IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 10, Issue 3 (Mar. - Apr. 2013), PP 78-82
www.iosrjournals.org

Web-Based System for Software Requirements Quality Analysis Using Case-Based Reasoning and Neural Network

ABM Tariqul Islam[1], Hajar Mat Jani[1], Adibah Shuib[2]
[1](*College of Information Technology, Universiti Tenaga Nasional, Malaysia)*
[2]*(Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Malaysia)*

Abstract : *The overall success of a software project depends on the quality of its software requirements specifications (SRS). Hence, it is very important to get the requirements correct from the onset of the project. This research paper proposes a web-based system to perform SRS quality analysis using Case-Based Reasoning (CBR) and Artificial Neural Network (ANN). CBR is an AI technique that learns and deduces solutions based on past cases or experiences that are stored in a case base. However, when the case base becomes very large, the "Retrieve" phase of CBR becomes very tedious. So, in this research, we use ANN to improve the retrieval phase within the CBR. ANN measures the similarity of the new case against all existing cases in the case base. This results in a more efficient method of performing quality analysis for a given SRS document.*

Keywords **–** *Software Requirements Specifications (SRS), Quality Analysis, Case-Based Reasoning (CBR), Artificial Neural Network (ANN), Similarity Measurement*

I. INTRODUCTION

Case-Based Reasoning (CBR) is the process of solving a new problem based on past experiences or cases [1]. A single case describes one particular situation or case and is independent from another case. Each case records several features and specific values occurred in that particular situation. So, when a new problem arises, it is compared against all existing cases at hand, and the best solution or suggestion is derived from the system. CBR mimics human problem-solving methodology. A CBR system reasons and learns by recalling and remembering past experiences.

Software Requirements Specifications (SRS) encapsulates the high level description of the overall end system. It determines and decides the fundamental building blocks on which the end system will act on. Usually requirements come from the end user, which means software requirements identify what the system must do and voice how the end user wants the system to be. So in essence, it is crucial to get the requirements right before designing the system.

The quality of the end system [2] depends on the quality of its requirements [3]. We can decide if a single requirement is of quality or not based on some quality attributes and quality indicators, which have already been identified by previous researchers. If the requirement meets high quality indicators in all its quality attributes only then we can pass it as a high quality requirement.

A web-based system is a collection of applications and services that resides on a web server and accessible over the Internet. The primary benefit of a web-based system is that it does not depend on the client's platform since the application is usually accessible via standard web browsers.

Neural Network (NN) is a computational model that simulates the human brain. It is often called Artificial Neural Network (ANN). The purpose of ANN is to recognize patterns in large scale data sets. To enable ANN to recognize patterns, first, it must be trained with sample inputs and desired output patterns. Being an adaptive system, once the ANN has been trained, it is able to detect similarities among inputs even though a particular input may have not been experienced before. ANN can provide optimal solutions or make suitable prediction by analyzing and finding patterns in the new problem.

This paper introduces a web-based system that performs quality analysis of SRS in an efficient way using CBR and ANN.

II. BACKGROUND

At the initial stage of a software project, the common process of collecting software requirements is via face-to-face meeting with the client (customer, end user) and documenting the requirements in natural language (e.g. English). As the project commences, new requirements come in with high priority or old requirements get ignored, which might easily create interdependency among the requirements and even a well-written requirements specifications document sometimes creates ambiguity and fails to express priority and dependency of the requirements.

Later, Unified Modeling Language (UML) is introduced to manage software requirements. UML is an object modeling language that soon became the industry standard to manage software intensive systems. Use case diagram can be used to represent a particular requirement as a single atomic business function of the system. Since use cases are not expressed in descriptive language, rather in diagrams, they can provide better picture of dependency and priority of the requirements. There are plenty of tools already available in the market to manage software requirements. *RequireIt* (from Telelogic AB, later acquired by IBM), *Rational RequisitePro* (from IBM), *SLATE* (System Level Automation Tool for Engineers from SDRC), *DOORS* (Dynamic Object-Oriented Requirement System from Telelogic AB), *CASE Spec* (from Goda Software) are some tools that use word processors, relational databases or object-oriented databases for managing software requirements.

These tools are mostly commercial and enterprise standard tools, which not only assist in the requirements phase but also help manage the whole software development life cycle (SDLC). These commercial tools are all very feature rich and help document and manage software requirements in the best ways possible. But these tools do not necessarily provide quality analysis of the requirements at hand. At any point of the SDLC, a system developer might want to reflect on the quality of requirements rather than how well he or she managed the requirements so far at that point of time. The proposed system is a web-based application where the systems developers can perform quality analysis on software requirements for their various projects.

III. RESEARCH OBJECTIVE AND OUTCOME

The goal of this research study is to propose a web-based system that will facilitate users (Software Systems Developers) to sign up, log in and perform quality analysis on their software requirements specifications (SRS). Research on quality attributes and quality indicators is outside the scope of this project since there has been extensive research on this premise and some solid discovery has already been made. This project will base itself on the research work done in [4, 5] regarding SRS quality analysis using a set of selected quality attributes and quality indicators.

This research study aims to take advantage primarily of CBR, which is a widely acclaimed research area in computational science. The main objective is to implement and use CBR process for the quality analysis of SRS within the web-based system. Implementation of ANN is at the "*Retrieve*" [1] phase of CBR for similarity measurement of a new case against the existing case base or knowledge base, which holds all past experienced cases. Basically, a CBR system or cycle has four main steps, which are "*Retrieve*", "*Reuse*", "*Revise*", and "*Retain*".

The final outcome of this research study is a standard web application that will capture quality attributes and quality indicators inputs from users, process it with an efficient algorithm, and present a quality analysis report of the SRS to the user. The algorithm, which incorporates CBR and ANN, is the main focus of this research study.

IV. METHODOLOGY

The overall methodology for this research project is *quantitative*, but some elements of *qualitative* research methodology are also used since this research is related to SRS quality analysis. The length of time taken to find a solution using a given case base and the proposed AI techniques will be measured for performance analysis and benchmarking of the proposed system.

Gathering required information for this project is divided into two stages. First, during literature review, where a study on what has already been done in this research premise is carried out. Second, during requirements gathering process in which requirements details are collected from the system users (when they access and use the system). It is necessary to capture requirements information from the live user to ensure the growth of the case base over time. In a typical scenario, the live user is presented with a set of HTML forms to provide software requirements information. Then, the system starts processing the answers from the user, performs a full cycle of CBR (with embedded ANN), and provides a solution. A new case is added to the case base if the system hits the "*Retain*" step of CBR.

Initially, the system will start off with an empty case base, which is not very convenient. To overcome this issue, a decent number of cases (past experienced cases) need to be added to the case base in order to train the ANN and make it work with CBR.

V. THE PROPOSED WEB-BASED SYSTEM

A web application comes with some pros and cons on its own. The main advantage is that it is independent of the client's platform. It can also be accessed from anywhere, and can respond to almost any appropriate client. The disadvantages are network latency, server downtime and browser compatibility issues to name a few. However, the advantages seem to be overwhelming and the disadvantages can be controlled and further minimized if we follow some standard web application design and implementation best practices [6].

The proposed system as illustrated in Fig. 1 will follow the MVC (Model-View-Controller) design pattern [7]. In computing, design patterns refer to reusable solutions to solve commonly encountered problems in a particular context without having to reinvent the wheel. By nature, a well-known and widely used design pattern is the best solution for a problem if that design pattern validates that particular problem context.

MVC (Model-View-Controller) design pattern has established itself as the de facto standard for web applications in recent years. By definition, MVC is a three-tier system with separation of concerns and loose coupling among the tiers. Separation of concerns simply means that there should be as minimal overlap as possible among the tiers and MVC executes it nicely.

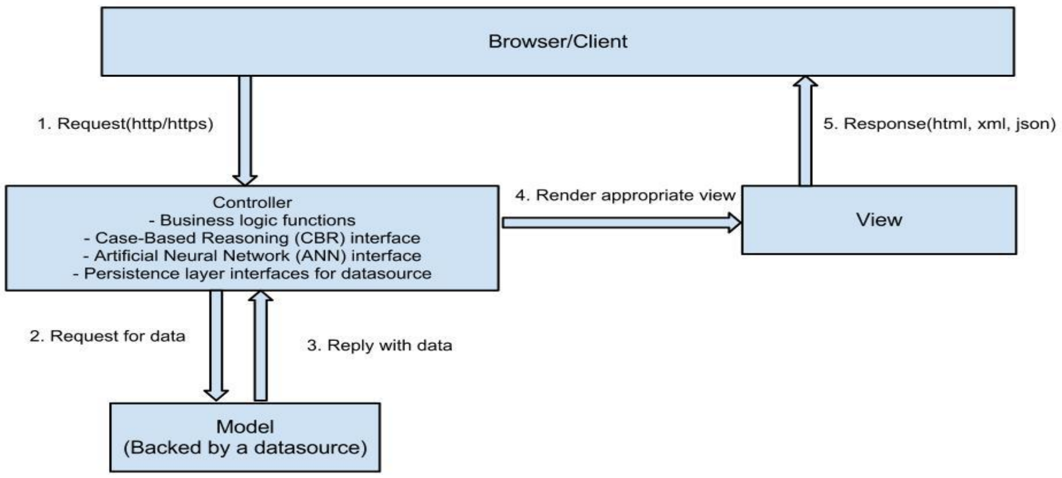
Browser/Client
1. Request(http/https)
5. Response(html, xml, json)
Controller
- Business logic functions
- Case-Based Reasoning (CBR) interface
- Artificial Neural Network (ANN) interface
- Persistence layer interfaces for datasource
4. Render appropriate view
View
2. Request for data
3. Reply with data
Model
(Backed by a datasource)

Figure 1. Model-View-Controller (MVC) design pattern

In MVC, the core business logic is consolidated in the Controller and the Controller usually receives all HTTP requests. After a particular Controller has received a request, it asks the Model(s) for data feed and renders a view for that request. Model is usually backed by data source(s). The Controller can support multiple Views for the same request and renders the appropriate one based on the request context and business logic.

The system will require the user to sign up and create at least one software project to perform quality analysis on the requirements specifications of that project. The system will allow the user to create multiple projects, but each project should be unique on its own and will be treated as a single case within the CBR system. After creating a project, the user will be asked a series of questions regarding quality attributes and quality indicators for the project's requirements. The system will capture all answers and apply the core algorithm comprising of CBR and ANN and present an analysis report to the user.

CBR is a four-step cycle comprising of "*Retrieve*", "*Reuse*", "*Revise*", and "*Retain*"[1], which nicely fits into the problem domain of this project. After collecting all of the required information from the user, the system will match the new problem case against previously experienced cases in the case base, and fetch all cases with most similarities [8]. The similarity measurement needs to be intelligent and highly optimized [9] because the size of existing cases in the case base will grow over time, and it will be unwise to match a new problem case against a million existing cases, which will result in serious performance drawback.

The system will assign scores to all existing cases based on their features and group the cases based on the scores. So, while performing similarity measurement, the system will only match the new case against groups or classes rather matching against all existing cases. The groups can be further grouped at another level to narrow down the search space even further. ANN engine will be implemented on top of this grouped data structure to perform similarity measurement via pattern recognition (also known as classification [9, 10, 11]) for the incoming case. The ANN engine will follow supervised learning paradigm, which is a process where ANN is trained with training data sets for pattern recognition [12]. Each example or case in the training data set is a pair comprising of sample inputs and output vectors. Learning from the sample data, ANN can perform efficiently when encountered with actual problems in real application. In the proposed web-based system, the case base will be the training data set for the ANN. So, when new cases are added to the case base, the ANN engine also needs to be retrained accordingly.

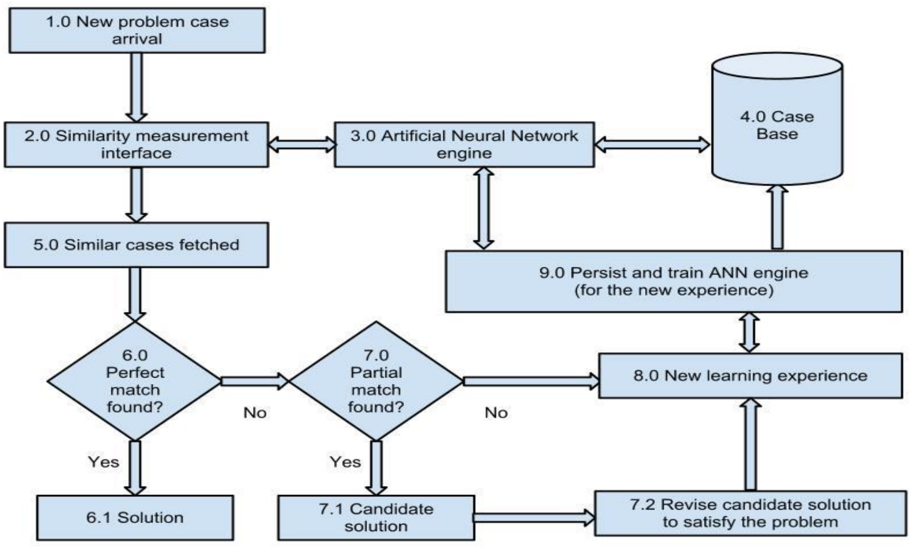
1.0 New problem case arrival
2.0 Similarity measurement interface
3.0 Artificial Neural Network engine
4.0 Case Base
5.0 Similar cases fetched
9.0 Persist and train ANN engine (for the new experience)
6.0 Perfect match found?
7.0 Partial match found?
8.0 New learning experience
No
No
Yes
Yes
6.1 Solution
7.1 Candidate solution
7.2 Revise candidate solution to satisfy the problem

Figure 2. Framework: high level process flow of the proposed system

As soon as the search algorithm finds a *perfect match* it will discard all partial matches found so far. So, the result set of similarity measurement search will contain either a *perfect match* or a number of cases with *partial match*. If a *perfect match* is found in the result set, the system will take it as the target solution and CBR for this problem will end here ("*Reuse*").

If the search does not yield a *perfect match* but manages to find some partial matches above the minimum threshold value (The system administrator will be able to specify a minimum *partial match* threshold value), the case with the highest *partial match* will be taken as a target candidate solution. The candidate solution will then be revisited and a new version will be adapted ("*Revise*") to satisfy the problem case and considered as a target solution. The target solution (revised version) is now a newly experienced case, and it will be appended to the existing case base ("*Retain*").

It is possible that the search algorithm might return an empty result set, which means neither a *perfect match* nor any *partial match* is found in the search space. It is likely to happen when the system initially starts off with a blank case base or the problem case is unique on its own. The system will consider the problem case as a new learning experience, which is also the target solution, and append it to the existing case base ("*Retain*").

After finding a solution, the system will present the SRS quality analysis report to the user. The report will be graphical with charts and graphs for better readability and understandability for the user. The user will be able to export the report to a portable document format if necessary.

The framework of the entire proposed system is shown in Fig. 2. A prototype of the web-based SRS quality analysis system has been developed (on-going), and some of the main features within the proposed framework such as the input data screens and simple analyses of new cases involving SRS quality analysis have been successfully implemented and tested. But, more time is needed in determining the effectiveness of the algorithm that uses both CBR and ANN in the analysis process of SRS.

5.1.Prospective use of the system

The system will build a database of solutions that is referred to as the case base or knowledge base, which holds all past cases or experiences with regards to SRS quality analysis. This case base will continue to grow over time. After a certain point, it is expected to become entirely comprehensive, and from here machine learning can take place efficiently.

Along with being capable of analyzing the quality of SRS for a new set of inputs related to a given software project, it should be able to serve as an interactive guide to anyone who is new to designing and developing software systems. This would allow the end user to have a more intimate understanding of their systems. It will also help systems developers gain more understanding of the prevailing trends and demands of SRS, and how they can be met in the most efficient and economical manner. In essence, this proposed system will bridge the gap between the end user and the systems developers by allowing them to attain clarity about the scope and requirements of a project.

5.2.Future work

In the future, the system may be used in academic circles, such as universities, as a teaching and learning aid in classes. When learning about software development process, the system may be used as a good online reference by the students. Teachers and instructors do not have to spend too much time explaining about the criteria of good quality SRS since the web-based system has all the details of what a good quality SRS document should have. Furthermore, the inherent nature of CBR and ANN lends them as great instructional tools as it is fundamentally designed on human behavior in such a way that the students will be able to learn efficiently by recalling and remembering past experiences, and this will definitely enhances students' learning process. This method of learning is very similar to the way human beings learn.

VI. CONCLUSION

It is crucial to diagnose the correct SRS right at the beginning of a project. The proposed method should increase the efficiency of the "*Retrieve*" phase of CBR by replacing the traditional linear search with ANN. The search takes place during similarity measurement between the new case and all past cases that are stored in the case base. Systems that yield partial matches should produce solutions very close to the original problem or new case. This can be used to extrapolate the SRS of a new project. In addition, for a web-based system to be usable, the user interface of the system should be simple, efficient, and easy to use.

Acknowledgement

This research study is funded by the Fundamental Research Grant Scheme (FRGS), Ministry of Higher Education (MOHE), Malaysia.

REFERENCES

[1] A. Aamodt, and E. Plaza, Case-based reasoning: foundational issues, methodological variations, and system approaches, AI Communications 7(1), 1994, 39–59.

[2] A.C. Gillies, *Software quality: theory and management* (London: Chapman & Hall, 1992).

[3] IEEE-Std-830-1998, *IEEE Recommended practice for software requirements specification*, doi:10.1109/IEEESTD.1998.88286.

[4] H.M. Jani, S.A. Mostafa, Implementing case-based reasoning technique to software requirements specifications quality analysis, *International Journal of Advancements in Computing Technology (IJACT), 3(1),* 2011, 23-31.

[5] H.M. Jani, Applying case-based reasoning to software requirements specifications quality analysis system, *Proc. 2nd International Conference on Software Engineering and Data Mining (SEDM)*, IEEE/AICIT, 2010, 140-144.

[6] S. Fowler, and V. Stanwick, *Web application design handbook: best practices for web-based software* (San Francisco, CA: Morgan Kaufmann Publishers, 2004).

[7] A. Leff, J.T. Rayfield, Web application development using the Model/View/Controller design pattern, *Proc. 5th IEEE Int. Enterprise Distributed Object Computing Conference*, 2001, 118-127.

[8] E. Armengol, S. Ontanon, and E. Plaza, Explaining similarity in CBR, *ECCBR 2004 Workshop Proceedings*, 2004, 155-164.

[9] T. Warren, L.Z. Zhang, and C. Mount, Similarity measures for retrieval in case-based reasoning systems, *Applied Artificial Intelligence*, *12(4)*, 1998, 267–288.

[10] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern classification*, (2nd Edition, New York: John Wiley and Sons, 2001).

[11] G. P. Zhang, Neural networks for classification: a survey, *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 2000, 451-462.

[12] C.M. Bishop, *Neural networks for pattern recognition* (Oxford: Clarendon Press, 1995).