Universidade Federal de Pernambuco
Centro de Informática

Bacharelado em Ciência da Computação

Análise de extensão do CSDiff para uso em linguagens com poucos separadores sintáticos

José Gabriel Silva Pereira

Trabalho de Graduação

Recife
27 de Abril de 2023

Universidade Federal de Pernambuco
Centro de Informática

José Gabriel Silva Pereira

Análise de extensão do CSDiff para uso em linguagens com poucos separadores sintáticos

Trabalho apresentado ao Programa de Bacharelado em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Paulo Henrique Monteiro Borba*
Co-orientadora: *Paola Rodrigues de Godoy Accioly*

Recife
27 de Abril de 2023

Agradecimentos

TODO: agradecimentos

Resumo

A prática de desenvolvimento de software, há muito tempo deixou de ser uma tarefa que era realizada por somente uma pessoa, pois, com o avanço da tecnologia, sistemas cada vez mais complexos foram sendo criados fazendo com que muitas pessoas trabalhassem no mesmo projeto. Por conta disso, ferramentas de controle de versionamento de código foram criadas, permitindo que múltiplos desenvolvedores trabalhassem modificando o mesmo trecho de código simultaneamente. Porém, essas modificações simultâneas podem gerar conflitos quando feitas em um mesmo pedaço de código, o que impacta negativamente na produtividade de um time. Ao decorrer do tempo, diversas formas de como detectar conflitos na junção de versão de códigos foram criadas, dentre elas: linha a linha, estruturada e semiestruturada. Neste trabalho, é proposto uma extensão para uma ferramenta semiestruturada de detecção de conflitos ja existente, o *CSDiff* [1], de forma que ela utilize indentação como separador da linguagem, permitindo assim que, durante a detecção de conflitos em linguagens com poucos separadores sintáticos, ainda seja possivel permitir uma redução de falsos conflitos, consequentemente melhorando a produtividade de um time.

Palavras-chave: Processo de Merge, Desenvolvimento Colaborativo, Merge Textual, Merge Estruturado, Separadores sintáticos

Abstract

The practice of software development has long ceased to be a task performed by only one person, as with the advancement of technology, increasingly complex systems have been created, causing many people to work on the same project. Therefore, code version control tools were created, allowing multiple developers to work on modifying the same piece of code simultaneously. However, these simultaneous modifications can generate conflicts when made on the same piece of code, negatively impacting a team's productivity. Over time, several ways of detecting conflicts in the merging of code versions have been created, including line-by-line, structured, and semi-structured. In this work, an extension is proposed for an existing semi-structured conflict detection tool, the *CSDiff* [1], so that it uses indentation as a language separator, thus allowing, during conflict detection in languages with few syntactic separators, a reduction in false conflicts, consequently improving a team's productivity.

Keywords: Merge Process, Collaborative Development, Textual Merge, Structured Merge, Syntactic Separators.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

TODO: deixar merge em italico

TODO: ajeitar referencias de acondo com joc

Com o crescimento da complexidade dos sistemas de software, surge a necessidade de que múltiplas pessoas trabalhem num mesmo projeto. Essas modificações, com o objetivo de trazer mais produtividade, costumam ser executadas de forma paralela e podem acontecer em trechos de código em comum [1]. Tendo como objetivo auxiliar os desenvolvedores a controlar e versionar suas modificações no código, ferramentas de controle de versão de código foram criadas. Essas ferramentas auxiliam a reduzir o trabalho extra quando se trata de modificações paralelas que precisam ser unidas [2]. O processo de unir duas modificações paralelas de código é chamado de merge [3].

No processo de merge, quando dois desenvolvedores modificam o mesmo trecho de código e essas mudanças interferem uma na outra, é gerado um conflito. Esses conflitos, quando detectados, algumas vezes precisam ser resolvidos por um ou ambos os desenvolvedores, o que acaba impactando na produtividade, dado que resolvê-los geralmente é uma tarefa tediosa e que demanda tempo [4] [5]. Além do impacto na produtividade do time, quando esses conflitos não são detectados pela ferramenta de merge, ou quando são detectados e mal resolvidos, eles podem levar à introdução de bugs dentro do código, o que influencia na qualidade do produto final [6].

A abordagem de merge mais utilizada na indústria atualmente é o merge não estruturado [7], que se utiliza de uma análise puramente textual, equiparando linha a linha trechos do código para detectar e resolver conflitos. Porém, por não utilizar a estrutura do código que está sendo integrado, por muitas vezes essa abordagem gera falsos conflitos. Ao observar isso, pesquisadores propuseram ferramentas que se baseiam na estrutura dos arquivos que estão sendo integrados, criando uma árvore sintática a partir do texto dos arquivos e de sua linguagem de programação [8]. Essas abordagens são chamadas estruturadas e semiestruturadas.

Estudos anteriores [8] [9] [10] compararam as duas abordagens (estruturada e semiestruturada) em relação à não estruturada e mostraram que, para a maioria das situações de merge dos projetos, houve uma redução de conflitos em favor da semi ou da estruturada. Essa redução se dá por conta de falsos conflitos que possuem resolução óbvia, como por exemplo, quando os desenvolvedores adicionam dois métodos diferentes e independentes numa mesma região do código [11].

Esse benefício advém da exploração da estrutura gerada pela análise sintática, também chamada de análise gramatical. Ela envolve o agrupamento dos tokens do programa fonte em frases. Cada linguagem possui conjuntos de tokens, onde alguns servem como divisores de elementos sintáticos e escopo semântico, como por exemplo as chaves (’’, ’’) numa linguagem

como Java. Estes tokens, especificamente, são definidos aqui simplesmente como separadores sintáticos. TODO: transformar o "separadores"em negrito

A solução não estruturada mais utilizada, o Diff3, se baseia somente na quebra de linha como o divisor de contexto para detecção de conflitos. Assim, o algoritmo de merge compara as linhas mantidas, adicionadas, e removidas por cada desenvolvedor e, com base nisso, reporta conflito quando as mudanças ocorrem em uma mesma área do texto, isto é, quando não há uma linha mantida que separa as mudanças feitas por um desenvolvedor das mudanças feitas pelo outro.

Como forma de melhorar os resultados do Diff3, o CSDiff, proposto em trabalhos anteriores, utiliza-se dos separadores mencionados acima para dividir o contexto de cada linha de código. Assim, o algoritmo de merge consegue, por exemplo, resolver conflitos em uma mesma linha, contanto que esses conflitos estejam separados por um dos separadores definidos.

Contudo, o CSDiff possui limitações, pois linguagens como Python, possuem poucos separadores (seu principal separador é a própria indentação do código, que não é considerado pelo CSDiff atual). Este trabalho propõe uma modificação para o CSDiff, que utiliza a indentação como um separador sintático, de forma a tentar resolver esse problema, e analisa os resultados em comparação ao merge não estruturado puramente textual. Em particular, investiga-se as seguintes perguntas de pesquisa:

1) PP1: A nova solução de merge não estruturado, utilizando indentação, reduz a quantidade de conflitos reportados em comparação ao merge puramente textual?
2) PP2: A nova solução de merge não estruturado, utilizando indentação, reduz a quantidade de cenários com conflitos reportados em comparação ao merge puramente textual?
3) PP3: A nova solução de merge não estruturado, utilizando indentação, reduz a quantidade de falsos conflitos e cenários com falsos conflitos reportados (falsos positivos) em comparação ao merge puramente textual?
4) PP4: A nova solução de merge não estruturado, utilizando indentação, amplia a possibilidade de comprometer a corretude do código, por aumentar o número de integrações de mudanças que interferem uma na outra, sem reportar conflitos (falsos negativos), além de aumentar cenários com falsos negativos?
5) PP5: A nova solução de merge não estruturado, utilizando indentação, demonstra um aumento de produtividade considerando o ato de resolver conflitos de merge?

Os resultados obtidos mostram que além de amentar a quantidade de conflitos reportados (como esperado dado os resultados dos trabalhos anteriores), o merge não estruturado utilzando separadores e indentação, demonstra

TODO: dados necessarios: aproximacao em % da quantidade de reducao de cenarios com conflitos ou de conflitos em si.

Os conceitos utilizados para esta análise são explicados no capítulo (TODO: adicionar capituli aqui)

CAPÍTULO 2

Motivação

2.1 Merge Estruturado

2.2 Merge Semiestruturado e Estruturado

2.3 Merge Não Estrurado com Comparadores

CAPÍTULO 3

Solução

3.1 CSDiff

3.2 Implementação

CAPÍTULO 4

Avaliação

4.1 CONCEITOS

4.1.1 Cenário de Merge

4.1.2 Falso Positivo Adicionado

4.1.3 Falso Negativo Adicionado

4.1.4 Resultado Errado de Merge

4.2 PERGUNTAS DE PESQUISA

4.2.1 A nova soluÿcao de merge nao estruturado, utilizando separadores, reduz a quantidade de conflitos reportados em comparaÿcao ao merge puramente textual?

4.2.2 A nova soluÿcao de merge nao estruturado, utilizando separadores, reduz a quantidade de cent'arios com conflitos reportados em comparaÿcao ao merge puramente textual?

4.2.3 A nova soluÿcao de merge nao estruturado, utilizando separadores, reduz a quantidade de falsos conflitos e cent'arios com falsos conflitos reportados (falsos positivos) em comparaÿcao ao merge puramente textual?

4.2.4 A nova soluÿcao de merge nao estruturado, utilizando separadores, amplia a possibilidade de comprometer a corretude do ct'odigo, por aumentar o nt'umero de integraÿcoes de mudanÿcas que interferem uma na outra, sem reportar conflitos (falsos negativos), alt'em de aumentar cent'arios com falsos negativos?

4.2.5 PP5 sobre aumento de produtividade

TODO: adicionar uma subsubsection pra cada possivel situacao desse role, após explicar o role

4.3 METODOLOGIA

4.4 RESULTADOS

TODO: copiar as subsections do perguntas de pesquisa aqui

4.5 DISCUSSÃO

4.6 AMEAÇAS A VALIDADE

CAPÍTULO 5

Trabalhos Relacionados

CAPÍTULO 6

Conclusão

Bibliografia

[1] J. Clementino, P. Borba e G. Cavalcanti, “Textual merge based on language-specific syntactic separators,” em *Brazilian Symposium on Software Engineering*, 2021, pp. 243–252.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).