those recorded assets is performed. Risk analysis enables an organization to appreciate the importance of the value of IT assets to be secured and to find the security holes in a cost-effective manner.

*Correspondence to: Professor Ingoo Han, Graduate School of Management, Korea Advanced Institute of Science and Technology, 207-43 Cheongryangri-Dong, Dongdaemun-Gu, Seoul, 130-012, Korea. E-mail: ingoohan@msd.kaist.ac.kr

However, the tasks of risk analysis for IT require 'deep knowledge' and include various decision problems. This is the reason that has caused risk analysis to be viewed negatively. The case-based reasoning (CBR) technique is good for the risk analysis process because it is useful for tasks that are experience-intensive, that lead to inconsistent outcomes, that have incomplete rules to apply, and that are hard to acquire domain experience.

This study intends to develop a risk analysis system in an EC environment using the CBR technique. The next section introduces the basic concepts of risk analysis in an EC environment and the following section describes the application of the CBR technique to risk analysis. The architecture and the implementation details of the proposed system are shown in the fourth and fifth sections. The final section explains the pros and cons of this study and further research directions.

RISK ANALYSIS OF EC

Risk Analysis

Kailay and Jarratt (1995) stated that the risk is the potential for damage to a system or associated assets that exists as the result of the combination of a security threat and a vulnerability. The risk is the combination of threats, vulnerability and asset value. The risk model is presented in Figure 1. The term vulnerability is a weakness in the security system that might be exploited to cause loss or harm (Pfleeger,

1989). Threats are defined as the sources or circumstances that have the potential to cause loss or harm (Kailay and Jarratt, 1995; Pfleeger, 1989). Risk analysis is a systematic process to examine the threats facing the IT assets and the vulnerabilities of these assets and to show the likelihood that these threats will be realized. Generally, the risk analysis process is composed of three steps (Cerullo and Shelton, 1981; Loch *et al.*, 1992; Rainer *et al.*, 1991).

Risk analysis begins with the identification of IT assets. However, not all the assets require protection, therefore the boundary of the review should be established during asset identification. After the boundary is specified, the overall worth of the identified assets should be assessed.

The next step is to identify all possible threats to the identified assets and to note vulnerabilities. As with the IT assets, all the threats will not necessarily be realized for each identified asset. Only those threats that are likely to occur in any given organization need be identified. The identified threats are assessed as the likelihood of occurrences in accordance with the related vulnerabilities. The final step is the analysis of the risk in the current IT. The impact of the threats is analyzed in this step. This assessment should take into account the asset value within the review boundary and the identified threats and vulnerabilities. The assessed impact leads to risk measures.

Various risk analysis methodologies used currently are categorized into quantitative and qualitative. The quantitative methodologies usually calculate the impact and frequency of threats mathematically. Most quantitative risk analysis methodologies regard the loss exposure as a function of the vulnerability of an asset to a threat multiplied by the probability of the threat becoming a reality. These methodologies include Annualized Loss Expectancy (ALE), the Courtney method, the Livermore Risk Analysis Methodology (LRAM), the stochastic dominance method, ALE using PERT, the simulation technique, etc. (Cerullo and Shelton, 1981; Rainer *et al.*, 1991).

The qualitative methodologies are, on the other hand, based on the assumption that a certain threat or loss cannot be appropriately

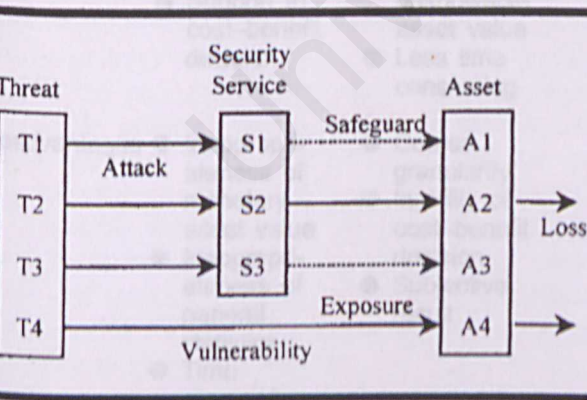


Figure 1 Risk model

expressed in dollar amounts or discrete events so precise information may be unobtainable. The qualitative methodologies, therefore, attempt to express the risk in terms of descriptive variables rather than in precise dollar terms. These methodologies include the Delphi technique, the scenario analysis method, the fuzzy metrics approach, the comparison risk ranking method, and the questionnaire approach (Rainer *et al.*, 1991).

One set of methodologies stated above cannot completely dominate the other. Table 1 summarizes the advantages and the disadvantages of each set of methodologies.

Risk Analysis in EC

EC based mainly on Internet technology is changing the way that people access and purchase information, communicate with each other, and acquire and pay for goods. Internet provides a new way to establish the computer-based resources that can be accessed by consumers as well as business partners around the world.

To facilitate and encourage EC, it will be

Table 1 Advantages and disadvantages of two sets of methodologies (Suh and Han, 1997)

	Quantitative methodologies	Qualitative methodologies
Advantages	<ul style="list-style-type: none"> ● Applicability to all assets ● Mathematical foundation ● Support to cost-benefit decision 	<ul style="list-style-type: none"> ● Simple risk calculation ● Usability to the irrelevant or unknowable asset value ● Less time consuming
Disadvantages	<ul style="list-style-type: none"> ● Inappropriateness of monetary asset value ● Inappropriateness of general statistics ● Time consuming 	<ul style="list-style-type: none"> ● Coarse granularity ● Inability of cost-benefit decision ● Subjective result

necessary to assure that the information on the network is safe and can be accessed only by an authorized recipient. To meet these needs, the Secure Electronic Transaction (SET) addresses seven major business requirements (Yang, 1996):

- Provide confidentiality of payment information and enable confidentiality of order information that is transmitted along with the payment information
- Ensure the integrity of all transmitted data
- Provide authentication that a cardholder is a legitimate user of a branded payment card account
- Provide authentication that a merchant can accept branded payment card transactions through its relationship with an acquiring financial institution
- Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an EC transaction
- Create a protocol that neither depends on transport security mechanisms nor prevents their use
- Facilitate and encourage interoperability among software and network providers.

The important assets to be secured in EC include hardware, software, network and data assets. The major hardware assets include EC servers, EC client systems, point of sale (POS) equipment and other specialized equipment that enable the implementation of EC. The major software assets are EC-enabled 'core process' applications, EDI translation software, mail interface software, operating systems and smartbots, etc. The major elements of network assets are router, firewall and protocols such as TCP/IP and HTTP. The sensitive data assets include user names, passwords, and credit card numbers, etc.

The critical threats to the security in EC such as network failure, internet viruses, and network wiretapping are composed of external and internal attackers against an organization. External attackers exploit the vulnerabilities to the components of EC. According to the 1997 *Computer Crime and Security Survey* of the Computer Security Institute, more than 80% of respondents perceive disgruntled employees as

a likely source of attack and more than 50% also consider corporate competitors a likely source.

CASE-BASED REASONING (CBR)

Basic Concepts of CBR

A case is a prior experience and, therefore, is situation-specific and domain-dependent. A casebase is the collection of cases (Brown and Gupta, 1994). A casebase is to a CBR system as a knowledge base is to a rule-based system. The CBR technique is one of the major artificial intelligence (AI) methodologies and is mostly applied to the problem-solving and learning area.

The fundamental principle of the CBR technique is similar to that of the human reasoning process. Humans use analogical reasoning in complex situations, which employs solutions to past problems to solve current ones. While humans use analogical reasoning, the limitation of the human brain does not take all past cases into consideration. As the number of cases increases, humans seem to use cases most recently solved or that seem most important. However, the CBR system can overcome this limitation and use all past cases in its reasoning, potentially making more effective decision. It can use successful cases to solve current problems or failed cases to adjust solutions to them.

The CBR process is generally composed of three stages: remembering, applying, and learning (Brown and Gupta, 1994). Remembering is the case-retrieval process, which locates and retrieves relevant and useful past cases. Applying is the process of case usage. In this stage, the CBR system applies the cases that have been retrieved to find an effective solution to the current problem. Learning is the process of casebase enhancement. At the end of each problem-solving session, incorporating new case and problem-solving experiences increases the casebase. The CBR process applied to risk analysis is shown in Figure 2.

When the CBR system is presented with a new problem, it selects past cases that are simi-

lar to the current problem and proposes a solution based on solutions to the selected past cases. Once the system solution is evaluated, the evaluation results are reported to the system. The system updates its casebase by capturing and storing important lessons learned during the problem-solving process.

CBR Approach in Risk Analysis

Despite the significance of performing risk analysis, there are several factors that cause this process to be viewed negatively. Most risk analysis processes use questionnaires for gathering asset, threat, and vulnerability data. These questionnaires are often very long and difficult to answer, making risk analysis time consuming and very costly. Moreover, the traditional process of risk analysis is not able to handle the contingency of organization, such as organizational culture, characteristics of IS, and characteristics of organization. Generally, the tasks of risk analysis for an information system require 'deep knowledge' and include various difficult problems.

Risk analyzers have been found to reason by analogy using prior experiences rather than by IF-THEN rules. This propensity of risk analyzers is same as the reasoning process of case-based reasoning. The CBR system solves new problems by recalling and adapting previous solutions. So, CBR is useful for tasks using predicates that are ill defined, that lead to inconsistent outcomes, and which have incomplete rules to apply (Morris, 1994). Moreover, CBR is good for the tasks that are experience-intensive and hard to acquire domain experience such as risk analysis.

This study proposes a CBR system designed to evaluate the risk of an organization in an EC environment. The system integrates the structured procedures of risk analysis and reasoning capability of CBR, and, it is hoped, will aid the decisions of risk analyzers.

Morris (1994) was the first to apply the CBR technique in the IT audit area. He developed a SCAN system using the CBR technique to evaluate IT controls as a part of the internal auditing process. The evaluation of internal controls is a context-sensitive, subjective, and

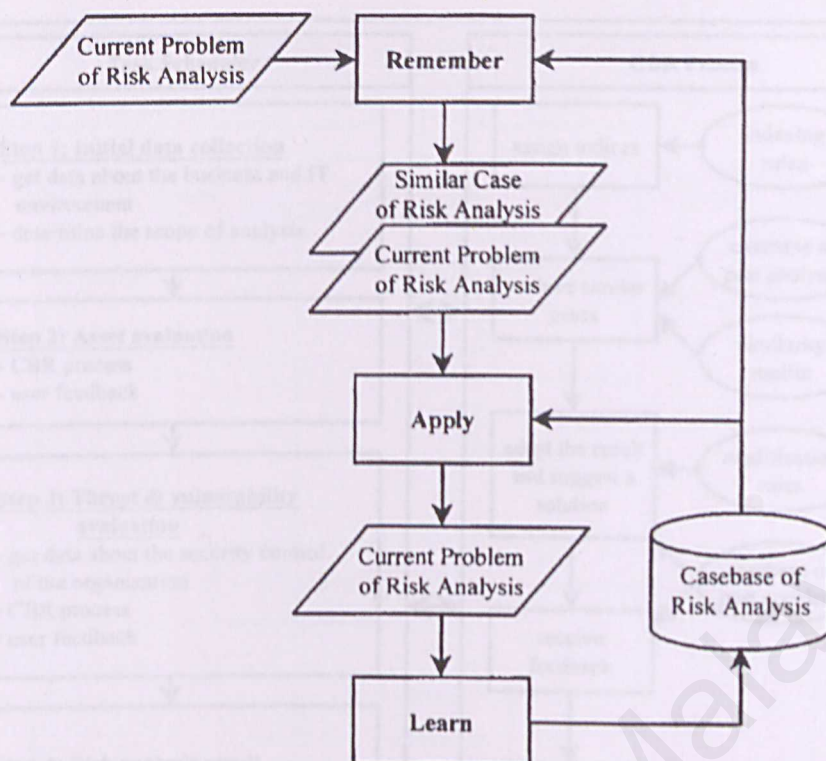


Figure 2 CBR process applied to risk analysis

non-deterministic task that requires considerable professional judgment. These characteristics make the CBR technique suitable for internal control evaluation.

The SCAN system was based on the previous control failure cases to construct the casebase. This system made an auditor recall the previous control failure and provided the auditor with a pattern of successful controls compared with the current controls. The SCAN system took what inexperienced auditors collected as input data and produced a report similar to that of the practicing auditors. Because of this feature, the users of this system were not required to interpret or preprocess the input data or to provide reliability or probability estimates. That is, the SCAN system was helpful to inexperienced auditors by providing a set of control alternatives.

SYSTEM ARCHITECTURE

Process of the System

The proposed system in this study has three sub-goals, which are asset analysis, threat analysis and vulnerability analysis. The process is composed of four steps as shown in Figure 3. First, the system collects data about the business and IT environment of an organization by asking questions. As the data are collected, the first sub-goal, 'asset analysis', is posted to the task scheduler, which keeps track of the current sub-goals of the system. If the memory provides a relevant case at this point, the system focuses on the analysis of assets of the previous cases to see whether anything can be adopted from it.

When the first sub-goal is achieved, it may

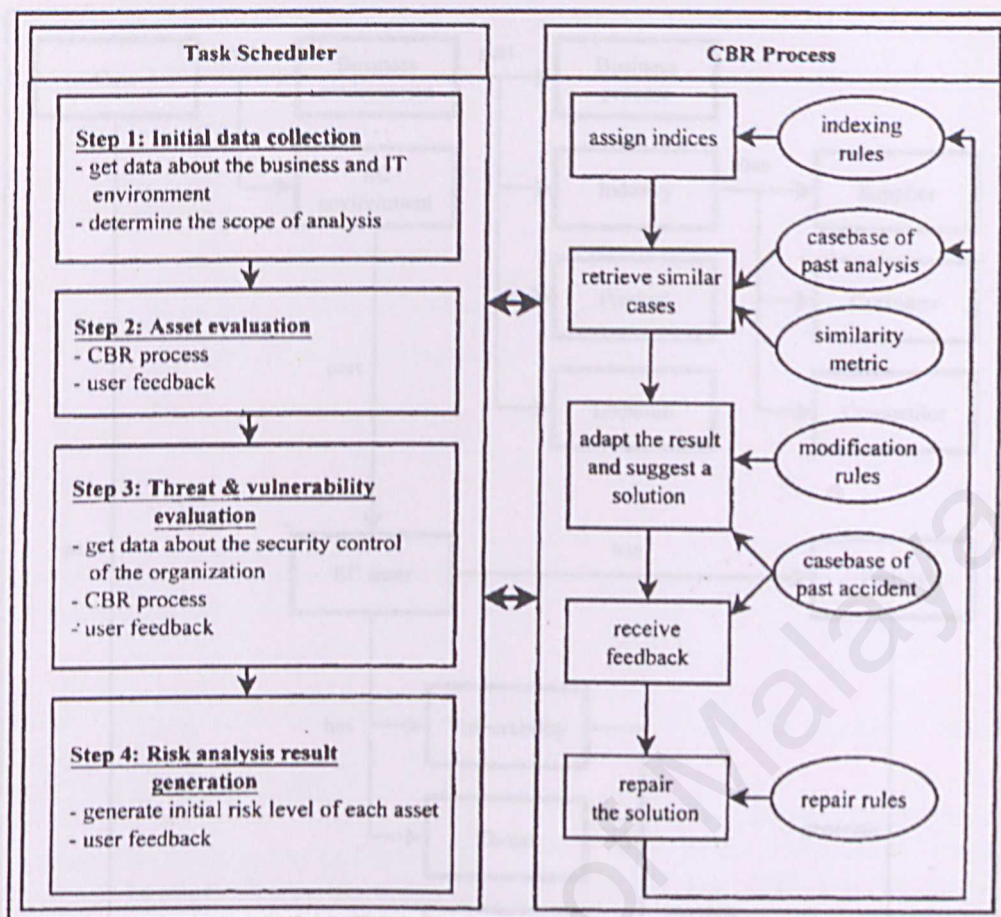


Figure 3 Process of the system

use the same case to reach the remaining sub-goals. During this process, the system may ask additional questions about the environment of the organization. If a case of a past security accident is recalled, the system attempts to find out whether it is possible for the accident to occur in the current case. Then the system produces initial results from the recall and adaptation process.

Then it receives feedback from risk analysis experts and users. This feedback tends to change the index structure and to augment knowledge regarding the cause of the failure and an explanation of whether and how it could have been prevented.

Copyright © 1999 John Wiley & Sons, Ltd.

Case Representation

The cases of this system are stored in two types of casebase. One is the casebase of past analysis inputted from the experiences of risk analysis. The other is the casebase of past accidents that is composed of the actual cases of the security accident. The cases are represented in the form of a frame, which is the most widely used knowledge representation method. The frame is a complex unit that represents situation or object in a domain (Luger and Stubblefield, 1993). The attributes of a frame are represented in the slots of the frame. The slot values may be values, pointers to other frames, or even

Int. J. Intell. Sys. Acc. Fin. Mgmt. **8**, 61-73 (1999)

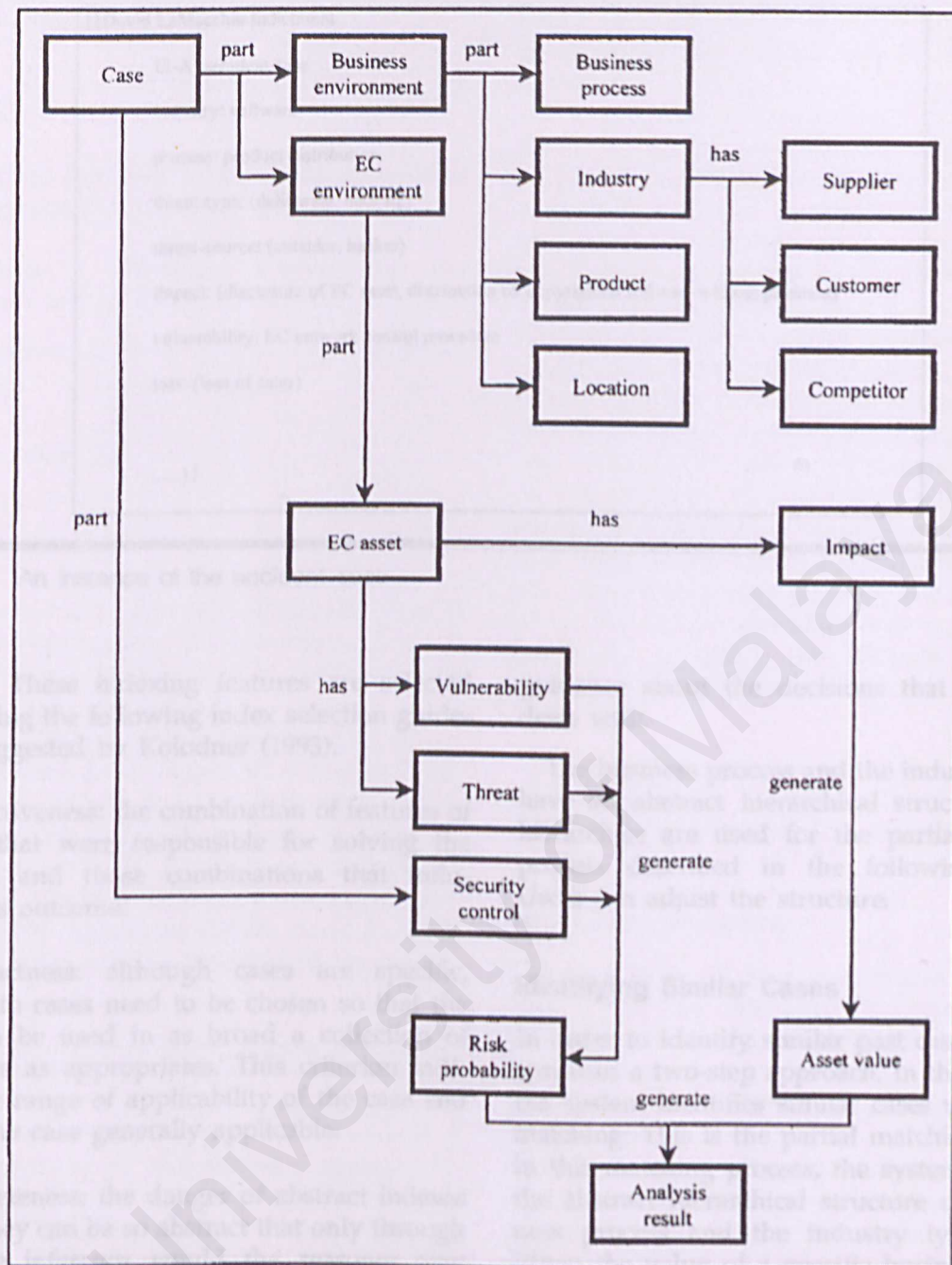


Figure 4 Internal structure of the casebase

attached procedures for performing some function. Figure 4 shows the internal structure of the frames for previous cases. An instance frame of a past security accident is shown in the Figure 5.

Case Indexing

The indexes of cases used in this system are the business process, industry type, data asset and the EC network environment of the organi-

```

{{David LaMacchia Indictment
  IS-A: accident case
  industry: software
  process: product distribution
  threat-type: (deliberate, hacking)
  threat-source: (outsider, hacker)
  impact: (disclosure of EC asset, distribution of copyrighted software without payment)
  vulnerability: EC network control procedure
  loss: (loss of sales)
  .....}}

```

Figure 5 An instance of the accident case

zations. These indexing features are selected referencing the following index selection guidelines suggested by Kolodner (1993).

- **Predictiveness:** the combination of features of a case that were responsible for solving the problem and those combinations that influenced its outcome.
- **Abstractness:** although cases are specific, indexes to cases need to be chosen so that the case can be used in as broad a collection of situations as appropriate. This criterion indicates the range of applicability of the case and makes the case generally applicable.
- **Concreteness:** the danger of abstract indexes is that they can be so abstract that only through extensive inference would the reasoner ever realize that a new situation has those descriptors. Thus, while indexes need to be generally applicable, they need to be concrete enough so that they can be recognized with little inference.
- **Usefulness:** indexes should be chosen to make the kinds of predictions that will be useful in later reasoning. Thus useful indexes are those that label a case as being able to give

guidance about the decisions that a reasoner deals with.

The business process and the industry frames have an abstract hierarchical structure. These hierarchies are used for the partial matching process described in the following section. Users can adjust the structure.

Identifying Similar Cases

In order to identify similar past cases, the system uses a two-step approach. In the first step, the system identifies similar cases using index matching. This is the partial matching process. In this matching process, the system considers the abstract hierarchical structure of the business process and the industry type frames. When the value of a specific business process and the industry type in the new case is different from the value of matching cases and their parent frames are identical to that of the new case, they can be selected as candidates. However, this inheritance mechanism is allowed only through one level.

Those partially matched cases are ranked using easy numeric matching and ranking algorithm called **nearest-neighbor matching**

(Kolodner, 1993). For each feature in the input case, the system:

- Finds the corresponding feature in the stored case
- Compares the two values with each other and computes the degree of match
- Multiplies a coefficient representing the importance of the feature to the match.

The system adopted the REMIND algorithm as follows:

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

where w_i is the importance of feature i , sim is the similarity function for primitives. f_i^I and f_i^R are the values for feature f_i in the input and retrieved cases, respectively.

In the REMIND system, both the importance and the dimensional degree of match are represented as numerical values between 0 and 1. Closer matches have values closer to 1: poorer matches closer to 0. Similarly, an importance ranking of 1 is higher than a lower importance ranking. The aggregate match score is computed by summing the products of the impor-

tance of each field multiplied by the degree of match of values in the field. To normalize the scores, they are divided by the sum of the importance ranking.

The proposed system adopted three different methods for computing the degree of similarity of two values based on the characteristics of each feature. These methods are represented in Table 2. These methods are represented in similarity functions sim in the above formula. Finally, the best-match case is modified and adapted using past security accident cases and modification rules. As stated above, the degree of similarity for business process and industry type is computed with a partial matching process. This is the method of comparison based on placement in an abstraction hierarchy. The comparison of distance on a quantitative scale is to compute the differences between the quantities and then assign the difference to a similarity. The comparison of distance on a qualitative scale quantifies the features and then computes the differences between the quantities.

IMPLEMENTATION OF THE SYSTEM

Initial Data Collection

The process for a new case begins when the user inputs the data about business and the IT environment of an organization. The major business data for case indexing includes the industry type and related processes of the organization.

They are represented in a hierarchical structure of frames. Each of these frames has the pointer of child frame. The users, as needed, can add new hierarchies. For example, when the user enters a new business process name, the system will ask the user to make the adjusted hierarchies of the business process type. In most EC cases, the processes to be inputted are likely to be related to the marketing and financial processes. The required data about the IT environment under EC for case indexing includes network, network operating system (NOS), file and DBMS. The sample

Table 2 Method of computing degree of similarity

Method	Feature
Comparison based on placement in an abstraction hierarchy	<ul style="list-style-type: none"> ● Business process ● Industry
Computation of distance on a quantitative scale	<ul style="list-style-type: none"> ● Size of the IT organization ● Type of H/W and S/W asset ● Sales revenue ● Number of customers
Computation of distance on a qualitative scale	<ul style="list-style-type: none"> ● Level of security control ● Degree of competition

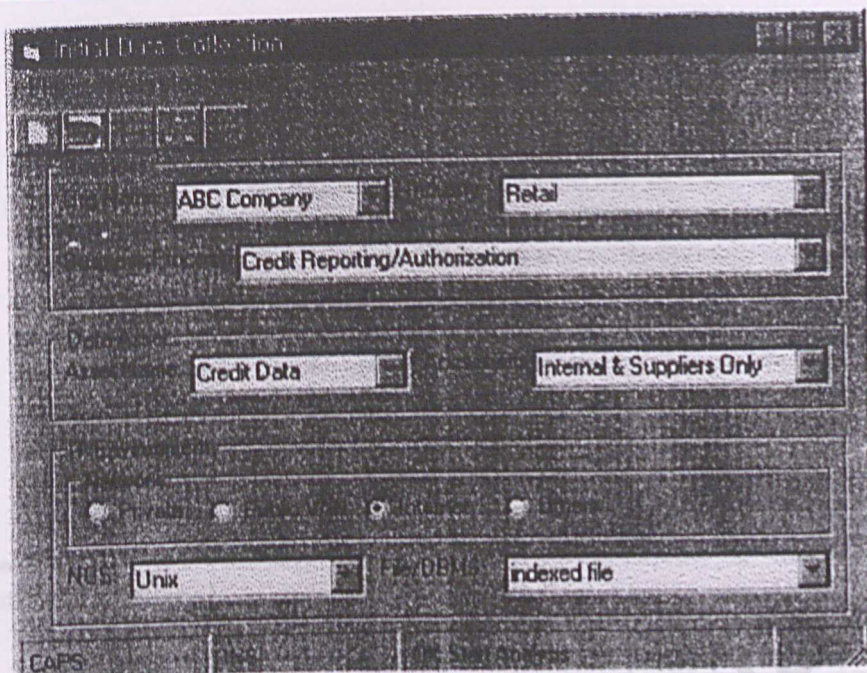


Figure 6 Initial data collection

screen for the initial data collection is shown in Figure 6.

Asset Evaluation

The goal of this step is to determine the resources or assets to be secured. This step identifies which data is critical, and thus which applications and servers need protection and monitoring. This step enables an organization to focus their budget and resources on the critical and sensitive properties of the organization and to establish a priority and required level of protection. When the user completes entering the initial data, the system identifies a set of similar cases, which contains the similar values in the features inputted from the system. Figure 7 shows an instance of the result generated by the system. The user can modify the value or examine the more detailed process of the retrieval.

Threat and Vulnerability Evaluation

If the process of asset evaluation has been completed, the system prompts the user with ques-

tions to determine the expected frequency of threats and the current control level against threat and vulnerability.

In particular, the security control in an EC environment must be more rigorous than that in a traditional environment. A robust security solution consists of three types of measure: protection, monitoring and validation.

- **Protective measures:** There are three categories of protective measures that are passive mechanisms to 'lock the door'. They are prevention, detection, and recovery. A security prevention mechanism enforces security during the system's operation by preventing a security violation from occurring. A detection mechanism detects attempts to violate security and successful security violations when or after they have occurred. A recovery mechanism is used after a security violation has been detected, restoring the system to a pre-violation state.

- **Validation measures:** Validation measures are proactive, that is, 'the lock is checked before'. Examples are active network path probing, compliance testing and audit review.

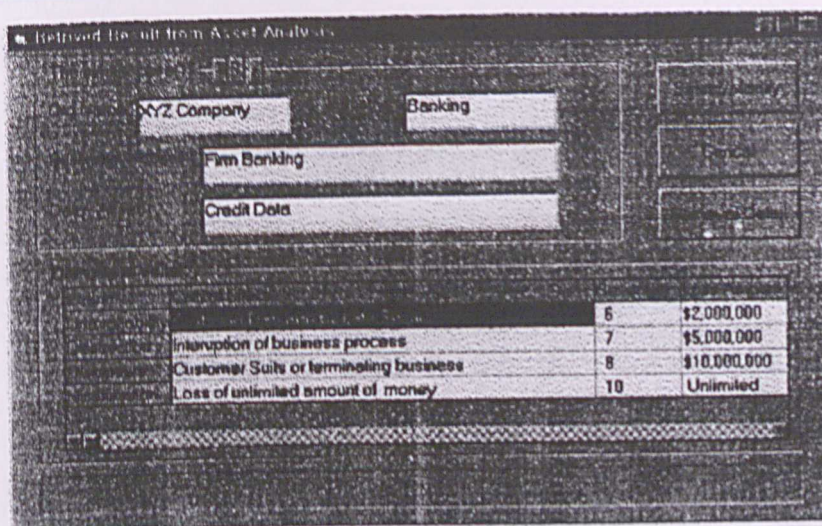


Figure 7 Evaluation of asset value

● **Monitoring measures:** Monitoring measures are reactive, that is, 'the security camera detects an intruder'. Examples include audit trails, activity monitoring; and policy breach detection.

The security control frame in the casebase contains slots for asking questions on these control measures. An example question of the validation measures is 'What's the primary method of encryption for the transmitted messages?'

When the user finishes answering the questions, the system identifies a set of similar cases which have the same values in the major attributes such as the industry, business process, asset type, network, NOS, file/DBMS, frequency of the threat, vulnerability and current control level. Based on the probability of the impact of the event on the asset value in the most similar case, the system generates the probability of the impact on unavailability, destruction, disclosure and modification, which are the major categories of threats to the security of an information system.

The system retrieves past security accident cases in order to adjust the probability generated in the above process. If the system find these cases similar to the current case, the probability or level of the related threat or vulner-

ability may be higher. An example case of a retrieved result from the threat and vulnerability analysis is shown in Figure 8.

Generation of Risk Analysis Result

Finally, the system generates the risk level for unavailability, modification, disclosure and destruction, which are popularly accepted risk classes (Pfleeger, 1989). The risk level is calculated by simply multiplying the severity of impact and the level of threat and vulnerability. The amount of annual loss expectancy (ALE) is estimated by multiplying the loss amount of the impact by the possibility of threat and vulnerability. Figure 9 shows an example of the result of risk analysis generated by the system.

CONCLUSION

The strong demand for the Internet and EC is emerging from the fast-changing environment of IT. Data integrity and confidentiality are more indispensable for the Internet and EC than for traditional systems due to open networks. However, the level of risk analysis and security control for the Internet and EC lags far behind demand. This study intends to

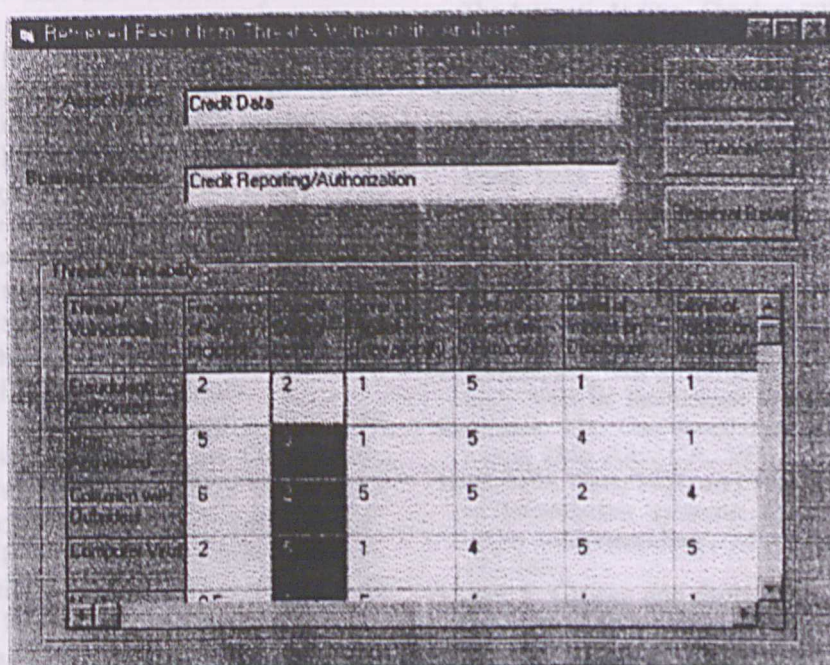


Figure 8 Threat and vulnerability analysis

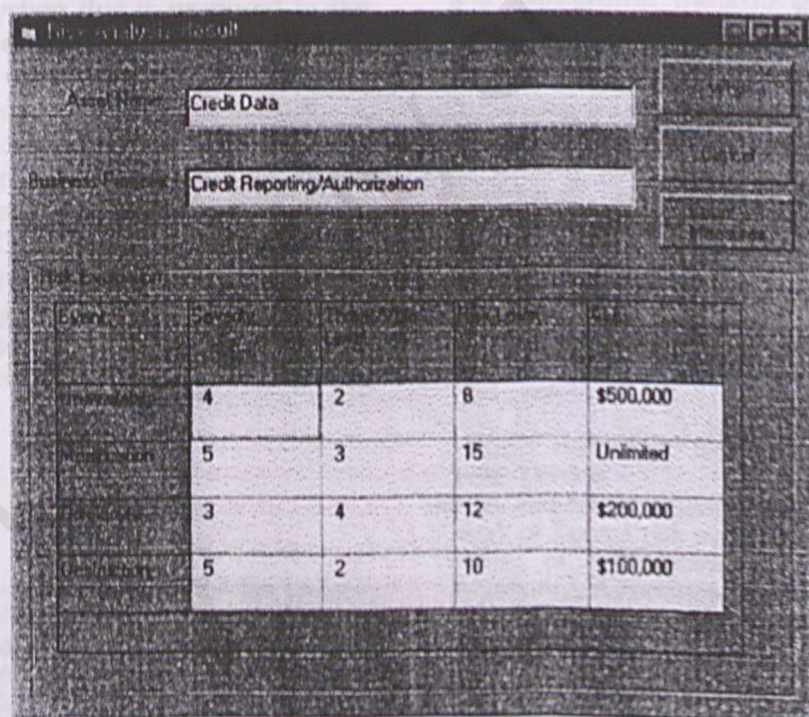


Figure 9 The initial result of risk analysis

develop a system that aids risk analysis in an EC environment.

Risk analysis for EC requires considerable professional judgment and knowledge of IT. Nonetheless, the immaturity of risk analysis for an EC system makes it difficult to afford expertise and knowledge. This is why this study takes advantage of the CBR technique. The benefits of this technique correspond to the above characteristics of risk analysis for EC and complement its immaturity. As the major casebase of CBR, this system uses the casebase of past risk analyses and security accidents.

This study is the first to apply the CBR technique for risk analysis of an EC system. The proposed system in this study provides a fast and cost-effective analysis using the reasoning ability of CBR, which comes from analogical reasoning of the past cases. Therefore it will become a useful instrument of risk analysis for novices in this area. In addition, the learning ability to update the casebase dynamically makes the system valuable in the fast-changing EC environment. Consequently, the performance of this system is expected to improve gradually as the casebase is updated.

However, the system that is proposed in this study is only a prototype. This prototype system has not been validated, nor applied to any organization or assessed for its superiority to traditional risk analysis methods. Morris (1994) showed the superiority of the CBR technique for risk analysis and this study only presents the possibility of CBR application to risk analysis in an EC environment. As a prototype, this system acts as a guiding light of risk analysis in an EC environment. In future, this prototype system will be advanced toward a system that can be applied to the real world.

References

- Benesko, G.G., 'Electronic commerce in the 21st century', available at http://www.rtc.com/cent_21.htm, 1994.
- Benesko, G.G., 'Electronic commerce the next generation', available at <http://www.rtc.com/ece.htm>, 1995.
- Brown, C.E. and Gupta, U.G., 'Applying case-based reasoning to the accounting domain', *Intelligent Systems in Accounting, Finance and Management*, 3, 1994, 205-221.
- Cerullo, M.J. and Shelton, F.A., 'Analyzing the cost-effectiveness of computer controls and security', *The Internal Auditor*, October 1981, 30-37.
- Ellsworth, J.H. and Ellsworth, M.V., *The Internet Business Book*, John Wiley, New York, 1994.
- Garfinkel, S. and Spafford, G., *Practical Unix & Internet Security*, 2nd edition. O'Reilly, New York, 1996.
- Icove, D. et al., *Computer Crime*, O'Reilly, New York, 1995.
- Kailay, M.P. and Jarratt P., 'RAMEX: a prototype expert system for computer security analysis and management', *Computers & Security*, 14, 1995, 449-463.
- Kalakota, R. and Whinston, A., *Frontiers of Electronic Commerce*, Addison-Wesley, Reading, MA, 1995.
- Kolodner, J.L., 'An introduction to case-based reasoning', *Artificial Intelligence Review*, 6, 1992, 3-34.
- Kolodner, J.L., *Case-Based Reasoning*, Morgan Kaufmann, Palo Alto, CA, 1993.
- Loch, K.D., Carr, H.H. and Warkentin, M.E., 'Threats to information systems: today's reality, yesterday's understanding', *MIS Quarterly*, June 1992, 173-186.
- Luger, F. George and Stubblefield, A. William, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Benjamin/Cummings, New York, 1993.
- Morris, B.W., 'SCAN: a case-based reasoning model for generating information system control recommendations', *Intelligent Systems in Accounting, Finance and Management*, 3, 1994, 47-63.
- Moses, R.H. and Glover, I., 'The CCTA risk analysis and management methodology (CRAMM) Risk management Model', Working Paper, 1988.
- Norman, 'Norman risk analysis: the buddy system', available at <http://www.norman.com/buddy.html>.
- NSI, 'Securing Internet information servers', available at <http://nsi.org/library/compsec/secuinte.html>, 1994.
- NSI, 'Security issues in the use of Electronic Data Interchange', available at <http://nsi.org/library/comm/secured.txt>, 1991.
- Pfleeger, C.P., *Security in Computing*, Prentice Hall, Englewood Cliffs, NJ, 1989.
- Rainer, R.K. Jr, Synder, C.A. and Carr, H.H., 'Risk analysis for information technology', *Journal of Management Information Systems*, 8, No. 1, Summer 1991, 129-147.
- Solms, R.V. et al., 'A framework for information security evaluation', *Information & Management*, 26, 1994, 143-153.
- Suh, B. and Han, I., 'Design of the conceptual framework of Korean risk management system: identification of information system threats', Korea Accounting Association. Conference on AIS, 1997.
- Trio, N.R., 'Internet security', available at <http://www.zurich.ibm.com/white-paper.html>.
- Yang, C.Y., *Secure Electronic Transaction (SET) Specification Book 1: Business Description*, Available at <http://www.vitedu.tw/~cyyang/set/bus/SETBUS.htm>, 1996.
- Zelevnik, M.P., *Security design in distributed computing applications*, PhD thesis. The University of Utah, 1993.



Case-based reasoning for antibiotics therapy advice: an investigation of retrieval algorithms and prototypes

Rainer Schindler^a, Leibar Gicli^b

^a Institute for Medical Informatics and Biometrics, University of Zurich, Rindlimbergstrasse 71, 8093 Zurich, Switzerland
^b Rishon LeZion, Israel

Received 20 May 2000; received in revised form 7 December 2000; accepted 20 May 2001

Abstract

We have developed an antibiotic therapy advisor system called XCONS for patients in an intensive care unit (ICU) who have caught an infection or bacterial complication. Therapy advice for such critically ill patients is usually very specific and a few actual pathogens still have to be identified by the laboratory. We use an expert system that is based on a central background knowledge and known resistances. The system contains a spectra and the resistance information are periodically updated from laboratory reports. In order to speed up the process of finding suitable therapy recommendations, we have applied several retrieval (IR) techniques. As all required information should always be up-to-date in medical expert systems, new cases should be immediately incorporated into the case base and outdated ones should be deleted or moved. For reasons of space flexibility, retrieval steps in the process of the case base should be modified. To fulfill these requirements we propose that specific single cases should be generalized to more general prototypical cases and that replacement techniques (rules) should be used. In this paper, we present and compare results of different generation strategies for generalized cases (prototypical antibiotics cases) and we compare retrieval cases for two indexing retrieval algorithms: simple indexing, which is appropriate for small and medium case bases, and tree-based retrieval, which is appropriate for large case bases. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Antibiotics; Decision support; Expertise; Intelligence; Case-based reasoning; Retrieval

1. Introduction

Severe bacterial infections are still a life-threatening complication in intensive care medicine, correlating with a high mortality [1]. Identification of bacterial pathogens is

Corresponding author. Tel.: +41(0)43 734 7330; fax: +41-043-734 7129.
E-mail address: schindler@infocenter.unizh.ch (R. Schindler).



Case-based reasoning for antibiotics therapy advice: an investigation of retrieval algorithms and prototypes

Rainer Schmidt*, Lothar Gierl

*Institute for Medical Informatics and Biometry, University of Rostock, Rembrandtstrasse 16/17, D-18055
Rostock, Germany*

Received 22 May 2000; received in revised form 7 December 2000; accepted 2 May 2001

Abstract

We have developed an antibiotics therapy advice system called ICONS for patients in an intensive care unit (ICU) who have caught an infection as additional complication. Since advice for such critically ill patients is needed very quickly and as the actual pathogen still has to be identified by the laboratory, we use an expected pathogen spectrum based on medical background knowledge and known resistances. The expected pathogen spectra and the resistance information are periodically updated from laboratory results. To speed up the process of finding suitable therapy recommendations, we have applied case-based reasoning (CBR) techniques. As all required information should always be up to date in medical expert systems, new cases should be incrementally incorporated into the case base and outdated ones should be updated or erased. For reasons of space limitations and of retrieval time an indefinite growth of the case base should be avoided. To fulfill these requirements we propose that specific single cases should be generalised to more general prototypical ones and that subsequent redundant cases should be erased. In this paper, we present evaluation results of different generation strategies for generalised cases (prototypes). Additionally, we compare measured retrieval times for two indexing retrieval algorithms: simple indexing, which is appropriate for small and medium case bases, and tree-hash retrieval, which is advantageous for large case bases. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Antibiotics; Decision support; Artificial intelligence; Case-based reasoning; Retrieval

1. Introduction

Severe bacterial infections are still a life-threatening complication in intensive care medicine, correlating with a high mortality [1]. Identification of bacterial pathogens is

* Corresponding author. Tel.: +49-381-394-7307; fax: +49-381-394-7203.
E-mail address: schmidt@medizin.uni-rostock.de (R. Schmidt).

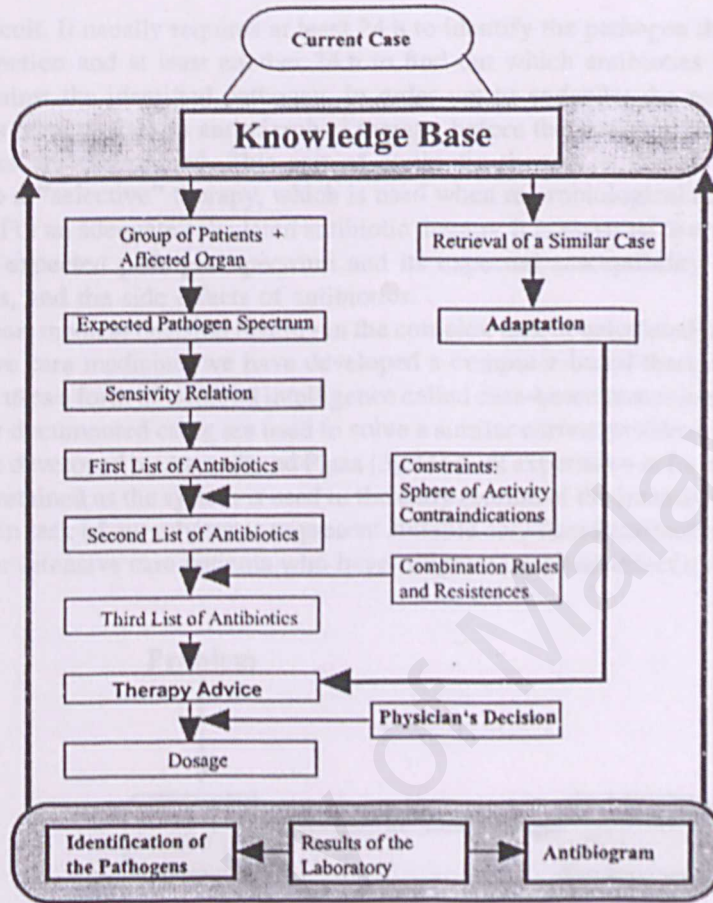


Fig. 2. Overview of ICONS.

complication. Since, for such critical patients, physicians cannot wait for the laboratory results, we use an expected pathogen spectrum based on medical background knowledge. Each recommended antibiotics therapy should completely cover this spectrum. Furthermore, as advice is needed very quickly we speed up the process of computing recommended antibiotic therapies by using CBR methods (the right path in Fig. 2). This means that we search for a previous similar patient and transfer the therapies suggested for his situation to the current patient. These previous therapies are then adapted to take account of any differences between the situations of the previous and current patients.

Within the ICONS project the adviser was originally developed for an ICU in Munich [2]. We evaluated the quality of the antibiotic therapy recommendations and the user friendliness of the system within this unit. Recently, we have converted ICONS to fit the environment in an ICU in Rostock. The change of ICU means that some other antibiotics are available, and some local information is partly different (frequently observed pathogens, slightly different resistances). So, now we are evaluating the therapy recommendations again.

often difficult. It usually requires at least 24 h to identify the pathogen that is responsible for an infection and at least another 24 h to find out which antibiotics have therapeutic effects against the identified pathogen. In order not to endanger the patient, physicians sometimes have to start an antimicrobial therapy before the responsible pathogen and its sensitivities are determined. This sort of antibiotic therapy is called “calculated”, in contrast to a “selective” therapy, which is used when microbiological results are already available. For an adequate calculated antibiotic therapy, it is essential to access information about the expected pathogen spectrum and its expected susceptibility, existing contra-indications, and the side effects of antibiotics.

To support medical decision making in the complex task of calculated antibiotic therapy in intensive care medicine, we have developed a computer-based therapy adviser, called ICONS. It uses a form of artificial intelligence called case-based reasoning (CBR). In CBR, previously documented cases are used to solve a similar current problem. Fig. 1 shows the CBR cycle developed by Aamodt and Plaza [3]. Medical experience in form of cases is automatically retained as the system is used in the daily routine of the intensive care unit (ICU).

The main task of our adviser is to present suitable calculated antibiotics therapy advice (Fig. 3) for intensive care patients who have caught a bacterial infection as an additional

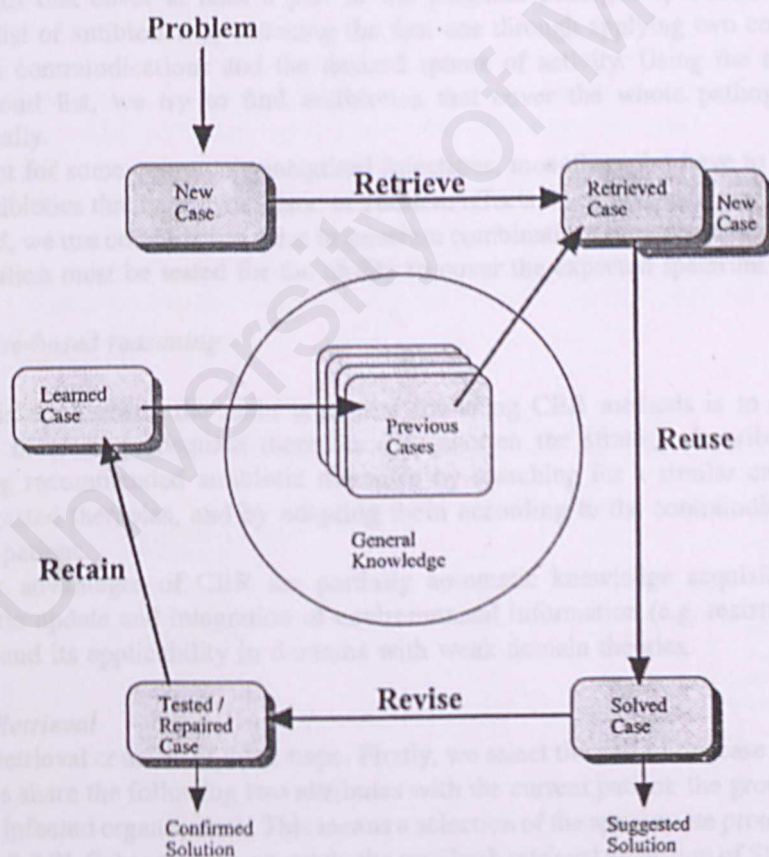


Fig. 1. Case-based reasoning cycle.

Furthermore, we have recently evaluated the CBR method within the ICONS system. Here we present results concerning retrieval algorithms and the storage architecture, which generates prototypical cases automatically.

2. Materials and methods

2.1. Strategy for selecting recommended antibiotic therapies

As ICONS is not a diagnostic system, we do not attempt to deduce evidence for diagnoses based on symptoms, frequencies or probabilities, but instead pursue a strategy that can be characterised as follows: find all possible solutions, and subsequently reduce them using the patient's contraindications and the requirement to completely cover the calculated pathogen spectrum (establish-refine strategy).

Firstly, we distinguish between different groups of patients (infection acquired in or outside the ward, respectively the hospital; immunocompromised patients). An initial list of antibiotics is generated by a susceptibility relation, which for each group of pathogens provides all antibiotics that usually have therapeutic effects. This list contains all antibiotics that cover at least a part of the potential pathogen spectrum. We obtain a second list of antibiotics by reducing the first one through applying two constraints: the patient's contraindications and the desired sphere of activity. Using the antibiotics on this second list, we try to find antibiotics that cover the whole pathogen spectrum individually.

Except for some community-acquired infections, monotherapies have to be combined with antibiotics that have synergistic or additive effects. If no adequate single therapy can be found, we use combination rules to generate combinations of antibiotics. Each possible combination must be tested for the ability to cover the expected spectrum completely.

2.2. Case-based reasoning

In this application, the main argument for using CBR methods is to speed up the process of finding adequate therapies. We shorten the strategy described above for selecting recommended antibiotic therapies by searching for a similar case, retrieving its suggested therapies, and by adapting them according to the contraindications of the current patient.

Other advantages of CBR are partially automatic knowledge acquisition, partially automatic update and integration of environmental information (e.g. resistance developments) and its applicability in domains with weak domain theories.

2.2.1. Retrieval

The retrieval consists of three steps. Firstly, we select the part of the case base in which all cases share the following two attributes with the current patient: the group of patients, and the infected organ system. This means a selection of the appropriate prototype tree (see Section 2.2.3). Subsequently, we apply the tree-hash retrieval algorithm of Stottler et al. [4] for nominal valued contraindications and the similarity measure of Tversky [5] for the few

integer valued contraindications. Furthermore, we use an adaptability criterion, because not every case is adaptable [6]. The attributes used for the retrieval are the contraindications, which work as constraints on the set of possible antibiotics. It is, therefore, obvious that we should use only former cases whose contraindications are shared by the current patient. To guarantee this condition the adaptability criterion has to be checked during retrieval. This can be considered as an example which supports the belief of Smyth and Keane that the similarity assumption alone is often inappropriate and that retrieval should also take adaptability into account [7].

2.2.2. Adaptations

In ICONS three different sorts of adaptations occur: a CBR adaptation to obtain sets of calculated advisable therapies for current patients (Fig. 3 shows the presentation of such a set), an adaptation of chosen therapies according to laboratory findings and a periodical update of laboratory information (resistance situation, frequently observed pathogens).

2.2.2.1. Case-based reasoning adaptation. Each contraindication restricts the set of advisable therapies. During the retrieval we require that the retrieved case does not have any additional contraindications besides those of the current case. Otherwise the solution set for the current case would be inadmissibly reduced by the additional contraindications of a previous case.

Because of this criterion, the adaptation of a previous similar case is rather simple. It is simply a matter of transferring the set of advisable therapies and if necessary of reducing this set according to the additional contraindications of the current case.

2.2.2.2. Adaptations of chosen therapies to laboratory findings. Adaptations of laboratory findings do not really belong to the CBR paradigm. However, they are based on case information. The goal of the main part of ICONS is to present advisable therapies before the laboratory results are known. When these results become available later on, the initial therapy has to be adapted to them.

Restrictions	Pathogen Spectrum	Additional Therapies	Own Creation	
ADVISABLE THERAPIES:				Price (DM/Day)
LINCOSAMIDE	+ GYRASEHEMMER :			92 to 201
<input type="checkbox"/> CLINDAMYCIN	+ CIPROFLOXACIN			
PENICILLINE	+ AMINOGLYCOSIDE :			48 to 111
<input type="checkbox"/> PIPERACILLIN	+ GENTAMICIN			65 to 141
<input type="checkbox"/> PIPERACILLIN	+ TOBRAMYCIN			166 to 231
<input type="checkbox"/> PIPERACILLIN	+ AMIKACIN			66 to 121
<input type="checkbox"/> AUGMENTAN	+ TOBRAMYCIN			168 to 201
<input type="checkbox"/> AUGMENTAN	+ AMIKACIN			11 to 15
<input type="checkbox"/> TAZOBAC	+ GENTAMICIN			28 to 42
<input type="checkbox"/> TAZOBAC	+ TOBRAMYCIN			129
<input type="checkbox"/> TAZOBAC	+ AMIKACIN			75 to 141
<input type="checkbox"/> MEZLOCILLIN	+ TOBRAMYCIN			177 to 221
<input type="checkbox"/> MEZLOCILLIN	+ AMIKACIN			

Fig. 3. Part of an advisable antibiotics therapy presentation.

There are two sorts of findings. After about 24 h the pathogen that is responsible for the infection is identified. If the identified pathogen does not belong to the calculated pathogen spectrum and if this pathogen is not sensitive to the initial therapy — according to the systems sensitivity information, then new specific advisable therapies against this pathogen have to be computed.

After about another 24 h the sensitivity test results (antibiogram) of this pathogen against the various antibiotics become available. If the laboratory sensitivity test results show that the identified pathogen is, in contrast to the system's sensitivity information, not sensitive to the initial therapy, it leads to the same task: new "selective" advisable therapies, which have therapeutic effects against the identified pathogen, have to be computed.

When new cases are incorporated into the system, their laboratory findings must be taken into account and consequently the sensitivity information has to be updated. Additionally, the expected pathogen spectra might change over time. For both laboratory information sources used by the system (sensitivity information, and expected pathogen spectra) we have implemented a periodical update. This can be seen as another form of adaptation that is not based on single cases, but on statistical evaluation of specific information from a number of cases.

2.2.2.3. Resistance information. It might seem to be a contradiction that, in contrast to the system's current sensitivity information, laboratory tests can show that a pathogen is not sensitive to an antibiotic. However, as pathogens are never exactly alike, but always slight mutations, the sensitivity information is based on a percentage value. For example, nowadays for many problematic pathogens only 80 or 90% sensitivity can be observed to the strongest antibiotics. So an observed sensitivity higher than 66% is usually already considered as "sensitive".

To make information about the current local resistances available, we have implemented an Intranet information program based on JAVA. The laboratory database, which contains the antibiograms (the results of the sensitivity tests of the various antibiotics applied to the identified pathogen), is evaluated monthly. The results are presented in two forms: a table shows the current resistance situation in percent and in absolute numbers (Fig. 4) and

Pseudomonas 04/96 - 03/97						
Antibiotics	Sensitive		Resistant		Not tested	
	percent	absolute	percent	absolute	percent	absolute
Tobramycin	94,98	625	5,02	33	2,81	19
Amikacin	91,22	613	8,78	59	0,74	5
Ciprofloxacin	81,66	552	18,34	124	0,15	1
Gentamicin	77,99	528	22,01	149	0,00	0
Imipenem	71,20	482	28,80	195	0,00	0
Ofloxacin	69,13	468	30,87	209	0,00	0
Ceftazidim	68,89	465	31,11	210	0,30	2

Fig. 4. Top part of the tabular presentation of the resistance situation (the antibiotics are sorted according to their percentage values).

graphs show courses of resistance developments. The user can switch between the two presentations. In the graphical menu, the physician can select the pathogen (only one for each presentation), the time period to be considered, and the antibiotics. Up to eight antibiotics can be graphically presented simultaneously.

2.2.3. Prototypes

Since in an incrementally working system the number of cases increases continuously, storing each case would slow down the retrieval time and exceed any space limitations. We, therefore, decided to structure the case base by prototypes and to store only those cases that differ significantly from their prototype. Though the general use of prototypes was introduced early in the CBR field [8,9], it is still mainly applied in the medical domain (e.g. [10–13]). Our prototype architecture is chiefly based on experience with a diagnostic application [14], where we create prototypes that include the most frequent features of the corresponding cases. In other words, the features of a prototype are those shared by most of their cases. This idea is based on empirical research [15], which indicates that people consider cases to be more “typical” when the number of features shared between the presented case and the “normal” case increases.

In diagnostic applications, prototypes correspond to typical diseases or diagnoses. So, for antibiotic therapies, prototypes are expected to correspond to typical antibiotic treatments associated with typical clinical features of patients [16]. However, as the attributes are contraindications that are responsible not for the generation, but for the restriction of the solution set, this is only partly true. We have investigated the growth of a hierarchical prototype structure built up from a randomly ordered stream of cases.

2.2.3.1. Selection of a prototype tree. In ICONS there is not just one prototype tree, but a forest of trees, which are all independent from each other. A specific tree can be generated for each affected organ system combined with each group of patients. So, for nearly 20 organ systems and five patient groups there are nearly 100 possible prototype trees. We generate them dynamically only when required. For example a tree for “community-acquired kidney infections” will be generated as soon as the first data input occurs from a patient who has a kidney infection which he has acquired outside the hospital.

Since all cases within the same prototype tree belong to the same group of patients, and the same organ system is affected, it follows that the same expected pathogen spectrum deduced from background knowledge has to be covered. Cases within the same prototype tree are only discriminated from each other by their contraindications. These are allergies against specific antibiotics, reduced organ functions (kidney and liver), specific diagnoses (e.g. CNS disease), special blood diseases, pregnancy and the patient's age.

2.2.3.2. Generating prototypes. The aim of our concept of prototypical cases is to structure the case base, to keep the prototypes always up to date and to erase redundant cases. As the prototypes are generated incrementally and as they should always contain the typical features of their cases, we use two threshold parameters:

1. The parameter “minimum frequency” determines how (relatively) often a contraindication has to occur in the set of cases to be incorporated into the prototype.

used in my system

2. The parameter “number of cases” determines the required number of cases that are necessary to fill a prototype or to create an alternative prototype. The lower this threshold the more prototypes are created and the fewer cases are stored.

First, all cases are stored below the prototype they belong to. If the threshold “number of cases” is reached after storing a new case below a prototype, the prototype will be “filled”. At this point, every contraindication that occurs in the prototype’s cases at least as often as the “minimum frequency” threshold will be included into the prototype. Subsequently, the “filled” prototype can be treated like a case. The same holds for prototypes as for cases: each contraindication restricts the set of advisable therapies. The contraindications of a prototype are those that occur most often within its cases. So from the viewpoint of frequency they are the typical ones. Those cases that have no additional contraindications in comparison with their prototypes are erased.

When new cases are added later on to an already filled prototype, the observed frequencies may change and consequently the contraindications of the prototype may have to be recomputed. If the contraindications of a prototype change, the suggested antibiotic therapies have to be recomputed, too. In addition, all cases must be inspected again to determine whether they need to be stored.

We create an “alternative” prototype below an already existing prototype if for the latter enough cases exist (which means the threshold “number of cases” is reached) that have at least one contraindication in common, which the already existing prototype does not include. We generate the alternative prototype using those cases that share at least one contraindication not included in the existing prototype. We place this new prototype in the hierarchy directly below the already existing prototype. Alternative prototypes differ from their superior prototypes by their contraindications and, therefore, also by their sets of advisable antibiotic therapies.

3. Results

3.1. Comparison of two indexing retrieval algorithms

The case attributes are the possible contraindications (e.g. penicillin-allergy or pregnancy). As these are unordered nominal values, we did not consider retrieval algorithms like CBR retrieval nets [17], which are appropriate for ordered nominal values, nor nearest neighbor algorithms [18], which are appropriate for metric values, but only indexing algorithms.

Originally, we applied the tree-hash retrieval algorithm developed by Stottler et al. [4], which is advantageous for large case bases. Recently we compared it with a simple indexing algorithm. The results show that the answer to the question: which algorithm works faster? depends on the definition of large.

As we are going to present retrieval times, we now give details of the hardware and software we used:

- Computer: Power Macintosh;
- Operating system: Mac OS D1-8.1;

- Memory size: 64MB;
- Virtual memory size: 74.8MB;
- Programming language: Macintosh Common Lisp (MCL), Version 3.0;
- Memory size reserved for MCL 3.0: desired size, 4.1MB; minimal size: 3.6MB.

3.1.1. Tree-hash retrieval

The tree-hash retrieval algorithm was designed for the retrieval of cases whose attributes have qualitative values. In the following, we use the terms of the authors of the tree-hash retrieval algorithm. From a rather spatial point of view, they see attributes as dimensions.

In a pre-processing step, a tree is constructed. For each possible combination of dimensions a node is set up. For example, in the two-dimensional case with dimensions d_1 and d_2 , the possible dimension combinations are $\{d_1, d_2, d_1d_2\}$. The algorithm attempts a retrieval based on each of these combinations and computes a similarity measure. In our example, a first retrieval is performed based on d_1 , meaning that d_1 is the only dimension that must produce an exact match. The same is done for d_2 . Retrieval based on d_1d_2 indicates that both dimensions must match the dimensions of a case in the case base.

If a hashing scheme is used, the retrieval time does not increase with the number of cases, but increases with the number of dimensions. When this number is rather low (up to 10, at most 15), the algorithm is very fast.

The tree-hash pre-processing step generates all possible combinations of dimensions and represents them as nodes in a tree. If the dimensions are given in decreasing order of importance in a dimension list, the dimensions are placed accordingly in the tree. The following algorithm generates the tree structure displayed in Fig. 5, when the number of dimensions, D , equals three.

- Root = d_1
- call tree (Root, 1)
- function tree (t , n)
- if $n = D$ then return t
- else
- leftson (t) = call tree ($d_n + 1$ concatenated to back of t , $n + 1$)
- rightson (t) = call tree ($d_n + 1$ concatenated to back of t without its last dimension, $n + 1$)

To provide efficient lookup, a unique pointer is created for each case and placed in a hash table $2^D - 1$ times, once for each possible combination of attribute values, ignoring order.

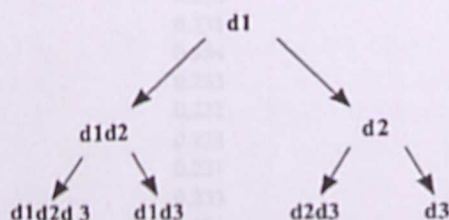


Fig. 5. Example of hash-tree.

This results in a large hash table, which encompasses all cases, organised by every possible ordering of dimensions. When a node is visited, the case can be hashed using the dimensions represented by the node. Retrieval simply consists of visiting the nodes of the tree and performing a hash look-up based on the dimensions of the node.

Instead of visiting every node in the tree, it is more efficient to search the tree in a specified order, because some nodes do not have to be visited. Traversal of the tree begins at the root. At each node, an attempt is made to hash retrieve a case based on the dimensions of that node. If the retrieval is successful, it continues down the left branch. If unsuccessful, it continues down the right branch, because in the left branch no success is possible and it would be a waste of time to visit such a left branch. This process is continued until a leaf node is reached. At this point, the similarity value of the node of the last successful retrieval is stored. Then the algorithm backtracks to the last successful node, and continues to its right son. This process is repeated until there are no successful nodes left to backtrack to. The stored similarity values are compared and the best one is selected.

This process can be shortened. During the pre-processing, the highest possible score of each node is stored. Since the highest possible score at any node is the maximum of the possible scores of its subnodes, nodes and their subnodes which have a lower highest possible score than the similarity value already reached do not have to be visited.

3.1.2. Comparison with a simple indexing algorithm

Simple indexing does not construct a hash-tree, but hashes directly. We use a table to store attribute values with pointers to lists of those cases that contain the values. At runtime, we set up a second table. During the retrieval it contains information like “caseno. → sum of (so far) indexed weights”. The cases with the same values as the query case are looked up in the first table and their weights are increased in the second one. At the end, the cases with highest weights are retrieved. For both algorithms, we have used a test set of 100 cases, which were incrementally incorporated into the system (Table 1).

The results can be summarised as follows: simple indexing works much faster than the tree-hash algorithm. However, for simple indexing the retrieval time increases with the number of stored cases, by approximately 0.001 s per 15 cases. As tree-hash does not

Table 1

Retrieval times for tree-hash and simple indexing retrieval algorithms

Case nos.	Average retrieval time for tree-hash	Average retrieval time for simple indexing
2–15	0.233	0.043
16–30	0.231	0.044
31–45	0.234	0.045
46–60	0.233	0.046
61–75	0.232	0.047
76–80	0.233	0.047
81–85	0.231	0.048
86–90	0.233	0.048
91–95	0.231	0.049
96–100	0.232	0.049

depend on the number of stored cases, both algorithms should have equal retrieval times when the case base contains roughly about 3000 cases. However, this result depends on our concrete implementation in Macintosh Common Lisp (MCL 3.0). As Lisp does not support pointers, which are useful for operations on trees, graphs, etc. an implementation in other programming languages like C or PASCAL might come up with better results for the tree-hash algorithm.

Consequently, the initial overhead for tree-hashing in comparison to simple indexing is outweighed by faster retrieval times for huge case bases. In our implementation, “huge” refers to case bases with a few thousands of items.

3.2. *Prototype generation strategies*

The general idea of our concept is to keep the prototypes always up to date. They should contain the typical features of their cases. We have tested two contrasting policies for deleting redundant cases and a strategy of keeping all cases. Our evaluation had two aims. First, we wished to find a strategy that best fits the two contrasting aims of finding many adaptable cases or prototypes and requiring little memory. Secondly, we wished to find good settings for the threshold parameters.

Normally, cases without additional features in comparison to their prototype are redundant, because they do not contain any additional information [19]. However, in our application the attributes are contraindications, which are not used to generate a solution, but to restrict a solution set. This means they are applied as constraints. A case with fewer contraindications than its prototype has a greater chance of being adaptable to a query case, because only a case without additional contraindications in comparison to the query case is adaptable.

We have, therefore, tested two opposing strategies: firstly, deleting cases without additional attributes and secondly, deleting only cases with additional attributes. Additionally, we have tested a strategy without deleting any cases at all.

The memory size without any stored cases is about 2.248MB for all three strategies. The argument about the memory might seem to be unreasonable, because the differences between the strategies are only about 40KB for 75 test cases. However, we performed our tests in just one of about 100 possible parallel sets. A total of 75 cases in each set might lead to differences of up to 4MB. This leads to the question of whether our system should require about 12 or about 16MB memory. Certainly, problems should not occur until the number of cases per set exceeds 75.

Without generating any prototypes at all, for 51 of the 75 test cases a similar adaptable case can be found. As prototypes are treated like cases, this number can be exceeded.

3.2.1. *Strategy A: deleting cases without additional attributes*

We have tested the strategies with 75 cases, which were incrementally incorporated into the system. For strategy A, we varied the threshold parameter “number of cases”, which indicates how many cases are necessary to generate a prototype. The second threshold parameter “relative frequency” was set to 33%, which means that a contraindication is incorporated into a prototype if at least a third of its cases have this contraindication.

Table 2
Test results for strategy A

	Setting number of cases = 2	Setting number of cases = 3	Setting number of cases = 4	Setting number of cases = 5
Memory size (MB): after 75 cases	2.392	2.390	2.401	2.402
Number of prototypes	9	7	8	8
Number of stored cases	53	57	62	63
Number of deleted cases	22	18	13	12
Number of adaptations	12	26	31	31

The results (Table 2) can be summarised as follows: the more cases necessary to generate a prototype (this is achieved by increasing the value “number of cases”) the higher the number of stored cases and the higher the number of retrieved adaptable cases. After a while there is only little to be gained by increasing this threshold parameter any further (4th setting). A surprise is the big increase in the number of retrieved adaptable cases in the second setting compared with the first one. This cannot be simply explained by the four additionally stored cases, but by the following two phenomena. Firstly, those cases that have no additional information (contraindications) in comparison to their prototype are deleted. This means that the deleted cases would be more likely to be adaptable to future queries. Secondly, under the second setting the prototypes are generated later and consequently cases are deleted later as well.

One aim of using prototypes is the hope of reducing the memory size. For strategy A, this benefit does not occur, because the storage requirement for prototypes is bigger than for cases. This is because prototypes contain some additional information: the intersection of advisable therapies for their cases (cases only contain additional specific therapy suggestions), observed frequencies of contraindications of their cases, etc.

3.2.2. Strategy B: deleting cases with additional attributes

Our aim with strategy B, was to keep in the case base those cases that have a higher chance to be adaptable. These are cases with few contraindications. We, therefore, adapted a strategy opposite to strategy A, namely, deleting cases with additional information (contraindications) to their prototype. As many cases are deleted, we set the threshold parameter “number of cases” to the value two. Here, we varied the second parameter “relative frequency”, which determines the frequency with which contraindications have to be observed among the cases to be incorporated into a prototype.

The difference between the results for the two settings for strategy B, is rather small (Table 3). With a smaller relative frequency (2nd Setting) more contraindications are incorporated into the prototypes. So, fewer stored cases have additional contraindications in comparison to their prototypes and consequently fewer cases are deleted. Furthermore, fewer prototypes are generated, because the prototypes cover more cases. The memory size is nearly the same and the number of retrieved adaptable cases is exactly the same for both settings.

In comparison to strategy A, it is noticeable that about the same number of prototypes have been generated, but much more cases have been deleted. Though those cases which

Table 3
Test results for strategy B

	Parameter setting: relative frequency = 33%	Parameter setting relative frequency = 25%
Memory size (MB): after 75 cases	2.373	2.368
Number of generated prototypes	9	6
Number of stored cases	20	25
Number of deleted cases	55	50
Number of adaptations	14	14

have a bigger chance to be adaptable remain in the case base, the number of retrieved adaptable cases slightly increases in comparison to the first setting of strategy A, but the number is not as high as in the other settings of strategy A.

So, the strategy of keeping those cases that are easily adaptable results in such a small case base that only few adaptable cases can be retrieved.

3.2.3. Strategy C: all cases remain in the case base

For strategy C, no cases are deleted at all. We have evaluated the same threshold parameter settings as for strategy A. It can be seen that many more adaptable cases can be retrieved in comparison to the corresponding settings of strategy C, while the memory requirement increases only slightly (Table 4).

Since two cases are sufficient to generate a prototype in the first setting, many prototypes are created and the memory requirement increases correspondingly. It is a little surprising that fewer adaptable cases are retrieved, but this is because a hierarchy with three levels of prototypes has been generated, and since the prototypes are treated as cases, the right prototype on each level has to be determined to be the most similar case.

Really surprising is the big increase of retrieved adaptable cases in the third setting. There are two possible explanations. Firstly, as the number of generated prototypes decreases, the prototype hierarchy is simpler and it is easier to find the appropriate case. Secondly, and probably the main reason, the number of cases which are necessary to generate a prototype is higher (four), so that more cases are considered when a generated prototype is filled, and consequently fewer contraindications are incorporated into the

Table 4
Test results for strategy C

	Setting number of cases = 2	Setting number of cases = 3	Setting number of cases = 4	Setting number of cases = 4
Memory size (MB): after 75 cases	2.439	2.426	2.421	2.419
Number of prototypes	19	10	8	7
Number of stored cases	75	75	75	75
Number of deleted cases	0	0	0	0
Number of adaptations	29	32	52	51

prototype. This means the prototypes themselves become more adaptable. However, when the number of generated prototypes decreases, there are fewer cases available to be used for adaptation (4th Setting).

3.2.4. Summary of the evaluation results for the prototype strategies

Keeping all cases in the case base increases the memory requirement, but increases the number of retrieved adaptable cases dramatically. Considering the number of retrieved adaptable cases, strategy A provides results that are nearly as good as for strategy C, but the achieved reduction is rather small. Keeping more adaptable cases (strategy B) results in a small case base, but only few adaptable cases can be found.

Too many prototypes should be avoided, because a complex hierarchy results in difficulties in finding the desired case. This means the threshold parameter “number of cases” should not be set too low.

The most preferable setting is the third one of strategy C. Only if the memory limitations become a real problem should strategies that delete redundant cases be considered.

4. Discussion

In this paper, we have briefly presented the application of CBR in our antibiotic therapy adviser ICONS; we have compared two retrieval algorithms, and have presented results for different prototype strategies and different settings.

Simple indexing works much faster than the tree-hash algorithm. However, since under simple indexing the retrieval time increases with the number of stored cases and the tree-hash algorithm does not depend on the number of stored cases, both algorithms should have equal retrieval times when few thousands of cases are stored.

As physicians reason with prototypical and exceptional cases anyway and as medical knowledge based systems should take the reasoning of physicians into account [20], the creation of prototypes seems to be an appropriate learning technique for medical domains. So far only few CBR approaches to therapeutic tasks are known. Some of them also use prototypes (e.g. [12,13,21]), some apply schemata (e.g. [16]).

However, the uniqueness of our prototype architecture results from the fact that the attributes, which are used to determine similarity, are not responsible for the generation, but the restriction of solutions.

The aim of our concept of prototypical cases is to structure the case base, to keep the prototypes always up to date and to erase redundant cases. The best strategy to find many adaptable cases is obviously to keep all cases in the case base. However, every stored case increases the memory requirement of our system by approximately 1.7KB. This might lead to performance problems for much bigger case bases, keeping in mind that our test set of 75 cases covers just one out of a set of more than 80 medical areas.

The best settings, whether all cases are retained (strategy C) or cases without additional information (strategy A) are deleted, are those where for the threshold parameter “number of cases” is set sufficiently high. This results in more retrieved adaptable cases.

Acknowledgements

We thank Robin Boswell at the School of Computer and Mathematical Sciences, The Robert Gordon University, Aberdeen, Scotland, for helping with the English language. The German Ministry for Research and Technology funded the earlier part of this research. It was part of the MEDWIS project of the MEDIS Institute of GSF, Neuherberg, for research on medical knowledge bases. The current part of this research is funded by the German Research Society (DFG).

References

- [1] Bueno-Cavanillas A, et al. Influence of nosocomial infection on mortality rate in an intensive care unit. *Crit Care Med* 1994;22:55–60.
- [2] Heindl B, et al. A case-based consiliarius for therapy recommendation (ICONS): computer-based advice for calculated antibiotic therapy in intensive care medicine. *Comput Methods Prog Biomed* 1997;52:117–27.
- [3] Aamodt A, Plaza E. Case-based reasoning: foundational issues, methodological variations, and system approaches. *Artif Intel Commun* 1994;7(1):39–59.
- [4] Stottler RH, Henke AL, King JA. Rapid retrieval algorithms for case-based reasoning. In: *Proceedings of the International Joint Conference on Artificial Intelligence. IJCAI-89*, Detroit, Michigan, 1989. p. 233–37.
- [5] Tversky A. Features of similarity. *Psychol Rev* 1977;84:327–52.
- [6] Smyth B, Keane MT. Retrieving adaptable cases: the role of adaptation knowledge in case retrieval. In: Richter MM, et al., editors. *Proceedings of the 1st European Workshop on Case-Based Reasoning*. University of Kaiserslautern, Kaiserslautern, 1993. p. 76–81.
- [7] Smyth B, Keane MT. Adaptation-guided retrieval: questioning the similarity assumption in reasoning. *Artif Intel* 1998;102:249–93.
- [8] Schank RC. *Dynamic memory: a theory of learning in computer and people*. New York: Cambridge University Press, 1982.
- [9] Bareiss R. *Exemplar-based knowledge acquisition*. San Diego: Academic Press, 1989.
- [10] Evans CD. A case-based assistant for diagnosis and analysis of dysmorphic syndromes. *Med Inform* 1995;20:121–31.
- [11] Turner R. Organizing and using schematic knowledge for medical diagnosis. In: *Proceedings of Case-Based Reasoning Workshop*. San Mateo: Morgan Kaufmann, 1988. p. 435–46.
- [12] Bichindaritz I. From cases to classes: focusing on abstraction in case-based reasoning. In: Burkhardt H-D, Lenz M, editors. *Proceedings of the 4th German Workshop on Case-Based Reasoning*, University of Berlin, Berlin, 1996. p. 62–9.
- [13] Bellazzi R, Montani S, Portinale L. Retrieval in a prototype-based case library: a case study in diabetes therapy revision. In: Smyth B, Cunningham P, editors. *Proceedings of 4th European Workshop on Case-Based Reasoning*. Berlin: Springer, 1998. p. 64–75.
- [14] Gierl L, Stengel-Rutkowski S. Integrating consultation and semi-automatic knowledge acquisition in a prototype-based architecture: experiences with dysmorphic syndromes. *Artif Intel Med* 1994;6:29–49.
- [15] Rosch E, Mervis CB. Family resemblances: studies in the structure of categories. *Cognitive Psychol* 1975;7:573–605.
- [16] Schmidt R, Pollwein B, Gierl L. Experiences with case-based reasoning methods and prototypes for medical knowledge-based systems. In: Horn W, et al., editors. *Proceedings of Artificial Intelligence in Medicine*. Berlin: Springer, 1999. p. 124–32.
- [17] Lenz M, Auriol E, Manago M. Diagnosis and decision support. In: Lenz M, et al., editors. *Case-based reasoning technology, from foundations to applications*. Berlin: Springer, 1998. p. 51–90.
- [18] Wess S, Althoff K-D, Derwald G. Improving the retrieval step in case-based reasoning. In: Richter MM, et al., editors. *Proceedings of 1st European Workshop on Case-Based Reasoning*. University of Kaiserslautern, Kaiserslautern, 1993. p. 83–8.

- [19] Kolodner J. Case-based reasoning. San Mateo: Morgan Kaufmann, 1993.
- [20] Strube G, Janetzko D. Episodisches Wissen und fallbasiertes Schließen: Aufgaben für die Wissensdiagnostik und die Wissenspsychologie. *Schweizerische Zeitschrift für Psychologie* 1990;49:211–21.
- [21] Camargo KG, et al. Designing nutritional programs with case-based reasoning. In: Gierl L, et al., editors. *Proceedings of 6th German Workshop on Case-Based Reasoning*. University of Rostock, Rostock, 1998. p. 141–7.

Case-Based Reasoning (CaBR) systems, by their nature, have a built-in set of test cases in their case library. Effective use of this internal feature can facilitate the validation process by minimizing the involvement of domain experts in the process. This can reduce the cost of the validation process, and eliminate the subjective component introduced by experts. This article proposes a validation technique which makes use of the case library to validate the CaBR system. Called the *Case Library Based Test Technique* (CLST), it evaluates the correctness of the retrieval and adaptation functions of the CaBR engine with respect to the domain represented by the case library. It is composed of two phases, 1) the Retrieval Test, and 2) the Adaptation Test. A complete description of the technique as well as an application of the technique to validate an existing CaBR system are discussed in this paper.

1. INTRODUCTION

Validation of knowledge-based system has received considerable attention from researchers in the last several years [Gupta, 1995]. The importance of ensuring that fielded knowledge-based systems operate correctly and as intended has been recognized by developers as well as users. However, the majority of the reported validation work to date has centered around rule-based systems. This may be because in comparison to other artificial intelligence techniques, rule-based expert systems are the most mature as well as the most commercially available. In fact, the majority of the fielded systems in existence today are of this type.

Published literature that deals with validation of Case-Based Reasoning (CaBR) systems is indeed scarce. O'Leary addresses the problem of CaBR validation in his 1993 article [O'Leary, 1993], and provides a readable overview of the problem by discussing the issues involved. We skip such discussion and refer interested reader to that source for an in-depth analysis of these issues. However, O'Leary stops short of actually proposing and testing a detailed evaluation technique. This paper describes a technique that can be used to validate a Case-based Reasoning system which is intended for involvement of an expert.

In addition to the work by O'Leary cited above, Simoudis [1990] contained simple retrieval with domain specific validation of retrieved cases to produce a tool for CaBR systems. The validation, however, was not considered as an independent phase in the system development. It is designed into the retrieval phase, and is called a *validated retrieval model* in CaBR.

Farr [1993] investigated systematic examination of the design decision in a CaBR system. He emphasized the complexity of the domain choice for a case-based system and system behavior criteria.

The above methods described required the derivation of a complex mathematical model to serve as the validation criteria. Additionally, other published validation efforts for case-based systems, Protes, HYPO, and Clavier, (as discussed in [O'Leary, 1993]) made extensive use of experts. None of these systems take advantage of the unique characteristic of CaBR, which is that the expertise is built-in from explicit historical cases.

VALIDATION TECHNIQUES FOR CASE-BASED REASONING SYSTEMS

Abstract

Case-Based Reasoning (CaBR) systems, by their nature, have a built-in set of test cases in their case library. Effective use of this unusual feature can facilitate the validation process by minimizing the involvement of domain experts in the process. This can reduce the cost of the validation process, and eliminate the subjective component introduced by experts. This article proposes a validation technique which makes use of the case library to validate the CaBR system. Called the *Case Library Subset Test Technique* (CLST), it evaluates the correctness of the retrieval and adaptation functions of the CaBR engine with respect to the domain as represented by the case library. It is composed of two phases, 1) the Retrieval Test, and 2) the Adaptation Test. A complete description of the technique, as well as an application of the technique to validate an existing CaBR system are discussed in this paper.

1.0 INTRODUCTION

Validation of knowledge-based system has received great attention from researchers in the last several years.[Gupta, 1992] The importance of ensuring that fielded knowledge-based systems operate correctly and as intended has been recognized by developers as well as users. However, the majority of the reported validation work to date has centered around rule-based systems. This may be because in comparison to other artificial intelligence techniques, rule-based expert systems are the most mature as well as the most commercially available. In fact, the majority of the fielded systems in existence today are of this type.

Published literature that deals with validation of Case-Based Reasoning (CaBR) systems is indeed scarce. O'Leary addresses the problem of CaBR validation in his 1993 article [O'Leary, 1993], and provides a valuable insight into the problem by discussing the issues involved. We skip such discussions and refer the interested reader to that source for an in-depth analysis of these issues. However, O'Leary stops short of actually proposing and testing a detailed evaluation technique. This paper describes a technique that can be used to validate a Case-based Reasoning system with little need for involvement of an expert.

In addition to the work by O'Leary cited above, Simoudis [1990] combined simple retrieval with domain specific validation of retrieved cases to produce a tool for CaBR systems. The validation, however, was not considered as an independent phase in the system development. It is designed into the retrieval phase, and is called a validated retrieval model in CaBR.

Ram [1993] investigated systematic evaluation of the design decision in a CaBR system. He emphasized the complexity of the domain choice for a case-based system and system behavior criteria

The above methods described required the derivation of a complex mathematical model to serve as the validation criteria. Additionally, other published validation efforts for case-based systems, Protos, HYPO, and Clavier, (as discussed in [O'Leary, 1993]) made extensive use of experts. None of these systems take advantage of the unique characteristic of CaBR, which is that the expertise is built-in from explicit historical cases.

Taking advantage of this feature of CaBR systems, Yi [1995] developed a set of algorithms in her work to build and validate a case-based reasoning system to help predict software development cost. The retrieval and adjustment algorithms in the case library were implemented to meet a specified *Minimum Relative Error (MRE)*. The MRE is the percentage difference of the system estimation to real software cost. However, her work came short of actually developing a full validation technique for general use.

The developers of Battle Planner (as discussed in [O'Leary, 1993]) also make use of the case library as a source of expertise by using some of the cases for testing. We expand upon this idea in our work.

The gold standard in most knowledge-based system validation efforts is considered to be the expert's knowledge. Some problems with this criteria, however, is that it is typically quite costly to involve experts due to their general unavailability and high salaries. The research presented in this paper, however, minimizes the need for intensive domain expert involvement in the validation process. The gold standard chosen for validation will be the case library itself, as it represents the explicit collection of historical results. The technique, called the *Case Library Subset Test, (CLST)* uses Yi's Relative Error (RE) as the comparison medium. It does, however, use experts to a small degree in determining the validation criteria to be employed. However, as this is typically within the purview of the user/purchaser, and not of the development team, we can safely state that expert involvement is not necessary for this procedure. This novel technique is evaluated in the re-validation of the *Case-Based Appraiser (CBA)* [Gonzalez, 1992; Laureano-Ortiz, 1990], a CaBR system used to appraise real estate property.

2.0 CASE LIBRARY SUBSET TEST METHODOLOGY

This section describes the proposed validation technique for CaBR systems called the *Case Library Subset Test (CLST)* technique.

The main concept underlying this validation method is the selection of a subset of cases from the case library and using this subset as a test set to evaluate the effectiveness of the system's retrieval and adaptation features. The comparison standards of the test set are considered to be correct because they are part of the case library. But first, the validation criteria has to be selected, as it affects the final correctness of the systems. This process is described below.

2.1 Determination of Validation Criteria

The first task is to develop a validation criteria. This consists of determining two basic parameters, the *Result Acceptability Criteria (RAC)*, and the *System Validity Criteria (SVC)*. The RAC serves to determine whether an individual test case has been solved correctly by the CaBR system. It mandates that the distance between the system solution to a test case, and the benchmark standard to which it is compared be calculated. If the solution is provided in numerical terms, then the Relative Error (RE) can be the percent difference between the two quantities. If, on the other hand, the output of the CaBR system is symbolic or Boolean, then *optimal*, *acceptable* and *unacceptable* solutions may be defined as the benchmark standard may allow. The SVC serves to determine whether, in light of the executed and evaluated suite of test cases, the system can be considered valid. The SVC requires that upon completion of all testing, the percent of all acceptable test cases be greater than its value before the CaBR system can be considered valid. The RAC and the SVC are typically obtained from either experts or users, and it may be defined in the requirements specification. Upon selection of the above validation criteria, the CLST technique begins as described below.

2.2 Description of the Case Library Subset Test

These are described in more detail below.

2.2.1 CaBR Retrieval Test.

Case indexing and case classification issues are intended to improve the effectiveness and efficiency of case retrieval and to reduce the complexity of similarity calculations. The correctness of the retrieval process is, therefore, one of major concerns in CaBR systems. The CaBR Retrieval Test is designed to evaluate the correctness of the retrieval function. The indexing system used, although not evaluated independently, is clearly part of the retrieval evaluation test, and deficiencies in indexing will show up as poor retrieval performance. The comparison function is also likewise validated.

Briefly, the Retrieval Test requires that each historical case in the case library "spawn" a test case identical to itself in all ways. A pointer to the historical case is maintained for the purpose of comparison later. This process generates a set of test cases, not only for the retrieval test, but also for the adaptation test as will be seen later. As part of the retrieval test, each test case is, in turn, presented to the CaBR system as the current case. The CaBR system goes through the comparison and retrieval processes, arriving at an internal list of library cases ranked in decreasing order of similarity. In order for any test case to be marked as successfully executed, the historical case which spawned the current test case should be found as the top-ranked historical case in this internal list, and the similarity distance should be the minimum allowed in the chosen measuring scheme (or very close to it).

2.2.2 CaBR Adaptation Test

The Retrieval Test ensures that the comparison and retrieval functions are correctly carried out. It is the purpose of this test to ensure that adaptations are properly made from valid retrieved cases. Therefore, the Adaptation Test should only be done after a successful Retrieval Test.

The test case set used here is the same as that of the retrieval test (e.g., spawned from each historical case in the case library). The significant difference is that in the Adaptation Test, the historical case corresponding to the test case being presented to the CaBR system is removed from the case library. Thus, if a case library has N cases in it, the modified case library will only contain N-1 cases at all times. The outputs of this test include retrieved cases, the final solution, and its RE. Although the test case is not longer in the case library, the CaBR system retrieves the most similar case from the case library and adjusts the closest matching case(s) with the adaptation strategies to obtain the final solution to this test case. Since the retrieval process has already been validated, this test isolates and evaluates the adaptation process of the CaBR system

3.0 IMPLEMENTATION AND EVALUATION OF THE CASE LIBRARY SUBSET TEST TECHNIQUE

It is important that any new concept in science and engineering be evaluated to determine its effectiveness in solving the problem it addresses. An evaluation technique should be no exception. In this section we briefly describe the steps taken to evaluate the CLST technique described in the previous section. The testbed chosen to carry out this evaluation is a CaBR prototype system for residential property appraisal called the *Cased-Based Appraiser (CBA)* [Gonzalez, 1992; Laureano-Ortiz, 1990]

3.1 The Case-Based Appraiser System

An prototype that automates property appraisal using a CaBR

approach therefore was developed by Laureano-Ortiz [1990]. Several attributes in the cases are used to calculate the price of the property. Some of these are the living area, number of bedrooms, number of bathrooms as well as others.

The CBA System works by determining the most similar cases to the current property, adjusting these cases to account for any remaining similarities, and then obtaining the appraised value using one of two widely accepted methods in property appraisal. Refer to [Laureano-Ortiz 1990] and to [Gonzalez, 1992] for more details on this system.

3.2 Determination of Validation Criteria

The task of determining the validation criteria was the first undertaken. A questionnaire was sent to individuals knowledgeable in the field of appraisal with the following questions:

Question 1: "What is the maximum acceptable Relative Error of a CBA system?" The maximum acceptable error range refers to the percent difference between appraised price and the actual real sold price. This corresponds to the Result Acceptability Criteria (RAC) defined in Section 2 above

Question 2: "What percentage of the correct appraised cases in a CBA system is considered reasonable?" This question asks for the Correctness Ratio (CR) in all appraised cases. This criteria corresponds to the System Validity Criteria (SVC) seen in section 2 above

In regards to the RE, the majority of the responders felt that 20% was appropriate, based on the limited set of attributes considered. Human bargaining, the seller's economic situation, and the various marketing factors are not considered in the CBA analysis. Yet, the actual price is strongly affected by those factors. Therefore, it was decided that 20% RE would be set as the RAC (the threshold between acceptable and unacceptable results from the CBA).

Likewise, the total CR was determined to be a minimum of 80% SVC for a valid system. That is, 80% of the subject properties to be appraised were valued at a price less than 20% different than the actual sale price. It is necessary to note here, however, that the CBA system sometimes displays a dummy "-1" as the result when the subject case does not have any similar cases in the case library (e.g., little similarity between the most similar historical case and the current case). This can be quite a normal occurrence in CaBR systems, and these subject properties should not be considered when determining the CR.

4.0 EVALUATION OF RESULTS FOR THE CASE LIBRARY SUBSET TEST TECHNIQUE

This section evaluates a rewritten version of the CBA prototype (in C/C++) with the Case Library Subset Test validation model. Since the new C/C++ prototype used exactly the same algorithms and data types as the original lisp-based version by Laureano-Ortiz, we shall assume that they functionally identical. The CLST validation model is designed to empirically demonstrate that the CBA system works correctly for property appraisal. However, the true purpose of this section is to evaluate the validity and usefulness of the Case Library Subset Test Technique itself, as its effectiveness in validating the CBA will be compared with Laureano-Ortiz's [1990] original expert-based validation of the lisp-based CBA. We refer the reader to [Xu, 1995] for all the raw data pertaining to the results shown in this section.

4.1 Retrieval Test

The results of the Retrieval test on the CBA case library showed that in

100% of the test cases, the case in the case library corresponding to the test case was chosen as the closest match. This indicates that retrieval was done properly. Furthermore, the average RE for the Retrieval Test was 8.239%. The reader should note that the appraisal value is computed by averaging the several most similar cases in the library, rather than exclusively using the most similar one, even if it is identical. Thus, the RE for this test with the appraisal testbed should not be expected to be 0%.

4.2 Adaptation Test

In the Adaptation test, the retrieval technique is also inherently evaluated, as the RE is also calculated. However, the emphasis here is on the adaptation aspect. The average RE for the Adaptation test was computed to be 13.2057%. A higher RE than for the Retrieval Test is to be expected, as the case base is somewhat less similar to the test cases by virtue of removing the case corresponding to the test case.

4.3 Original, Expert-based Validation of CBA

Laureano-Ortiz [1990] evaluated his original CBA system according to the traditional method of comparing the CBA's output to the domain expert's appraisal results for the same set of test cases. In his validation exercise, seventy (70) test cases were presented to the domain experts, and their appraised values for those test cases were recorded. The same test cases were presented to the original CBA, and its results were recorded. Using the original data, the RE for each of the 70 test cases was computed as part of the present investigation.

The relative error (RE) was computed for the results obtained in the original CBA validation. We found that 11.4% of the test cases had a computed RE of more than 20%. This calculation excluded cases that resulted in a "-1" as indicated above. We also calculated the average RE of the original CBA validation test (excluding the "-1" answers) to be 9.91123%.

The average RE of the Test Case set used by Laureano-Ortiz (computed to be 9.9%) was deemed to be acceptable by the experts that were involved in the original validation process. The case library (107 cases) and the test case set (70 cases) used in that evaluation did not have any cases in common. However, they were obtained from real-world data, and thus realistic in their makeup. Laureano-Ortiz [1990] concluded his CBA system test by stating that the "CBA does a good job given the limitations it has in its condition of prototype in its early stage: small case base, small number of represented figures and lack of better sources of cases."

In the CLST, the same case library used in the original evaluation (107 cases) was employed. However, the test case set was the case library itself. Since the case library is real-world data, it can be said that these test case sets were generally of equivalent makeup as the 70 case test set used in the original evaluation. Thus, the use of the case library itself as the source of test cases can be considered to be equally realistic as the original (70 case) test case set, and thus acceptable. The average RE was computed to be 8.2% for the Retrieval Test, and 13.2% for the Adaptation Test. Since the Adaptation Test is more similar to the original test run by Laureano-Ortiz than the Retrieval Test, that is the number to which we compared the performance of the CLST technique. However, it should be noted that the Adaptation Test always removes the most similar of cases in the case library, thus introducing a slight disadvantage. Nevertheless, regardless of which number is used for comparison, the numbers are quite close to each other, suggesting strongly that the CLST technique is an effective way of validating CaBR systems.

5.0 SUMMARY AND CONCLUSION

In this investigation, a new method called Case Library Subset Test is presented for the purposes of validating Case-Based Reasoning Systems without the need for involving a domain expert. A prototype system which carries out the CLST testing technique automatically was designed, built and successfully demonstrated on a testbed CaBR system, the Case-Based Appraiser (CBA) for appraising single family residential property.

The implementation of Case Library Subset Test techniques presented here is a realization of a new validation idea. We believe that the methodology presented here is not only applicable to small CaBR system like the CBA, but also to validation of more complex systems.

6.0 REFERENCES

- [Gonzalez, 1992] Gonzalez, A. J. and Laureano-Ortiz, R., (1992) "A Case-Based Reasoning Approach to Real Estate Property Appraisal", *Expert Systems with Applications*, Vol. 4, No. 2, pp. 229-246.
- [Gupta, 1992] Gupta, U. G. (ed), (1992) Validating and verifying knowledge-based systems, Las Alamedas, CA: IEEE Computer Society Press.
- [Laureano-Ortiz, 1990] Laureano-Ortiz, R., (1990) "Application of case-based reasoning techniques to the automation of single-family residential property appraisals", Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, December.
- [O'Leary, 1993] O'Leary, D. E., (1993) "Verification and validation of case-based systems" *Expert Systems with Applications*, Vol. 6, pp. 57-66.
- [Ram, 1993] Ram, A., "Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases", *Machine Learning*, 1993, pp. 7-54.
- [Simoudis, 1990] Simoudis, E. and Miller, J., (1990) "Validated retrieval in case-based reasoning", *Proceedings of the 1990 National Conference on Artificial Intelligence*, AAAI Press, pp. 310-317.
- [Xu, 1995] Xu, L., (1995) "Validation techniques for case-based reasoning systems", Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, Fall Term.
- [Yi, 1995] Yi, Y., (1995) "CASHEEW", Master's thesis, University of Central Florida of Department of Electrical and Computer Engineering, Spring Term.

Avelino J. Gonzalez / Lingli Xu

Electrical and Computer Engineering Dept.
University of Central Florida
Orlando, FL 32816-2450

Uma M. Gupta

Decision Sciences Department
East Carolina University
Greenville, NC 27858-4353