Proceedings of the International Conference , "Computational Systems for Health & Sustainability"
17-18, April, 2015 - by R.V.College of Engineering,
Bangalore,Karnataka,PIN-560059,INDIA

A Novel Refactoring Appraoch to Remove the Smels Present in Programming Languages

K. Malini
Software Engineering, Anna University
Sri Ramakrishna Engineering College, Coimbatore
Tamil Nadu, India-641022

Dr. N. Rajkumar
Professor & HOD(Software Engineering)
Sri Ramakrishna Engineering College, Coimbatore
Tamil Nadu, India-641022

Abstract: Mash ups are the one which is created as a new source by using the knowledge of already existing sources. The Mash ups are used to create and develop new innovative ideas from the already existing knowledge's. However the mash up creation by users may tend to some deficiencies which need to be concentrated to avoid the complexity. There may be a possibility of creation of smells during the mash up creation process which need to be addressed well in order to avoid the software failure. This is avoided by introducing the methodology called refactoring approach which tends to find and eliminate the possible smells which may occur at the time of mash up creation. However in programming cannot support the programming languages with different object behavior in different places. The refactoring cannot be applicable for the smells identified in the new type of programming language. It is overcome in our work by analyzing the object behavior when it is used in different places. Based on this knowledge, refactoring has been applied with the consideration of the moving the objects across different classes and modules and predict the behavior changes occurred in the programming languages. By doing so, efficient refactoring can be done which can be used for any type of programming languages in order to avoid the smells at the time of mash up creation. The experimental conducted were proves that the proposed methodology lead to better performance than the existing approach in terms of mash up creation.

Keywords: Mash up, Refactoring, Smells, Behavior change.

INTRODUCTION

Software engineering is the process of gathering, analyzing and developing new innovative ideas. The software engineering tends to maintain the good qualified software development. It is a one of the most popular approach in which the already existing knowledge of program will be utilized to create new ideas.

Mash up is the most improved and accepted approach in today world through which burden of users can be reduced considerably and the more knowledge can be learned. From the name itself it is explained clearly that the mash ups are the one which is used to gather and integrate the knowledge from many sources. The important characteristics of mash ups are visualization, integration and aggregation process through which one can learn the most useful information. It is the one which can be used to further produce the most essential information. There are various types of mash ups are present in today world, those are:

- Business mash up
- Consumer mash up
- Data mash up

Business mash up is the one which is used to combine the resources of current application and the data from outside resources to create a new source of language. It can be useful for the business team in order to make collaboration among the people and the developers.

Consumer mash up is created by analyzing the public sources and combining the features from those data sources. Consumer mash up can be used to create a new source for the satisfaction of the users.

Data mash up is the one which is created by combining the knowledge from the data driven applications like media and information in order to produce a new type of information. This data mash up is useful for the person who tends to do a research process.

The characteristics of the mash up creation may suffer from various deficiencies. One of the important deficiencies which may occur at the time of mash up creation was smell creation. This smell creation may affect the execution of mash up in the worst manner. Thus smell creation during mash u need to be considered more for the efficient processing. And another limitation which may occur is the data from different sources cannot be incorporated together due to the different nature of the data. This problem needs to be addressed in order to enable an incorporation of the different type of data sources.

The main contribution of this work is to introduce the efficient refactoring approach by using which the new code can be constructed from the already existing codes without changing the behavior. It can lead to an efficient construction of the innovating methodology form the existing approaches. And also this work focus to support an different types of programming languages with different behavior by using the same refactoring approach.

The organization of this work is given as follows: Section 1 gives the brief introduction about the refactoring and its usage in today world. Section 2 gives an analysis of the related researches which has been conducted already. Section 3 gives a brief description about the proposed methodology for achieving effective refactoring. Section 4 discusses as experimental test which has been conducted to prove the effectiveness of the proposed methodology. And finally in section 5 results that were obtained are concluded.

RELATED WORKS

In [1], refactoring with the satisfaction of user is achieved by considering the smells present in the mash up. This work aims to reduce the smell which occurs during the run time of

mash up creation and remove it during the mash up creation. This process is tested in the yahoo pipe environment. Pipes are nothing but the interface to connect the processing of the new module and old module. The well efficient pipe can combine the new module with the old module in the efficient manner. The various types of smells are discussed in this work for the construction of the good mash up creation environment.

In [2], refactoring approach in the object oriented programming languages is discussed in the detailed manner which aims to provide suitable environment for the mash up creation. The object oriented programming languages is the combination of various functions and modules together in order to create useful information. There will be a presence of more interrelationship among the various functions and modules where the input and output details will be shared. In this environment, mash up creation will be more difficult process which needs to consider more. This work aims to build a refactoring for the object oriented programming language in which more suitable environment can be created.

In [3], new methodology for mash up creation is introduced in which mash up is created with the consideration of the user wish. In this work, at each and every iteration mash up are created with the consideration of the users wish. The mash up will be created by analyzing the nature of the existing code and the new code. The result obtained after combination of the old code with the new code, it will be analyzed and revised based on the user requirement. After gathering the requirements from the users, the efficient mash up can be created with the consideration of the user requests. And also this work considers the abstraction details of the programming language through which one can obtain the good refactoring code.

In [4], tool was developed through which the advice for the good mash up creation can be obtained. It is done to evaluate the more suitable environment through which one can obtain the more efficient and useful information and one can evaluate the various information of the coding structure. MashUp advisor aims to provide ideas for the good mash up creation by analyzing the factor of nature of programming, the type of abstraction present in the environment and then the different fields through which one can obtain the good mash up environment. It is enabled by predicting the nature and flow of information transaction among the various constraints.

In [5], automatic creation of mash up is discussed through which the complete coding can be analyzed and mash up can be created. To do so, the entire environment will be analyzed and predicted the nature of a coding and requirement that has been obtained. Matchup methodology is introduced to compare the existence features with the following new idea. Match up module is used to identify the similar measures which are present in the coding and thus it is obtained by predicting the various features. Data model and ranking metric is used to predict the nature of the coding environment which exists in the various places.

In [6], mash up creation for the intranet application have been discussed. Intranet application is the one which is used to provide the network services for the organization or for the one coverage area. There will be more communication will arise among the different system that are connected together for sharing the network information. In this phase, mash up creation will be difficult where objects behavior will be different in different places. This factor needs to be considered in mind in order to predict the nature of the mash up which is going to be proposed. The mash up creation must provide useful information for the new innovative ideas.

REFACTORING METHODOLOGY

Mash up is the one which intends to create a new source from the presence of the existing sources. It is used to make use of already existing sources and reduce the burden of developers to create new innovative ideas. The refactoring is one of efficient methodology which aims to create a new source from the already existing source without changing the behavior of the code. The main limitation which may occur at the time of mash up creation was the smell creation which needs to be avoided in order to give an efficient code. In the existing work, the different types of smells are identified and that are refactored to create a good mash up. However, existing work cannot support the programming languages where there is a lot of communication exists among different modules and behavior of objects will be different in different classes. In the proposed work, the refactoring approach for the programming languages is introduced which aims to eliminate the smells which may occur due to the different behavior of objects and provide an efficient mash up.

Refactoring in the programming languages are done with the consideration of the program behavior. While combining the new features with the already existing code, behavior of the existing code should not be changed. It is the most essential thing which needs to be considered in mind at the time of refactoring process. The refactoring which is compiled in our work to reach an efficient mash up are given as follows

- Constructing an super class form the more than two available class
- Improving the performance of class by introducing the sub class in which conditional test will be eliminated
- Placing the parent class into the class from which it is inherited
- Replacing an member variables and functions of class
- Inter changing the original code with the function call
- Changing the meaning and name of the classes and variables

The way to do this refactoring is discussed in the following sub section in the detailed manner.

A. Constructing an super class form the more than two available class

The construction of the super class from the existing classes is done provide a new class in which behavior of all of the existing class can be preserved. It is achieved by analyzing

and predicting the behavior and nature of the available classes and integrates them in to the new class. It is done by extracting the inter relation present among the objects of the different classes and presenting them in the hierarchical manner. It is done so that the super class can be created efficiently by abstracting the related objects of different classes together.

B. Improving the performance of class by introducing the sub class in which conditional test will be eliminated

In this step, refactoring is applied by considering the conditional statements which are exist on the super class. The super class may consist of various sub conditions which will be more time consuming process. And also, the user may want the new refactoring code among the set of conditional which is present in the super class. It will be more burdens to retrieve and present the super class with the more conditions. To overcome this problem, the following phase refactoring aims to present a class with various conditional statements as the single class. For example, the switch statement can be represented as the sub classes, where each and every case is defined as the sub class. By doing so, the complexity and burden of the code handling can be reduced considerably.

C. Placing the parent class in to the class from which it is inherited

In object oriented programming languages there may be a possibility of presence of various sub classes which are inherited from the single sub classes. It will be more complex process, where there is a presence of many sub classes. It can be aggregated together in order to create an single aggregated class by using which behavior of every sub classes can be obtained in the single place. However it will lead to creation of smells due to the nature of presence of the different behavior of the classes.

D. Replacing an member variables and functions of class

In object oriented programming languages, different classes will consists of the different member and attributes. Each and every attribute is used to create a more suitable environment which intends to provide different behavior. These attributes from different class may be wanted to provide an single behavior based on the user requirement. To achieve this, the movement of member class attributes and function into the single class is done with the consideration of the behavior changes. And also consistences among the different attributes are taken in mind in order to avoid the creation of smells in the mash up creation process.

E. Inter changing the original code with the function call

The user may want to have an behavior of the programming language but they may try to eliminate some part of coding and introduce a new concept instead of it. But it will create more smells by replacing the function with the different function due to the change of behavior. This is done by creating the new pipeline which intends to avoid the smell creation and supports the integration of different behavior of function with the same.

F. Changing the meaning and name of the classes and variables

In some cases, the new programming may be wanted create by without changing any of the behavior of the coding. It can be done by renaming or replacing the attribute with the help of another attribute. It is done by considering the behavior similarity level of each and every attribute with the other attributes. It is done by changing the class name, variable name, member function name, data type of variable, access control mode etc.,

Thus, efficient refactoring on the programming language is achieved by following the above methodology. By following this technique efficient refactoring can be achieved without the presence of the smells and also these refactoring methodology can be applied to any type of programming language in which inter relationship are present.

EXPERIMENTAL RESULTS

The refactoring procedure proposed in this work is done using the programming languages with many module and functions. In our work, java programming is considered in which refactoring is done. The experimental test were conducted to prove that the effectiveness of the algorithm by comparing it with the existing methodology in terms of the processing time. The comparison is shown in the following graph.

A. Processing Time

The processing time is the time taken to construct the refactoring process. The less processing time will lead to a more refactoring and support of many modules in the particular period of time. The processing time obtained in the existing work and proposed work is shown in the following table.

Table 1. Processing Time For Each Data

Number of Data	Processing Time in ms	
	Existing work	Proposed work
50	22	17
100	28	19
150	33	22
200	37	25
250	41	27
300	46	32
350	49	37
400	54	42

The graphical result for the comparison of proposed methodology with the existing method is shown in the following figure 1.

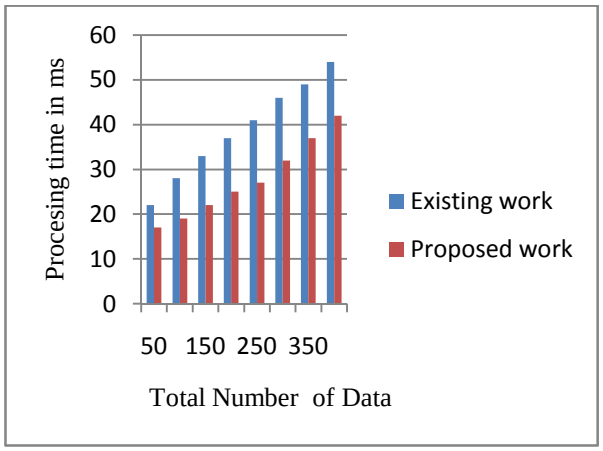

Fig.1. Efficiency of the process compared.

The above graph proves the proposed methodology constructs a refactoring better than the existing methodology. In the x axis total number of data is taken and in the y axis processing time is taken. The comparison proves that the existing methodology consumes more processing time whereas the proposed methodology works efficiently.

CONCLUSION

Refactoring is a methodology which is used to construct a new source from the already existing sources by preserving the behavior of the coding. In this work, the refactoring approach for the programming language is introduced which aims to reduce the problem of not supporting the different programming languages. The proposed work can tolerate the programming language with more condition which aims to improve the performance and reduce the complexity level. The experimental test conducted were proves that the proposed methodology can provide better results than the existing methodology.

REFERENCE

[1]. Kathryn T. Stolee, Sebastian Elbaum, "Refactoring Pipe-like Mashups for End-User Programmers", Software Engineering (ICSE), 2011 33rd International Conference on 21-28 May 2011

[2]. William F. Opdyke,"Refactoring Object-Oriented Frameworks", University of Illinois at Urbana-Champaign Champaign, IL, USA, 1992

[3]. Anton V. Riabov, Eric Bouillet, Mark D. Feblowitz, Zhen Liu and Anand Ranganathan, "Wishful Search: Interactive Composition of Data Mashups", Web Engineering - Web Service Composition, 2008

[4]. Hazem Elmeleegy Anca Ivan, Rama Akkiraju, Richard Goodwin, "MashupAdvisor: A Recommendation Tool for Mashup Development", 2008 IEEE International Conference on Web Services

[5]. Ohad Greenshpan, Tova Milo, Neoklis Polyzotis, "Autocompletion for Mashups", Proceedings of the VLDB Endowment, Volume 2 Issue 1, August 2009

[6]. David E. Simmen, Mehmet Altinel, Volker Markl, Sriram Padmanabhan, Ashutosh Singh, "Damia: Data Mashups for Intranet Applications", Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Pages 1171-1182.