Volume 2, Issue 12, December 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Refactoring Methods and Tools

Anshu Rani
Department of Computer Engineering
UCOE , PU Patiala, India

Harpreet Kaur
Department of Computer Engineering
UCOE , PU Patiala, India

ABSTRACT: Producing software is a very complex process that takes a considerable time to evolve. Poorly designed software systems are difficult to understand and maintain. Software maintenance can take up to 50% of the overall development costs of producing software. One of the main attributes to these high costs is poorly designed code, which makes it difficult for developers to understand the system even before considering implementing new code. In the context of software engineering process, Software Refactoring has a direct influence on reducing the cost of software maintenance through changing the internal structure of the code, without changing it external behavior . Refactoring is a technique for restructuring an existing body of code, Its heart is a series of small behavior preserving transformations. Each transformation called a 'refactoring' does little, but a sequence of transformations can produce a significant restructuring. Since each refactoring is small, it's less likely to go wrong. In this paper we disscussed some refactoring methods. we reviewed various refactoring tools,and studied their features.

Keywords-Introduction ,Activities ,Methods, Tools.

1. Introduction

Software evolves and its complexity is increased over the time. When project fails for reasons that are primarily technical, the main reason is often uncontrolled complexity [4]. Managing complexity is the most important technical topic in software development . Some characteristics of design quality are also characteristics of a good program: reliability, performance, and so on [4]. Internal characteristics of the design are, for instance, minimal complexity, ease of maintenance, loose coupling, extensibility, reusability,portability [4].. Agile development is performed iteratively and in the every iteration it is necessary to reconsider the program structure being exposed to some change. Refactoring means improving the design of existing code. In the context of test driven development, code is refactored to removed duplication and express intent as soon as it is written and passes the tests. Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Refactoring is the process of changing the design of existing software modifying the structure of its code without affecting its external behavior. Refactoring: Improving the Design of Existing Code. While refactoring can be applied to any programming language, the majority of refactoring current tools have been developed for the Java language. One approach to refactoring is to improve the structure of source code at one point and then extend the same changes systematically to all applicable references throughout the program. The result is to make the code more efficient, scalable, maintainable or reusable, without actually changing any functions of the program itself. Refactoring as proposed by Fowler not only covers the mechanics of restructuring, but also addresses the following issues:

2. Why Do We Need Refactoring?

Refactoring is used to improve code quality, reliability, and maintainability throughout the software lifecycle. Code design and code quality are enhanced with refactoring. Refactoring also increases developer productivity and increases code reuse Continuous design allows one to add more flexibility into the design, by adding to an initially simple design as the need arises, instead of having a big upfront design. Thus the design will evolve as the code grows. There is a shift from building software towards growing it. The process of refactoring can be used to contribute to these evolving states of the code[3]. Refactoring improves the design of software Without refactoring the design of the program will decay. Poorly designed code usually takes more to do the same things, often because the does the same thing in different places Refactoring makes the code easier to understand In most software development environments, somebody else will eventually have to read the code so it becomes easy for others to comprehend. Refactoring helps to find bugs .It helps in finding the Bugs present in the program. Refactoring helps to program fast[6]

3. Refactoring Activities

The refactoring process consists of a number of different activities, each of which can be automated to a certain extent:
(1) Identify where the software should be refactored;
(2) Determine which refactorings should be applied to the identified places;
(3) Guarantee that the applied refactoring preserves behavior;
(4) Apply the refactoring;

© 2012, IJARCSSE All Rights Reserved

(5) Assess the effect of refactoring on software quality characteristics;
(6) Maintain consistency between refactored program code and other software [9]

4. Refactoring Techniques

A. Composing methods

1) Extract Method: Extraction of a piece of code into a separate method. You have a code fragment that appears in multiple places within the code .You have a code fragment that can be grouped together. Turn this fragment into a method whose name explains the purpose of this method.[2]

```
void printOwing() {
    printBanner();

  //print details
    System.out.println ("name:          " + _name);
    System.out.println ("amount        " + getOutstanding());
```

```
void printOwing() {printBanner();
printDetails(getOutstanding());
}

void printDetails (double outstanding) {
    System.out.println ("name:          " + _name);
    System.out.println ("amount        " + outstanding);
}
```

2)Replace Temp with Query: Replacement of references to the temporary variable with the method calls. It facilitates method extraction. Extract the expression into a method. Replace all references to the temp with the expression. The new method can then be used in other methods.[2]

```
double basePrice = _quantity * _itemPrice;
if (basePrice > 1000)
        returnbasePrice* 0.95;
else
        returnbasePrice* 0.98;
```

```
if (basePrice() > 1000)
        returnbasePrice() * 0.95;
  else
        returnbasePrice() * 0.98;
...
double basePrice() {
    return_quantity*_itemPrice;
}
```

3)Inline Method: A method's body is just as clear as its name. Put the method'sbody into the body of its callers and remove the method.[4]

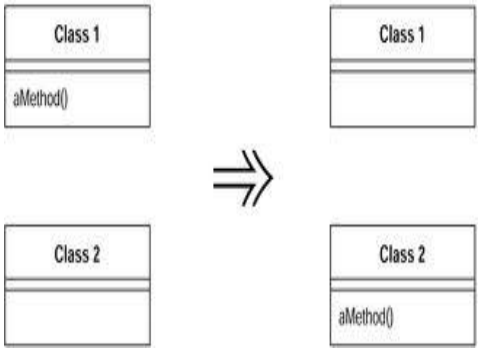
```
int getRating() {
  return(moreThanFiveLateDeliveri
 s()) ? 2 : 1;
}
 boolean moreThanFiveLateDeliveries() {
  return_numberOfLateDeliveries >
   5;
```

© 2012, IJARCSSE All Rights Reserved

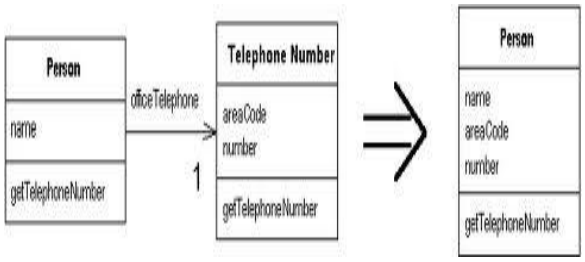
```
}
                    int getRating() {
            return(_numberOfLateDeliveries > 5) ? 2 : 1;
}
```

B. Moving features between objects

1)Move Method: Moving method to another class when a class has too much behaviour or when classes collaborate a lot and are too highly coupled. Moving method is the bread and butter of refactoring.[5]

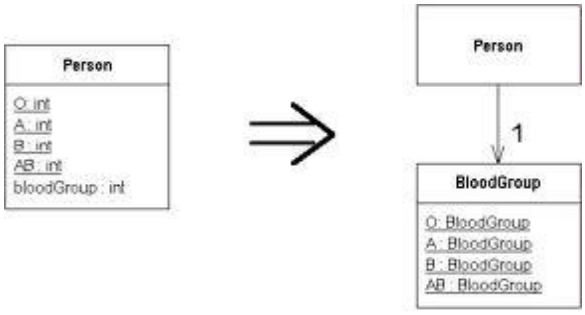


2)Inline Class: A class isn't doing very much. Move all its features into another class and delete it.



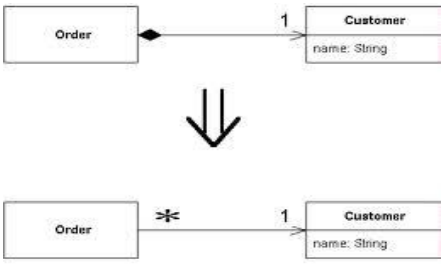
C.Organizing data

1)Replace Type Code with Class: Replacement of state constants with type safe Enum. Replace the number with a new class. [4]



2)Change Value to Reference: When a class has many equal instances then is better to replace it with a single object or turn the object into a reference object.[7]

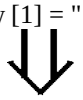
© 2012, IJARCSSE All Rights Reserved



3)Replace Array With Object: You have an array in which certain elements mean different things.Replace the array with an object that has a field for each element.[4]

```
String[] row = new String[3];

row [0] = "Liverpool"
;
row [1] = "15";
```



```
Performance row = new
  Performance();

row.setName("Liverpool");

row.setWins("15");
```

5. Refactoring tools

A. INTELLIJ IDEA

IntelliJ IDEA was the first Java™ IDE to extensively implement many of the refactorings worked out and recommended in the ground-breaking book Refactoring: Improving the Design of Existing Code *by* Martin Fowler et al. Since then, code refactoring has been a major focus of development, with every release to date adding new and/or expanded refactorings. This page details some of the more significant Java refactoring features of IntelliJ IDEA (you will actually find many more in the product, plus code refactorings for JSP, XML, CSS and HTML, and JavaScript.[4]

1)Extract Method: Select a block of code and invoke Extract Method to turn it into a method. It will automatically analyze the code to detect which variables of the original method must be passed to the extracted method as parameters, which variables being modified by this code fragment are used afterwards. In the displayed dialog you will be able to set the name for the new method, as well as re-order and possibly specify better names for its parameters.[2]
2)Extract Method Object: When you are extracting methods with multiple return values, IntelliJ IDEA automatically offers you to invoke the Extract Method Object refactoring, instead of giving up and resorting to manual refactoring. This new refactoring moves method into an ad-hoc created new class, converting all the local variables to its fields, so letting you further decompose the method into other methods on the same object.[2]
3)Inline Superclass: Inline Superclass is another extension to IntelliJ IDEA Inline refactorings family. When class A extends class B, you can use the Inline Superclass refactoring to push members from class B down to A and remove the class B. All usages of class B and its members are then replaced with class A and its members[2]
4)Replace Method Code Duplicate : When this refactoring is called on a method, IntelliJ IDEA will locate all places in the current file where the selected method code is fully repeated and turn all such code blocks into the corresponding method calls[2].
5)Move Inner Class to Upper Level: Use this refactoring to move an inner class to the same level with its enclosing class. In case the inner class uses members of the enclosing class, there is a possibility to pass a reference to the enclosing class as an argument to the moved class' constructors. It is especially useful to apply this refactoring to large inner classes so that the code becomes more readable.[2]
6)Introduce Constant: Use the refactoring Introduce Field as a «shortcut» to rapidly create static final fields. As a particular case, this refactoring covers the Replace Magic Number With Constant refactoring
7)Inline Method : This refactoring is opposite to Extract Method. It will inline all invocations of a certain method and remove the method declaration. When invoked on a particular method's usage, there is an option to inline only this usage and keep the method. This refactoring is often useful as a part of complex refactorings.[2]

© 2012, IJARCSSE All Rights Reserved

8)Replace Temp With Query: Select some local variable in the editor and invoke Replace Temp With Query. This will extract the variable's initializer expression into a method, and replace all references to the variable with calls to the extracted method. The declaration of the variable will be removed[2].

B. Eclipse

Eclipse is the multi-language software development enviounment.it is written mostly in java . It can be used to develop applications in Java and, by means of various plug-ins other programming language like including R,ruby Ada ,C,C++.COBOL,FORTAN,php etc

Eclipse requires Java to run, so if you don't already have Java installed on your machine, first install a Java 6 SDK. Please note: The DIL/NetPC DNP/2486 MAX-Linux comes with a Java 6 runtime environment. Your PC as a development system needs the same Java version. You can download Java SDKs from http://java.sun.com. Look for the Java 6 J2SE SDK (Software Development Kit)

1)Simple Name Change: In eclipse We can build, run, test without committing the rename change to the StarTeam repository.[3]

2)Extract Method: A useful refactoring is to mark code and create a method from the selected code. For this mark the coding of the "for" loop, right click and select Refactoring → Extract Method.

3)Rename Method: Renaming something can be a beast of a task when you have a large program. Eclipse makes this absurdly easy to do. Right-click on almost anything and choose **Refactor > Rename** [7].

6. Comparision Between Intellij idea And Eclipse

TOOL	System Requirement	Methods For Refactoring	Language Supports
Intellij Idea	MicrosoftWindows,8/7/Vista/XP 1GBRAMminimum, 300 MB hard disk space +at least 1 G for caches	Extract Method Extract Method Object Inline Superclass Replace Method Code Duplicate Inline Method Replace Temp With Query Introduce Constant	Java JSP, XML, CSS and HTML, and JavaScript.
Eclipse	Eclipse requires Java to run system needs the same Java version 315 MB available hard disk space MicrosoftWindows,8/7/Vista/XP	Extract Method Rename Simple Name Change Extract Method Object	R,ruby Ada, Java, PHP C,C++.COBOL,FORTAN,php

7. Conclusion

Software Refactoring is an important area of research that promise substantial benefits to software maintenance. Refactoring is a process that improves the quality and allows developers to repair code that is becoming hard to maintain, without throwing away the existing source code and starting again. We can return with a well structured and well designed code after proper application of refactoring techniques. Refactoring naturally fits in an agile software development process. It forms even one ofthe cornerstones of the extreme Programming process; together with unit testing. The research in software engineering continues to be very active .[9]

REFERENCES

[1] Www.Google.Com
[2] D.M. Coleman, D. Ash, B. Lowther, and P.W. Oman. *Using metrics to evaluate software system* IEEE Computer, 27(8):44–49,August 1994.
[3] William G. Griswold. Program *Restructuring as an Aid to Software Maintenance*. PhD thesis,University of Washington, August 1991.
[4] E. Casais. *Automatic reorganization of object-oriented hierarchies:* a case study. Object Oriented Systems, 1:95–115, 1994.
[5] Borland. Borland Together ControlCenter, January 12 2004.
[6] Borland. Borland JBuilder, Janurary 12 2004.
[7] Don Roberts. *Practical Analysis for Refactoring.* PhD thesis, University of Illinois
[8] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification.* Addison Wesley, 1996.
[9] AspectJ homepage: http://www.aspectj.org

© 2012, IJARCSSE All Rights Reserved