Reusability of a software using Feature Oriented Domain Analysis Method

S. Malathi and Dr. P. Sudhakar

Department of Computer Science and Engg.,
Annamalai University.

Received 28 January 2017; Accepted 22 April 2017; Available online 1 May 2017

Address For Correspondence:
S. Malathi, Department of Computer Science and Engg.,
E-mail: ekpmmanikandan@gmail.com

ABSTRACT
Feature Oriented Domain Analysis [FODA] is a Rule Engine and also called FODA utility tool used for implementing Refactoring the code and then reuse. To illustrate the application of domain analysis to a representative class of application software systems. Refactoring is a technique to keep the code cleaner, simpler, extendable, reusable and maintainable to achieve the quality of the code. It also transforms a program to improve its internal structure, design, understand ability or other features without affecting its external behaviour. Successful software reuse requires the Systematic discovery and exploitation of commonality across related software systems. This paper reveals the design of FODA and its implementations establish methods for performing a Feature Oriented Domain Analysis and describe the products of the FODA process. Using FODA rule set in further Refactoring and Reusability of any application software code. While this is a good start, I am going to implement the Simple Text Editor Application in java swing technology using Eclipse IDE with the help of FODA Rule Engine.

KEYWORDS: FODA, Domain, Feature, Software Refactoring, Reusability, Eclipse.

INTRODUCTION

For Pietu pohjalainen presented FODA methodology implementation achieving successful software for Refactoring the code and Reusability. Refactoring and Reuse of software is one of the most promising solutions so it is called “software crisis”. They are used to construct different feasible configurations of refactoring, reusable architectures. The use of the “features“ is introduced by the fact that customers and engineers speak of product characteristics in terms of features the product has to be delivered. Features of a domain characterized each variant product in the domain and the code that implements the characterizing features should be package, manage, restructuring and reusing as software modules. Domain is defined in terms of a set of current applications or further another application which share a set of capabilities of common data. Domain Analysis is defined as the process of identifying, collecting, organizing and representing the relevant information. That is to meet a customer and collect the correct requirements of customer need. If any changes of customer need, the developer is to be modified. The term Feature is a prominent or distinctive user-visible aspect, or characteristics information of software or software systems.

Related Work:
Feature Oriented Domain Analysis was originally presented by kang et al (1990). Its later proponents include c zranecki and Eisenecker present a language for defining FODA models and algebra for transforming as counting the various needs, such as counting the number of model conforming configurations and expanding the model language expressions [1]. Feature Oriented Domain Analysis Expression was presented by pietu pohjalainen present a FODA is used to express requirements on different possible configurations of a concept.
To build a bridge between the configurations model and actual implementation present a regular expressions related language that can be used to define FODA models [2]. B.Ramalkshmi and Gayathri Devi (2015) presented Refactoring is done to improve the quality of a software system, structure. Which tends to degrade as the system evolves. While manually determining useful Refactoring is a challenging. Searched-based techniques can automatically discover useful Refactoring. Refactoring approach uses the concept of pareto optimality which naturally applies to search-based Refactoring. Before Refactoring is done, the test case should be generated. A formal written test-case is characterized by a known input and by an expected output, which is worked out before the test is executed [3]. Woo-chang shin and Jung kyu Rho (2014) presented to enhance productivity, active tool support is necessary. Especially, a Refactoring tool that can alter the internal structure of software to more easily understandable and modifiable structure hugely affects software maintenance productivity. This work proposes a code model to support software maintenance tool developers to easily access and handle software source code. Also, this work shows the implementation method of software Refactoring operation using code model [4]. T.Pandiyavathi presented the aim in finding the Restructuring candidates which have to be rearranged, thereby applying changes to the code on those parts using the tool helps in ordered arrangement of the source code. Applying modularization to the Restructuring candidates will lead to decrease in the human effort as well as tool effort in Restructuring. Unwanted evolution of new errors will be eliminated [5].

Reusability Using Refactoring:

In today’s intense world of software engineering one of the most pressing challenges is how to make software easier and how to maintain while keeping cost is low. Bouliba Ben Ammar and Mohamed Tahar Bhiri (2014) presented Refactoring is the process of changing a software system in such a way that it does not change the external behaviour of the system and to improves the internal quality of the structure. It does not changing functionality, it just says that it’s a different activity try to change it functionality. In general, refactoring does not modify the overall structure of a program a. It tends to focus on the design details of individual modules and an local data and functions defined within a modules. A design that produces the same function as the original program but with a result is high quality.

a. Need for refactoring:

1. Refactoring improves the design of a software code.
2. Refactoring makes software easy to understand.
3. Refactoring helps find the errors.
4. Refactoring helps to program run faster.
5. Software Reusability

A. Ravi and K. Nirmala (2015) presented Software Reusability is generally considered a way to solve the software development crisis. When we solve a problem, we try to apply the solution to similar problems because that makes our task simple and easy. This software reusability can improve software productivity. Software Reuse has become a topic of much interest in the software community due to its potential benefits, which include increased product quality and reduced cost and schedule.

In Vikshant Khanna and Parul Mohindru (2014) presented Software Reuse is the process of implementing or updating software systems using existing software components. A good software reuse is the process facilitates the increase of productivity quality, reliability and decrease of costs and implementation time. An initial investment is required to start a software reuse process, but that investment pays for itself in a few reuses. In short, the development of reuse process and repository produces a base of knowledge that improves in quality after every reuse, minimizing the amount of development work required for future projects and ultimately reducing the risk of new projects that are based on Repository knowledge. By considering all these things, we propose a methodology to reuse the software in a much better way. The aim of the paper can be achieved by implementing software refactoring, reusability using Feature Oriented Domain Analysis. Reduced the Lines of code, reduced the memory space.

Rajender Nath et al (2014) proposed to serves as a roadmap to software development industry to help in reducing the size of code and hence the efforts and development time which is indirectly responsible for efficient maintenance method for software reusability. Automation of this approach in the industries can cause to control the software crisis which is a big demand of each IT industry. It is observer that the work purposed clustering method is working well in the case of live products. if we want to develop another project which will have the similar properties, then the development time and effort will be reduced, so the maintenance will be effective which will cause the less cost of development.

Surbhi and Maggo proposed the approach provides support for reusability evaluation at functional level rather than at structural level. The automation support for this approach is provided in the form of tool named
Introduction Of Foda [Feature Oriented Domain Analysis]:

This section gives a fast course to FODA method and present how refactoring the code can be used to check whether certain applications and conforms to the specified model. Fig. 4 shows an example of a FODA model, as presented by Kyo c.Kang et al. As a FODA model, Specifies possible configurations for a car, which has mandatory (It means compulsory) parts of a car body, transmission can exclusively be automatic or manual (Alternative) while the engine can be an electric engine or gasoline driven or both (Optional). Optionally the car in question can pull a trailer.

Fig. 1: Example FODA model

When implementing a system that has some of its requirements expressed is as FODA models, One of the problems is how to check for conformity: when Given a list of the parts of a car, The system need to decide whether this list is a car defined by the model of a car or not in the model, The system might need to decide whether this list is a car defined by the model, our approach is to translate the model in to a Refactoring the code and then Reusability of the software code.
Proposed Work:
In our work to develop FODA Rule set tool or FODA Rule Engine or Simply FODA tool. In previous literature using Refactoring tool cannot find errors exactly which part of the source code. And unable to support separable of modules. In our proposed work FODA tool to separate the modules. To find errors exactly on which part of source code. Because, FODA tool support the big modules is divided in to sub-modules. Using this tool refactoring the source code and then reuse the software.

FODA Tools Supports Software Development

Implementation Of Foda:

<table>
<thead>
<tr>
<th>Technologies used</th>
<th>XML (Xtensible Markup Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLs</td>
<td>XSD (XML Schema Definition)</td>
</tr>
<tr>
<td>XSDs</td>
<td>Java – for XML parsing and validating</td>
</tr>
<tr>
<td>XML parsers and validators</td>
<td>Apache Ant – for build</td>
</tr>
<tr>
<td></td>
<td>Eclipse IDE</td>
</tr>
</tbody>
</table>

Components of FODA tool

| XMLs                        | Feature list xml - Master repository of the features |
|                            | Foda rule set xml - Repository of relationship between all the features defined in the master feature list |
|                            | Project feature list xml - Contains the list of features along with its inclusion status |

| XSDs                        | Feature list xsd |
|                            | Foda rule set xsd |
|                            | Project feature list xsd |

XML parsers and validators

<table>
<thead>
<tr>
<th>Parsers [Parses xml files]</th>
<th>[This component is the core of our foda rule engine project]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional Validators [Validates the definition of xml files]</td>
<td>Transformers [Generates new files, if required]</td>
</tr>
<tr>
<td>Error Logger - Write the error logs to an file</td>
<td></td>
</tr>
</tbody>
</table>

A. Build and Execution:
The java source is compiled in to an jar file via Apache Ant. Ant build file is defined with targets of java source compilation and jar file creation. The jar file (FODA tool) is copied to destination folder (dest). The schema definition files are copied to input folder. Sample XML files are defined for all the xml components – feature list, foda rule set and project feature list. These files are also copied to input folder. The FODA tool is executed with the input folder (which contains the XSDs and XMLs) as argument in the command line tool. The implementation of a foda tool as shown in Fig 2 and Fig 3.
Advantages of FODA:
1. Separation of concern.
2. Ease of maintenance.
3. Ease of debugging.
4. Easy addition of new feature.
5. Less software breakage.
6. Highly customisable.

Conclusion:
In general, the FODA provides a detailed overview of the problem solved by software in a given domain. Tool development is used to support the software development domain analysis methods. The complexity of even a well understood application, such as that of window managers, establishes the need for tools to handle the volume and variety of information a domain analysis can generate. The FODA method must also extend to provide automatic support for the application to support the user decisions. Applying the FODA method in new application will support the separation of the method and give validation to the approach. In our paper FODA Rule set is defined and developed successfully. This Rule set is used for further Research work for example software refactoring and reusability of the software code.

REFERENCES