Kevin Nguyen, Jacqueline Wong - Team 4
CS 6301 - Software Comprehension and Analysis

Project Proposal - Common Coupling using Class-Level and Method-Level Granularity

Topic Description

We have chosen for our project to create a tool to detect common coupling. As coupling is a tool commonly used to detect impact analysis, our approach on how we create the tool will keep this in mind. First, common coupling is described as the occurrence "when two modules share the same global data." By identifying common coupling, we also identify modules which may be impacted should that global data change. We also identify possible modules which may be impacted by other modules.

For example, in the following image, we show three classes. The first class GlobalClass has global variable x. The second class Getter is a class with method divideByX() which returns 10 divided by GlobalClass.x. The final class Main simply prints out the result of Getter.divideByX().

```
public class GlobalClass
{
    public static int x = 5;
}

public class Getter
{
    public static int divideByX()
    {
        return 10 / GlobalClass.x;
    }
}

public class Main
{
    public static void main(String [] args)
    {
        System.out.println(Getter.divideByX());
    }
}
```

Now we introduce the class Setter. This class sets GlobalClass.x to 0. If we call Setter.setX() and then call Getter.divideByX(), we are now dividing by 0, throwing an error. In this case, the introduction of Setter sharing global variable x impacted the functionality of Getter.

```
public class Setter
{
    public static void setX()
    {
        GlobalClass.x = 0;
    }
}
```

The next step to consider is the language we will be using as global variables may have variations between languages. In this project, we will be analyzing and identifying common coupling only with Java systems. Furthermore, we would like to do this at a class-level and method-level granularity.

If we wanted to identify common coupling between classes, global variables in Java would be class variables instantiated with public and static. Public allows the variable to be called globally in all files while static allows the variable to be called without the need of a class instance. As seen in the above example, usage of the same global variables could also identify classes which impact each other.

Kevin Nguyen, Jacqueline Wong - Team 4
CS 6301 - Software Comprehension and Analysis

Using common coupling at method-level granularity would be limited to within a single class and which methods are related to each other within the file. The way common coupling would be used here is that in this scenario, class instance variables would be considered the global variables to methods. We then can identify methods within a class related to each other more than other methods.

For example, in the following image, we have two methods setX() and returnX(). These two share the instance variable x, and in the eyes of these two methods, x is like a global variable to them. We also establish that setX() and returnX() relate to each other within this class.

```
public class Example {
    private int x;

    public void setX()
    {
        x = 5;
    }

    public int returnX()
    {
        return x;
    }
}
```

By considering common coupling at the class-level and method-level granularity, we can establish relations within a class as well as between classes. Although out of the scope of this particular project, this could also potentially be used in conjunction with other coupling methods in future projects (Ex. Content coupling establishing impact relations between methods across classes + common coupling furthering those impact relations between methods within a class).

Technologies to be Used

- Java + Eclipse AST - Language + library we will use to write the common coupling detection tool.
- Various Java Systems - These will be analyzed using the common coupling detection tool.

Deliverable List and Dates

Date	Deliverable List
4/1	Finalize project outline based on professor recommendations. Begin coding.
4/12	Project Update Presentation Date. Have most or all of the code finished.
4/14	Intermediary Project Report Due. Preliminary results for 1-3 systems.
4/21	Complete final code revisions and finalize presentation-ready results.
4/26-4/28	Final Project Presentation Date.
5/3	Final Project Report Due.