Feature Envy Factor

A Metric for Automatic Feature Envy Detection

Kwankamol Nongpong
Department of Computer Science
Assumption University
Bangkok, Thailand
Email: kwan@scitech.au.edu

***Abstract*—As a software system evolves, its design get deteriorated and the system becomes difficult to maintain. In order to improve such an internal quality, the system must be restructured without affecting its external behavior. The process involves detecting the design flaws (or code smells) and applying appropriate refactorings that could help remove such flaws. One of the design flaws in many object-oriented systems is placing members in the wrong class. This code smell is called Feature Envy and it is a sign of inappropriate coupling and cohesion. This work proposes a metric to detect Feature Envy code smell that can be removed by relocating the method. Our evaluation shows promising results as the overall system's complexity is reduced after suggested Move Method refactorings are applied.**

Keywords-feature envy; code smells; refactoring; design flaws; software quality; software metric

I. INTRODUCTION

The evolution of a software system is a long and continuous process. Software system usually goes through a series of small and big changes over a period of time. When a software system gets larger, new features are added to the system, its design generally gets worsened. It gets more complicated and difficult to understand. Furthermore, if the design is not regularly revised, the system will become difficult to manage and maintain.

To promote continuous design revision, the process of locating and removing code smells should be transparent and seamless to the software developer. The process includes identifying code smells and fixing them using behavior-preservation transformations or refactoring. Refactoring is usually initiated by the developer. Most software developers only refactor their code when it is really necessary because this process requires in-depth knowledge of that particular module of the software system. While many experienced developers can recognize the pattern and know when to refactor, novice programmers may find this process very challenging.

Fowler et al. [4] describe a set of code smells and how to resolve them with corresponding refactorings. A number of studies [16, 19] also confirm that code smells have a great impact on software maintainability and proper use of refactorings can actually help improve the software qualities of the system.

II. FEATURE ENVY

One of the main design flaws in object-oriented systems is misplacing the class members or making the class responsible for things that should be handled by other classes. In object-oriented systems, classes must be loosely coupled and highly cohesive. The code smell that involves wrong placement of the class members is called Feature Envy.

Feature Envy code smell is a sign of inappropriate coupling between classes. It occurs when a class member is more interested in some other classes than the class that it is currently defined in. The higher the coupling between classes, the higher the number of classes are affected when changes are made to the system. A small premeditated change in a highly coupled system could result in a long series of unanticipated changes in a lot of classes. Hence, class interdependence should be kept to the minimum if possible.

The rule of thumb for this code smell is that if the feature does not really belong to the class it is defined in, it should then be given a new home.

A. Coupling and Cohesion Measures

Since Feature Envy code smell concerns coupling and cohesion, we look at some existing coupling and cohesion measures. Chidamber and Kemerer [2] defined coupling as a situation when methods declared in one class use methods or instance variables defined by the other classes. They propose a metric called Coupling Between Objects (CBO) counts the number of other classes to which it is coupled.

There are several classes of cohesion measures: structural metrics [2, 5], slice-based metrics [10, 13], and information retrieval approach [8, 14]. Lack of Cohesion in Methods (LCOM) and its variants are the most investigated structural cohesion metrics.

It has been observed that the granularities of existing structural coupling and cohesion metrics are at the package or the class level but Feature Envy occurs inside the class, or more specifically, at the method level. Hence, such metrics are not applicable for use in Feature Envy detection.

978-1-4799-6049-1/15/$31.00 ©2015 IEEE