Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Título da monografia

Thiago Gomes Cavalcanti
Vinícius Correa de Almeida

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientador
Prof. Dr. Rodrigo Bonifácio de Almeida

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Computação — Licenciatura

Coordenador: Prof. Dr. Wilson Henrique Veneziano

Banca examinadora composta por:

Prof. Dr. Rodrigo Bonifácio de Almeida (Orientador) — CIC/UnB
Prof. Dr. Professor I — CIC/UnB
Prof. Dr. Professor II — CIC/UnB

CIP — Catalogação Internacional na Publicação

Cavalcanti, Thiago Gomes.

Título da monografia / Thiago Gomes Cavalcanti, Vinícius Correa de Almeida. Brasília : UnB, 2015.
23 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2015.

1. language design, 2. language evolution, 3. refactoring,
4. microrefactoring, 5. java

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Título da monografia

Thiago Gomes Cavalcanti
Vinícius Correa de Almeida

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof. Dr. Rodrigo Bonifácio de Almeida (Orientador)
CIC/UnB

Prof. Dr. Professor I
CIC/UnB

Prof. Dr. Professor II
CIC/UnB

Prof. Dr. Wilson Henrique Veneziano
Coordenador do Curso de Computação — Licenciatura

Brasília, 31 de março de 2015

Dedicatória

Dedico a....

Agradecimentos

Agradeço a....

Abstract

Resumo.....

Palavras-chave: language design, language evolution, refactoring, microrefactoring, java

Abstract

Abstract.....

Keywords: language design, language evolution, refactoring, microrefactoring, java

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o passar do tempo as linguagens de programação evoluem entretanto não sabe-se ao certo como os softwares projetados e implementados há alguns anos acompanharam tais atualizações. Conforme explicado por [1], tal evolução faz com que características obsoletas sejam mantidas e raramente são removidas de uma linguagem o que acarreta em um aumento da complexidade, aprendizagem e da manutenção do software. Isso naturalmente aumenta a dificuldade de desenvolvimento o que resulta em um aumento de dificuldade de aprendizagem de determinada versão já ultrapassada de uma linguagem e faz com que a equipe alterne entre propriedades atuais e antigas as quais passam a ser quase um dialeto da linguagem implicando no aumento de tempo para conceber um projeto e consequentemente gerindo aumento no custo final projeto.

Uma decisão não é tão simples de manter uma porção do código congelado, sem evolução, ao longo projeto devido alguma restrição técnica. O que infelizmente acarreta em uma estagnação de todo um sistema pois não é somente o projeto afetado, mas sim uma toda infraestrutura como compiladores, banco de dados e sistema operacional e que se de alguma forma vierem a ser atualizados com esta porção código estagnado pode ocasionar problemas como uma queda significativa de desempenho ou até mesmo o sistema parar de funcionar. Devido a esses problemas de código não atualizado com as versões com estruturas mais atuais a proposta da realização de refatoração através de ferramenta a serem desenvolvidas que visem atacar esse gargalo deixado por código obsoleto.

A base para tal trabalho será o desenvolvimento de algumas ferramentas que auxiliem em mudanças em um código legado para reduzir suas estruturas obsoletas e com baixa performance. Essas ferramentas tem como base a sua construção em linguagem *Java* com o intuito de tornar o processo de construção mais ágil e posteriormente aberto para o acoplamento de novos módulos. A construção de uma arvore sintática é um passo onde é feito para cada arquivo de código .Java e posteriormente de todos os arquivos .Java contidos em qualquer projeto para posterior análise. Este parser implica em listar todos os arquivos Java e gerar uma Árvore Sintática Abstrata (AST) para que posterirormente haja um percorrimento de blocos de codigo afim de compará-los como estes com a versão atual.

O processo de refatoração tem como motivação a reestruturação de codigo, de forma que o codigo considerado pelo processo, morto,duplicado ou com perca de desempenho

não haja codigo morto, duplicado ou com perca de desempenho em um determinado trecho de codigo. Esse processo não tem como premissa a atualizaçao do codigo para novas estruturas de versões da liguagem mais recentes. Essa nova abordagem de refatoração tem com motivaçao a retirada de codigo obsoleto, devido a novas abordagens e estruturas das novas versões, e com baixo desempenho de sistemas sem prejuizo na sua engenharia e funcionamento.

Devido a esse tipo de refatoração de código visa a evolução do código para a uma mais recente com estruturas onde não haja perca de desempenho devido a mudança e também a atualização do código legado para estruturas modernas. Tais estruturas antigas com a ação dessa proposta de refatoração tendem com o tempo sair das estruturas providas pela linguagem Java como aconteceu com Fortan 90 como visto em [1] e essa abordagem tem o intuito de diminuir a quantidade de estruturas antigas que não fazem mais sentido pois novas estruturas realizam as mesmas funções no interpretador da linguagem Java. As modificações proposta pela ferramenta de refatoração não deverá trazer com que haja perca de desempenho ou aumento da complexibilidade do código portanto deixando como uma sugestão a alteraçao do codigo ou segmento de codigo para o usuário a adesão das propostas de refatoração ou não.

A árvore de sintaxe abstrata (AST) é uma estrutura de dados que representa estruturas de cadeias sintaticas onde é respresentada por um esqueleto semantico da liguagem em questão. É constituída atravez de um framework do ambiente de desenvolvimento integrado chamado Eclipse onde o nome desse framework é chamado de EclispeJDT onde traz métodos implementados pelo próprio framework para o percorrimento e ações na árvore sintática. A ideia é transformar inicialmente qualquer código fonte java em uma árvore sintática e devido a isso, o mapeamento da arvore já construída que é muito conveniente para inspecionar o código fonte de um arquivo ou de um projeto com vários arquivos em diferentes diretórios. Com isso é possível realizar ou sugerir ao usuário modificações no código fonte através desta árvore construída e isto seria referenciado automaticamente no código fonte.

A proposta é criar ferramentas de análise estática para códigos fonte da linguagem Java para que possa-se apurar projetos pequenos e posteriormente em projetos *open-sources* para a verificação da existência de alguma defasagem de estruturas entre qualquer versão da linguagem Java que fora concebido para a versão atual e estável na qual a linguagem se encontra. O desenvolvimento das ferramentas será com a versão mais atual da linguagem Java que neste momento é o Java versão 8 para verificar se os softwares desenvolvidos está versão nesta versão e como acompanharam a evolução de decadas de novas versões sem atualização do codigo por motivo de enegenharia outro qualquer.

Capítulo 2

Computação quântica

texto.... referência [**?**]

Referências

[1] Jeffrey L. Overbey and Ralph E. Johnson. Regrowing a language: refactoring tools allow programming languages to evolve. (10):493–502, October 2009. 1, 2