



كلية الدراسات العليا



Sudan University of Science and Technology

College of Graduate Studies

A Comparison between Similarity Metrics in Case-Based Reasoning Systems (Applied to Self Healing in Mobile Bank Applications)

مقارنة بين معايير التشابه في أنظمة الاستنتاج المعتمد على الحالة (بالتطبيق على الشفاء الذاتي في تطبيقات الهاتف المحمول)

A thesis Submitted in Partial Fulfillment of the Requirements of M.Sc in Information Technology

Prepared by

Doha Nadir Ezeldeen Idries

Supervisor:

Dr .Nadir Kamal Salih

December 2020

الآية :-

قال الله تعالى :

﴿ وَمَا أَرْسَلْنَا مِنْ قَبْلِكَ إِلَّا رِجَالًا نُوحِي إِلَيْهِمْ فَاسْأَلُوا أَهْلَ الذِّكْرِ إِنْ كُنْتُمْ لَا تَعْلَمُونَ ﴾

سورة النحل الايه: 43

Dedication

**To My Mom's Precious Soul ... God's Mercy on Her...To My Dear Father to My
Dear Brother, Sister and Friends.**

A Special Dedication To My Grandmother (Salma Yousef Sukkar).

To My Supervisor Dr: Nadir Kamal Salih.

To All My mates, And Everyone Who Helped Me Complete This Work.

Thank You So Much

Acknowledgement

Thank God, Glory be to Him, who has enabled us to complete this study and who favors good works. I extend my sincere thanks to Sudan University of Science and Technology and distinguished professors in particular.

I would like thank Dr. **Nader Kamal**, for accepting the supervision of this study and devoting his precious time to guide me through this work.

I also thank Dr. Halima Mustafa Elbasheir for her support throughout the study.

ABSTRACT

Mobile bank applications provide many services to customers and this study supports the model of simulation of the application's performance.

There are many problems threatening applications, such as reliability, connection and confidentiality problems, and problems related to the system's behavior during service interruptions or lack of availability at the time specified for the customer, and these are the main problems that affect the reliability of the system.

In This research, the main concept is autonomic system with its different parts in general ,and self-healing in more details, It Provides a mechanism for self-Healing using an inference algorithm based on previous cases (Case-based reasoning). The algorithm enables the model of simulation system to detect and identify the problem and then self-treat it, and it is able to examine and treat itself without external interference. This mechanism is applied to the mobile bank application as a case study of an application that contains sensitive data. This algorithm was implemented using Eclipse using Java language and used the case database to save a set of cases similar to the occurrence of the problem, The algorithm selects the best solution to solve the problem among the set of available solutions using a set of functions to calculate the similarity distance between these cases and accuracy and error rate.

المستخلص

تطبيق بنك الهاتف المحمول يقدم العديد من الخدمات للعملاء وهذه الدراسة تدعم نموذج محاكاة اداء التطبيق باستخدام خوارزميه الاستدلال المبني على الحالات .

هناك العديد من المشكلات التي تهدد التطبيقات، مثل الموثوقية، ومشكلات الاتصال ، ومشكلات سلوك النظام خلال انقطاع الخدمة أو عدم توفرها في الوقت المحدد للعميل ، وهي من أكثر المشكلات التي تؤثر على موثوقية النظام.

يوفر هذا البحث مفهوم عام عن الانظمة اللارادية بصورة عامة واستخدام مفهوم الشفاء الذاتي بصوره خاصه واستخدام آلية تعمل للعلاج ذاتيًا باستخدام خوارزمية الاستدلال المبني على الحالات السابقة (Case-based reasoning).

تمكن الخوارزمية نموذج محاكاة النظام من اكتشاف وتحديد المشكلة ومن ثم معالجتها بنفسه، وسيكون قادرًا على فحص ومعالجة نفسه دون تدخل خارجي. يتم تطبيق هذه الآلية على تطبيق البنك المحمول كدراسة حالة لتطبيق يحتوي على بيانات حساسة. تم تنفيذ هذه الخوارزمية على برنامج Eclipse باستخدام لغة Java في البرمجة، واستخدمت قاعدة بيانات الحالة لحفظ مجموعة من الحالات المشابهة لحدوث المشكلة. وتعمل الخوارزمية على استنتاج أفضل خيار لحل المشكلة من بين مجموعة الحلول المتاحة. باستخدام مجموعة من الوظائف لحساب التوافق والدقة بين هذه الحالات .

Content

-: الأية.....	I
Dedication.....	II
Acknowledgement.....	III
ABSTRACT.....	IV
المستخلص.....	V
List of Tables.....	IX
List of Abbreviations.....	X
CHAPTER I.....	1
1 INTRODUCTION.....	1
1.1 Background.....	1
1.2 Problem statement.....	1
1.3 Proposed Solution.....	1
1.4 Methodology.....	2
1.5 Objectives.....	2
1.6 Research Questions.....	2
1.7 Scope.....	2
1.8 Thesis Outlines.....	3
Chapter 2.....	4
2 Literature Review.....	4
2.1 Introduction.....	4
2.2 Self-Healing.....	4
2.2.1 Autonomic Computing Systems and Applications.....	5
2.2.2 Application quality in self -healing:.....	7
2.2.3 Autonomic Computing Research Issues and Challenges.....	7
2.2.4 Self- Healing type:-.....	8
2.2.4.1 Self- Healing in networks:.....	8
2.2.4.2 Self -Healing web service:-.....	9
2.2.4.3 Self-healing Smartphone software via automated patching:-.....	9
2.3 Case-based reasoning algorithm Model.....	13
2.4 Developing and Testing Environment.....	13

2.5	General background of CBR.....	13
2.5.1	CBR using database technology:.....	14
2.5.2	Components of CBR.....	14
2.5.2.1	Case:.....	14
2.5.2.2	Case base:-	14
2.5.2.3	Case index:.....	14
2.5.3	Solving the problem by using CBR.....	15
2.5.4	This cycle comprises four activities.....	15
2.5.5	CBR CASES	17
2.5.5.1	Retrieval Case:.....	17
2.5.5.2	Reuse and revision in CBR Case	18
2.5.5.3	Revise case.....	18
2.5.5.4	Retain case.....	18
2.5.6	Benefit and limitations of CBR:	19
2.6	Previous studies:-	19
2.6.1	Self-management:-.....	20
2.6.2	Self-managing computing system:-	20
2.6.3	Implementing an automation computing evaluation scale to generate recommendations:-.....	20
2.7	Summary	22
CHAPTER 3		23
3	Methodology	23
3.1	Introduction.....	23
3.2	Case-based reasoning.....	23
3.3	The functions used to calculate the similarity distance between these cases and accuracy: ..	27
3.4	Root Mean Square Error (RMSE):-	29
3.5	Software	29
3.6	Summary	30
CHAPTER 4		31
4	Implementation of self-healing in bank system.....	31
4.1	Introduction.....	31

4.2	Development Environment	31
4.3	System of Interface	31
4.4	Attribute weights interface:-	34
4.5	The new cases of self-healing system and test interface:-	35
4.6	Result Interface	36
4.7	Self-Healing using CBR algorithm model: -	40
4.8	Problem solving and result:-	40
4.9	Accuracy measurement.....	41
4.10	Root Mean Square Error (RMSE):-	45
4.11	Discussion.....	47
4.12	Summary	47
CHAPTER 5		48
5	Conclusions and recommendations.....	48
5.1	Conclusions.....	48
5.2	Recommendations.....	48
References:-		49

List of Tables

Table 2. 1 Four aspects of self-management as they are now and would be with autonomic computing	4
Table 2. 2 previous works.....	10
Table 3. 1 four important attributes in self healing algorithm.....	30
Table 4. 1 values of new and old cases.....	37
Table 4. 2 Manhattan function calculation code.....	37
Table 4. 3 Euclidean function calculation code.....	37
Table 4. 4 Canberra Function calculation code	38
Table 4. 5 Squared Chord function calculation code.....	39
Table 4. 6 Squared chi-squared function calculation code	40
Table 4. 7 show the similarity distance	41
Table 4. 8 shows the error rate for all similarity functions when diagnosis case. The error rate percentage has been calculated using RMSE formula.....	46
Table 4. 9 Shows the error rate for all similarity functions when diagnosis cases the error rate percentage has been calculated using RMSE formula.....	46

List of Abbreviations

Figure 2. 1 Top 8 mobile operating systems evolution between March 2011 – March 2012	5
Figure 2. 2 an autonomic element.....	7
Figure 2. 3 the CBR cycle.....	16
Figure 2. 4 Relationship between problem and solution spaces in CBR, Adapted from (Leake, 1996b).....	17
Figure 3.1 case base organization.....	23
Figure 3. 2 The flow chart of interruption of the process in the system and the recovery of a similar case.....	24
Figure 3. 3 mobile bank framework using CBR.....	25
Figure 3.4 the sequence of interruption operation to buy electricity.....	26
Figure 4. 1 self –healing interfaces model.....	332
Figure 4. 2 Self-healing framework using CBR	33
Figure 4. 3 Show cases of the system	34
Figure 4. 4 attribute weights of self-healing system interface.....	35
Figure 4. 5 the new cases of self healing system and test interface.....	36
Figure 4. 6 result of self healing system interface using Manhattan function.....	36
Figure 4. 7 Result of similarity distance interface using Euclidean function	38
Figure 4. 8 Result of similarity distance interface using Canberra function	39
Figure 4. 9 Result of similarity distance interface using squared chord function	39
figure 4. 10 Result of similarity distance interface using squared chi-squared function.....	40
Figure 4. 11 Euclidean function accuracy	42
Figure 4. 12 Manhattan function accuracy	43
Figure 4. 13 Canberra function accuracy.....	43
Figure 4. 14 Chi-chord function accuracy	44
Figure 4. 15 Sugared chi-chord function accuracy	44
Figure 4. 16 Similarity functions accuracy rate.....	45
Figure 4. 17 Error rate of similarity functions using RMSE.....	47

CHAPTER I

1 INTRODUCTION

1.1 Background

Smartphone's with open source operating systems are becoming increasingly popular now a days. Android is one of the most popular open source operating systems for mobile platforms. Android offers a set of properties to protect phone applications, with the basic functionality of system utilities and middleware in form of Virtual Machine (VM) [1].

The materials and applications of self-healing are no longer illusion and we are not far from the days when human-made materials can restore their structural integrity in case of failure.

Self-healing is a part of the autonomic system and can be defined as the ability of a material to heal, recover or repair from the damages automatically and autonomously without any external intervention .Many common terms such as self-healing, autonomic healing, and self-repair are used to determine such property in materials [2].

1.2 Problem statement

Mobile application users suffer from many possible service interruptions due to loss of connection, server down, service not available or login problem .These problems affect the application's reliability, especially when the application deals with sensitive data such as mobile bank applications.

1.3 Proposed Solution

To develop a mobile bank application that applies the concept of self-healing on some behavioral problems. This autonomic system uses the (CBR) algorithm to solve the problem and uses dataset which was collected from Faisal Islamic Bank (FIB). And works to choose an optimal solution from among the set of solutions stored in the cases base. Comparison between a set of functions was used to measure the similarity and accuracy between these cases.

1.4 Methodology

Case-based reasoning (CBR) can mean adapting old solutions to meet new demands, using old cases to explain new situations; using old cases to discover new solutions, or reasoning from precedents to interpret a new situation.

1.5 Objectives

The objectives of the research are to:

1. Improve the healing mechanism to identify the problem without human intervention.
2. Treatment of the problem without human intervention.
3. Determines the optimal solution to solve the problem automatically.
4. Comparison of functions to measure similarity, accuracy and error rate.
5. Using the algorithm reduces the processing time for the behavioral errors that may occur in the system.

1.6 Research Questions

In this section we will explain the problems of research that can occur in applications and some of the tools that used to solve the problems which will be mentioned:

- **Research question 1:** How can self- healing concept be used in mobile application bank to detect, determine and solve errors?
- **Research question 2:** How can self-healing be used to protect applications and reduce service interruption in mobile bank applications?

1.7 Scope

In this thesis, five functions are used to calculate similarity distance, accuracy and error rate .These functions are: Manhattan, Euclidian, Canberra, Squared Chord and Squared Chi-Chord.

1.8 Thesis Outlines

The thesis is organized as follows:

Chapter One: Introduction

This chapter explains background information about self- healing and includes the problem statement, proposed solution and objectives.

Chapter Two: Literature Review

Consists of the literature review and evaluation of previous research of CBR technique in self healing in mobile application.

Chapter Three: Methodology

Is a guideline for how the problem was sloved with specific components of CBR such as phases, tasks, methods, techniques and tools using in the project.

Chapter Four: Implementation of self -healing

The implementation of the research project is presented; interfaces design, the result of the project and discussion of the outcomes of the project.

Chapter Five: Conclusions and Recommendations

The conclusions of the research and suggestions of some future work in order improve the project. This chapter will briefly summarize the overall project.

Chapter 2

2 Literature Review

2.1 Introduction

This chapter reviews the autonomic computing study and some previous studies that discuss the subject of self-management and the classification of self-healing.

2.2 Self-Healing

Ghosh, S.K. et al [2] Define self-healing as the ability of a material to heal recover or repair damages automatically and autonomously, that is, without any external intervention. Incorporation of self-healing properties in manmade materials very often cannot perform the self-healing action without an external trigger[2].

Table 2. 1 Four aspects of self-management as they are now and would be with autonomic computing:-

Concept	Current computing	Autonomic computing
Self-configuration	Corporate data centers have multiple vendors and platforms. Installing, configuring, and integrating systems is time consuming and error prone.	Automated configuration of components and systems follows high-level policies. Rest of system adjusts automatically and seamlessly.
Self-healing	Problem determination in large, complex systems can take a team of programmers weeks.	System automatically detects, diagnoses, and repairs localized software and hardware problems.
Self-protection	Detection of and recovery from and cascading failures is manual.	System automatically defends against malicious attacks or cascading failures. It uses early warning to anticipate and prevent system wide failures.
Self-optimization	Systems have hundreds of manually set, nonlinear tuning parameters, and their number increases with each release.	Components and systems continually seek opportunities to improve their own performance and efficiency.

The reliability, approach and structure in Autonomic Computing Systems and the self-healing concept:-

The reliability and security of software application is one of the most important characteristics of software quality because it describes the ability of the software to work without any failure and protect the user data. Mobile applications are compatible with desktop applications in terms of rich interfaces and functions, and become one of the most widely use applications. Mobile applications should provide special applications to avoid failure and maintain a high level of reliability and security because mobile operating systems provide limited services or access to administrative or regulatory tools that allow a user with an IT background to access temporary or permanent data. A high level of software reliability is directly affected by a low level of failure [3].

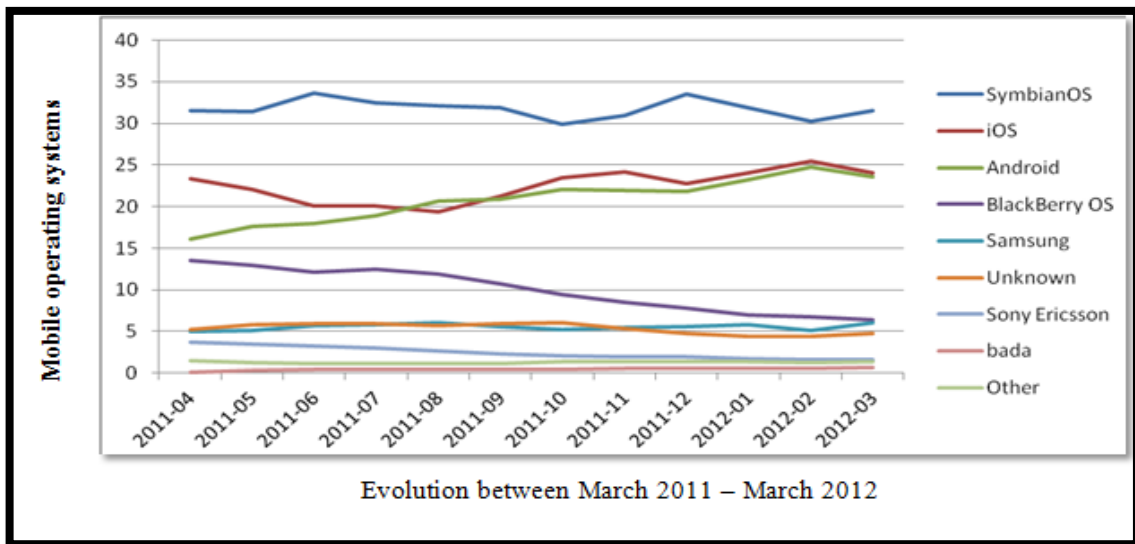


Figure 2. 1 Top 8 mobile operating systems evolution between March 2011 – March 2012

2.2.1 Autonomic Computing Systems and Applications

Autonomic applications and systems are composed from autonomic elements, and are capable of managing their behaviors and their relationships with other systems applications in accordance with high-level policies. [4]

Autonomic systems/applications exhibit eight defining characteristics [7]:

Self-Awareness: An autonomic application/system “knows itself” and is aware of its state and its behaviors. Self-Healing: An autonomic application or system should be able to detect and recover from potential problems and continue to function smoothly. Self-Configuring: An autonomic application system should be able to configure and reconfigure itself under varying and unpredictable conditions. Self-Optimizing: An autonomic application or system should be able to detect suboptimal behaviors and optimize itself to improve its execution. Self-Protecting: An autonomic application/system should be capable of detecting and protecting its resources from both internal and external attack and maintaining overall system security and integrity. Context-Aware: An autonomic application system should be aware of its execution environment and be able to react to changes in the environment. Open: An autonomic application/system must function in a heterogeneous world and should be portable across multiple hardware and software architectures. Consequently it must be built on standard and open protocols and interfaces. Anticipatory: An autonomic application/system should be able to anticipate to the extent possible, its needs and behaviors and those of its context, and be able to manage itself proactively.

Managed Element: This is the smallest functional unit of the application and contains the executable code (program, data structures) (e.g., numerical model of a physical process). It also exports its functional interfaces, its functional and behavior

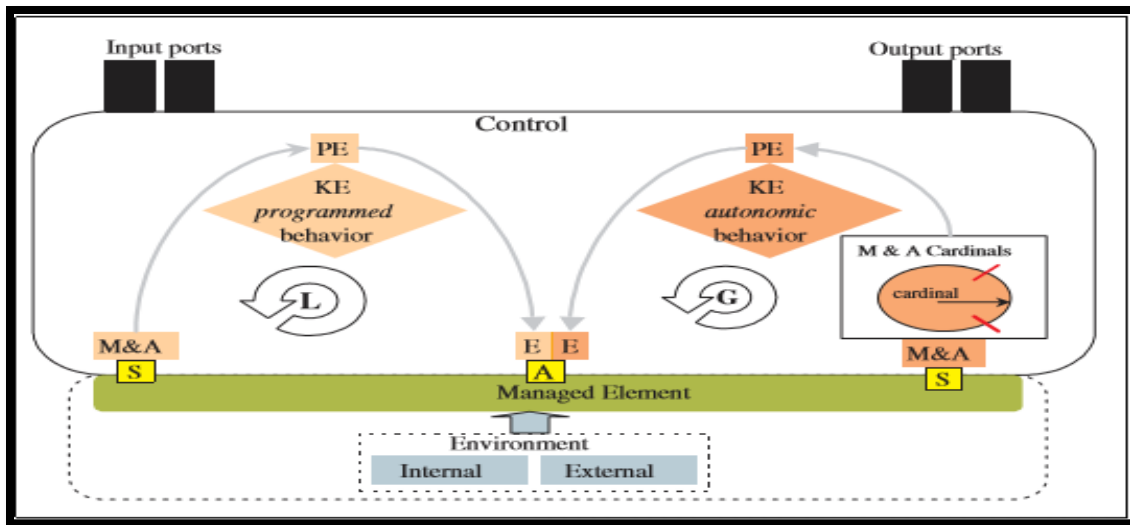


Figure 2. 2 An autonomic element

2.2.2 Application quality in self -healing:

Functionality: -

Is a set of functions that satisfy specific application needs in terms of appropriateness, the accuracy, interoperability, security and compliance with standards in the area.

Reliability: -

Is describes the application ability to work correctly in difficult or special conditions. High tolerance to errors and recoverability are two of the requirements that the application in question must have.

Usability: -

Is defined by the effort required to use the software by users, Operability is high because the time it produces the desired result is very short.

Efficiency: -

Software application is by the correlation optimum between the consumption of resources and complexity or difficulty of the problem to solve.

2.2.3 Autonomic Computing Research Issues and Challenges

Conceptual Challenges: Conceptual research issues and challenges include:

Defining appropriate abstractions and models for specifying, understanding, controlling, and implementing autonomic behaviors The following four aspects are very important for self-Healing [17]

Monitoring: the system should be monitored always to observe the normal comportment of the application.

Interpretation: is important to determine the moment when is a problem with the system, a mechanism to detect whether the system is changed in a bad way.

Resolution: is the proposed mechanism for repairing the system.

Adaptation or reconfiguration: is the final phase of self-healing and consist in implementing the mechanism for recovery the damaged modules of the system.

Architecture Challenges: Autonomic applications and systems will be constructed from autonomic elements that manage their internal behavior and their relationships with other autonomic elements in accordance with policies that humans or other elements have established.

MAPE cycle:-

The engine uses the Symptom database, an XML file that contains symptoms and recovery procedures. By searching for the symptom database, one or more routes can be found so that following these directions will eventually recover from the current situation.

Once the diagnostic drive has found some recovery procedures, The base engine is used as a Web service to map the MAPE cycle to determine actions that can be taken according to system policies[11]

2.2.4 Self- Healing type:-

2.2.4.1 Self- Healing in networks:

KATHLEEN S. TOOHEY et al [12] Self-healing polymers composed of microencapsulated healing agents exhibit remarkable mechanical performance and regenerative ability^{1–3}, but are limited to autonomic repair of a single damage event in a given location. Self-healing is triggered by crack-induced rupture of the embedded capsules thus, once a localized region is depleted of healing agent further repair is precluded. Re-mendable polymers^{4,5} can achieve multiple healing cycles, but require external intervention in the form of heat treatment and applied pressure .

Christopher J. Hansen et al [13] Micro vascular Substrate Fabrication: Micro vascular interpenetrating networks are fabricated using a three-axis robotic deposition stage (ABL9000, Aerotech Inc.) whose motion is controlled by customized software

(RoboCAD version 3.1). Wax and pluronic inks are housed in syringes that are fitted with 330 and 100mm nozzles (EFD Inc.) respectively.

2.2.4.2 Self -Healing web service:-

Self -healing :describes any device or system that has the ability to perceive that it is not operating correctly and, without human intervention, make the necessary adjustments to restore itself to normal operation.

Reliability : is the ability to perform independently of the current execution circumstances, which permits to guarantee business continuity.

How the functionality of a Web service is achieved?

The control behavior guides in a controlled way the progress of executing the operational behavior. Self- healing approach that is built upon a set of dedicated modules that would support the reliability of Web services. These modules are part of a “control interface” that ensures the monitoring of a Web service’s behavior, the catching of errors, and the recovery from these errors. The design of this control interface complies with Aspect-Oriented Programming (AOP) and Case-Based Reasoning (CBR). [14]

2.2.4.3 Self-healing Smartphone software via automated patching:-

Efficiency our approach incurs a one-time overhead for model extraction, via static analysis, to enable rollback point detection. Model extraction time is solely depending on the app's binary size and code complexity, and as it is a one-time cost incurred before running the app, we drove the apps to crash points via systematic exploration. Depending on the bug, exploration time will vary, though techniques such as targeted exploration [17] or fast forwarding record-and-replay [10] can significantly accelerate the procedure. The time-to-crash is presented For example, for Face book Mobile, the actual fault was in the initial login screen, and hence the systematic exploration time (4 seconds) was much lower than for the other apps. [5]

Table 2. 2 previous works

Table (2.2) shows the analysis of previous works:

Number of reference	Year	Title	Methodology	Result
5	2014	Towards self-healing smartphone software via automated patching	Frequent app bugs and low tolerance for loss of functionality create an impetus for self-healing smartphone software. We take a step towards this via on-the-fly error detection and automated patching. Specifically, we add failure detection and recovery to Android by detecting crashes and "sealing off" the crashing part of the app to avoid future crashes. In the detection stage, our system dynamically analyzes app execution to detect certain exceptional situations. In the recovery stage, we use byte code rewriting to alter app behavior as to avoid such situations in the future.	They take architecture Centered around detecting crashes and in response applying seal-off patches. A model is constructed rest, via static analysis on the app byte code); the model includes rollback (resume) points where the app will be driven when recovering from a fault. Next, the app is executed, either manually by users or automatically via systematic exploration tools and its execution logis monitored via dynamic analysis.
6	2008	Autonomic Management Policy Specification: From UML to DSM	Autonomic computing is recognized as one of the most promizing solutions to address the increasingly complex task of distributed environments' administration. We designed such a component-based autonomic management framework, but observed that the interfaces of a component model are too low-level and difficult to use.	in this paper, we present our experience in designing the Tune system and its support for management policy specification, relying on UML diagrams and on DSMLs. We analyse these two approaches, pinpointing the benefits of DSMLs over UML.
7	2013	Self –management (Computer Science)	Self-management is the process by which computer systems shall manage their own operation without human intervention. Self-management technologies are expected to pervade the next generation of network management systems.	four functional areas: Self-configuration Self-healing Self-optimization Self-protection

8	2004	Self-Managing Systems: A Control Theory Foundation	The properties of feedback systems are used in two ways. The first relates to the analysis. Here, we are interested in determining if the system is stable as well as measuring and/or estimating its steady state error, settling time, and maximum overshoot. The second is in the design of feedback systems. Here, the properties are design goals. That is, we construct the feedback system to have the desired values of steady state error, settling times, and maximum overshoot. More details on applying control theory to computing systems can be found in [14].	This paper explores the extent to which control theory can provide an architectural and analytic foundation for building self-managing systems, either from new components or layering on top of existing components. Further, we propose a deployable test bed for autonomic computing (DTAC) that we believe will reduce the barriers to addressing key research problems in autonomic computing. The initial DTAC architecture is described along with several problems that It can be used to investigate.
9	2011	Method, apparatus, and program for implementing an automation computing evaluation scale to generate recommendations.	A method, in a data processing system, for assessing automated computing capabilities, the method comprising: receiving information about automated computing capabilities of a customer; assigning a maturity level from a set of maturity levels to each of a plurality of assessment categories based on the information about the automated computing capabilities of the customer; and Providing information for achieving a target level of automated computing to the customer.	The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.
22	2018	Automatic software repair a bibliography. ACM Computing Surveys (CSUR), 51(1), pp.1-24.	<u>APPROACH :-</u> Detection: our framework works on a dynamic analysis of systems behavior and app output. Recovery. The recovery mechanism works in two phases. First, after detecting the crash point, we determine the safe retreat point. Second, we re-launch the app to a secure nearby	They introduced a methodology that uses automatic error detection and spot creation to provide a degree of self-healing capability for Android applications. The use of dynamic analysis to identify crash points and problems that can occur, static analysis for the determination of undo points, and binary rewriting of sealo_ routes

			<p>point so users can continue Work and interact with the application.</p> <p><u>IMPLEMENTATION:-</u> Platform. We implemented our approach and conducted experiments on a Motorola Droid Bionic phone, running An-droid 2.3.4</p> <p><u>Tools:-</u> Used the SATG extraction component (static analysis-based) of A3E. To drive exploration, we used the systematic exploration component of A3E. We wrote the main instrumentation code in Java.</p>	<p>associated with fault points so that applications can continue to operate even after a malfunction, albeit limited Careers. By testing the actual bugs in For many common applications, we show that our approach is e_ective e_cient.</p>
23	2002	Reflection, self-awareness and Self -healing in Open ORB. In Proceedings of the first workshop on Self-healing system	<p><u>Implementation :-</u> Were developed using Python, the architecture has been re-implemented with the explicit goal of provide a high performance implementation of our reflective middleware technology.</p>	<p>This paper provides an analysis of thinking and its potential in supporting self-healing systems. Openness and Re-initialization is a clear Programming models are used to build distributed applications.</p>
23	2002		<p><u>Ransomware:-</u> is a class of malware that encrypts valuable files found on the victim's machine and asks for a ransom to release the decryption key(s) needed to recover the plaintext files.</p> <p><u>Problem Statement :-</u> Were the first to analyze a large corpus of ransomware samples. The authors suggest that the file system is strategic point for monitoring the typical ransomware activity.</p>	<p>Proposed a way to make modern operating systems more flexible for malicious encryption Attacks, by detecting behaviors similar to ransom and return Its effects preserve the integrity of user data.</p>
24	2016	Automatic Runtime Recovery via Error Handler Synthesis	<p><u>APPROACH :-</u> This paper proposes Ares, a novel, practical approach for ARR. Our key insight is leveraging a system's inherent error handling support to recover</p>	<p>an automatic runtime recovery system implemented on top of the industry-strength Java Hot Spot VM and the Android ART VM. as it requires only minimal modifications to the</p>

			<p>from unexpected errors.</p> <p><u>Implementation :-</u> Implemented Ares on two popular platforms: Java Hot Spot VM and Android ART. This design decision enables our approach to be a drop-in substitute for standard VMs, and easy to deploy in production environments without complex configurations</p>	<p>runtime),and efficient (as it intercepts only the exception handling mechanism, incurring no overhead on normal program execution). Thus, it can be seamlessly integrated into production environments. it is able to successfully recover from 39 of the bugs. To ensure reproducibility, we have released all the source code and data used in this pa-per, and more details about our evaluation can be found.</p>
--	--	--	--	--

2.3 Case-based reasoning algorithm Model

In this research, the concept of a self-healing algorithm was applied and the use in model of simulation of mobile bank as a case of study using a case base of stored cases as problems and inferring the most appropriate solutions from the base of the stored cases to solve the problems, and help in achieving treatment of interruption of the process or lack of service for the customer.

2.4 Developing and Testing Environment

IBM provides the Emerging Technologies Toolkit (ETTK) that can be used in building a proof-of-concept prototype. The toolkit contains a logs adapter called Generic Log Adapter for Autonomic Computing (GLA) that translates legacy logs into CBE format. Moreover, Log and Trace Analyzer (LTA) can be used to implement the diagnosis engine as it provides the means for performing a root-cause analysis by correlating events and showing the interactions among the logs that contributed to the problem.[10]

2.5 General background of CBR

Any programs we write must automatically do case-based reasoning that will need to be seeded with a representative store of experiences. Those experiences (cases) should cover the goals and sub goals that arise in reasoning and should include both successful and failed attempts to achieve those goals.[15].

2.5.1 CBR using database technology:

At the simplest form, CBR could be implemented by used database technology. Databases are efficient means of storing and retrieving large volumes of data. If the problem was described. Could make well formed queries it would be straightforward to retrieve cases with matching descriptions. The problem with using database technology for CBR is that databases retrieve using exact matches to the queries [16].

The quality of case based reasoning solutions depends on four things: [15]

1. The experiences it's had.
2. Its ability to understand new situations in terms of those old experiences.
3. Its depend on adaptation.
4. Its depend on evaluation.

2.5.2 Components of CBR

2.5.2.1 Case:

The case contains two components a description of the problem and the solution. Therefore it is defined as an example of the problem [18]

2.5.2.2 Case base:-

The case base contains the experiences and conforms to one of the four sources of knowledge required in a CBR.: They are the vocabulary ,the case-base ,the similarity measure and adaptation containers.

1. The first, the vocabulary contains the terms which support the others.
2. The case-base comprehends what is in a case and how cases are organized.
3. The similarity measure contains knowledge to determine the similarity between two cases in the retrieval phase.
4. The solution adaptation container contains knowledge to adapt past solutions to new problems in the reuse stage [19]

2.5.2.3 Case index:

Kolodner identifies indexing with an accessibility problem [20] that is, with the whole set of issues inherent in setting up the case base and its retrieval process so that the right cases are retrieved at the right time. Thus, case indexing involves assigning indices to cases to facilitate their retrieval. CBR researches proposed several guidelines on indexing [21]. Indexes should be:

1. Predictive of the case relevance
2. Recognizable in the sense that it should be understandable why they are used
3. Abstract enough to allow for widening the future use of the case base concrete (discriminative) enough to facilitate efficient and accurate retrieval.

2.5.3 Solving the problem by using CBR

Solving a problem by CBR involves obtaining a problem description, measuring the similarity of the current problem to previous problems stored in a case base with their known solutions, retrieving one or more similar cases, and attempting to reuse the solution of one of the retrieved cases, The solution proposed by the system must be evaluated [17]

In problem solving CBR, old solutions are used as inspiration for new problems. Since new situations rarely match old ones exactly, however, old solutions must be fixed to fit new situations. In this step, called adaptation, the ballpark solution is adapted to fit the new situation. There are two major steps involved in adaptation: figuring out what needs to be adapting and doing the adaptation [15].

2.5.4 This cycle comprises four activities

The classic definition of CBR was coined by Riesbeck and Schank [5]: Case based reasoning solves problems by using or adapting solutions to old problems. Note that this definition tells us what a case based reasoning does and not “how” it does what it does. Conceptually CBR is commonly described by the CBR cycle.

1. Retrieve similar cases to the problem description.
2. Reuse a solution suggested by a similar case.
3. Revise or adapt that solution to be appropriate for the new problem if necessary.
4. Retain the new solution once it has been confirmed or validated [16]

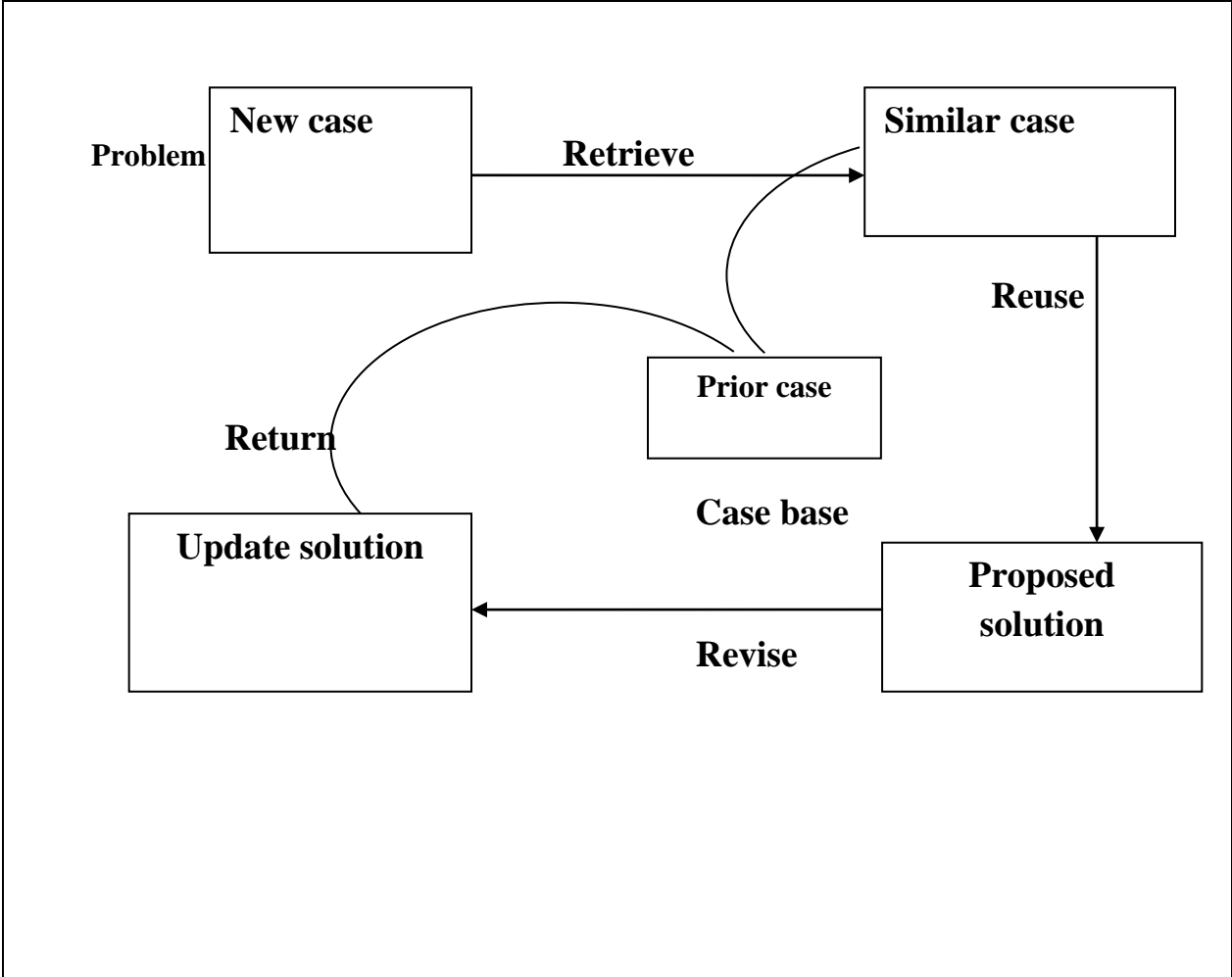


Figure 2. 3 the CBR cycle

In Figure 2.3 the retrieval distance R increases as the similarity between the input problem description and a stored problem description decreases (lower similarity means greater distance). A common assumption in CBR is that the retrieval distance R is commensurate with A , the adaptation distance or effort.

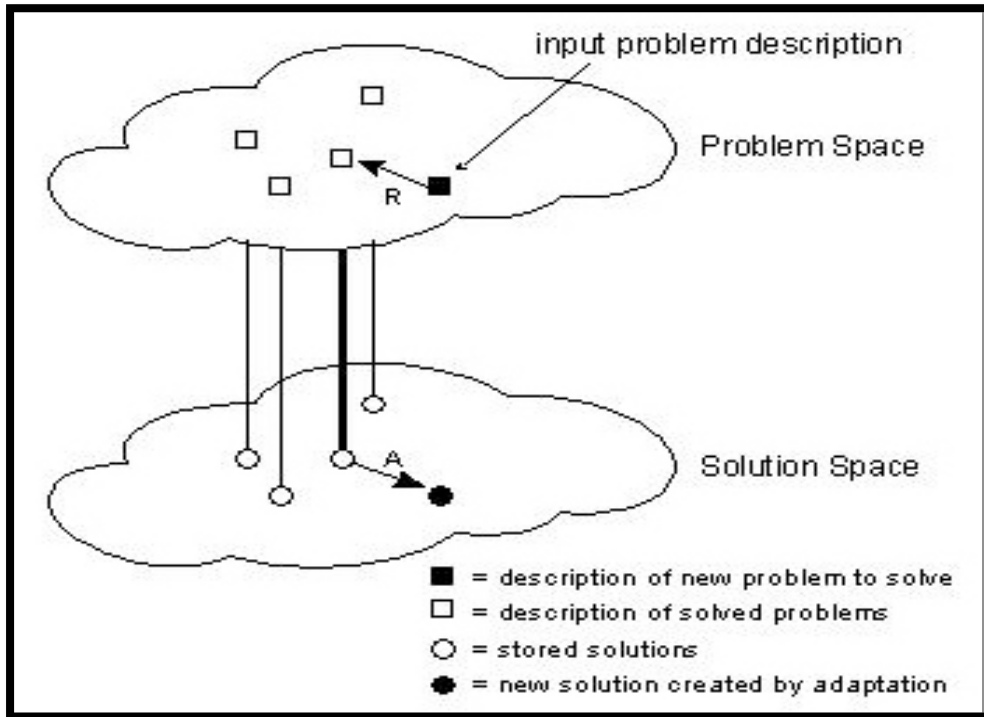


Figure 2. 4 Relationship between problem and solution spaces in CBR, Adapted from (Leake, 1996b).

2.5.5 CBR CASES

2.5.5.1 Retrieval Case:

The process of retrieving a case or set of cases from memory In general, it consists of two sub steps:

❖ Recall previous cases:

The goal of this step is to retrieve "good" cases that can support reasoning that comes in the next steps. Good cases are those that have the potential to make relevant predictions about the new case. Retrieval done by using features of the new case as indexes into the case base. Cases labeled by subsets of those features or by features that can be derived from those features are recalled.

❖ Select the best subset:

This step selects the most promising case or cases to reason from those generated in step 1. The purpose of this step is to winnow down the set of relevant cases to a few

most-on-point candidates worthy of intensive consideration. Sometimes it is appropriate to choose one best case. [15]

2.5.5.2 Reuse and revision in CBR Case

The reuse process in the CBR cycle is responsible for proposing a solution for a new problem from the solutions in the retrieved cases. Reuse appears second, after retrieve, and is followed by revise and retain. Reusing a retrieved case can be as easy as returning the retrieved solution, unchanged, as the proposed solution for the new problem. This is often appropriate for classification tasks, where each solution (or class) is likely to be represented frequently in the case base, and therefore the most similar retrieved case, if sufficiently similar, is likely to contain an appropriate solution. But reuse becomes more difficult if there are significant differences between the new problem and the retrieved case's problem. In these circumstances the retrieved solution may need to be adapted to account for these important differences [17]

2.5.5.3 Revise case

The review stage consists of evaluating and resolving the status of the reuse phase. If the result is successful, the system will learn from success (state retention), otherwise it will be necessary to fix the case resolution using domain-specific knowledge.

In CBR systems, the solution is successful or wrong. If successful, they can be retained, inserted in the base if necessary, or should not be. But when the solution fails, the system is also concerned about retention because of failure; there is an investigation task to learn more about the issue.

2.5.5.4 Retain case

This phase retains the solution and adds it to the status rule once this solution is verified. This allows the system to learn from its experiences, and there are two types of adaptation.

❖ Structural Adaptation:

In structural adaptation, formulas and rules are directly applied to stored solution in CBR library. When case is applied on these rules and formulas. Then, CBR system adapts this case and match with new problem.

❖ **Derivational Adaptation:**

It is a technique to reuse the rules and formulas to produce a new solution of a current problem. Solutions which are retrieved must be stored as additional case in the CBR library so it reproduces new solution to new case.

2.5.6 Benefit and limitations of CBR:

1. The test can be performed by person with relatively little experience and training.
2. You can test with simple, portable equipment.
3. You can run tests in either the field or the lab for design, construction control, or evaluation of existing construction.

Limitation:-

1. The lab and field compaction methods are not identical. However, Comparative tests indicate that reasonable correlation of results can be obtained from field compact materials and Samples compacted under similar conditions.
2. Because many of the procedures are of an arbitrary nature, you must run the test to exact standards in order for the design tables to be valid.
3. The lab and field compaction methods are not identical.
4. Defer: lazy problem solvers simply store the presented data and generalizing beyond these data is postponed until an explicit request is made.

2.6 Previous studies:-

Azim, M.T. et al [5] Our first application uses dynamic analysis to detect when an application has entered an error and determines the finished part of the application; then executes the recovery process, either by eliminating transient errors and continuing to operate at full capacity, or reversing to a safe state followed by closing the finished part, Limited mode while avoiding further breakdowns. They take approach Android platform ,The Android platform consists of the Android OS (a Linux kernel customized for smart phones),the Dalvik VM (virtual machine) and apps, written mostly in Java, that are compiled to a byte code format named DEXand execute in the Dalvik VM. Apps can also embed native code (written in C/C++) that is executed directly by the OS.

They take architecture Centered around detecting crashes and in response applying seal-off patches. A model is constructed rest, via static analysis on the app byte code); the model includes rollback (resume) points where the app will be driven when recovering from a fault. Next, the app is executed, either manually by users or automatically via systematic exploration tools and its execution logis monitored via

dynamic analysis. When a failure is detected, we employ byte code rewriting (code generation and instrumentation)

Jiang, M. et al [6] In order for the system to be self-healing, the system must be able to detect the errors that have occurred and how to correct them. Self-healing software systems. The model-based approach is used to classify software failures and determine their arrangements at the model level.

The basis of autonomic computing draws on biological analogies to describe an autonomic computer as a computer that is self-governing, in the same manner that a biological organism is self governing.

2.6.1 Self-management:-

Puviani et al [7] is the process by which computer systems shall manage their own operation without human intervention. The growing complexity of modern networked computer systems is currently the biggest limiting factor in their expansion. The increasing heterogeneity of big corporate computer systems, the inclusion of mobile computing devices, and the combination of different networking technologies like WLAN, cellular phone networks, and mobile ad hoc networks make the conventional, manual management very difficult, time-consuming, self-management has been suggested as a solution to increasing complexity in cloud computing.

2.6.2 Self-managing computing system:-

Hellerstein et al [8] A method, computer program product, and data processing system for constructing a self-managing distributed computing system comprised of “autonomic elements” is disclosed. An autonomic element provides a set of services, and may provide them to other autonomic elements. Relationships between autonomic elements include the providing and consuming of such services. These relationships are “late bound,” in the sense that they can be made during the operation of the system rather than when parts of the system are implemented or deployed. They are dynamic, in the sense that relationships can begin, end, and change over time. They are negotiated, in the sense that they are arrived at by a process of mutual communication between the elements that establish the relationship.

2.6.3 Implementing an automation computing evaluation scale to generate recommendations:-

Barel, M.A., et al [9] Analysis module analyzes the automation capabilities of the customer based on survey answers. Automation capabilities of an enterprise include, for example, the ability to be self-configuring, the ability to be self-healing, the ability to be self-optimizing, and the ability to be self-protecting. Across the four automation capabilities, there are several key operational areas where one can assess

automation maturity. These operational areas are used as automation assessment categories in accordance with an exemplary embodiment of the present invention. The automation assessment categories may include, for example, problem management, availability management, security management, solution deployment, user administration, and performance and capacity management.

Problem management is the act of identifying, isolating, and resolving issues that might negatively impact IT service delivery. Availability management is the act of ensuring that required IT services are available, as needed, to ensure business continuity. Security management is the act of securing critical business resources and data against attacks and unauthorized access from both external and internal threats. Solution deployment is the act of planning, testing, distributing, installing, and validating the deployment of new IT solutions, including the IT infrastructure elements, in a manner that is the least disruptive to operational services. The ability to roll back to a prior functioning environment if a change is unsuccessful is also necessary.

Martin Monperrus et al [22] This article presents a survey on automatic software repair. Automatic software repair consists of automatically finding a solution to software bugs, without human intervention. Detection: our framework works on a dynamic analysis of systems behavior and app output.

Recovery. The recovery mechanism works in two phases. First, after detecting the crash point, we determine the safe retreat point. Second, we re-launch the app to a secure nearby point so users can continue Work and interact with the application.

Implementation Platform. We implemented our approach and conducted Experiments on a Motorola Droid Bionic phone, running An-droid 2.3.4

Tools Used the SATG extraction component (static analysis-based) of A3E. To drive exploration, we used the systematic exploration component of A3E. We wrote the main instrumentation code in Java.

Blair, G.S. et al [23] Proposed a way to make modern operating systems more flexible for malicious encryption Attacks, by detecting behaviors similar to ransom and return Its effects preserve the integrity of user data. Were developed using Python, the architecture has been re-implemented with the explicit goal of provide a high performance implementation of our reflective middleware technology. Problem Statement Were the first to analyze a large corpus of ransomware samples. The authors suggest that the file system is strategic point for monitoring the typical ransomware activity.

Tianxiao Gu et al [24] Software systems are often subject to unexpected runtime errors. Automatic runtime recovery (ARR) techniques aim to recover them from erroneous states and maintain them functional in the field. This paper proposes Ares, a novel, practical approach for ARR. Our key insight is leveraging a system's inherent error handling support to recover from unexpected errors.

Implemented Ares on two popular platforms: Java Hot Spot VM and Android ART. This design decision enables our approach to be a drop-in substitute for standard VMs, and easy to deploy in production environments without complex configurations.

Exception Handling Mechanism The Java programming language has a built-in exception handling mechanism. The basic construct is a try-catch block, composed of a try block and a catch block that catches exceptions thrown inside the try block. A catch block declares a catchable exception type Eh , that is, any exception of type Eh or a subtype of Eh is catchable by the catch block. We denote a try-catch block as (ls, le, lc, Eh) (Try-Catch Block).

2.7 Summary

With the development of mobile applications, there is a need to increase the guarantee that the application will not fail to perform its tasks, and the number of mobile applications increases constantly because Android is open source, and therefore programmers must use available technologies that guarantee them the treatment and self-healing of applications, and the implementation methodology will be discussed in the next chapter.

CHAPTER 3

3 Methodology

3.1 Introduction

This chapter discusses the methodology of our system, self-healing on mobile bank using case-based reasoning. The case study MOBILE BANK

3.2 Case-based reasoning

Case-based reasoning can mean adapting old solutions to meet new demands; using old cases to explain new situations; using old cases to discover new solutions; or reasoning from precedents to interpret a new situation[15]

The following figure (3.1) shows the organization of case base:

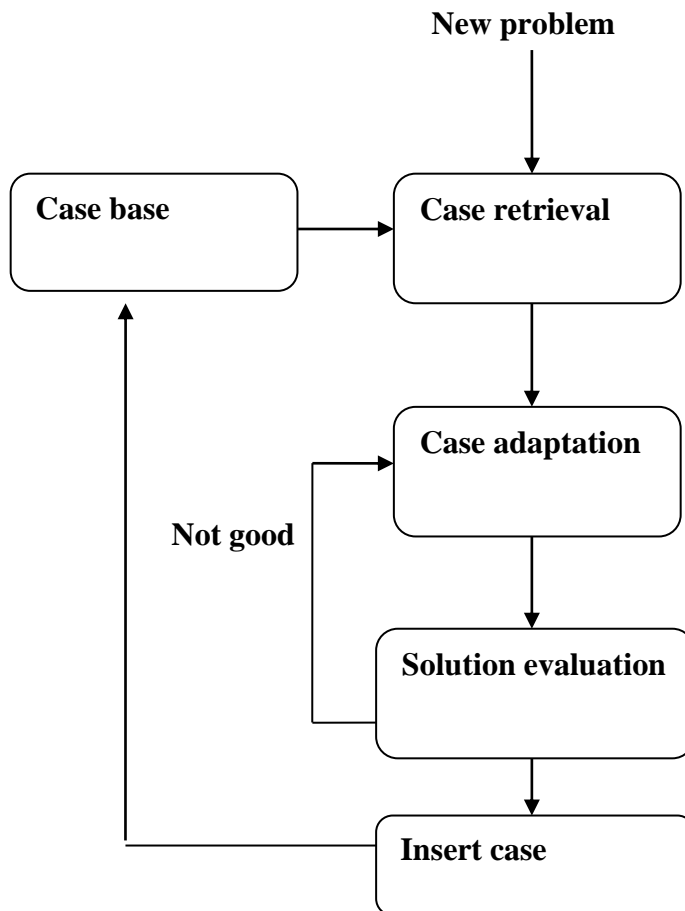


Figure 3.1 case base organization

Flow chart showing the interruption of the process in the system and the recovery of a similar case:-

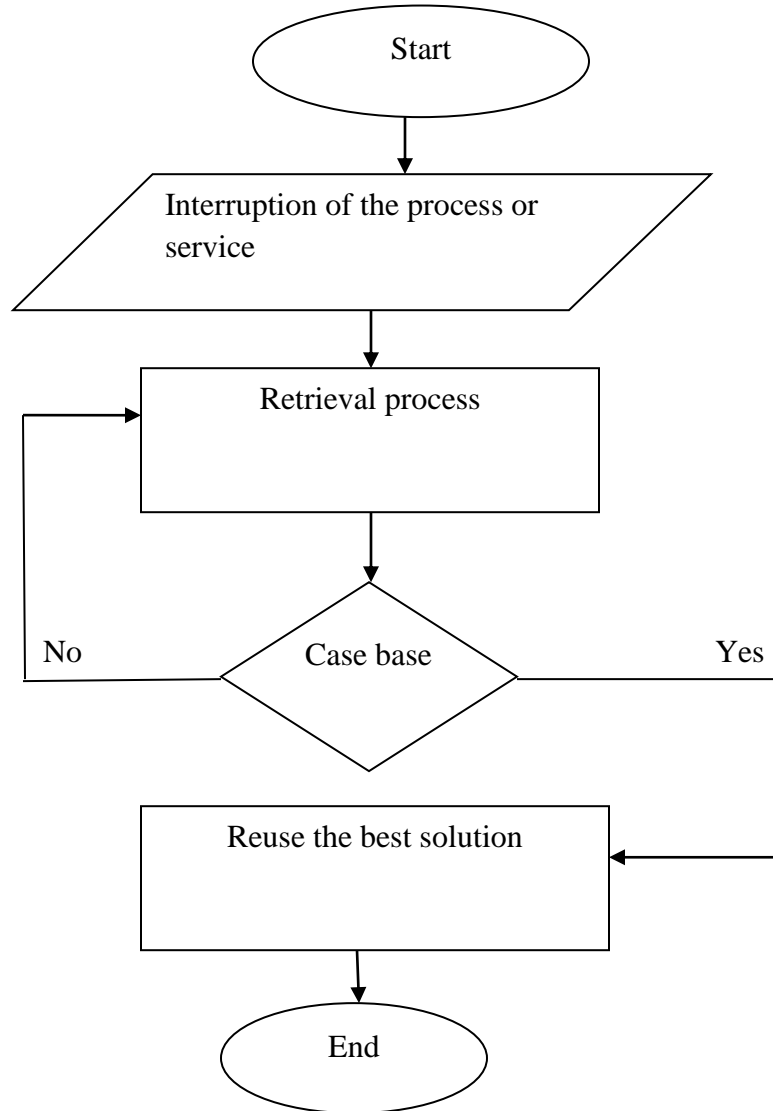


Figure 3. 2 The flow chart of interruption of the process in the system and the recovery of a similar case

The customer purchases electricity, and when the process is interrupted, service retrieval occurs and the problem is determined from case base if a similar case is found, the optimal solution is assigned to solve the problem.

Below figure (3.3) shows the implemented framework of the self-healing on mobile application (mobile bank FIB) using CBR:

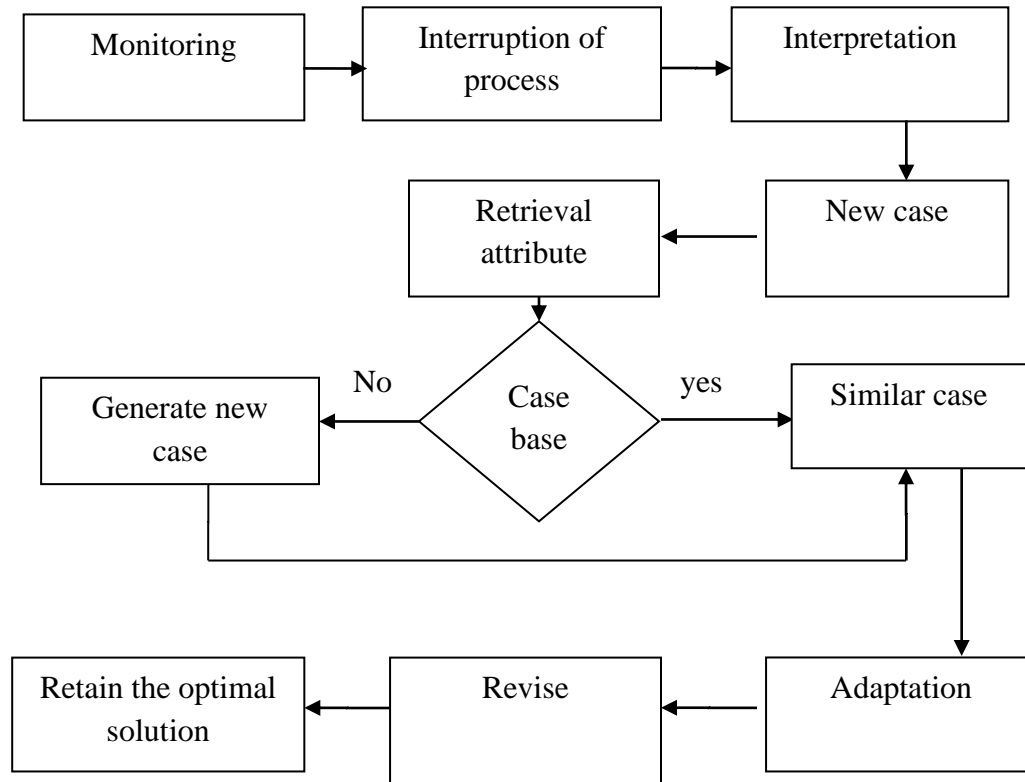


Figure 3. 3 mobile bank framework using CBR

Flow chart showing the sequence of request and interruption of service from the customer:

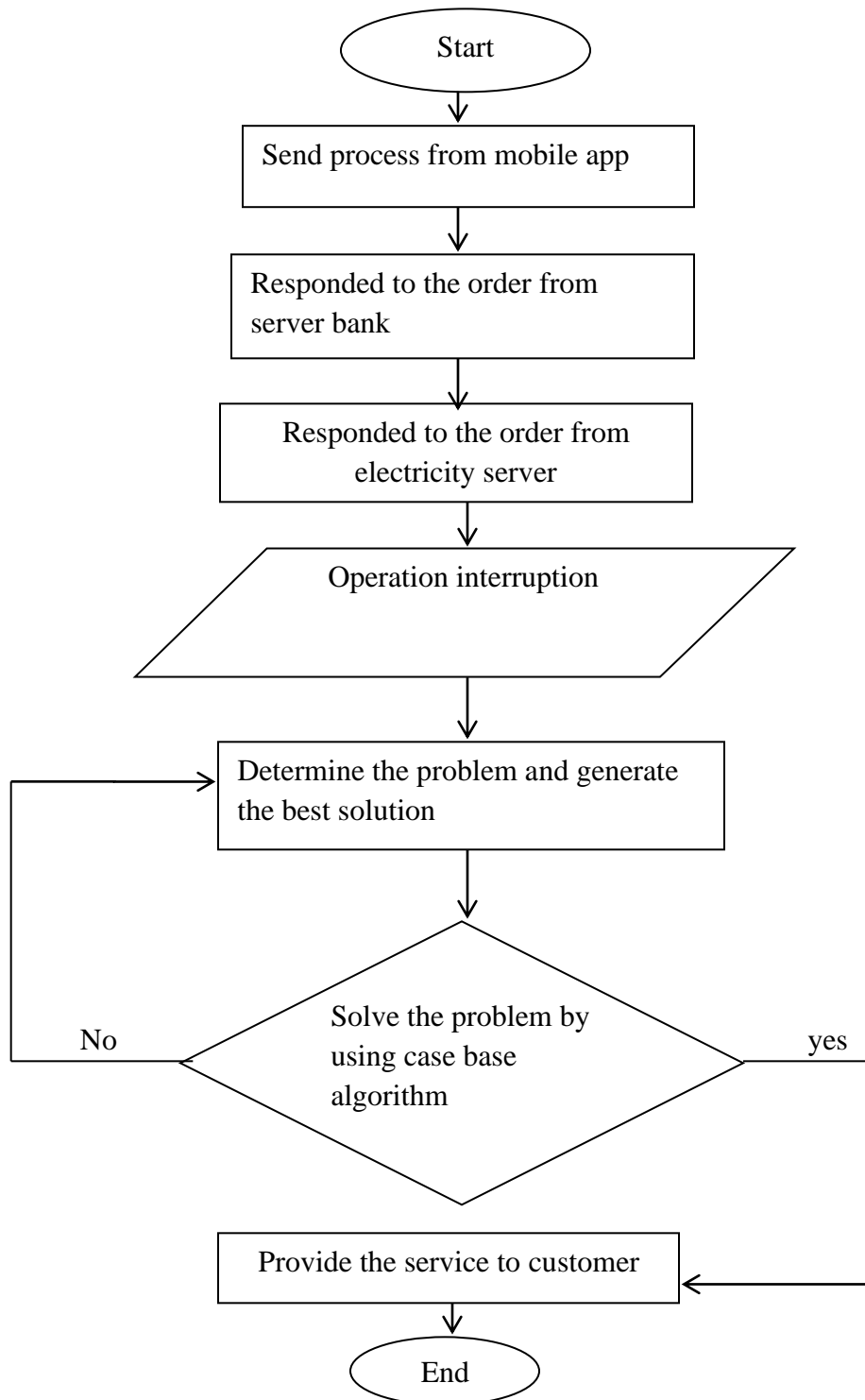


Figure 3. 4 the sequence of interruption operation to buy electricity.

3.3 The functions used to calculate the similarity distance between these cases and accuracy:

The similarity will be calculated using:-

1. Manhattan:-

Following equation (3.1) describes this function:

$$D = \sum_{i=1}^n | x_i - y_i | \quad ..(3.1)$$

Where:

d: Manhattan distance

x_i : New case

Y_i : Old case

n: number of compared cases

2. Euclidean:-

Following equation (3.2) describes this function:

$$d = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (3.2)$$

Where

d: Euclidean distance

x_i : New case

y_i : Old case

n : number of compared cases

3. Canberra:-

Following equation (3.3) describes this function:

$$d_{ij} = \sum_{k=1}^n \frac{|x_{ik} - x_{jk}|}{|x_{ik} + x_{jk}|}$$

(3.3)

Where:

d_{ij} : Canberra distance

x_{ik} : New case

x_{jk} : Old case

n : Number of compared cases

4 Squared Chord:-

Following equation (3.4) describes this function:

$$d = \sum_{i=1}^n (\sqrt{x_i} - \sqrt{y_i})^2$$

(3.4)

Where:

d: Squared chord distance

xik: New case

xik: Old case

n: Number of compared cases

5 Squared chi-Chord.

Following equation (3.5) describes this function:

$$d = \sum_{i=1}^n \frac{(x_i - y_i)^2}{(x_i + y_i)} \quad (3.5)$$

Where:

n: number of compared cases

x_i: New case

y_i: Older cases.

3.4 Root Mean Square Error (RMSE):-

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

3.5 Software

Software that will be used for developing this application is Eclipse for creating the interface and code. The Eclipse use Java as the programming language.

Table 3. 1 four important attributes in self-healing algorithm

Name of attributes	Type of value
Connections	String
Database server	String
Update session	String
Result message	String

3.6 Summary

using comparison between different functions to calculate the similarity and accuracy, The success of this concept in model of simulation system will lead to a major shift in the world of applications where the behavior of the problem will be determined and self -healing without the need of any external intervention and the completion of application tasks and providing services without any interruption and as little time as possible

CHAPTER 4

4 Implementation of self-healing in bank system

4.1 Introduction

This chapter will demonstrate the implementation stage of developing the self-healing in mobile application system, and discussing the results.

Development involves designing a self-healing of problems related to system behavior and improvement of the entire application. The method that used in the development of the application and the database were discussed here. The self-healing in mobile application was developed using Eclipse. The application source code was created by using the Java programming language and the case-base was created using text files, The mobile bank application provides a set of services and processes to end users. In this research, we highlight the interruption of the process or service to the customer and we will use a mechanism that helps solve this problem related to the behavior of the system. The mechanism depends on storing a set of previous solutions and works to deduce the best solution from among the similar solutions.

4.2 Development Environment

This application was developed using Eclipse using the Java programming language ,Window 10 is used as on operating system with Intel(R) processor and 3 GB of RAM to develop the application environment. This application used self-healing on mobile application which has been collected from Fasial Islamic bank(the problem of behavior error that related to attribute using in CBR algorithm as(connection , database server ...)and the proposal solution which entered in model of simulation) . This dataset is used as the case base for the model.

4.3 System of Interface

The interface allows the application to interact with the system and others. This system consists of three interfaces, the first interface to measure the weights of tests, the second interface is the new case for test cases, and the third interface is the results of the test.

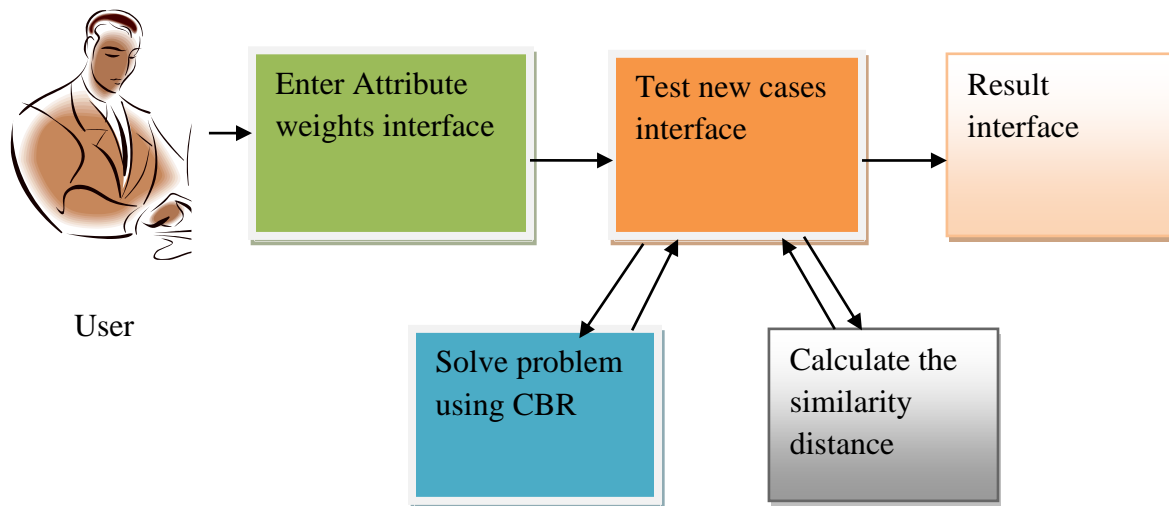


Figure 4. 1 self –healing interfaces model

Below figure (4.2) shows the implemented framework of the self-healing system using CBR:

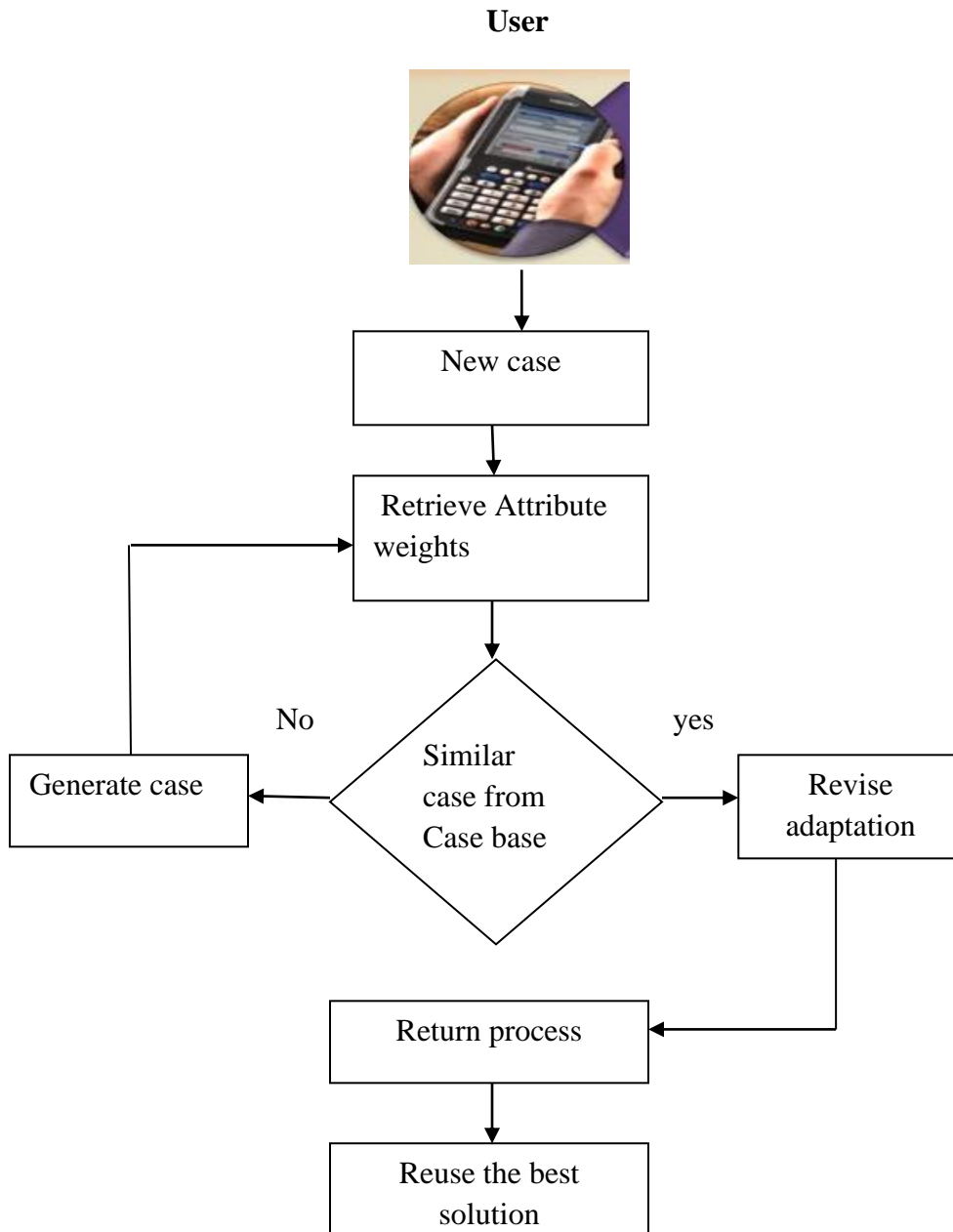


Figure 4. 2 Self-healing framework using CBR

Following figure (4.3) in this figure I have build the cases from real-time problems of the mobile bank, we used comma (,) to delaminate the cases values, and display them in the interface of case-base:

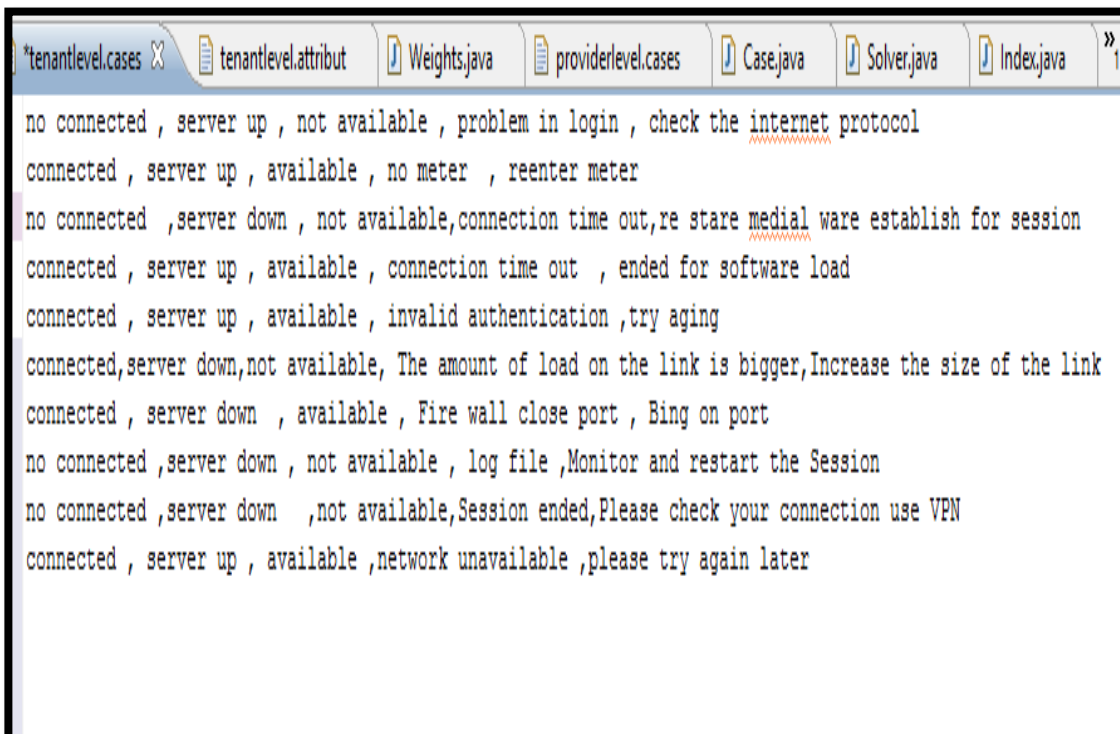


Figure 4. 3 Show cases of the system

- The way I used in indexing to match cases with each other is a brute-force(the search is done sequentially from the first case to the last case , then the match value for the case that achieves self healing is saved at the end of search).

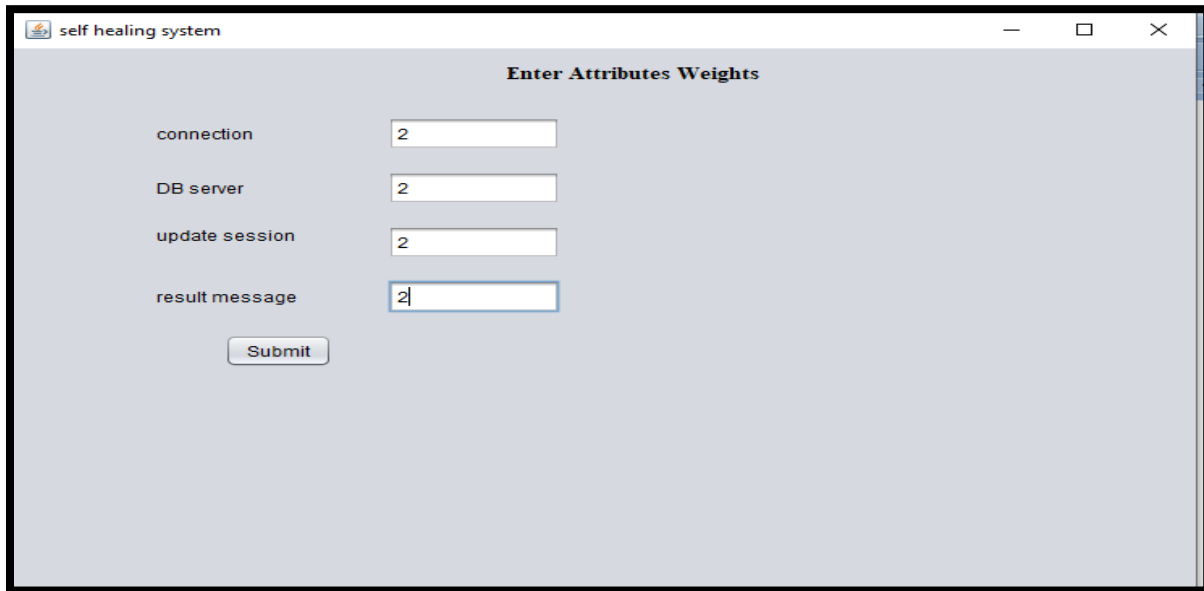
4.4 Attribute weights interface:-

This is the first interface in the model. It consists of four text fields which are used as the input of weights for connection, Database server, update session and result message. And a button takes this action. Weight is used in Manhattan and Euclidian functions to calculate the distance between new cases and stored cases.

- The attribute determined based on the affect of problems that face the application .

- The weights determined based on the importance of the parameter affecting (values of attributes) if the problem was more important the weights take number 1 and if it less important take number 2.

Following figure (4.4) is the test weight interface in our system.



The screenshot shows a window titled "self healing system" with a subtitle "Enter Attributes Weights". It contains four text input fields, each with the number "2" entered. The fields are labeled "connection", "DB server", "update session", and "result message". Below the fields is a "Submit" button.

Figure 4. 4 attribute weights of self-healing system interface

4.5 The new cases of self-healing system and test interface:-

This is the page used to input the new cases of self-healing system and information about their attribute. From the testing interface, after clicking the solve button, the input will be compared with the case-base, retrieve the data from the case-base to the calculate the 'similarity. After the calculation, it will display the result and the similarity from the previous case.



Figure 4. 5 the new cases of self healing system and test interface

4.6 Result Interface

This is the result interface where the result and the new cases will be displayed .It displays the result of best matching case and similarity of the case to the previous case. User can choose either to retain the case or exit the interface.

- Through the solver file matching cases are retrieved after determining the weights and entering the new values for the case the existing case is retrieved in the cases base and then all the attributes are matched to the old and new cases.

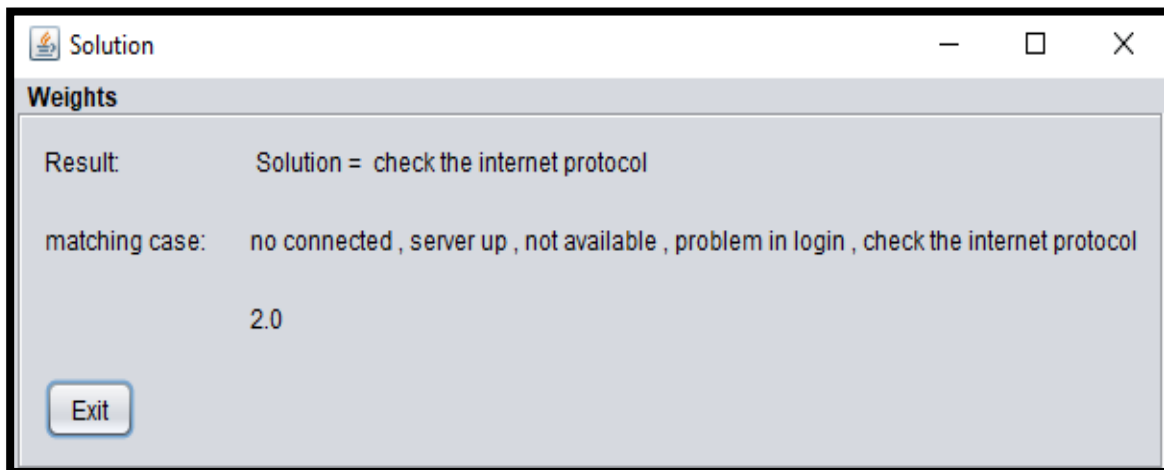


Figure 4. 6 result of self-healing system interface using Manhattan function

The retrieval technique, cases is retrieved from the solver file, after determining the weights and entering the values for the new cases, the existing case is retrieved in the cases base, and then all parameters are matched to the old and new cases.

The following tables describe the functions used and the codes that have been implemented:-

table (4.1) shows the code reading the value of the value of the new attribute from the user and storing it in the new attribute Value variable, and reading the value of the old Attribute from the case-base and store it in CaseAttributeValue:

new and old cases
<pre>newAttributeValue = Double.parseDouble(this.newCaseAttributeValues[j]); caseAttributeValue = Double.parseDouble(theCase.attributeValues[j]);</pre>

Table 4. 1 values of new and old cases

1. The first function:

To calculate the similarity distance between these cases is Manhattan function. The variable this distance stores the similarity distance which is calculated by subtract newAttributeValue form CaseAttributeValue multiplied by weight of the new case attribute.

The following table (4.2) shows the code that calculates the distance using Manhattan function:

<pre>if(choice=='m')// manhattan this.distances[i] += Math.abs(newAttributeValue - caseAttributeValue) * this.bobot[j];</pre>

Table 4. 2 Manhattan function calculation code

2. The second function is Euclidean :-

<pre>if(choice=='e') //euclidean this.distances[i] += Math.sqrt(Math.pow((newAttributeValue - caseAttributeValue),2) * this.bobot[j]);</pre>
--

Table 4. 3 Euclidean function calculation code



Figure 4. 7 Result of similarity distance interface using Euclidean function

3. The Third function is Canberra:-

```
if(choice=='c')// canberra
    this.distances[i] += Math.abs((newAttributeValue - caseAttributeValue))/(newAttributeValue + caseAttributeValue)
```

Table 4. 4 Canberra Function calculation code

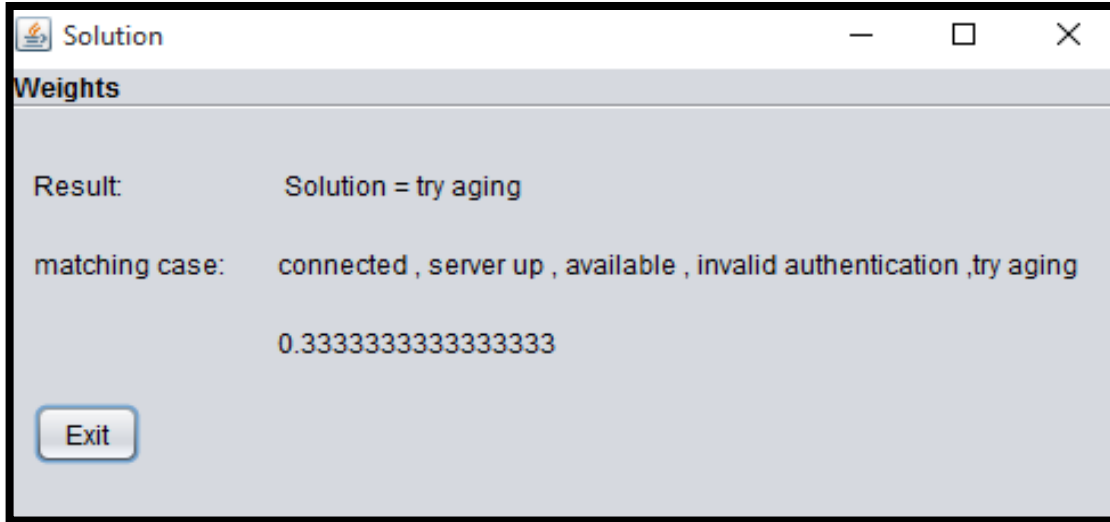


Figure 4.8 Result of similarity distance interface using Canberra function

4. The forth function is Squared Chord:-

```

if(choice=='s')//squared chord
    this.distances[i] += Math.pow((Math.sqrt(newAttributeValue) - Math.sqrt(caseAttributeValue)),2) ;
    .....

```

Table 4. 5 Squared Chord function calculation code

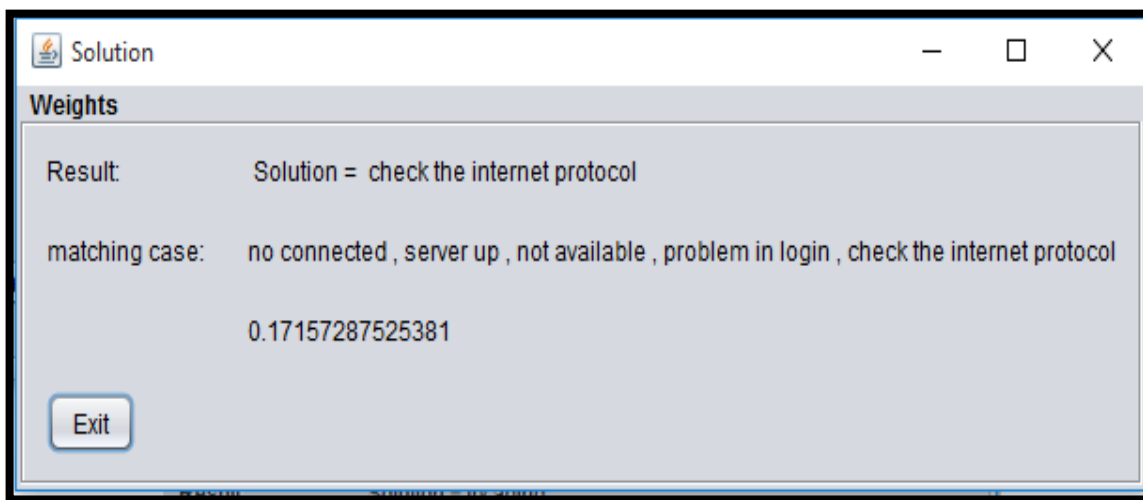


Figure 4. 9 Result of similarity distance interface using squared chord function

5. The fifth function is squared chi-Chord:-

```
if(choice=='i')//squared chi-squared
    this.distances[i] += Math.pow((Math.abs(newAttributeValue - caseAttributeValue)),2)/(newAttributeValue + caseA
```

Table 4. 6 Squared chi-Chord function calculation code

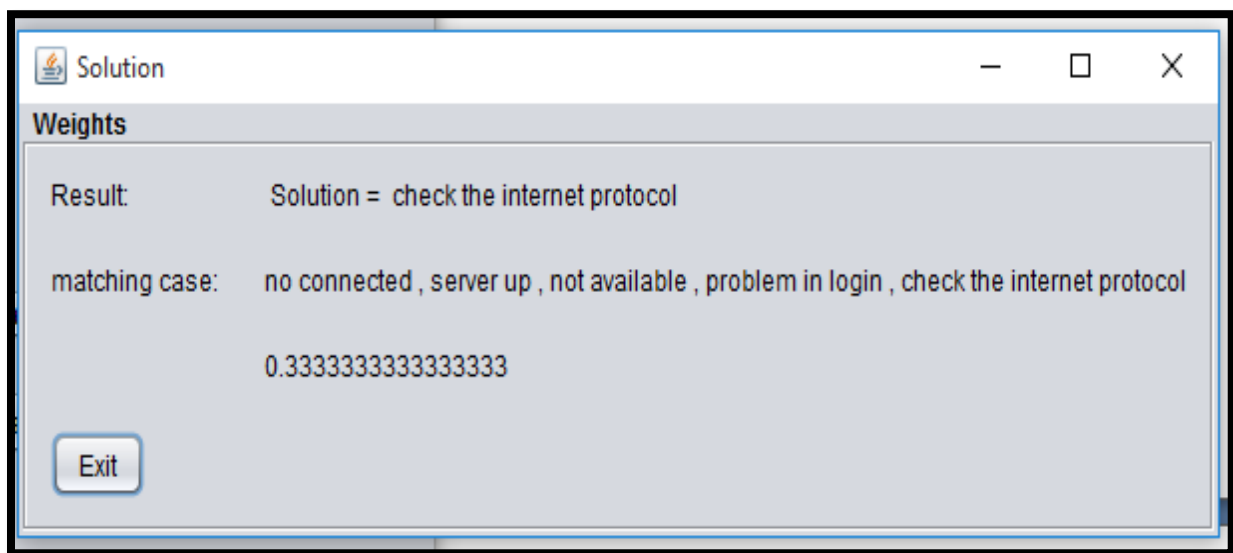


Figure 4. 10 Result of similarity distance interface using squared chi-Chord function

4.7 Self-Healing using CBR algorithm model: -

In CBR, there are four components that are important during the prediction which are Retrieve, Reuse, Revise and Retain. In the self-healing system, retrieve is referring to give a target problem, retrieve cases from memory that are relevant to solve it.

4.8 Problem solving and result:-

The four attributes or features used are Connection, Database server, Message result, and update session measurement as a system input that uses analogy in problem solving and inference to match a previous case of the system with the new problem to

find a solution. After the similarity is found, the similarity will be calculated using these functions Manhattan, Euclidean, Canberra, Squared Chord and Squared chi-chord functions. The most important features (weight) are determined and it will be used in similarity computation.

4.9 Accuracy measurement

The measure accuracy is computed in Equation (4.1) as follow:

$$\text{Accuracy} = \frac{\text{Correct case}}{\text{Total Testing Cases}} * 100$$

CBR algorithm in this app will bring accuracy over 40% to determine system problems, and 60% max. The accuracy of all functions used to calculate the similarity distance is shown in Table (4.7) below:

Function	Accuracy
Manhattan	40%
Euclidian	40%
Canberra	60%
Squared Chord	40%
Squared chi-Chord	60%

Table 4. 7 show the similarity distance

The following figure (4.11) is a chart that shows the accuracy rate using Euclidean functions and determines number of test cases which have been tested using the similarity and determines the correct result.

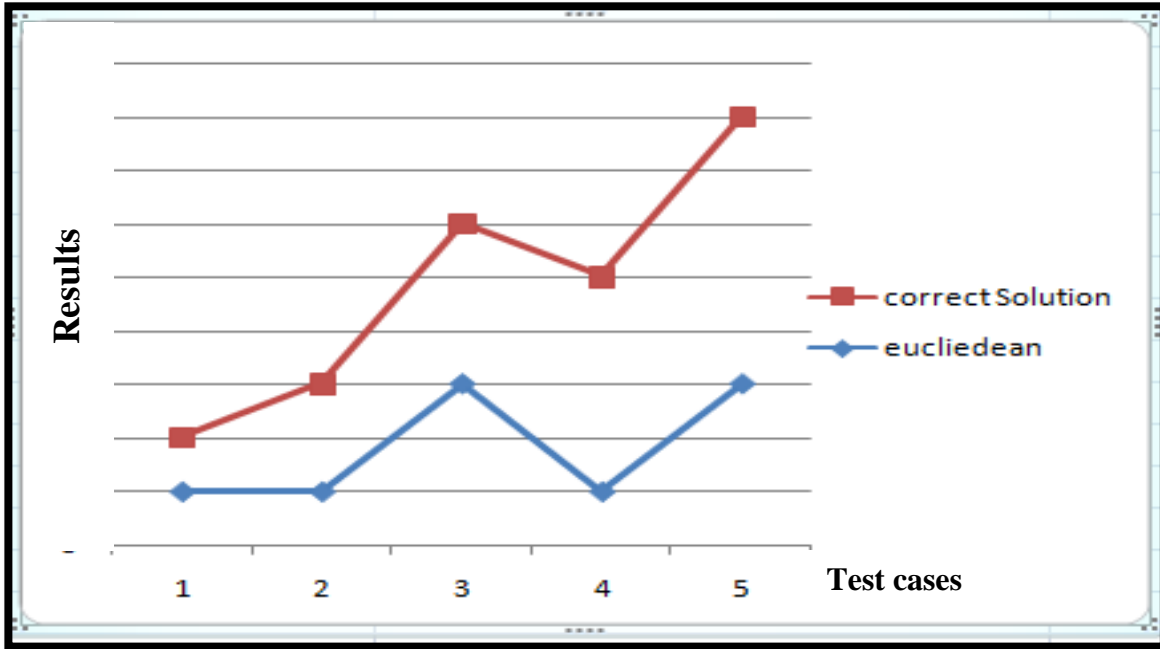


Figure 4. 11 Euclidean function accuracy

The following figure (4.12) is a chart that shows the accuracy rate using Manhattan functions and determines number of test cases which have been tested using the similarity and determine correct result.

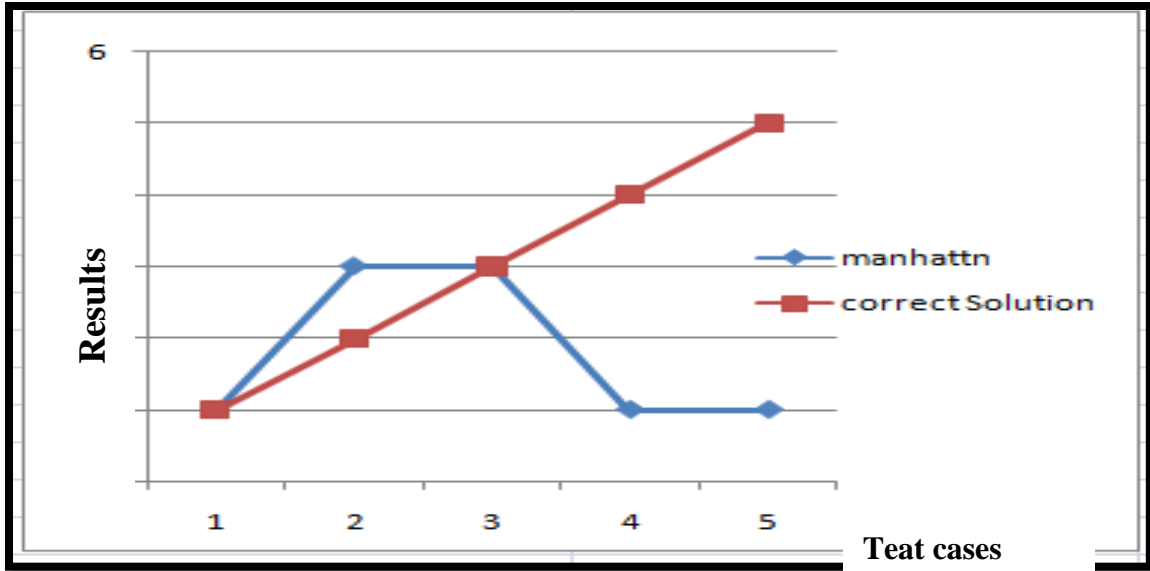


Figure 4. 12 Manhattan function accuracy

The following figure (4.13) is a chart that shows the accuracy rate using Canberra functions and determines number of test cases which have been tested using the similarity and determines the correct result.

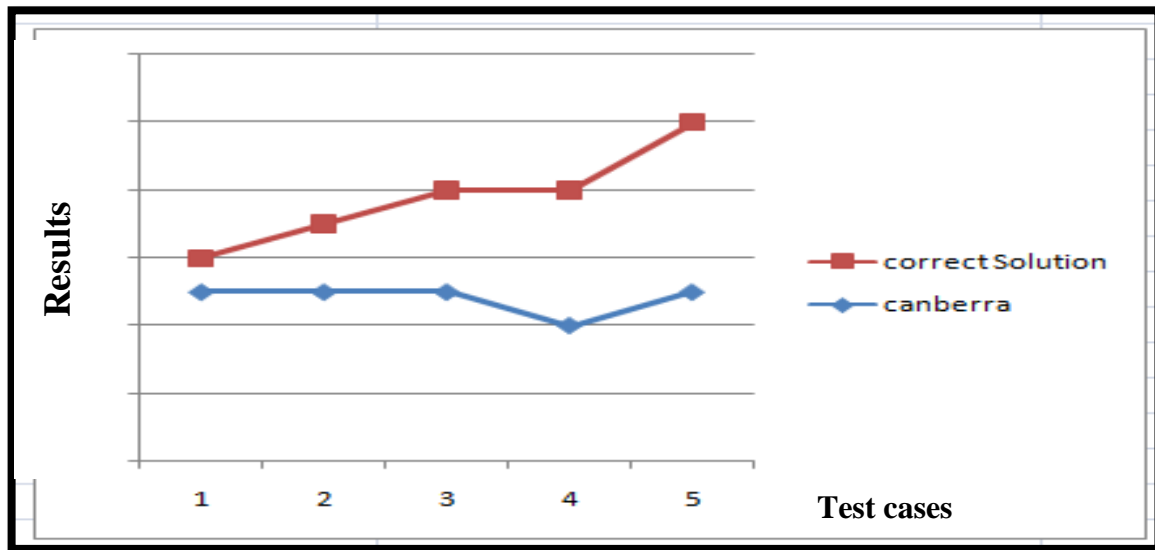


Figure 4. 13 Canberra function accuracy

The following figure (4.14) is a chart that shows the accuracy rate using chi-chord Functions and determines number of test cases which have been tested using the similarity and determines the correct result.

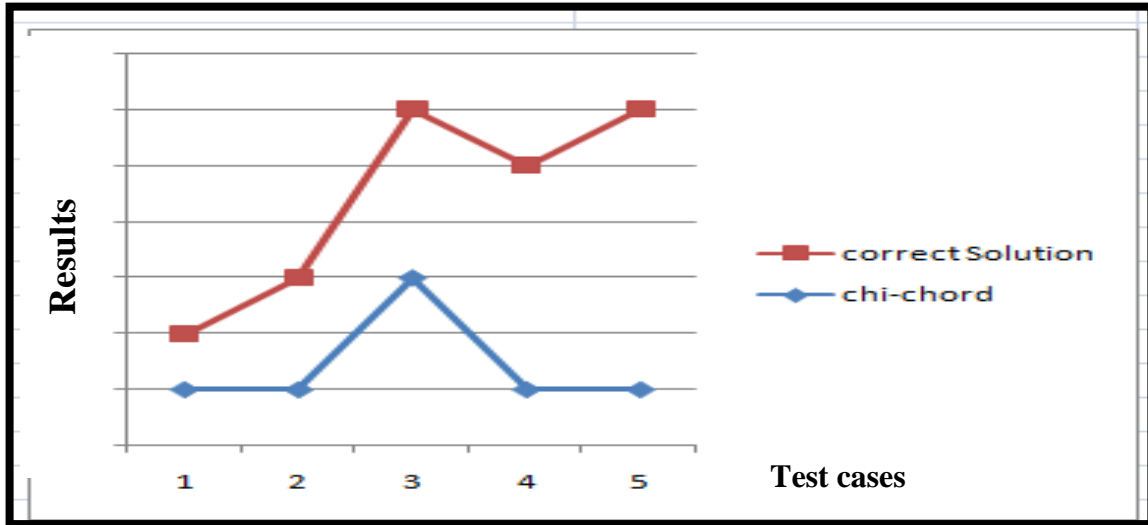


Figure 4. 14 Chi-chord function accuracy

The following figure (4.15) is a chart that shows the accuracy rate using sugared chi-chord Functions and determines number of test cases which have been tested using the similarity and determines the correct result.

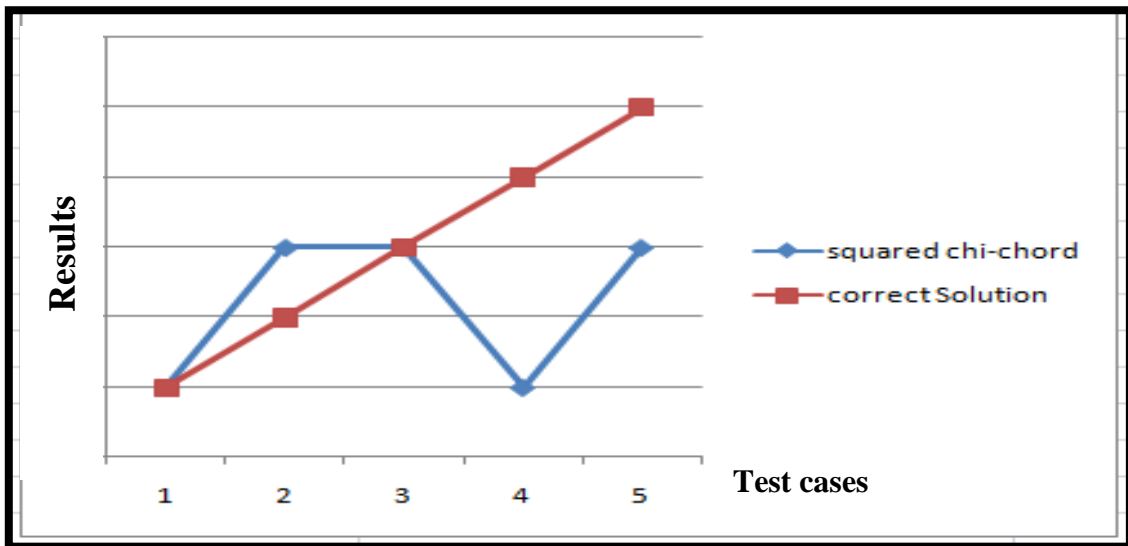


Figure 4. 15 Sugared chi-chord function accuracy

The following figure (4.16) is a chart that shows the accuracy rate using all Functions and determines number of test cases which have been tested using the similarity and determines the correct result.

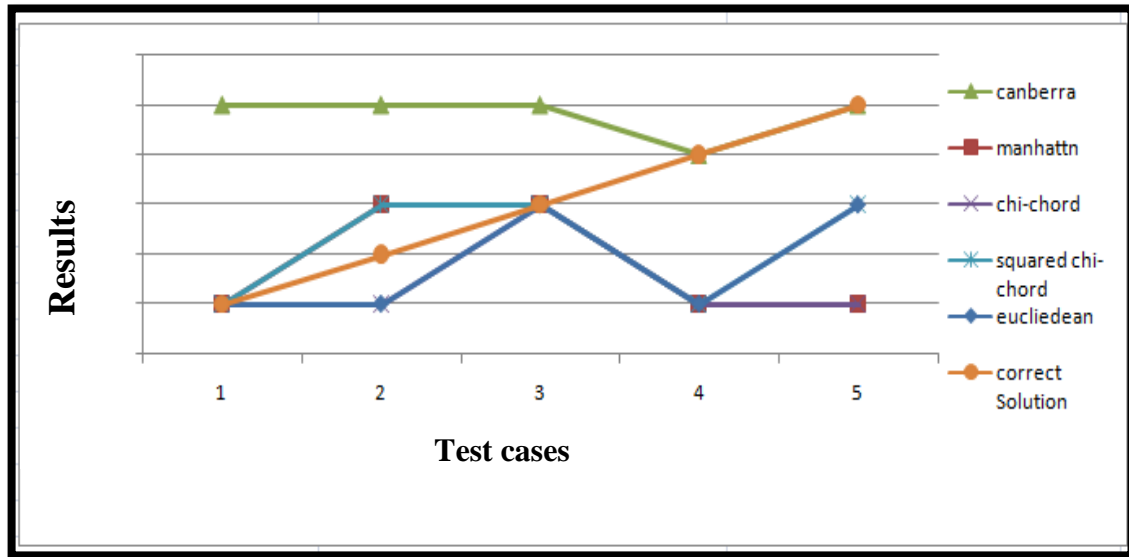


Figure 4. 16 Similarity functions accuracy rate

4.10 Root Mean Square Error (RMSE):-

The Error rate is computed in Equation (4.2) as follow:

$$\text{REMS (X1, X2)} = \sqrt{\frac{\sum_{i=1}^n (x_{i1} - x_{2i})^2}{n}}$$

(4.7)

The following table (4.8) show estimating error rate percentage for all similarity functions is the case number from case-base and the similarity value.

Case1	Case2	Case3	Case4	Case5	chose n	function
1	2	2	1	2	5	manhattn
0.3	0.3	0.3	0.3	0.3	5	cenberra
1	1.4	1	1.41	1.41	5	eucliedean
0.17	0.17	0.16	0.17	0.17	5	chichord
0.3	0.3	0.3	0.3	0.3	5	squared chi-chord

Table 4. 8 Show the error rate for all similarity functions. The error rate percentage has been calculated using RMSE formula.

The following Table 4.9 Show Error rate calculated using RMSE formula:-

Function	Error rate
Manhattan	0.774597
Canberra	0
Euclidean	0.315024
chi chord	0.004472
squared chi-chord	0

Table 4. 9 Show the error rate for all similarity functions the error rate percentage has been calculated using RMSE formula.

The following figure 4.17 Show that the function which gives error rate. The x is the number of tested cases and y is the error rate percentage for all similarity functions.

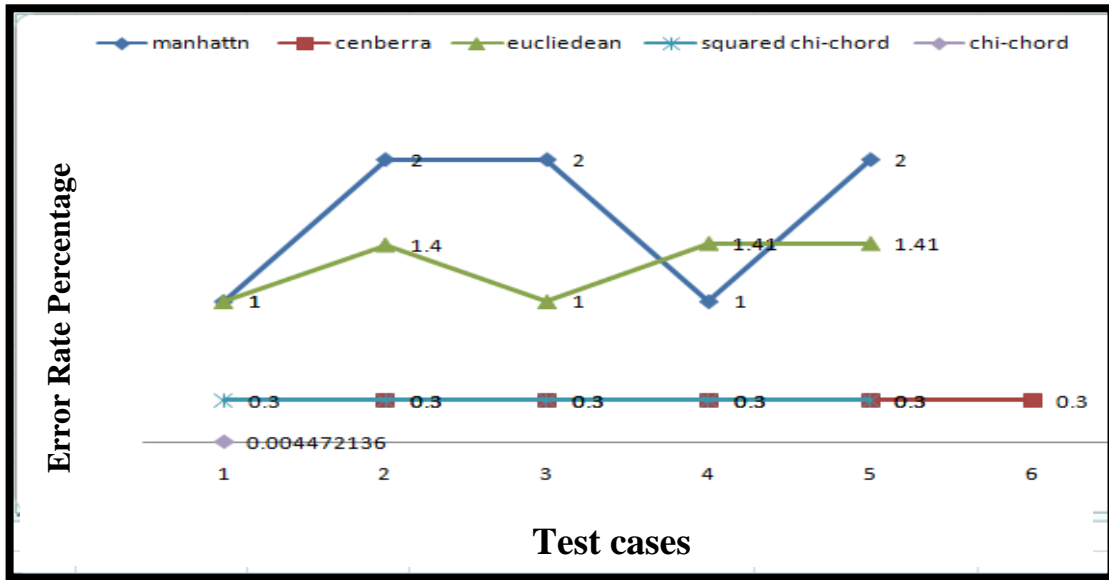


Figure 4. 17 Error rate of similarity functions using RMSE

4.11 Discussion

The success of this work will permit to leverage the development of CBR systems in mobile application. The above results have provided some indications on the factors affecting the performance of the CBR-system, such as the range of values affecting the similarity distance between two cases.

Chi-squared test is the best to use to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more categories.

The model of simulation will provide benefit from the development of CBR systems in mobile applications in general and the development of the mobile bank system in particular by reducing the dispute that usually occurs when the aforementioned causes occur and it becomes possible to correct all error of this through a set of sentences.

4.12 Summary

Self-healing in mobile app system use model of simulation by CBR algorithm to design the development environment, designing of interface is discussed. The next chapter will discuss about the conclusion and recommendation.

CHAPTER 5

5 Conclusions and recommendations

5.1 Conclusions

The goal of self- healing systems is to provide applicable, highly accessible and reliable systems. It offers properties such as adaptation, transformation, self-preservation, etc. The mechanism used for self- healing and adaptation in this study is case base reasoning, but this study is still in its infancy ,The result of using the self-healing algorithm in an application that contributes to:-

1. Increased self-healing in applications.
 2. Reducing the manual dispute process.
 3. Solve the problems based on the conclusion of the solution stored in the case base and use the optimal solution to solve the problem.
 4. Reduces the time to process the behavioral errors that can occur in the system.
 5. Provided continuity of service to the customer by using algorithm to measure error similarity.
- This proposed solution can be generalized to solve the problems that may occur in the rest of the banking transactions such as (balance transfer - transfer to a card), taking into account the differences in the transactions affecting those problems.

5.2 Recommendations

There are some suggestions and recommendations that should be done in order to improve the application as follow:

1. Development of the application by linking the algorithm to the Mobile Bank application.
2. For next version this application can be implemented inside the mobile.

References:-

1. Powar, S. and Meshram, B.B., 2013. Survey on Android security framework. *International Journal of Engineering Research and Applications*, 3(2), pp.907-911.
2. Ghosh, S.K. ed., 2009. *Self-healing materials: fundamentals, design strategies, and applications* (pp. 138-217). Weinheim: Wiley-vch.
3. Kephart, J.O. and Chess, D.M., 2003. The vision of autonomic computing. *Computer*, 36(1), pp.41-50.
4. Begum, S., Ahmed, M.U. and Funk, P., 2009. Case-based systems in health sciences: a case
4. Parashar, M. and Hariri, S., 2004, September. Autonomic computing: An overview. In *International workshop on unconventional programming paradigms* (pp. 257-269). Springer, Berlin, Heidelberg.
5. Azim, M.T., Neamtiu, I. and Marvel, L.M., 2014, September. Towards self-healing smartphone software via automated patching. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering* (pp. 623-628).
6. Combemale, B., Broto, L., Crégut, X., Daydé, M. and Hagimont, D., 2008, September. Autonomic management policy specification: From uml to dsml. In *International Conference on Model Driven Engineering Languages and Systems* (pp. 584-599). Springer, Berlin, Heidelberg.
7. Puviani and Friel (2013). "Self-Management for cloud computing". *Proceedings of Science and Information Conference (SAI), 2013: 940–946 – via IEEE Xplore.*
8. Hellerstein, J., Kephart, J., Lassetre, E., Pass, N., Safford, D., Tetzlaff, W. and White, S., International Business Machines Corp, 2004. Self-managing computing system. U.S. Patent Application 10/252,247.
9. Barel, M.A., Carter, S., Crosskey, J.P., Ernest, L.M., Evans, D.H., Ford, L.L., Lilies, R.C., Spence, D. and Swett, A.L., International Business Machines Corp, 2011. Method, apparatus, and program for implementing an automation computing evaluation scale to generate recommendations. U.S. Patent 8,019,640.

10. Saha, G.K., 2007. Software-implemented self-Healing system. *CLEI Electronic Journal*, 10(2).
11. Chan, K.M. and Bishop, J., 2009, May. The design of a self-healing composition cycle for Web services. In 2009 ICSE workshop on software engineering for adaptive and self-managing systems (pp. 20-27). IEEE.
12. Toohey, K.S., Sottos, N.R., Lewis, J.A., Moore, J.S. and White, S.R., 2007. Self-healing materials with microvascular networks. *Nature materials*, 6(8), pp.581-585.
13. Hansen, C.J., Wu, W., Toohey, K.S., Sottos, N.R., White, S.R. and Lewis, J.A., 2009. Self-healing materials with interpenetrating micro vascular networks. *Advanced Materials*, 21(41), pp.4143-4147.
14. Karray, M.H., Ghedira, C. and Maamar, Z., 2011, March. Towards a self-healing approach to sustain web services reliability. In 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (pp. 267-272). IEEE.
15. Watson, I., 1995, January. An introduction to case-based reasoning. In *UK Workshop on Case-Based Reasoning* (pp. 1-16). Springer, Berlin, Heidelberg.
16. Watson, I., 1999. Case-based reasoning is a methodology not a technology. In *Research and Development in Expert Systems XV* (pp. 213-223). Springer, London.
17. De Mantaras, R.L., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., T COX, M.I.C.H.A.E.L., Forbus, K. and Keane, M., 2005. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3), pp.215-240.
18. López, B., 2013. Case-based reasoning: a concise introduction. *Synthesis lectures on artificial intelligence and machine learning*, 7(1), pp.1-103.
19. Richter, M.M., 1995. The knowledge contained in similarity measures.

20. Kolodner, J.L., 1996. Making the implicit explicit: Clarifying the principles of case-based reasoning. *Case-based Reasoning: Experiences, Lessons and Future Directions*, pp.349-370.
21. Watson, I. and Marir, F., 1994. Case-based reasoning: A review. *Knowledge Engineering Review*, 9(4), pp.327-354.
22. Monperrus, M., 2018. Automatic software repair: a bibliography. *ACM Computing Surveys (CSUR)*, 51(1), pp.1-24.
23. Blair, G.S., Coulson, G., Blair, L., Duran-Limon, H., Grace, P., Moreira, R. and Parlavantzas, N., 2002, November. Reflection, self-awareness and self-healing in OpenORB. In *Proceedings of the first workshop on Self-healing systems* (pp. 9-14).
24. Gu, T., Sun, C., Ma, X., Lü, J., & Su, Z. (2016, September). Automatic runtime recovery via error handler synthesis. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 684-695). IEEE.

