UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM ESTUDO SOBRE A ABORDAGEM SEMIESTRUTURADA PARA RESOLUÇÃO DE CONFLITOS NO GIT

PROPOSTA DE TRABALHO DE GRADUAÇÃO

Aluno	Guilherme José Carvalho Cavalcanti	{gjcc@cin.ufpe.br}
Orientador	Paulo Henrique Monteiro Borba	{phmb@cin.ufpe.br}
Co-Orientadora	Paola Rodrigues Godoy Accioly	{prga@cin.ufpe.br}

RECIFE, NOVEMBRO DE 2013.

Sumário

1 – Contexto

Atualmente, o desenvolvimento de grandes projetos de software tem como característica essencial o desenvolvimento paralelo por equipes de desenvolvedores. Como esse desenvolvimento paralelo é estruturado e suportado impacta profundamente na qualidade, cronograma e sucesso do projeto [1]. Problemas de coordenação de atividades e consequentes conflitos ocorrem frequentemente e estudos demonstram que modificações paralelas são as principais causas de defeitos nos sistemas. Problemas de coordenação durante o desenvolvimento paralelo lideram os problemas de integração de código, com isso, os desenvolvedores têm dificuldade em identificar o impacto de suas alterações e gastam bastante tempo coordenando seus esforços. [1,3,6]

Durante o desenvolvimento paralelo de software, a ocorrência de conflitos surge como um dos principais problemas porque os desenvolvedores possuem cópias inconsistentes do software compartilhado. Tais conflitos são dispendiosos, pois eles atrasam o projeto enquanto o conflito é investigado e resolvido [2]. Um bom desenvolvimento paralelo de software só é viável graças a uma gerência de configuração e o consequente uso de sistemas de controle de versões. Sistemas de controle de versões permitem ao desenvolvedor baixar arquivos e modificá-los na sua área de trabalho local, que é periodicamente sincronizada com o repositório que contém a versão base. Enquanto o protocolo de sincronização permite um rápido desenvolvimento paralelo, ele também permite que os desenvolvedores façam modificações conflitantes inadvertidamente [3].

Os sistemas de controle de versões podem ser, segundo a abordagem para resolução de conflitos, (1) *não estruturados*, que são puramente baseados em texto e resolvem conflitos via similaridade textual; (2) *estruturados,* que são adaptados para linguagens de programação específicas e usam o conhecimento específico da linguagem para resolução de conflitos [5,7]; e, recentemente, surgiu a abordagem (3) *semiestruturada* que herda o ponto forte de ambos: a generalidade dos sistemas não estruturados e a expressivade dos sistemas estruturados; a ideia desta abordagem é prover informação estrutural sobre os artefatos do software, de modo que os conflitos sejam resolvidos

automaticamente, e quando a informação não for suficiente, aplicar a resolução textual usual no conflito [5].

Dentre os sistemas de controle de versões, um dos mais populares e difundidos da atualidade é o *git,* principalmente entre a comunidade open source [4]. O *git* pertence à classe de sistema de controle de versões *não estruturados*, que são baseados em texto e peca por não utilizar conhecimento sobre os artefatos da linguagem, o que impossibilita a resolução de certos tipos de conflito, como conflitos causados pela ordem de elementos (classes, métodos, etc.).

2 – Objetivos

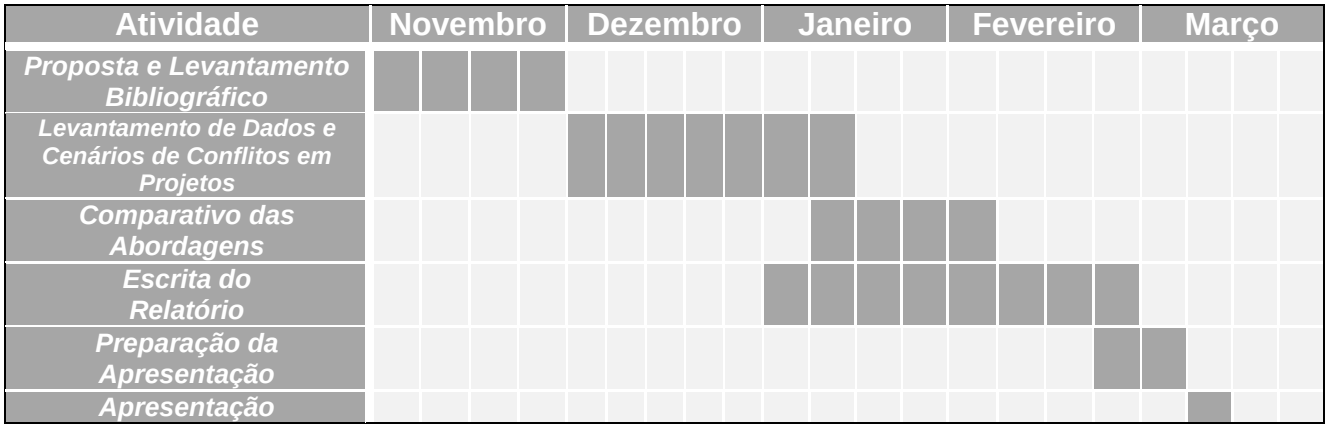
O objetivo deste trabalho é a realização de um estudo empírico sobre as atividades de *merge* e resolução de conflitos em projetos de software que utilizam o *git* como sistema de controle de versões comparando a abordagem *não estruturada* com a abordagem *semiestruturada* para quantificar os benefícios da abordagem *semiestruturada* para resolução de conflitos.

Para este fim, deverão ser utilizadas ambas as abordagens em um determinado número de cenários e, por fim, comparado o número resultante de conflitos.

Este trabalho também visa servir como justificativa para o fomento de ferramentas que permitam o uso da abordagem *semiestruturada* na resolução de conflitos de projetos de software que utilizem o git, que é nativamente *não estruturado,* como sistema de controle de versões. Com advento de tais ferramentas, a atividade de resolução de conflitos estaria cada vez mais automatizada permitindo, desta forma, a redução de atrasos, esforços e, consequentemente, custos causados por conflitos durante o desenvolvimento paralelo de projetos de software.

3 – Cronograma

O cronograma a seguir tem o objetivo de guiar o processo de desenvolvimento deste trabalho de graduação.



4 – Referências

[1] D. E. Perry, H. P. Siy, e L. G. Votta, "Parallel Changes in Large-Scale Software Development: An Observational Case Study," *ACM Trans. Softw. Eng. Methodol,* vol. 10, no. 3, pp. 308–337, Jul. 2001.

[2] Y. Brun, R. Holmes, M. D. Ernst, e D. Notkin, "Proactive detection of collaboration conflicts," *19th ACM SIGSOFT symposium*, pp. 168–178, 2011.

[3] B. K. Kasi e A. Sarma, "Cassandra: Proactive Conflict Minimization through Optimized Task Scheduling," *35th International Conference on Software Engineering*, pp.732-741, 2013

[4] Christian Bird, Peter C. Rigby, Earl T. Barr, David J. Hamilton, Daniel M. German, e Prem Devanbu. 2009. "The promises and perils of mining git". *6th IEEE International Working Conference on Mining Software Repositories*, pp.1-10, 2009.

[5] S.Apel, J.Liebig,B.Brandl,C.Lengauer,C.Kästner, "Semistructured Merge: rethinking merge in revision control systems",em: *ESEC/FSE, ACM*, pp.190–200, 2011.

[6] C. R. B. de Souza and D. Redmiles, "An Empirical Study of Software Developers' Management of Dependencies and Changes," *30th Intern. conf. on Soft Eng.*, pp. 241–250, 2008.

[7] Tom Mens. "A state-of-the-art survey on software merging".*IEEE TSE, 28(5)*: pp. 449– 462, 2002.

5 – Assinaturas

Paulo Henrique Monteiro Borba
Orientador

Paola Rodrigues Godoy Accioly
Co-Orientadora

Guilherme José Carvalho Cavalcanti
Aluno