A-Team: Kevin Shiah, Edward Law, Jacky Liao, Risa Newyear-Ramirez

Project Goal

We are going to be analyzing the creation date of the classes by looking at the Github commit history (using Custom-github-analyzer), the degree of coupling of each package (using JDepend), and the code quality of each package (using PMD). We will be analyzing 1) whether classes that are created earlier have more problems with code quality, and 2) whether coupling of packages is associated with code quality (i.e. whether loose coupling of packages is associated with less problems in the code). We will produce an animated real-time visualization of each class for the source repositories “Sponge” and “twoway-View”. The visualization will display classes as dots surrounded by a bigger circle representing the package that the class belong to. Density of the dots is defined by the tightness of coupling (a group of dots clustered together represent that they are tightly coupled). Code quality will be represented by the number of circles around the dots (more circles around a class means that the class has more potential errors. You could imagine that the section of visualization where there are many circles overlapping with each other and messy will illustrate that it is closely coupled and error proned) . Order of travelling through the dots will be determined by the sequence of class creation.

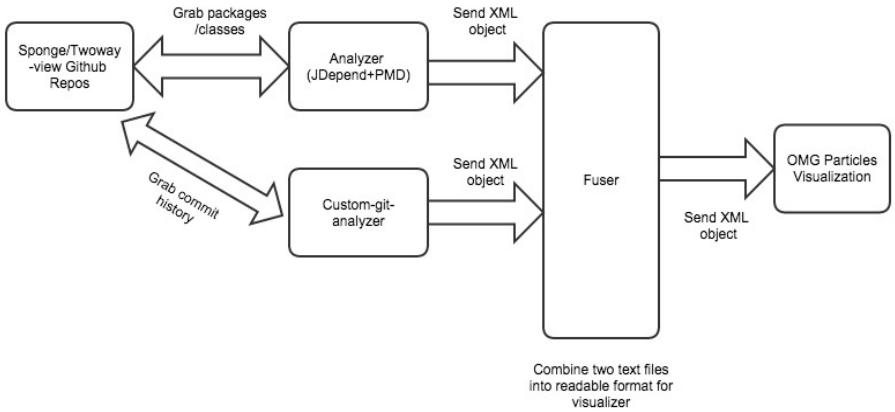
Repository

https://github.com/edwardlaw8/CPSC410

Current Architecture

JDepend and PMD are subclasses of a java class Analyzer. They look at both the Sponge and twoway-view repositories. The Analyzer outputs the number of violations in the code (from PMD) as an integer and the dependencies of each package, including afferent and efferent couplings (from JDepend) as an XML object. In PMD, a violation of code quality could mean unused variables, unused imports, too many static imports, etc.
Custom-github-analyzer uses JQuery to fetch the git commit history from both Sponge and twoway-view and outputs the sequence of class creation to an XML object. The different variables and objects from Custom-github-analyzer, PMD, and JDepend will then be read and compiled by our Fuser, which will make it into a format readable by our visualizer, OMG Particles.

Architecture Corresponding Files



Kevin: He will work on getting the Visualizer to accept the output from the Fuser, and to get the Visualizer to meaningfully display the information we want.
Edward: He will work on getting the Fuser to accept the two analyzers' outputs from Custom-git-analyzer, PMD and JDepend, and combine them into a readable format for the Visualizer. At this time, we are combining the outputs into a single XML object.
Jacky: He will work on grabbing only the number of violations PMD counts in the package files outputted by PMD, and then returning the integer.
Risa: She will work on grabbing only the information related to afferent/efferent coupling and dependencies between each package from the output of JDepend and putting this information into an XML Object.

```
</Package>

<Package name="org.gradle.cli">
    <Stats>
        <TotalClasses>24</TotalClasses>
        <ConcreteClasses>18</ConcreteClasses>
        <AbstractClasses>6</AbstractClasses>
        <Ca>1</Ca>
        <Ce>3</Ce>
        <A>0.25</A>
        <I>0.75</I>
        <D>0</D>
        <V>1</V>
    </Stats>
```

Fig 1. Sample output of JDepend (by running Runner.java)

```
        <create-date>2014-09-08T01:56:37Z</create-date>
    </1>
    <2>
        <class>SpongePluginContainer</class>
        <create-date>2014-09-08T01:56:37Z</create-date>
    </2>
    <3>
        <class>SpongePluginManager</class>
        <create-date>2014-09-08T01:56:37Z</create-date>
    </3>
    <4>
        <class>SpongeCoremod</class>
        <create-date>2014-09-08T02:34:00Z</create-date>
    </4>
    <5>
        <class>SpongeEventManager</class>
        <create-date>2014-09-08T04:14:35Z</create-date>
    </5>
```

Fig 2. Sample output of custom-git-analyzer

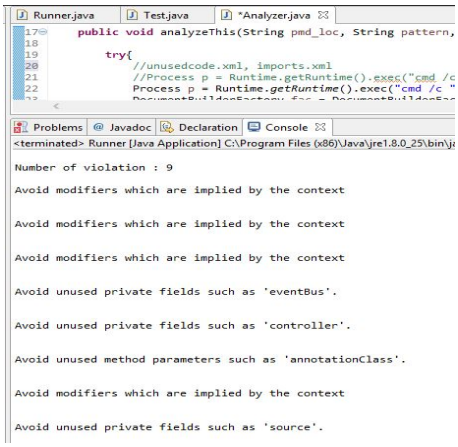


Fig 3. Sample output of PMD (by running Runner.java)

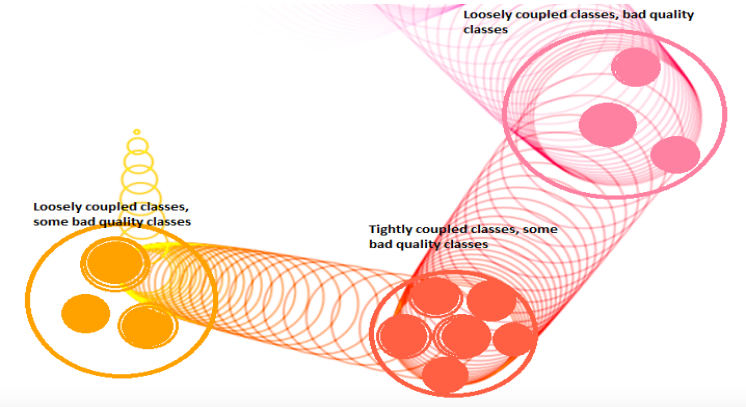


Fig 4. Sample Visualization Output