Estudo de Falsos Negativos em uma Ferramenta de Merge Semi-estruturado

Leonardo Ribeiro Borges[1]

[1] *Universidade Federal de Pernambuco, Pernambuco, Brasil*

December 15, 2018

Abstract: Version control tools, which most commonly use unstructured integration strategies, promote much ease and agility in collaborative development environments. However, a semi-structured integration approach is a more adequate alternative, since it generates a considerable decrease in the number of conflicts found. Although the semi-structured merge generates fewer conflicts in some cases, there are few studies about its false negatives and its performance in business projects. This study proposes an analysis of the false negatives found in software developed in an industry collaborative environment. This analysis did not find any real false negative added by the semistructured one. However, the counting of false negatives showed imprecise when counting cases in which the semi-structured merge detection ignores the conflict in a correct way.

Resumo: Ferramentas de controle de versão, que que mais comumente utilizam estratégias de integração não estruturadas, promovem muita facilidade e agilidade em ambientes de desenvolvimento colaborativos. Porém uma abordagem de integração semiestruturada é uma alternativa mais adequada, pois gera uma diminuição considerável no número de conflitos encontrados. Por mais que o merge semi-estruturado gere menos conflitos em alguns casos, há poucos estudos a respeito dos seus falsos negativos e seu desempenho em projetos empresariais. Esse estudo propõe uma análise dos potenciais falsos negativos encontrados em um software desenvolvido em ambiente colaborativo da indústria. Essa análise não encontrou nenhum falso negativo real adicionado pelo semi-estruturado. Contudo, a contagem de falsos negativo mostrou-se imprecisa ao contabilizar casos em que a abordagem semi-estruturada ignora o conflito de forma correta

***Keywords*—** controle de versão, conflitos, ambiente de trabalho colaborativo, indústria de software

1 Introdução

Em um ambiente de desenvolvimento colaborativo cada membro de uma equipe deve integrar ao ramo do projeto em que estão trabalhando suas alterações e ao final de um prazo estipulado pela organização, as modificações concluídas pela equipe devem ser integradas à ramificação principal do projeto. Nesta etapa de junção que contém as alterações de todas as demais equipes, há a possibilidade de uma ou mais equipes terem modificado o mesmo trecho de código, ocasionando um conflito de integração.

Cada ferramenta de controle de versão contém uma abordagem para a mesclagem de código e a mais comum é a abordagem não estruturada que leva em consideração apenas a diferença entre o texto e o seu posicionamento. Essa abordagem imprecisa não considera o contexto de parte do código, gerando assim conflitos classificados como falsos positivos. Esses conflitos adicionais poderiam ser evitados, pois utilizando uma metodologia que contenha mais inteligência ao realizar o merge poderia oferecer uma otimização no tempo de desenvolvimento.

O processo de resolução de conflitos é uma tarefa que demanda certo tempo para as equipes tratarem devidamente, além de possuir um custo considerável no valor final da entrega. Outro caso que impacta na qualidade e integridade do produto é a não detecção de um conflito real, que pode ter um comportamento não esperado no projeto final. Ambos os casos citados são decorrentes devida à imprecisão da ferramenta de integração.

Uma alternativa ao merge não estruturado é a abordagem semi-estruturada, que diferente da primeira metodologia, essa estratégia entende parte do contexto do código. Ao realizar um merge a semi-estruturada elabora um arranjo dos elementos que compõem o arquivo analisado, como classes, métodos e interfaces. Essas estruturas permitem que a ferramenta tenha um conhecimento sobre qual parte do código ela está tratando. Ao definir o arranjo é executado o algorítmo Kdiff3, o mesmo que a abordagem não estruturada utiliza, dentro de cada estrutura. O estudo em questão é uma análise empírica sobre o desempenho do merge semi-estruturado com relção ao não estruturado em um projeto da indústria de softwares.

2 Motivação

Em um processo de desenvolvimento de software, questões como prazo e custo estão correlacionadas. Se o prazo não é cumprido como deve ser, o custo do projeto pode aumentar. Minimizar a perda de tempo com obstáculos no projeto é um dos focos a ser considerado durante o processo. Porém, tanto a existência de conflitos desnecessários como a não detecção de um conflito são problemas muito recorrentes na engenharia de software. Essas ocorrências, se não tratadas de forma rápida e eficiente, geram prejuízos no final do projeto.

As análises realizadas em estudos anteriores (Cavalcanti, Accioly, and Borba, 2017b) fizeram uso de repositórios bem avaliados do GitHub. Por mais que foi entregue uma gama de cenários para avaliar a ferramenta de merge semi-estruturado, eles podem não retratar o desempenho real em projetos de indústrias de softwares. Este contexto em específico contém padrões de projetos, diferenças na comunicação entre os membros da equipe e exige uma maior corretude nas ocorrências de defeitos encontrados no código. Por conta dessa singularidade na forma de desenvolvimento definida pela empresa, os repositórios, mesmo sendo reconhecidos com um bom desempenho, podem não encaixar-se corretamente no cenário inserido.

Ao se considerar a mudança de uma abordagem de merge, o número de falsos negativos adicionais é uma das características mais relevantes ao se avaliar uma ferramenta de controle de versão. Esses falsos negativos adicionais podem ocasionar comportamentos inesperados e indesejados do projeto, pois os conflitos reais ocorridos não serão indicados. Para ter o seu uso no cotidiano de de-

senvolvimento considerado viável o recurso deve apresentar o número de falsos negativos reais menor ou igual ao da abordagem não estruturada. Alguns casos que exemplificam bem o impacto negativo que um falso negativo real possa vir a causar, seria um merge que gere o arquivo final com a mesma declaração, podendo ocasionar uma quebra no build automático do projeto, ou novos trechos de código que modifiquem um mesmo elemento em diferentes áreas e gere um resultado final não esperado, podendo gerar perdas significativas, caso trate-se de uma operação financeira pro exemplo.

A abordagem de merge semi-estruturado S3M implementada por (Cavalcanti, Accioly, and Borba, 2017a) exibe um resultado impreciso em relação ao seu real ganho de desempenho, pois o número de falsos negativos reportados é superestimado pelo software de mineração. Ao analisar o resultado real manualmente da ferramenta, nota-se que o desempenho final é superior ao informado. Ou seja, a quantidade de falsos negativos declarado pela S3M é maior do que realmente é encontrado no repositório usado como amostra de análise.

Esse trabalho visa entender os possíveis falsos negativos reportados pela S3M em um projeto de indústria de software e classificá-los a fim de avaliar o desempenho d o recurso. Essa análise apresenta um grau de relevância para que problemas pontuais tenham sua devida correção e assim atue como um passo para extrair a melhor performance possível.

3 Estudo

O estudo em questão utilizou como base de análises um software de uma empresa que cedeu seu código e seu histórico no git. O projeto analisado foi desenvolvido predominantemente na linguagem Java e mantém-se em constante manutenção desde 2013. Inicialmente o projeto foi desenvolvido com o auxílio da ferramenta SVN e posteriormente migrado para GitLab, em abril de 2016. Seu último commit a ser analisado foi em junho de 2018. O projeto contém um total de 241 (duzentos e quarenta e um) merges realizados na master.

O processo de pesquisa fez uso da mesma ferramenta que executa a mineração no estudo de (Cavalcanti, Accioly, and Borba, 2017a), o S3M. O software de mineração reproduz todos os merges realizados no seguimento principal do projeto, denominado como master, utilizando ambas as abordagens não estruturada e a semiestruturada. O software ao reproduzir os merges realiza um refinamento dos conflitos para informar dos potenciais falsos negativos.

O minerador, após sua execução, gera diversos arquivos utilizados para análises. Um deles é o *conflicts.unstructured* que contém os conflitos que foram reportados pela abordagem não estruturada, mas que não foram detectados na semiestruturada. Todos os possíveis falsos negativos que foram reportados pelo S3M estão nesse arquivo, contudo parte dos conflitos que aparecem nesse arquivo foram tratados pelo minerador e não foram contabilizados como falsos negativos. O contrários ocorre no arquivo conflicts.semistructured, ou seja, este possui divergências reportadas pela abordagem semi estruturada, mas que não foram detectados na não estruturada. Se a mesma ocorrência estiver em ambas as abordagens o registro é feito no arquivo conflicts.equals.

A fim de identificar os potenciais falsos negativos encontrados foi necessário a implantação de uma pequena extensão no código do minerador que pudesse retornar um resultado mais aproximado do real número de falsos negativos contabilizados pela ferramenta. A ideia por trás dessa extensão, que facilitaria e daria uma maior confiabilidade na análise dos casos, consiste na lógica de imprimir em um arquivo texto os conflitos que foram analisados pela ferramenta toda vez que um potencial falso negativo fosse informado.

Os resultados da mineração inicialmente trouxeram um total de cento e quatro conflitos em um único arquivo que registra os commits que não foram reportados pelo semi-

Table 1: *Resultado do refinamento dos conflitos*

Arquivo	nº conflitos	FN presentes
Arquivo 1	5	1
Arquivo 2	9	3
Arquivo 3	2	2
Arquivo 4	1	1
Arquivo 5	9	9
Arquivo 6	59	2
Arquivo 7	3	2
Arquivo 8	2	2

estruturado em vinte e um cenário de merge que continham eventuais falsos negativos. Com a extensão desenvolvida, foram retornados oito arquivos que continham o indicativo da quantidade de potenciais falsos negativos. Totalizando o número de resultados dos oito arquivos obtivemos noventa conflitos. Numericamente a redução não parece significativa, contudo com a divisão dos conflitos nos oito arquivos e com o valor exato de potenciais falsos negativos que são reportados em cada um desses documentos, uma maior precisão foi apresentada na detecção do que entra na conta de potenciais falsos negativos da ferramenta como mostra na tabela 1.

Todos os arquivos, com exceção do seis, tiveram seus falsos negativos detectados e catalogados manualmente com base no comportamento conhecido da ferramenta. O documento seis trouxe um questionamento a respeito do funcionamento do minerador, devido ao vasto número de conflitos contabilizados pela ferramenta e o baixo número de falsos negativos reportados. Os conflitos existentes no arquivo possuem um comportamento semelhante a outros já registrados anteriormente como potenciais falsos negativos. A soma dessas diversas ocorrências que se assemelham foi superior ao número de potenciais falsos negativos a este documento, de forma que isso possa vir a representar um problema na fórmula que calcula os conflitos. A fim de validar a possível existência de um falso negativo real foram analisados todos os conflitos existentes no arquivo, para confirmar que mesmo que a contagem de potenciais falsos negativos esteja equivocada na conta os conflitos que ali estejam presentes não possam ser catalogados como um falso negativo real, fato que veio a se confirmar após a análise.

A fim de confirmar a hipótese de uma eventual falha na contagem dos falsos negativos, um outro estudo foi realizado com base nos conflitos que não pertenciam a esses oito arquivos listados, ou seja, os conflitos que certamente não foram contabilizados como falsos negativos. Foi selecionado uma amostra de cinco conflitos com comportamentos distintos para um análise manual com base no conhecimento prévio do funcionamento da abordagem semiestruturada. Como resultado obtivemos que dois dos selecionados a ferramenta não era inteligente suficiente para tratar devidamente as divergências.

4 Resultados

Após a execução da ferramenta de mineração obtivemos como resultado que o S3M foi executado em 186 arquivos do projeto, com 22 potenciais falsos negativos adicionais, 92 falsos positivos evitados e 137 conflitos gerados e 4288 linhas de código. Observando a abordagem não estruturada obtivemos 164 conflitos e 6200 linhas de código, o que representa uma redução de aproximadamente 16,64% no número de conflitos e uma redução de 30,83% das linhas de códigos a serem analisadas para a resolução dos conflitos.

Utilizando os resultados obtidos por (Cavalcanti, Accioly, and Borba, 2017a) pudemos observar que a redução no número de conflitos teve um valor percentual superior de 24%. Nenhuma conclusão pode ser extraída apenas dessa comparação devido ao espaço amostral limitado a um único projeto do estudo. Contudo é representada um indício de que os projetos com desenvolvimento num ambiente corporativo apresentam comportamentos distintos de repositórios abertos.

O resultado da tabela dois, que agrega o resultado final da análise, desconsidera dois

Table 2: *Análise dos potenciais falsos negativos*

Conflito não estruturado	104
Falso negativo reportados	22
Falsos negativos reais	0
Diferença no match	9
Espaçamento	9
Ordenamento	2

dos possíveis falsos negativos referentes ao arquivo seis conforme indicado na tabela um devido ao fato do tópico anteriormente explicado. Todos os conflitos que não foram classificados como falsos negativos se caracterizam como verdadeiros negativos, pois representam o funcionamento correto da abordagem semiestruturada que pode ter diferentes interpretações de acordo com o tipo do conflito.

4.1 Diferenças no match

O tipo denominado diferenças no match ocorre quando os conflitos, por mais que representem a mesma alteração de código, contém uma pequena diferença no texto dos conflitos. As Figuras 1 e 2 exemplificam bem um caso desse tipo, onde é possível ver que ambas as abordagens geraram um conflito essencialmente iguais, contudo o comentário que também veio a aparecer no conflito gerado pela abordagem não estruturada gerou uma diferença textual que impactou na contagem dos falsos negativos.

4.2 Ordenamento

Como dito anteriormente a abordagem semiestruturada identifica estruturas como classes, métodos, interfaces ou imports e separa essas estruturas para comparar as diferenças entre os arquivos. Essa abordagem trata em especial os conflitos denominados como ordenamento, pois a separação dos métodos evita que apenas a posição e a diferença de textos seja levada em conta, como é feito na abordagem não estruturada.

As estruturas montadas pela estratégia semiestruturada evita que o conflito exem-

```
<<<<<<< MINE (Current Change)

    int a = 0;
    int b = 1;

    return a + b;
=======

    int a = 10;
    int b = 1;

    return a - b;
>>>>>>> YOURS (Incoming Change)
```

Figure 1: *Conflito gerado pela abordagem semi-estruturado*

```
<<<<<<< MINE (Current Change)
    // Soma dos valores a e b
    int a = 0;
    int b = 1;

    return a + b;
=======

    int a = 10;
    int b = 1;

    return a - b;
>>>>>>> YOURS (Incoming Change)
```

Figure 2: *Conflito gerado pela abordagem não estruturado*

```
<<<<<<< MINE (Current Change)
    public void getA(){
        return a;
    }

    public void getB(){
        return c;
    }
=======
    public void getC(){
        return c;
    }
>>>>>>> YOURS (Incoming Change)
```

Figure 3: *Conflito de ordenamento gerado pela abordagem não estruturada*

plificado na Figura 3 ocorra em merges, caracterizando-se como um verdadeiro positivo da abordagem não estruturada. O minerador que foi utilizado para este estudo em questão não tem uma inteligência suficiente o bastante para detectar um conflito de ordenamento como esse. O que foi notado ao longo do estudo é que nos conflitos de ordenamento, apesar de não serem de fato falsos negativos, ocorreu certa imprecisão na contagem. Isso se deu ao fato de que conflitos com a mesma lógica uma hora eram contabilizados, outra hora não eram.

4.3 Espaçamento

Os conflitos que se caracterizam como espaçamento possuem causas semelhantes aos conflitos de ordenamento, de forma que a estratégia semiestruturada não reporta o conflito devida a sua estrutura elaborada. Conforme o exemplo na figura 4, a abordagem semiestruturada não seria afetada pelas quebras de linha, devido a montagem da estrutura. Entretanto, o minerador não faz um tratamento adequado para detectar e remover da conta conflitos com quebra de linha.

4.4 Propostas

Após a classificação e análise de todos os potenciais falsos negativos foi discutido propostas para promover uma aproximação real

```
<<<<<<< MINE (Current Change)
    public void exemplo(int a, int b, int c, int d){
=======
    public void exemplo(
        int a,
        int b,
        int c,
        int d) {
>>>>>>> YOURS (Incoming Change)
```

Figure 4: *Conflito de espaçamento gerado pela abordagem não estruturada*

maior dos potenciais falsos negativos aqui apresentados.

Uma das ideias impacta nos conflitos catalogados como de diferenças no match seria uma análise percentual da semelhança entre os textos. Como os conflitos de ambas as abordagens eventualmente aparecem com o texto levemente diferente a estratégia para detectar esse tipo de ocorrência poderia ocorrer da seguinte forma:

1. Montar duas estruturas, uma contendo todas as linhas que contem o conflito pela abordagem não estruturada e outra contendo todas as linhas pertencentes ao conflito gerado pela abordagem semiestruturada.
2. Cada elemento de uma estrutura deve se comparar com os nós da segunda estrutura que ainda não foram marcados e marca-los caso haja semelhança entre eles, até todos os nós da primeira estrutura tentem se comparar com os nós não marcados da segunda.
3. Contar os nós não marcados e marcados e a partir desses valores gerar um resultado percentual de similaridade do texto do conflito

Ao final desse procedimento teríamos o valor percentual de similaridade do texto do conflito que poderia ser utilizado como uma métrica que removesse os conflitos que atingissem um determinado percentual de equivalência da contagem dos potênciais falsos negativos. Consequentemente os resultados dessa função deve fazer suas modificações no log da ferramenta de mineração, pois conflitos que

tiverem o grau de similaridade alto deverão aparecer no arquivo conflicts.equals e serem removidos dos outros dois arquivos já citados.

Uma segunda abordagem que poderia oferecer uma maior precisão na contagem dos conflitos de ordenamento, que são conflitos que corretamente aparecem apenas na abordagem não estruturada, poderia ser resolvido criando estruturas semelhantes ao funcionamento do semi-estruturado, que poderia funcionar da seguinte forma:

1. Analisar apenas os conflitos que aparecem somente na abordagem não estruturada, e para isso seria necessário o uso da função apresentada anteriormente.
2. O conflito analisado seria dividido em estrutura como classes, métodos, interfaces ou imports de forma equivalente ao que o semi-estruturado faz.
3. Caso o passo dois não consiga gerar algumas dessas estruturas será retornado um valor falso e caso o passo dois consiga gerar estruturas para todo o texto do conflito será retornado o valor verdadeiro.

Com o valor retornado por essa função seria possível saber quais dos conflitos se caracterizam como conflitos de ordenamento e em caso positivo, seria removido da conta dos potênciais falsos negativos.

As propostas feitas possivelmente teriam impacto significativo sobre o número de potenciais falsos negativos registrados nesse estudo, entretanto resta saber se com essas modificações a estimativa do cálculo tornaria-se mais precisa ou se os resultados da mineração poderia não reportar algum falso positivo real, piorando a estimativa que calcula o piso dos falsos negativos

5 Ameaças a validade

O estudo em questão não visa abranger uma alta gama de resultados, o objetivo em questão é realizar uma análise focada e cautelosa de um repositório em particular, a fim de levantar questionamentos para pesquisas futuras e propor algumas melhorias pontuais a partir dos resultados obtidos. Isso acaba acarretando algumas ameaças a validade que serão descritas nesta seção.

O repositório analisado como já informado, não contém todo o histórico de merge do projeto, pois como o projeto teve uma migração para o Git em 2013, uma parte considerável de cenários de merge podem tornar a conclusão de desempenho da abordagem semi-estruturada para esse projeto um pouco mais fraca, pois o projeto ao ser migrado encontrava-se em um estágio mais maduro de desenvoltura, o que pode ser um fator que tenha ou não um peso na análise final.

O fato de após a análise ter sido deduzido que a fórmula para calcular os potênciais falsos negativos possa vir a apresentar falhas é outro fator que põe em risco a validade desse estudo, pois como dito anteriormente a análise foi feita em cima dos resultados indicados por esse cálculo, de forma que se vier a ser confirmada a eventual imprecisão dessa fórmula o estudo pode chegar a uma conclusão diferente da realidade.

6 Trabalhos Relacionados

Vários estudos como (Kasi and Sarma, 2013) e (Brun et al., 2011) analizam casos em que, após a mesclagem dos códigos, ocorreram falhas no build ou em testes do projeto. Isso pode a representar falha humana ou falsos negativos que foram integrados a ramificação. O estudo aqui realizado procura identificar e aprimorar a forma que os falsos negativos são detectados de forma que a abordagem semiestruturada mostre seu real ganho de desempenho considerando os falsos negativos reais adicionados.

A redução apresentada no número de conflitos por esse estudo mostra que a ferramenta em alguns casos oferece ganho de desempenho no desenvolvimento, contudo conforme (Apel et al., 2011) mostra, há casos em que o FSTMerge não reduz o número de falsos negativos e mesmo que representem uma parcela pequena do dos casos não bem suce-

dido, deve ser levado em conta na avaliação final da ferramenta.

Os estudos promovidos por (Cavalcanti, Accioly, and Borba, 2015), (Kasi and Sarma, 2013) e (Brun et al., 2011) relatam a frequência e as causas dos conflitos analisados, como resultado concluindo a constância dos mesmo em diversos projetos. Esse estudo visa promover um complemento aos trabalhos apresentados e uma base para pesquisas futuras, pois os resultados aqui apresentados mostram a imprecisão no cálculo dos potenciais falsos negativos.

Se o estudo em questão levasse em consideração as propostas por (Prudêncio et al., 2012), que considera o número de operações necessárias para a resolução do conflito, os casos analisados como potenciais falsos negativos adicionais obteriam um resultado diferente do relatado por (Cavalcanti, Accioly, and Borba, 2017a) que reporta que os falsos negativos reportados apresentavam uma maior dificuldade em sua resolução.

Trabalhos como os de (Adams and McIntos, 2016) e (Henderson, 2017) mostram que algumas empresas têm preferência por abordagens de controle de versão que utilizam apenas um ramo de desenvolvimento, a fim de evitar problemas com conflitos de merge. Isso mostra que conflitos no meio corporativo é muito relevante e tratado com cautela e se algumas empresas preferem perder alguns benefícios que as ferramentas de controle de versão oferecem como múltiplas ramificações do projeto, uma outra abordagem como a semi-estruturada pode vir a se tornar uma alternativa viável ao meio empresarial

7 Conclusão

Os resultados da análise realizados nesse estudo mostraram que todos os potenciais falsos negativos registrados se tratavam de imprecisões na fórmula que detecta os falsos negativos reais. O entendimento dos casos apresentados nesse estudo abrem margem para a possibilidade de melhorias nos cálculos elaborados no projeto, pois a ferramenta de mineração atualmente realiza-os de uma forma superestimada. Uma abordagem que trate especificamente os casos conhecidos e remova-os do resultado do cálculo realizado para detectar os potenciais falsos negativos, poderá gerar uma confiabilidade maior a respeito dos falsos negativos reais adicionados, o que é um fator crucial para a disseminação da ferramenta na indústria de software

os cálculos possam ser melhorados, pois a ferramenta de mineração atualmente realiza seus cálculos de uma forma superestimada e uma abordagem que trate especificamente os casos conhecidos e remova da conta de uma forma precisa e não estimada, gera uma confiabilidade maior dos resultados, o que é um fator crucial para a disseminação da ferramenta na indústria de softwares.

O estudo em questão por diversos fatores, como tamanho da amostra por exemplo, não pôde determinar se o uso do S3M é de fato adequado para desenvolvimento de software em indústrias. O que se pode afirmar é que para esse projeto analisado não houve ganho de falsos negativos reais pela ferramenta, o que caracteriza o desempenho acima do esperado para a ferramenta. Diversos fatores podem impactar nesse resultado obtido em particular, mas faz-se necessário um estudo que traga mais repositórios a serem analisados e possivelmente um estudo focado, levando em consideração o máximo de fatores que possam vir a impactar no desenvolvimento e no resultado final do código.

8 Agradecimentos

Em especial gostaria de agradecer ao professor Paulo Borba que orientou de forma presente e atenciosa todo o estudo, a Guilherme Cavalcanti que se dispôs a ajudar e compartilhar informações sobre seus trabalhos, a empresa que cedeu seu repositório e código fonte para a análise e a todo o Centro de Informática da Universidade Federal de Pernambuco que oferece toda estrutura necessária para o desenvolvimento de estudos como esse.

Referências

Adams, B. and S. McIntos (2016). “Modern Release Engineering in a Nutshell ś Why Researchers Should Care”. In: *Proceedings of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER’16).IEEE*.

Apel, Sven et al. (2011). “Semistructured Merge: Rethinking Merge in Revision Control Systems”. In:

Brun, Yuriy et al. (2011). “Proactive Detection of Collaboration Conflicts”. In: *Journal of Systems and Software*.

Cavalcanti, Guilherme, Paola Accioly, and Paulo Borba (Oct. 2015). “Assessing Semistructured Merge in Version Control Systems: A Replicated Experiment”. In: *Proceedings of the 9th International Symposium on Empirical Software Engineering and Measurement (ESEM’15).ACM*.

– (Oct. 2017a). “Evaluating and Improving Semistructured Merge”. In: *Proceedings of the ACM on Programming Languages*.

– (Oct. 2017b). “Online Appendix of the paper Evaluating and Improving Semistructured Merge”. In: URL: `https://spgroup.github.io/s3m`.

Henderson, Fergus (2017). “Software Engineering at Google”. In: *CoRR (2017)*.

Kasi, Bakhtiar Khan and Anita Sarma (2013). “Cassandra: Proactive Conflict Minimization Through Optimized Task Scheduling”. In: *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering(ESEC/FSE’11). ACM.*

Prudêncio, João Gustavo et al. (2012). “To Lock, or Not to Lock: That is the Question”. In: *Journal of Systems and Software*.