A Tool for Detecting Detects on Class Implementation in Object Oriented Program on the Basis of the Law of Demeter: Focusing on Dependency between Packages

RYOTA CHIBA, HIROAKI HASHIURA and SEIICHI KOMIYA
Graduate School of Engineering, Shibaura Institute of Technology
Komiya Lab, 13F, Kenkyu-to, 3-7-5 Toyosu, Koto-ku, Tokyo
JAPAN
{chiba, hashiura}@komiya.ise.shibaura-it.ac.jp, skomiya@shibaura-it.ac.jp
http://www.komiya.ise.shibaura-it.ac.jp/

Abstract: - Design methods, such as OOSE and OMT, are used in object-oriented development. However a common fault of these design methods is that design of classes becomes a problem, therefore we have suggested that lowering the coupling by paying attention to module partition have proposed the method of supporting class design. And extended that the Law of Demeter which is one of the laws to lower coupling, can also be applied to JAVA. In addition, about JAVA, when reusing the modules created by other persons, the application of use is higher with package units than the class units, so made it a rule to apply between packages. We developed the tool, which detected the points that violated this automatically as plug in of Eclipse and inspected it about the effectiveness. It experimented with a made tool , As a result, It was able to detect that violate the Law of Demeter and confirmed that the concrete can be detected that location.

Key-Words: - Class Design, Object Oriented Software Development, Law of Demeter, Eclipse

1 Introduction
In today's information-oriented society, the software development becomes large scale and becomes complicated. Therefore much knowledge becomes necessary for software development. Especially in the object-orientation, often used in recent years, although there are typical design methods such as OOSE (Object-Oriented Software Engineering) method and OMT (Object Modeling Technique) method as a common fault in these designs method, there are problems that the design of the class is difficult.

In addition, even if the usage of modeling languages such as object oriented programming language or UML are learned, advantages such as solidity, the expansibility that object-oriented technology originally have, are not obtained if used without thinking about the background. Therefore, the industries are demanding to the university for training of the talented people who have the knowledge and skill about software development. PBL (Project-Based Learning) is adopted as the exercise lessons of a software development and practical software design and development technology by object-orientation are made to study in this university.

It is necessary to learn methods (software design methods) to design good software to bring up talented persons with deep knowledge about the software development and high skills. Therefore it is necessary to develop tools supporting the acquisition of the software design methods. Therefore we use coupling [1] for an index to measure good points of the software design plan in this study. Coupling is strength of the cooperation between the modules, and it is said to be a weak moderate design coupling between the modules.

In order to lower coupling between modules pay attention to interdependent dependency relation between modules, and consider decreasing coupling by adding restrictions to the destination of messages. As for the technique of lowering coupling, it is based on the Law of Demeter. Interdependent dependency relations between modules are detected and develop the tools, which point out the parts, which need to correct the destination of messages.

"By this, I detect the point where the revision is necessary and state the place clearly. And
Locations of parts to be corrected are clarified and the information useful for future programs changes (specification changes included) is offered.

Finally shown below is the structure of this report. Chapter 2 shows the positioning of this study. Chapter 3 shows the Law of Demeter and the things, which were expanded in this study. Chapter 4 explains the summary of the developed tool and the implementation method, Chapter 5 explains whether the detection of points violating the Law of Demeter with the tool developed are possible or not. Chapter 6 gives the summary of this report.

2 The purpose of this paper

Myers[1] says that it is necessary to perform module partition adequately to make good software. One of the techniques to perform module partition is to think about module independence, and to divide. Myers[1] proposed a concept called module cohesion (module strength) and module coupling as a standard to measure program module independence. And points out that it is necessary to be loosely coupled module coupling in high degree module cohesion to raise module independence. Thus, by creating software accordingly, and as per the quality of the software of ISO/IEC 9126-1, it becomes possible to raise maintainability of software. In order to evaluate these modules independence quantitatively, Shyam et al. [2] have proposed defining and measuring various kinds of metrics'. However, in the (system) implementation work, it is necessary to revise (system) implementation so that a value of metrics measured by a tool is settled in appropriate numerical value, but whether the numerical values became appropriate and the specifications of design or procedure were not shown, and the effective means and method of that purpose were not shown also.

Therefore there were many places for the experience and the perception of the developers to make appropriate numerical values. Therefore the structure, which pointed out what kind of dependency relation occurred at which place in the source code by analyzing the source code concretely thought that it was necessary to solve this problem. When thinking about the approach with tool like this, the tool revise implementation flaw automatically is thought about, the tool points out implementation flaw concretely and does not perform automatic revision of the source code.

As for this, minimum dependence is needed for the programs, but the dependence needed and dependence un-necessary, should be judged by the intentions of the designers and the person of (system) implementation. In addition, our aim is the support of the acquisition of the software design method, because it is thought that it revises design flaw automatically, and it will not support the acquisition of the software design method.

3 The Law of Demeter

3.1 What is the Law of Demeter?

The Law of Demeter was proposed by Lieberherr [3][4] of Northeastern University in 1987 as a rule for producing the design proposal which may be set for object-oriented software. "The Law of Demeter, as shown in Fig. 1 is a rule which “messages can be sent from a standard object only to objects directly linked with the standard object, and cannot be sent from the standard object to other objects”. Specifically, the proposed design is not good idea to send a message similar to the dotted line in Figure 1."

Lieberherr et al. [3] called the good objects, which may directly communicate with standard objects as “friend”, and the objects called as friend are defined for C++ program.

![Fig. 1 The Law of Demeter](image)

This was shown by Larman [5] also as, "Low Coupling Pattern". That is it can be put in another way as the Law of Demeter being structured, which clarifies a part without the necessity of having dependency relation (being coupled).

3.2 Extended Demeter Law for Java language
The independent nature of the object becomes higher when the program is implemented according to this law, and the Law of Demeter is set in this way so that coupling defined by Myers falls down.

The concept called "friend" was defined in 1988 assuming the C++ of the object-oriented language [3] (that time the language JAVA was not there). Therefore friend is defined as in Fig. 2 in this study to be able to apply even JAVA.

![Fig. 2 The Law of Demeter for Java language](image)

Message passing of interdependent is allowed only, between the objects, which fulfill one of the above-mentioned conditions (interdependent dependency relation between classes is not restricted) was done.

Especially adding the definition (v) to the Law of Demeter, embedded class and data type used by Java, which are judged as part of OS, and not to regard them as dependency relation.

If the embedded class and data type of the JAVA are not made friend, when outputting in the standard output the methods etc. to perform string conversion even when called for make violations the Law of Demeter. Also when the time of outputting to standard output, the method, which performs string conversion, etc. are called, it is for breaking the Law of Demeter.

### 3.3 Extends The scope of the Law of Demeter

The Law of Demeter pays attention to interdependent dependency relation between classes, and there is an aim to lower coupling between classes by limiting the address of the message.

However, when the Law of Demeter is applied in the smallest units called the classes, the scale of the software development grows big, and the number (coupling numbers) of objects having a friend relation grows big for each object drastically.

On this account even objects in relations of friend are difficult to judge as the objects, which may cancel message passing or whether message passing is a really necessary object.

Therefore, in this study, we suggested the package, which compiled plural classes for a unit and not a class.

The reason is because there are many cases to reuse by package units than the class units when it reuses the module, which another person made in the JAVA.

When you reuse a package, often used without knowing the existence of all classes used in the package. To conduct internal inspections of packages created by others to point out, it means that the rewrite inside the package to be reused. (To ensure that this violates the spirit of hiding information.)

This is shown in Fig. 3. This way the communication over the classes in the same package is granted.

![Fig. 3 The Law of Demeter applied between packages](image)

### 4 Class design support by tool

The tool created by this research detects the place where the Law of Demeter violates by analyzing the source code. In this section, the necessity of the tool and its outlines are explained.

#### 4.1 The necessity to develop a tool

The measurement method of coupling placed between the packages, which are suggested, is based on the definition of 5 friends, defined in Fig. 2, and it is to investigate the forwarding address of the message in the source code.

For example, Object A can call any routine of Object B when Object A is made the instance of Object B. When the routine of the object, which Object B offers is called from Object B, just by investigating the source code of Object A, to which class the message was sent is unknown.

In other words you must investigate the contents of multiple classes to investigate one class and can be
counted manually if the scale of the source program is small, but it becomes difficult to examine the forwarding address of the message when the program becomes big.

In addition, you must distinguish it with complete qualifier when there are classes of the same name.

Therefore the addresses of all message passing defined in the package is detected with a tool automatically, and based on the extraction result it is judged whether to identify it for every package and every type, and it becomes necessary to check dependency relation between packages automatically.

In addition, not everybody can find the dependency relation, but for detection, identification the knowledge about the module independence is necessary.

Judgment is difficult for programming beginners we are going to support. Therefore, if the teacher side will perform the dependency detection, identification, it is impossible by manual labor because the number of the programs is equal to student group number.

4.2 The outlines of a tool to develop


IDE integrates the necessary editor, compiler, and debugger etc. required for programming tool with a unified interface.

Again Eclipse has the track record of being used for development by Java in many companies.

From such a background, we decided to perform (system) implementation of the tool as a plug in of Eclipse.

By the plug in style of Eclipse, Beginners are able to get the advice by this tool normally when implementation is done.

This tool is offered in form such as in Fig. 4.

When the directory of the source code of the tool is specified and the search package is specified, the analysis for the search object is done. The class violating the Law of Demeter according to the definition of friend as defined in chapter 3.2 and about its related source code detection of disputed points is done.

4.3 The functions of the tool to develop

This tool is a function to detect the classes, which violates the Law of Demeter by investigating the classes in the package which is specified.

When sending a message to the class except the definition of friend as defined in chapter 3.2 directly, the name of the class in the package and the line numbers of the related place of the source code in that class and the class name of the message passing are displayed.

This detection result detects the message passing violating the Law of Demeter to lower coupling between the modules and can make possible the revision of the program.

In addition, when the number of message passing destinations between the packages is counted, the revision of the coupling between packages can be done on the basis of our definitions.

5 An experiment, experimental results, and the review of them

In this section, by preparing the points of violation of the Law of Demeter as defined in chapter 3.2 and test whether violating points can be detected or not and confirm what is detected.

5.1 The purpose of an experiment

In this section, by preparing the points of violation of the Law of Demeter as defined in chapter 3.2 and test whether violating points can be detected or not and confirm what is detected.

5.2 Experimental methodology

As an example of the source code, arrange two points in a two-dimensional coordinate system and think about a problem to link two points in a line.
Make source codes shown in class diagram such as Fig. 5 and perform the analysis with the tool, which was developed.

The outline of source codes is as follows.

A LoD class acquires the coordinates of Point from a LineWrapper class. Therefore, when calling the x-coordinate of the 1st point from a LoD class, and is good if made line.

getP1X(). LineWrapper class when calculate the value of the x-coordinate of the 1st point, it becomes line.getP1().getX(), but since the Point class and LineWrapper classes are of the same packages, it can be considered that they are friend. Therefore, if the LineWrapper class is calling the coordinates of the Point class, it will not be against the Law of Demeter. However, AntiLoD class acquires the coordinates of Point from a Line class.

When AntiLoD class calls the x-coordinate of the 1st point it is necessary to make line.

getP1.getX(). Since Point and the AntiLoD class are not of the same package, this calling breaks the Law of Demeter.

5.3 Experimental results and the review of them

The results of the tool are shown in Table 1.

<table>
<thead>
<tr>
<th>Line number</th>
<th>message passing place</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>model.Point</td>
</tr>
<tr>
<td>10</td>
<td>model.Point</td>
</tr>
<tr>
<td>11</td>
<td>model.Point</td>
</tr>
<tr>
<td>12</td>
<td>model.Point</td>
</tr>
</tbody>
</table>

Since the LineWrapper class was not against the Law of Demeter, it was not detected.

It is possible to detect which portion of the source code of the class in violation of the Law of Demeter has sent the message which is breaking as mentioned above in this tool.

From this, in order to make modular coupling low, it could be shown that it is a tool, which can detect the message passing place in violation of the Law of Demeter and to which correction of a program can be urged.

5.4 Experimental results and the consideration of them

As a tool for investigating various kinds of metrics over IDE (Integrated Development Environment) which Shyam et al. [2] proposed, there are Metrics plugin for Eclipse [8] and Eclipse Metrics Plugin [9], etc.

By using these tools, it becomes possible to investigate metrics during development in real time.

Therefore at the time of development when judging module independence, useful information can be obtained.

Eclipse Metrics Plugin using and analyzing the source code in Fig.5, Table 2 results obtained.

<table>
<thead>
<tr>
<th>Ce</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>lod.AntiLod</td>
</tr>
<tr>
<td>3</td>
<td>model.LineWrapper</td>
</tr>
<tr>
<td>2</td>
<td>model.Line</td>
</tr>
<tr>
<td>2</td>
<td>model.Point</td>
</tr>
</tbody>
</table>

However, only by displaying the result of the number of this dependence, which dependency relation is appropriate and which is not, cannot be understood by reading the source code.

In addition, depending on a package, the point where the number of the dependence is enormous, it is troublesome to look for the part manually.

The tool we developed tools, Can be shown that the corresponding section of the code that violates the
Law of Demeter. And you know what that is like sending a message to that class anywhere.
We think that it is assistance to sort needed dependency relations and dependency relations not needed by the intention of the designer and the person of implementation in this study.

6 Conclusion and future work
When we thought about software design, it was necessary to perform module partition properly, and we paid our attention about the module independence, which was one of the point.
Because the Law of Demeter was a thing devised in the times of the C++, we expanded the definition of friend so that an application was enabled for the Java language.
Furthermore, we suggested a method to control complexity of the detection by assuming an application unit of the Law of Demeter a package and not a class, and to facilitate an application.
We performed implementation as plug-in feature of Eclipse to discover dependency relation between the packages based on this definition.
As a result, it became easy to detect dependency relation between the packages.
By making the list of the forwarding addresses of the message, and by judging from structure of the whole program what kind of position is there for revision points was made clear, it was possible due to the intention of selecting designers and implementers dependency.
As a result, It showed that implementation to help future program changes was enabled.
It is impossible to use unless if it is possible for the package division adequately to some extent on introducing this tool.
Therefore you must teach whether it is necessary to do the package division beforehand when you apply to programming beginners.
However, it is possible to divide naturally the package by referring to information of coupling of each package and by partitioning of the package with this tool.
In addition, in this study we applied the Law of Demeter between the packages, but when the same Law of Demeter is applied too between the classes, the number of related objects becomes bigger, and to distinguish the role of the messages between the objects, cannot grasp message relations unless make a rule to reduce messages, as it is difficult to make it appropriate coupling.
For future prospects, When implementing the plan as designed, they may violate the Law of Demeter.
In that case, because the disputed points of the object are the intention of the designer, if we convert designs plan such as class diagram made into XMI and do implementation according to the design plan. Cooper et al. [10], in order for implementation of a system to be as per specifications, think that it must follow the design documents faithfully when deviated from the design in the middle of implementation, the function to give warning becomes an interesting problem.

References: