# Um estudo sobre o efeito de Mover Classe na Arquitetura de Sistemas de Código Aberto

**Vagner Clementino**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

vagnercs@dcc.ufmg.br

***Resumo. TODO***

## 1. Introdução

A atividade de refatoração tem por objetivo alterar o código fonte de um software sem modificar o seu comportamento. Em última instância, refatorar visa melhorar a qualidade interna do sistema [Fowler 1999, Opdyke 1992]. Sua importância é reconhecida tanto na literatura quanto na indústria, no qual, nesta última, é possível verificar a existência de processos de desenvolvimento que incorporam a refatoração como atividade rotineira [Beck and Fowler 2000]. No últimos anos, pesquisas foram realizadas com o objetivo de entender com qual frequência os desenvolvedores aplicam os diferentes tipos de refatoração [Murphy-Hill et al. 2009]; a relação entre a atividade de refatorar e a correção de *bugs*[Kim et al. 2011] bem como nos resultados de testes de regressão [Kim and Rachatasumrit 2012]. No trabalho de kim et.al discute a percepção dos desenvolvedores sobre a refatoração [Kim et al. 2012].

Recentemente estudos vêm focando em entender as motivações que levam os desenvolvedores a realizem refatoração. Existe um consenso que a motivação original é remover porções de código com baixa qualidade conhecidos como *Bad Smells* [Fowler 1999]. Todavia, estudos demonstraram que o desenvolvedores refatoram para outros fins, como por exemplo, a refatoração *Extrair Método* pode ser utilizada para fins de extensão do sistema ou mesmo possibilitar compatibilidade com versões anteriores do código [Tsantalis et al. 2013]. É possível verificar a utilização do *Extrair Método* visando favorecer a reutilização de código no longo prazo,mesmo que inicialmente não seja esta a intenção [Silva et al. 2015].

Apesar da existência de estudos relativos à motivação da refatoração, ao bem do nosso conhecimento, não existem trabalhos que relacionem a refatoração *Mover Classe* com mudanças na arquitetura do sistema. Suspeitamos que o fato de um desenvolvedor mover classes em um número maior que padrão do sistema tem por objetivo alterar a arquitetura do sistema. Este tipo de ação é conhecida como *batching moving*. A mudança na arquitetura pode ter como objetivo, por exemplo, reorganizar o código existente em uma nova camada lógica ou ainda tratar o problema da *Erosão Arquitetural*. O processo de Erosão arquitetural é conhecido como os desvios ocorridos no código de um sistema que causam violação de alguma regra arquitetural previamente estabelecida[Perry and Wolf 1992].

Neste sentido, este trabalho se propõe em analisar a relação entre mudanças na arquitetura de um sistema e a refatoração *Mover Classe*. A fim de investigar tal relação

analisamos o histórico de versões de 04 sistemas de código aberto desenvolvidos em Java e procedemos com um *survey* com os desenvolvedores visando responder as seguintes questões de pesquisa:

Listar os sistemas escolhidos

**RQ1** Com qual frequência a refatoração Mover Classe tem por objetivo alterar a arquitetura do sistema (*batching moving*)?

**RQ2** Com qual frequência o desenvolvedor informa no *log de commit* que a refatoração teve por objetivo alterar a arquitetura do sistema?

**RQ3** Segundo dos desenvolvedores, Mover Classe foi efetivamente utilizado para alterar a arquitetura dos sistema?

Ao responder as questões proposta neste trabalho entendemos que iremos contribuir no aumento do entendimento das razões que levam os desenvolvedores a realizar refatorações. Esta informação poderá ser utilizada posteriormente na construção de ferramentas que ajudem os times de desenvolvimento em tarefe relativas à mudança da arquitetura do software. Além disso será possível verificar a relação entre a atividade de refatoração e a mudança de arquitetura de um sistema.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 descreve a metodologia utilizada neste estudo; a Seção 3 apresenta os resultados e responde as questões de pesquisas propostas; na Seção 4 realiza-se uma discussão sobre as ameaças à validade do trabalho; a Seção 5 apresenta os trabalhos relacionados à análise da atividade e detecção de refatoração; a Seção 6 sumariza o artigo e discute suas principais contribuições.

## 2. Metodologia

## 3. Resultados

## 4. Ameaças à Validade

## 5. Trabalhos Relacionados

## 6. Conclusão

## Referências


Beck, K. and Fowler, M. (2000). *Planning Extreme Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, USA.

Kim, M., Cai, D., and Kim, S. (2011). An empirical investigation into the role of api-level refactorings during software evolution. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 151–160, New York, NY, USA. ACM.

Kim, M. and Rachatasumrit, N. (2012). An empirical investigation into the impact of refactoring on regression testing. In *Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM)*, ICSM '12, pages 357–366, Washington, DC, USA. IEEE Computer Society.

Kim, M., Zimmermann, T., and Nagappan, N. (2012). A field study of refactoring challenges and benefits. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, pages 50:1–50:11, New York, NY, USA. ACM.

Murphy-Hill, E., Parnin, C., and Black, A. P. (2009). How we refactor, and how we know it. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 287–297, Washington, DC, USA. IEEE Computer Society.

Opdyke, W. F. (1992). *Refactoring Object-oriented Frameworks*. PhD thesis, Champaign, IL, USA. UMI Order No. GAX93-05645.

Perry, D. E. and Wolf, A. L. (1992). Foundations for the study of software architecture. *SIGSOFT Softw. Eng. Notes*, 17(4):40–52.

Silva, D., Valente, M. T., and Figueiredo, E. (2015). Um estudo sobre extração de métodos para reutilização de código.

Tsantalis, N., Guana, V., Stroulia, E., and Hindle, A. (2013). A Multidimensional Empirical Study on Refactoring Activity. *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, pages 132–146.