Reflexão 02 - Práticas Ágeis e Refactoring

Paulo Kiyoshi Oyama Filho

"Se refactoring é tão importante no contexto de práticas ágeis, se práticas ágeis são tão adotadas hoje em dia, por que um estudo empírico mostrou que 80% dos bad smells permanacem no código ao longo do tempo e por que refactoring é responsável por eliminar apenas 9% dos smells?"

A refatoração é uma área de muito estudo em computação, servindo até de assuntos para muitos livros. Ela busca transformar os códigos que já existem em código mais bem estruturados, organizados em suma busca melhorar o código. Contudo, como muitas dessas refatorações não tem o efeito desejado ou pior acabam por tornar o código mais completo e menos .

Primeiramente, para responder a reflexão temos o caso dos programadores que conhecem que há um *bad smell* e sabem que ali tem um problema porém não sabem como resolvê-lo ou não conseguem, isso acontece por causa da alta complexidade do código ou muitas das vezes a refatoração é tão grande e tão difícil que acaba por desanimar o programador, por exemplo em código duplicados, em que cada parte tem uma única diferença nesse código tornando díficil a generalização do mesmo. Além disso, também há os desenvolvedores que por falta de experiência não reconhecem um determinado *bad smell,* veja por exemplo que certos *smells* requerem do programador experiência e conhecimento, um exemplo claro são os *bad smells Long Class* (Classes longas) ou *Duplicated Code* (Código duplicado) contra os *Lazy Class* ou *Middle Man*, os dois primeiros por sua vez são visíveis, ou seja você consegue observar que uma classe é longa por meio do número de linhas do seu editor de textos, o mesmo vale para o código duplicado é vísivel quando você usa tem o mesmo código escrito em partes diferentes contudo, os dois últimos são difícies de se observar elas tem mais a ver com conceitos abstratos e alguns termos técnicos, olhe por exemplo a Lazy Class , ela se refere que classes sem muitas inmportância devem ser excluídas, mas como saber o que é uma classe importante ? Outro exemplo, é a Middle Man, esse smell argumenta que se determinada classe tem mais o objetivo de um ponte de conexão entre duas classe do que um classe com responsabilidade própria ela deve ser excluída, mas como é possível achar uma classe dessa, em um projeto grande e muita das vezes tem seus proprios campos e valores ?. Por conta disso, certos smell são reconhecidos por programadores experientes enquantos outros deixam

passar, há, por fim, o grupo que sabe que tem problemas porém não quer em refatorar por causa do esforço a ser feito, não querem refatorar por mexer em muitas partes no código ou com medo de “quebrar”, o que já está funcionando.

Ainda sim, há certas refatorações que por sua vez acabam por piorar o código já existente ou mover um bad smell para outro, o processo de refatoração deve ser feita de maneira estratégica, um exemplo de uma refatoração mal feita é uma extração de classe que se torna um *Middle Man*, ou um *extract method* que torna-se uma *Feature Envy* , portanto a refatoração é um processo complicado porém que possibilita quando bem feito, uma melhoria significativa do código.