CSE 403: Project Proposal

Wesley Cox, coxw2, 1336735
Kevin Klein, kevinkle, 1650003

1 Motivation

When introduced to a large and complex project for the first time, it takes developers a significant amount of time to learn its working. In specific, we want this project to aid in pointing out and understanding the implementations of the design patterns in the new project. Developers currently rely on either drawing handwritten graphs with manually gathered information. Simple generated control flow graphs have many details, but no insights in the higher level concepts. Stepping through execution or simply reading through the code over and over again can be just as tedious.

Helping the developer understand code by automating parts of the manual process of reading code is the aim of the project. Particularly, we want to focus on design patterns. Those are concepts the developer is usually familiar with. However their application can be very hard and tedious to understand without the necessary context information.Giving the developer this exact context information can contribute a lot to a better and faster understanding of libraries he wants to use, the reviews he is supposed to perform or the work he aims to do on a code base which has previously been established by fellow developers.

The graphical context information should include some information typically gained from conventional control flow graphs, such as inheritance, method calls and fields, as well as precise information regarding the instantiation of design pattern. In the example of a visitor pattern, this should reveal to him, when which visit or accept method is called on which object and what purpose it fulfills. This can include different scenarios for different dynamic types. We want to reduce the displayed information to a minimum. Another example would be callback functions, which can be used to avoid polling in the Model-Viewer-Controller pattern. If nested, they can be very tricky to understand, as they typically include a lot of inheritance and control flow changes. Hence the purpose of the provided information would be to bridge the gap between the abstractions, the developer is familiar with, with the particular instance of it in this piece of code.

2 Approach

The general approach of our project would be using machine learning to detect patterns based on control flow graphs.

First, our program will analyze the source code to generate some graph or other data structure based on traits that we base design patterns on. We consider using already existing tools for this purpose as 'standard' control graphs might be sufficient for the machine learning algorithms to perform well. If not, we would need to either extend existing projects or, in the worst case, completely develop it on our own. This step is not necessary in the first place, however we expect the machine learning algorithm to perform better on this than on pure source code.

Second, based on these generated structures, our program will then use previous trials to group components into clusters and identify patterns within/between them, i.e. unsupervised learning. If this appears to be unreliable, we may require a human label the modules themselves, then leave it to the program to detect only the patterns and classify them, i.e. supervised learning.

Third, we want to generate a visualization of the design pattern aspect as well as the contextual information. This requires not only the detection of the pattern through machine learning techniques, but also the recognition of the particular instantiation aspects.

This project will be restricted to Java, as it is a heavily typed language many of us are familiar with. Moreover, it may be helpful to further restrict the source code that can be analyzed efficiently by this project, such as possibly stricter control on packages or documentation.

3 Risks and challenges

The most difficult part of this project will probably be obtaining a good dataset. Each trial needs unique source code and the labeled modules/patterns it contains. We personally have very little experience with machine learning, so it is unclear how much quantity and variation our data needs to be effective. Talking about re-engineering in general, projects in the past have found difficulty differentiating between implementation details and important abstractions. This is where the machine learning comes into play; it tries to guess the human classification based on previous classifications.