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.
21 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 sabemos ao certo como os softwares projetados 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 no aumento da complexidade, da aprendizagem e da manutenção do software. Isso naturalmente aumenta a dificuldade de desenvolvimento o que resulta em um aumento de difuculdade de aprendizagem de determinada vesão 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 gerendo aumento no custo final projeto.

Uma decisão não tão simples é 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 afetedado, é toda uma 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 sérios problemas como uma queda significativa de desempenho ou até mesmo o sistema parar de funcionar.

A base para tal trabalho será um parse de todos os arquivos .java contidos em um projeto para posterior análise. Este parse implica em listar todos os arquivos java e gerar uma Árvore Sintática Abstrata AST e depois percorrer os statements e comparar como estes com a versão atual.

Árvores de sintaxe abstratas (AST) são a base de um poderoso framework o Eclipse IDE,e ainda de refectoring. A idéia de mapear inicialmente código fonte java em uma árvore sintática é muito conveniente para inspecionar o código fonte de um arquivo ou de um projeto. Com isso é possível realizar ou sugerir modificações nesta árvore e isto seria referenciado automaticamente no código fonte.

Nossa proposta é criar uma analisador estático para apurar projetos opensources e checar se existe alguma defazagem entre versão da linguagem que este fora concebido para a versão atual na qual a linguagem se encontra. Trabalharemos com a versão mais atual da linguagem Java que neste momento é 8 para checar os softwares desenvolvidos

com esta e como acompanharam tal evolução.

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] Max Schäfer and Oege de Moor. Specifying and implementing refactorings. (16):286–301, October 2010.