Universal Journal of Electrical and Electronic Engineering 6(3): 94-100, 2019 http://www.hrpub.org
DOI: 10.13189/ujeee.2019.060302

Conceptual Framework: Rapid Web Application Development by Using Component Reuse and Case Based Reasoning

Eka Angga Laksana*, Feri Sulianta

Department of Informatics, Widyatama University, Indonesia

Abstract Developing web application from scratch is a cumbersome task. There is a need for a fast and efficient methodology to develop web-based application to realize user's requirement. User's requirement and expectation have increased because of the complexity of business process. This paper introduces a new method which allows web developer to build application faster. Reuse-based software component concept is used to construct cases in case-based reasoning. Case-based reasoning method defined a mechanism for learning on the basis of the previous case in order to solve the current problem. That is the basic idea about how to build application generator as fast development by reuse collection of previous code component.

Keywords Application Generator, Case-based Reasoning, Learning, Reuse Component, Web Application

1. Introduction

Now day, application development becomes a very interesting topic for discussion. Currently the provision of service becomes a necessity of human life. Applications as a service provider also increase along with human needs. Software has become a part of human life and software's reliability is tested by how they react on specific environment under a given condition. To develop applications there is a need for methodology that has been conducted for many years covered by software engineering. Efficient software development method had become a necessity for developer to build a reliable system and also reduce the time of development. Problem arises when requirements increase, user expectation is high and the development time is limited. Therefore, there is a need for an existence of basic methodology associated with application development to solve the problem.

This problem occurs because time management lacks and client gives little time for developing application [1]. Web developers sometime rely on their experience rather than standard practice on web application development [2]. Web-based application become more complex along with heterogeneous end user, however the method that developer follows today is still poor. Many organization and industry succeeded in developing website and application, but the others failed and still struggled to face the potential failure [3]. They face some challenges when developing the system because web-based system is ad hoc, lack of systematic method and quality control and certainty procedure [4]. Website development is usually accomplished without following a well-designed process and lacks of appropriate tools to support [5]. Therefore, there is a need for rapid application development of web-based application that encompasses standard methodology on software engineering [2].

On the other hands, when developing system especially web-based application, there are a lot of tasks that are repeated frequently. On web-based application, some functions occur as basic system that common application has. For example, the usage of CRUD (Create, Retrieve, Update, and Delete) is always a standard requirement for database manipulation. Furthermore, creating menu, user form and authentication is also related to the way that the system must behave. Therefore, the method for developing web-based application from scratch is not efficient work, because we can reuse some basic component. Some patterns of the programming language can be learned because they share the same requirements. Pattern documentation can be made of both code itself and related requirement.

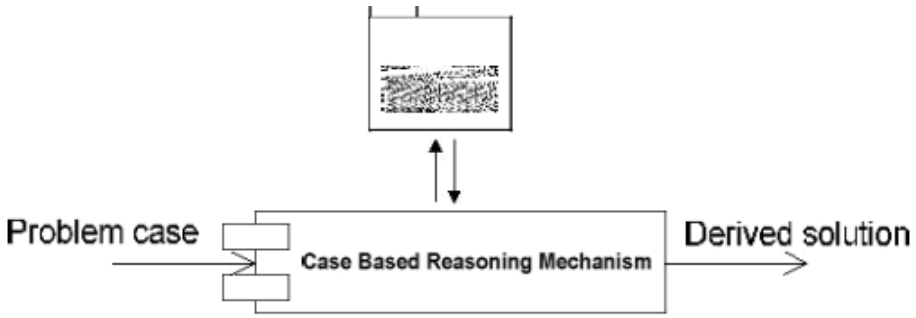

Figure 1. Case Based Reasoning Schema

Case-based reasoning has been used in various research fields in many years. Cased-based reasoning relies on previous problems and solutions and is adapted to solve current problems. Analogically, in the hospital when a new patient gives some symptoms documented on medical record, the doctor will check the similar symptom that happened previously on another patient. Then treatment for that patient will be conducted and if necessary, it is adapted properly to suit the new patient. Case-based reasoning tries to make a knowledge gathered from previous case; therefore all related cases must be documented properly to fit with the current problems as is shown in figure 1. The idea that case is not restricted to problems within specific domain, but generally can be implemented in software engineering field. In the software engineering fields, there are many problems that have been described in the previous paragraphs. This paper tries to make a conceptual framework for developing web-based application to minimize redundancy and increase development time by utilizing the methodology of case-based reasoning.

The structure of this paper is as follows: section 2 introduces related woks about application development and case-based reasoning; section 3 describes theory and concept about case-based reasoning, software pattern and software reuse; section 4 explains about conceptual framework for web application development by using case-based reasoning. Finally, section 5 is the conclusion of this paper.

2. Related Works

There are several studies related to web application development. A MVC model proposed by [6] to add benefit and maintainable code is different from [7] which makes effort estimation in web-based application development. Empirical study on Web usability has been investigated by [8] and some usability evaluation was given. Paper [9] focuses on the processes of estimating, planning and managing in Agile Development web project.

Case-based reasoning has been used in many different fields. Flood disaster forecast based on multi-agent approach was proposed by [10]. In software engineering field, non-functional requirement is analyzed to estimate software effort accuracy. Soil classification based on case-based reasoning proposed by [12] reported that those methods can significantly improve judging. A hybrid of case-based reasoning and neural network for credit risk analysis proposed by [13] can increase classifier performance.

3. Theory and Concept

3.1. Case Based Reasoning Life Cycle

Basically, problems solved by case-based reasoning can be divided into following four parts (as shown by figure 2) [14]:

a) *Retrieving* the case based on past experience that seemed similar to the problem.
b) *Reusing* the case by using the solution from the previously retrieved cases.
c) *Revisingand* adapt the solution in order to solve a new kind of problem.
d) *Retaining* the new solution when it's already validated and confirmed.

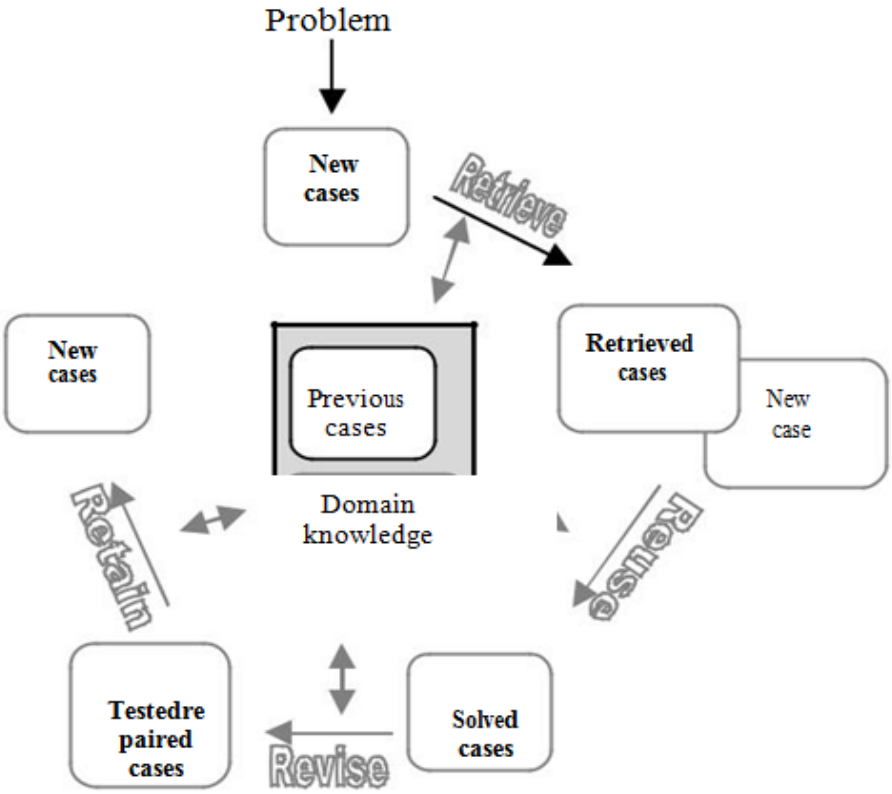

Figure 2. Case Based Reasoning Life Cycle

Although case-based reasoning has been applicable in various research fields, there are some characteristics which determine whether it can be implemented or not. Some analyses have been conducted to make sure that case-based reasoning applicable on problem within this domain. As is mentioned below, five points [15] [19] as requisition to implement case- reasoning on web development problem:

a) Does the domain problem have a model?

If the domain problem is unable to be modeled completely or there is no model of solution which leads to success or failure (medical diagnosis or weather forecast), case-based reasoning works based on the past experience even without understanding the mechanism. When developing web based application, previous component can be reuse and even without knowing the underlying mechanism completely.

b) Are there exception and novel cases?

In the condition when the new exception and experience encountered frequently, it would be difficult to store it as case in the system. Sometimes when developing an application, developer encounters a new problem. Unfortunately, developer must start again from scratch when encountering the same problem.

c) Do cases recur?

The key point is similarity; programming code can be divided into some components. The same case based on previous experience occur when developing the new project.

d) Is there any benefit in adapting past solution?

Modify solution that previously gathered to fit the current problem. In programming, there is reuse concept that is easily adapted to the requirement.

e) Are previous cases obtainable?

Recorder cases contain the relevant feature on both cases along with attribute and the solution. In programming, previous code can be recorded as case by following the design pattern.

3.2. Pattern and Software

Software can't be separated from pattern. The problem that encountered while developing software always appeared again and again. Patterns will help us to build body of solution and share experience and use that as body of knowledge.

What is fundamental to any science or engineering discipline is that a vocabulary is defined to describe its concept and try to make a language to be related them together. Formally, codifying these solutions and their relationship allow us to create body of knowledge which determines our understanding of software architecture that fits with user requirements [16]. Forming pattern on common programming language by revealing the structures and architecture mechanism helps us to understand the reason behind them. A pattern must be useful, because by learning pattern and transform it in real

word will add values to application developer and practitioner.

3.2.1. Element of a Pattern

According to Appleton [16], element of software pattern should contain the following parts:

a) Pattern Name,

A pattern name should be meaningful. It describes the actual concept and is understandable. Classification also can be shown in addition to its name.

b) Problem,

Describe the objective to reach within context and force. Problem shows clear statement which contains actual condition.

c) Context,

Precondition of the system before pattern applied to it. Show the initial condition before problem and its solution occur.

d) Forces,

Constraints that force certain rule that how they interact with another. Force the pattern to follow certain scenario which is frequently employed. A good pattern must have some forces to encapsulate it.

e) Solution,

Solution is the desired outcome which can solve the main problem. A set of instruction constructs the works product. Show the pattern structure by pictures, diagrams, and prose. The solution also shows not only static but dynamic behavior. Static structure tells about architecture of the pattern while dynamic behavior tries to make pattern come alive.

f) Examples,

Illustrating sample application of the pattern shows applied pattern on application and how it transforms in specific context. It makes pattern more understandable.

g) Resulting Context,

Resulting context is the condition that pattern has been applied to, including the result and another problem that emerging after implementation. Sometimes it is called 'resolution of force' because it tells how force has been resolved, which one is still unresolved and describes which pattern is applicable. Resulting context documentation helps to be correlated with the initial context of another pattern

h) Related Pattern,

Show relationships between patterns, within the same patterns of language or systems. Sometimes some patterns are related by same force. It shows compatibility with the other pattern based on the initial context or the result. Some patterns are treated as predecessors whose application potentially this pattern and successor pattern whose applications use it. An alternative pattern gives different solutions to the same problem by different forces and constraints.

3.3. Software Reuse

Software reuse has been used as an important tool for development in order to deal with software production, maintenance cost, fast delivery, and increase software quality. There are three approaches that classify reuse-based software engineering [17]: application system reuse, component reuse, object and function reuse. There are potential reusable entities on software system and component, but sometimes the differences of business requirement have a specific nature that forces them to adapt to the new situation.

Methodology on this paper can be classified as component reuse, the middle tier of reuse-based software engineering. Component reuse divides a system into some components which may be reused. Components can be arranged by software element pattern that has been described in the previous paragraphs. Software components are potentially reusable entities, but sometimes it is hard to modify them in specific situation [17]. This is the idea as the purpose of this paper is to make efficient web application development by fully understanding the benefit of component reuse and software pattern.

3.4. Application Framework

Application framework was commonly used in object-oriented development as key benefit of software reuse. It consists of abstract and concrete classes which are adapted and extended to build an application. In web application development, there is some open sources of application framework. Application framework helps to develop application faster and efficiently, but it still builds with small pieces of class and object. But the programmers still write the code from scratch when constructing application to fit the business requirement. On the other hand, according to software pattern concept it can be combined into compact element. For example, making CRUD (Create, Retrieve, Update, and Delete) by using application framework must follow the procedures based on MVC (Model, View, Controller) concept, write query, set database and so on but from the another point of view it also can be considered as pattern. Applications that are built by using framework can be extended for the basis for further reuse through the concept software product liner application families [17] [18].

3.5. Generator Based Reuse

Generator based reuse is an automated tool that implements reusable concept, which offers an easy way to develop application by integrating specific code [17]. Generator system gives a solution to specific application and is selected by developer to create a new system based on saved requirement as pattern as is shown in figure 3. The main purpose of this paper is to make automated tools

for developing web-based application by integrating some software components. The mechanism is to convert some software components (according to software concept pattern) into case as base of case-based reasoning methodology to create fully application generator. Software requirement can be treated as problem and the related code component as the solution. The code itself can be created from previously stored component as Case Based Reasoning concept. Section IV explains more specific explanation about how the concept to make web-based application generator by using Case Based Reasoning.

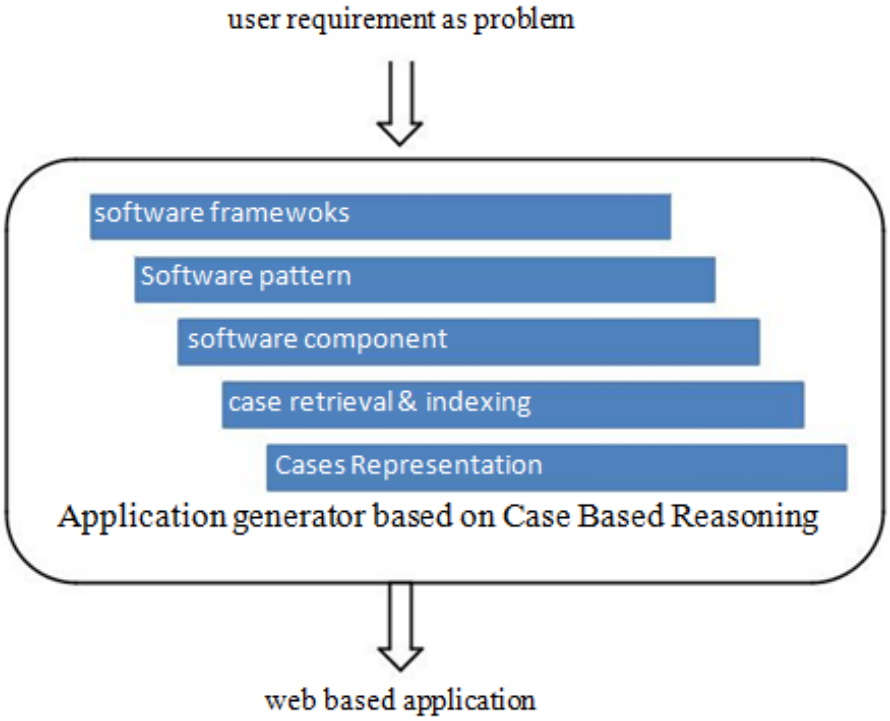

Figure 3. Application generator schema

Pattern name:	Case-name
Problem:	<problem definition>
Context	< precondition when the solution occur>
Resulting Context:	<post condition when the pattern has been applied>
Forces	<force description>
Examples	<example usage of the pattern>
Related pattern:	<show the related pattern>
Known uses	<prove that solution has been accepted>
Solution	<a set of solution that show the user & page behavior, use case, data behavior, architecture & security>

User Behavior	Page Behavior	Prototype	Data Behavior	Architecture	Security
User	User: Insert Update delete	Insert Update delete 1 Abc 2 Def 3 efg	Database table	Client | Server	Form filtering

Figure 4. Case Representation based on software pattern

4. Conceptual Framework for Web Application Development by Using Case Based Reasoning

Cases can be divided into two components: problem specification and solution. Problem specification is constructed by the set of attributes and values. The attribute of case shows that the case is unique and suitable to predict a solution to it. There are five steps [14] to build case-based reasoning system which is described below:

a) Case representation & Indexing

Case can be represented in many different formats. As the first step of case-based reasoning, it will lead to the success of this method. As is described before, a case consists of both problem and related solution. However, to make a case representation can be one of the most difficult tasks.

In this paper, case representation format has already been described on section 2. This representation is based on software pattern, the main element that construct software pattern consist of: pattern name, problem, context, resulting context, forces, example, related pattern, known uses and solution as is displayed in figure 4. Software component plays important role as the main component in software generator. Software component is the real programming coded by a specific programming language.

In this paper, making representation format could be done by learning the software pattern which has been described in section III. The pattern itself is adapted from [2] and [16].

Case indexing is a mechanism for assigning index to cases; it is useful for retrieval and comparison in the future steps. Indexes play an important role in the retrieval of the right time and case, because it will determine which context will be retrieved in the future. Indexed case will be stored in the system and will help system in fast searching toward the case.

b) Case retrieval

Case retrieval is a process of searching within the case base which is closest to the current problem. There must be selection criteria to make sure how a case suitable for retrieval and how the case based is searched. Standard case retrieval is to search for an entire case and then compare it to the current case. Retrieval methods that are commonly used is diverse, ranging from a simple nearest-neighbor to intelligent agent. Retrieval is the most important research area in case-based reasoning.

In this process, case-based reasoning adopts the similarity measure to greatly achieve retrieval performance. The effectiveness of similarity measurement is defined by usefulness of retrieved case to solve a new problem. Similarity measure will compare current case with previously indexed cased.

c) Case adaptation

In this step, the retrieved solution is transformed into a solution to the current problem. It has been debated that case adaptation is a key step of case-based reasoning because it adds intelligence to pattern matchers. After the adaptation has been applied, there is a checking to know whether any differences between case retrieved and current problem exist. Furthermore, it also needs to be considered if proposed solution is not working. At this point, the solution proposed is ready to be tested in the applicable domain.

d) Case-based learning

A case-based reasoning can be updated to make any new information uncovered as processing as a new solution. This information will be added to the system for two purposes: first, more case stored in case based will make a higher case match possibility; second, it will allow the system to improve the solution. Learning process may occur in many ways there is by newly add a problem, solution, case and the outcome.

e) Cased-based Maintenance

This step tells about performance of the system. When the number of cases stored in library increases, it will slow down the system performance. Removing redundant cases is one of the maintenance tasks; it will find an acceptable error of the system.

5. Conclusions

In software engineering field, how to reduce development time becomes an interesting topic [20]. This paper tries to build a fast web development application based on reuse concept and case-based reasoning. This method depends on the experience of software developer. Experienced developers know really well about software pattern from their previous projects. Software pattern can be transformed into specific format, consisting of pattern name, problem, context, resulting context, forces, example, related pattern, known uses and solution. Software component, as one of the software reuse concept, can be constructed by those pattern. There are relations between user requirement and software pattern. Basically the proposed idea in this paper is case-based reasoning which stores a set of cases which is constructed by combination of user requirement and related code. Therefore, a collection of pattern (which is gathered from previously project) is transformed into the case as case representation in case-based reasoning. Finally, a web-based application can be generated by retrieving similar case based on specific user requirement.

REFERENCES

[1] R. S. Pressman, “What a Tangled Web We Weave” no. February, 2000.

[2] R. S. Wahono, "Extensible requirements patterns of web application for efficient web application development," *First Int. Symp. Cyber Worlds, 2002. Proceedings.*, pp. 412–418, 2002.

[3] W. A. Development, "The Essence of Web Engineering," pp. 22–25, 2001.

[4] S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige, "Web Engineering: A New Discipline for Development of Web-based Systems," pp. 1–9.

[5] F. Coda, C. Ghezzi, G. Vigna, F. Garzoto, "Towards a Software Engineering Approach to Web Site Development". 1998

[6] D.-P. Pop and A. Altar, "Designing an MVC Model for Rapid Web Application Development," *Procedia Eng.*, vol. 69, pp. 1172–1179, 2014.

[7] D. Čeke and B. Milašinović, "Early effort estimation in web application development," *J. Syst. Softw.*, vol. 103, pp. 219–237, May 2015.

[8] A. Fernandez, S. Abrahão, and E. Insfran, "Empirical validation of a usability inspection method for model-driven Web development," *J. Syst. Softw.*, vol. 86, no. 1, pp. 161–186, Jan. 2013.

[9] C. J. Torrecilla-Salinas, J. Sedeño, M. J. Escalona, and M. Mejías, "Estimating, planning and managing Agile Web development projects under a value-based perspective," *Inf. Softw. Technol.*, vol. 61, pp. 124–144, May 2015.

[10] [10]B. Linghu and F. Chen, "An Intelligent Multi-agent Approach for Flood Disaster Forecasting Utilizing Case Based Reasoning," *2014 Fifth Int. Conf. Intell. Syst. Des. Eng. Appl.*, pp. 182–185, Jun. 2014.

[11] F. Fellir, "Analyzing the non-functional requirements to improve accuracy of software effort estimation through Case Based Reasoning," 2015.

[12] L. Qiu, "Automated Keys of Soil Diagnostic Horizons Based on Case-Based Reasoning," no. Cx.

[13] C. Silva, G. Vasconcelos, and G. Frana, "Case-based Reasoning Combined with Neural Networks for Credit Risk Analysis," 2015.

[14] S. K. Pal and S. C. K. Shiu, "Foundation of Case Based Reasoning". Willey-Interscience. 2004.

[15] J. L. Kolodner, "An introduction to case-based reasoning," *Artif. Intell. Rev.*, vol. 6, no. 1, pp. 3–34, 1992.

[16] B. Appleton, "Patterns and Software: Essential Concepts and Terminology," pp. 1–26, 2000.

[17] I. Sommerville, "Software Engineering, *Ninth Edition*".Adisson Wesley. 2011

[18] Hajirahimova, M., & Aybeniz, A. (2018). Current Approaches in Prediction of PVT Properties of Reservoir Oils. Review of Information Engineering and Applications, 5(2), 31-40.

[19] Hosseini, Z., Fazlollahtabar, H., & Mahdavi, I. (2014). Waste Management in Reverse Supply Chain Considering Pricing. Review of Industrial Engineering Letters, 1(1), 1-15.

[20] Jabarullah, N.H., Shabbir, M.S., Abbas, M., Siddiqi, A.F. & Berti, S. (2019) Using random inquiry optimization method for provision of heat and cooling demand in hub systems for smart buildings, Sustainable Cities and Society, 47, 101475.