Relevância Prática de Fatores que Contribuem para Conflito de Merge

Luís Henrique Delgado Santos
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
lhds@cin.ufpe.br

***Abstract*—Conflitos de merge, são ocorrências que atravessam gerações de desenvolvimento compartilhado. Apesar do surgimento de vários conjuntos de técnicas que tentam trazer boas práticas para a criação de software, ainda é comum a existência de conflitos e eles podem tornar-se grandes problemas na elaboração de um programa, pois além de se tornar algo frequente, isto é, acontecer mais vezes do que uma equipe espera que aconteça, fazendo parte do dia a dia de trabalho, ainda se torna necessário a dedicação maior de tempo para resolvê-los. Tempo esse que poderia ser usado em outras áreas do desenvolvimento, mas acaba por ter que ser destinado a esse fim. Este trabalho tem como objetivo observar a percepção na prática de alguns fatores apontados por estudos como possíveis contribuidores para a ocorrência de conflitos de merge, verificar se existem novos fatores encontrados na prática e descobrir quais são os fatores mais percebidos pelos desenvolvedores como ocasionadores de conflitos. Com isso, pretendemos possibilitar algumas sugestões com mais convicção para outros estudos e gerentes de projetos sobre medidas que se adotadas possam levar a equipe de desenvolvimento a ter menos conflitos de merge.**

***Index Terms*—fatores, conflitos, merge, prática, entrevista**

I. Introdução

Embora a criação de novos *branche*s e *forks* seja fácil e rápida com os sistemas modernos de controle de versão, o processo de *merge* geralmente consome tempo [1]. A palavra *merge*, como o nome sugere, significa fundir. Em computação, isso estaria ligado a fundir códigos. A necessidade de fundir códigos vem do fato do *software* ser desenvolvido, geralmente, por mais de uma pessoa. Então, para um melhor aproveitamento do tempo, cada pessoa faz uma espécie de cópia do código principal, elas trabalham em paralelo nas suas cópias, mas em algum momento é necessário juntar o código de volta. Nesse momento de junção podem ocorrer alguns problemas devido o que cada pessoa escreveu no seu código em específico, visto que cada uma pode ter escrito uma coisa diferente em uma mesma parte do código. Esse problema, explicado aqui de forma geral, é o que comumente chamamos de conflitos de *merge*. Vale ressaltar que algumas pessoas costumam chamar o processo de apenas fundir os códigos, mesmo que sem conflitos, de *merge*. Nesse estudo, utilizamos o termo *merge*, nas maioria das vezes, para nos referir a conflitos de *merge*. Os conflitos de *merge*, são ocorrências que atravessam gerações de desenvolvimento compartilhado.

Apesar do surgimento de vários conjuntos de técnicas que tentam trazer boas práticas para a criação de um *software*, ainda é comum a existência dos conflitos e eles podem tornar-se grandes problemas na elaboração de um programa, pois além de se tornar algo frequente, ainda se torna necessário a dedicação maior de tempo. Tempo esse que poderia ser usado em outras áreas do desenvolvimento, mas acaba por ter que ser destinado a esse fim. Esse tempo maior gasto com *merge* não acontece por desinteresse da área, mas pela maior dificuldade de resolver esse problema dos conflitos. Diversas abordagens já foram feitas, existindo até estudos que buscam estruturar esses conflitos para poder melhor enfrentá-los, mas nem assim parece ser um caminho fácil. Em Cavalcanti et al [2] é possível ver, por exemplo, um estudo para resolver conflitos sobre uma parte específica do estudo de *merge* que chamamos de *merge* semiestruturado.

A maioria dos estudos consiste em analisar repositórios e seus respectivos conflitos e a partir disso testar hipóteses ou fazer inferências. Tais estudos são importantes como o estudo de Leßenich et al [1] que chega a estudar até 7 possíveis indicadores (como: muitos commits dentro de um pequeno período de tempo são mais propensos a produzir conflitos do que o mesmo número de commits por períodos mais longos; mudanças maiores que modificam mais linhas de código são mais prováveis de causar conflitos do que alterações menores) que conflitos de merge podem acontecer. Apesar da importância desses estudos e seus métodos, um problema que existe é que fatores como a comunicação do time de desenvolvimento não são levados em conta, não permitindo verificar sua relvância no contexto. Estudos prévios realizados em Dias [3] sobre repositórios verificaram a relação de fatores de modularidade, tamanho e tempo na ocorrência de conflitos. Entretanto, um outro problema que existe é se na prática tais fatores são percebidos pelos desenvolvedores. Diante desses dois problemas, surgem os objetivos do nosso estudo, de teor exploratório e qualitativo, que seriam verificar a percepção dos desenvolvedores sobre os fatores já estudados na prática, observar se existem outros fatores, não perceptíveis somente pelas análises dos repositórios, mas sim pela experiência prática, que interferem nos conflitos de merge e identificar quais os fatores mais percebidos pelos desenvolvedores como causadores de conflitos.

Para atingir os objetivos desse estudo, realizamos 16 entre-

vistas com o intuito de identifcar práticas que causam/evitam conflitos de merge que estavam relacionadas com fatores de modularidade, tamanho e tempo e os novos fatores que surgissem de acordo com a conversa com os desenvolvedores. Para isso, transcrevemos as entrevistas, resumimos para o conteúdo relevante e criamos uma codificação que pudesse categorizar o conteúdo, fazer uma análise em busca das práticas e poder realizar uma relação com o tipo de fator que ela estaria relacionada. Esse processo de codificação e análise conseguiu ser aplicado em 4 das entrevistas. As demais não entraram nesse processo devido à viabilidade de escopo referente ao tempo desse trabalho se encaixar ao tempo de uma disciplina de conclusão de concurso.

Nossos resultados mostraram que foi possível identificar 4 novos fatores: estrutura de integração, comunicação, características das pessoas e codificação. Além disso foi possível observar que todos os fatores de tamanho sugeridos, sendo eles: quantidade de arquivos e linhas modificados, número de desenvolvedores e commits foram percebidos pelos entrevistados. Dos fatores de tempo: tempo de contribuição e delay entre o término de contribuições, apenas o primeiro foi percebido. E do fator de modularidade, onde se observava a interferência de trabalhar em slice vertical ou em módulos na ocorrência de conflitos, os desenvolvedores não perceberam o fator em sua experiência prática, pelo menos não de forma isolada.

Os estudos mostraram ainda os fatores mais percebidos pelos desenvolvedores como causadores de conflitos. Foram eles: tempo sem atualizar código, pessoas, abrir branches de maneira correta e modificações feitas por pessoas de outros times. Isso mostra que 75% desses fatores são relacionados aos fatores novos que nosso estudos apontaram.

O resto desse documento está organizado como segue. Seção 2 motiva nossos estudos e apresenta nossas questões de pesquisa. Seção 3 descreve nosso estudo, desde o processo de realização das entrevistas até o processo de análise das mesmas. Seção 4 mostra nossos resultados. Seção 5 mostra a relação dos nossos resultados com os trabalhos relacionados. A Seção 6 apresenta nossas conclusões.

II. Motivação

Como pode ser visto Leßenich et al [1], a maioria dos processos que envolvem a estrutura de integração como todo, são rápidos e simples de realizar. A própria questão em específico dos merges é projetada para seguir essa linha pensamento. Contudo, apesar dos estudos na área, a quantidade de questões intrísecas envolvidas por trás do processo de juntar o código são muito mais complexas que os demais processos. Apesar de podermos imaginar alguns desses fatores, não é possível garantir a abrangência dessa imaginação para um contexto maior do que o nosso.

Em seu estudo, Dias [3] analisou 20449 cenários de merge escolhidos aleatoriamente de 101 projetos Ruby e 25 projetos Python do GitHub. Todos os repositórios utilizavam estrutura MVC. Para Ruby era Rails e para Python era Django. Na análise, cada cenário de merge era analisado visando investigar a presença de fatores como modularidade (contribuições com merge e a não mudança no slice MVC que pertencem), tamanho (número de desenvolvedores, commits, arquivos e linhas modificadas em uma mesma contribuição de merge) e tempo (duração da contribuição e delay de contribuição). Estudos como o de Dias [3] servem para indicar vários nortes nas pesquisas dessa área. Isso porque além de apenas indicar fatores macro como modularidade, tamanho e tempo, eles possibilitam uma visualização melhor dos possíveis problemas por mostrarem uma granularidade maior do que pode ser a causa do conflito. Em modularidade, Dias [3] mostrou uma relação entre aumento de conflitos, a utilização de frameworks MVC e o trabalho não modular, no sentido de envolver arquivos do mesmo slice MVC. No fator tamanho, Dias [3] observa que contribuições que envolviam muitos desenvolvedores, commits, arquivos modificados ou linhas de código modificada levavam a conflitos de merge nos projetos Ruby. O mesmo acontecia para os projetos Python, exceto pelo fator linhas de código modificada. No fator tempo, é identificado que contribuição de desenvolvedores com longos períodos de duração levavam a conflitos e no quesito delay entre o término de contribuições, notou-se que o fator não remetia a conflitos.

Além do estudo de Dias [3], Cataldo e Herbsleb [4] também atentam para possíveis sugestões nas equipes de desenvolvimento. Como por exemplo, a designação de features procurar verificar estruturas de desenvolvimento do projeto para evitar algumas práticas que seus estudos apontam como possíveis causadoras de conflitos. Tais consequências mostram a passagem do âmbito do estudo do campo da importância dos mesmos para relevâncias práticas.

Motivados pelo estudo de Dias [3], mas com a curiosidade sobe a percepção dos desenvolvedores na prática sobre os fatores abordados no estudo, nós apresentamos a seguinte pergunta de pesquisa.

P1: Qual a percepção dos desenvolvedores sobre os fatores já estudados na prática?

Estudos como o de Accioly et al [5], que apesar de terem também um foco em outro tipo de conflito, abrem a possibilidade de pensarmos em novo fatores como possíveis causadores de conflitos. Se já existia uma dúvida sobre a relevância da comunicação para evitar o problema, com Accioly et al [5] é possível começar a identificar um outro fator quando vemos um de seus objetos de estudo: que tipos de mudança de código geralmente lideram a ocorrência de conflito? Motivados pelas possibilidade da existência de novos fatores e questões de relevância prática, nós apresentamos a segunda pergunta de pesquisa.

P2: Existem outros fatores que interferem nos conflitos de merge?

Com a curiosidade sobre o fator mais encontrado pelos desenvolvedores na sua experiência de conflitos de merge, surge a motivação para nossa terceira e última pergunta de pesquisa.

P3: Quais os fatores mais percebidos pelos desenvolvedores?

Sabendo os nossos problemas e motivações, passamos para a próxima Seção. Ela mostrará como foi feito nosso estudo.

III. Estudo

Para responder nossas perguntas de pesquisa, realizamos 16 entrevistas com o intuito de identifcar práticas que causam/evitam conflitos de merge que estavam relacionadas com fatores de modularidade, tamanho e tempo e os novos fatores que surgissem de acordo com a conversa com os desenvolvedores. Para realização das entrevistas, divulgamos pela internet através de e-mail e redes sociais chamadas explicando a mesma e em busca de voluntários. Após realizar um teste piloto com o nosso roteiro, fizemos alguns ajustes e começamos a entrevistar desenvolvedores. Apesar de chamar em vários momentos os entrevistados de desenvolvedores, vale ressaltar que o cargo desempenhado pelos entrevistados em suas respectivas empresas não necessariamente é esse, pode ser também gerente de projeto, gerente de configuração, líder técnico ou arquiteto de software.

O roteiro da nossa entrevista referente a esse estudo, para melhor ser entendido, pode ser divido em quatro partes. Havia uma primeiro momento com perguntas mais gerais, na qual se destacam perguntas que visavam saber práticas que causam, ou evitam, conflitos de merge, segundo a experiência do desenvolvedor, sejam essas práticas iniciativas dele ou recomendações da empresa e também perguntas que visavam saber quais práticas mais ocasinavam conflitos. A segunda parte estava voltada para a parte de modularidade. Para ser mais específico, para identificar se o entrevistado percebia se havia uma influência em trabalhar em slices MVC e a diminuir a ocorrência de conflitos. Como vimos na Figura 3, é importante destacar que nem todos os entrevistados usavam slice MVC em suas respectivas empresas. Dependendo das respostas prévias sobre dados demográficos do entrevistado e a compreensão dele sobre o assunto, poderia haver uma modificação nessa pergunta para se adaptar ao contexto de trabalho da empresa ou para explicar melhor o que se estava tentando dizer. Como, por exemplo, a mudança de termos para analisar o contrário, que seria o trabalho em módulos back-end, front-end, full-stack, etc. A Figura 1 ilustra um pouco isso. A terceira parte da entrevista visava identifcar se o entrevistado percebia a influência do fator tamanho na ocorrência de conflitos de merge. Do fator tamanho, era observado, individualmente, a quantidade de arquivos, de linhas modificadas, número de desenvolvedores e número de commits. A quarta parte procurava observar o fator tempo, mais especificamente o tempo de contribuição das integrações e delay entre o término de contribuições, como mostra a Figura 2.

Para entender o tempo de contribuição devemos pensar no tempo que um desenvolvedor leva entre pegar o trabalho de um repositório compartilhado, realizar suas modificações e devolver o trabalho para o repositório compartilhado. Para o delay entre o término de contribuições devemos pensar na data que um desenvolvedor devolve o seu trabalho para o

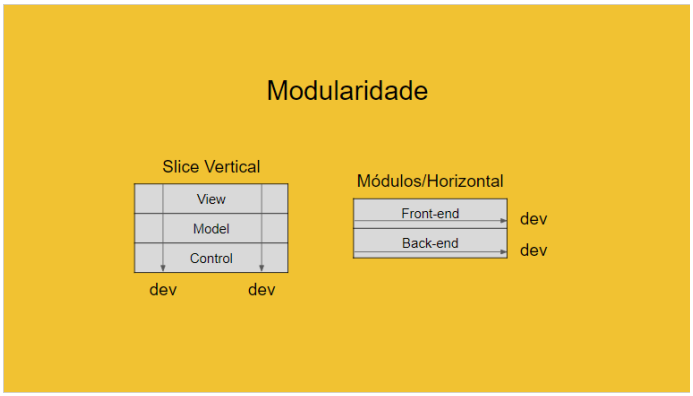

Fig. 1. Slice MVC/Vertical e Slice por Modulos/Horizontal - Modularidade

repositório compartilhado e o tempo que levou até o outro desenvolvedor mandar seu trabalho de volta para o repositório compartilhado. Essa diferença de dias é o delay. Na Figura 2 nós temos um exemplo. Nele, o primeiro desenvolvedor devolveu seu trabalho para o repositório principal no dia 10 enquanto que o segundo devolveu no dia 15, ou seja, o tempo de delay de contribuições entre eles é de 5 dias, que foi o tempo que levou entre a chegada das contribuições do primeiro desenvolvedor no repositório principal e a chegada das contribuições do segundo desenvolvedor.

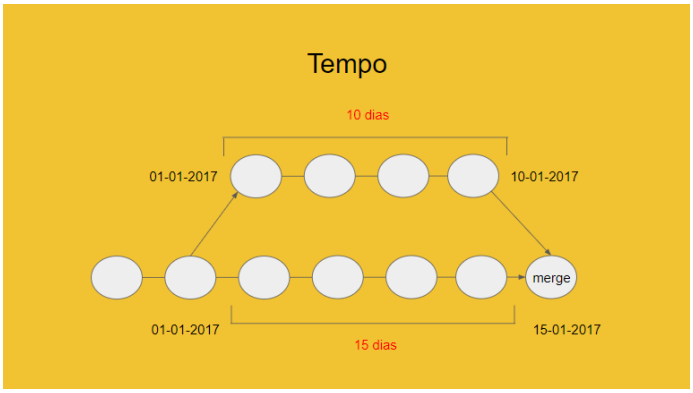

Fig. 2. Tempo de contribuição e delay entre o término de contribuições - O tempo de contribuição de um desenvolvedor na imagem foi de 10 dias. O delay entre o término de contribuições foi de 5 dias.

O passo seguinte do nosso estudo foi transcrever as entrevistas que estavam gravadas em áudio. Para cada entrevista, haviam duas formas de gravação. Uma era através de um gravador de notebook e outra era através de um gravador de celular. Assim, era escolhido a que tivesse com a melhor qualidade de áudio para ser feita a transcrição. Após a transcrição das 16 entrevistas, 4 foram selecionadas para serem submetidas aos passos seguintes. As demais não entraram nesse processo devido à viabilidade de escopo referente ao tempo desse trabalho se encaixar ao tempo de uma disciplina de conclusão de concurso. A Figura 3 mostra a caracterização dos entrevistados através de uma tabela. A etapa seguinte consistia em remover dos registros da entrevista qualquer

dado que pudesse identificar o entrevistado ou sua empresa. Para isso, foi utilizado termos como entrevistado 1, empresa 1, pesquisador 1 e assim sucessivamente. Após essa edição, foi o momento de fazer um resumo de cada entrevista para descartar trechos que não seriam relevantes para o contexto de computação como ciência, mas sem descaracterizar o sentido da entrevista, e posteriormente deu-se início a etapa de codificação.

	Cargo	MVC	Entrevista	Tam. Equipe	Equipe no Mesmo Espaço
Entrevistado 1	Desenvolvedor	Spring	Presencial	5 até 10	Sempre
Entrevistado 2	Gerente de Configuração/Líder Técnico/Arquiteto de Software/Desenvolvedor	Spring	Presencial	2 até 5	Quase sempre
Entrevistado 3	Gerente de Configuração / DevOps	Spring	Online	10 até 20	Metade
Entrevistado 4	Desenvolvedor	Não usa	Online	5 até 10	Metade

Fig. 3. Caracterização dos Entrevistados

Para a etapa de Codificação das Entrevistas, o processo foi dividido em sub-etapas. Para melhor entendimento, vamos explicar essa etapa através de um exemplo. Dado o seguinte trecho a seguir retirado de uma entevista.

"Tudo bem, mas é.. uma coisa que é muito comum ver é gente reclamando e você, quando você vai questionar. Assim, tipo, mas você tá trabalhando há quanto tempo nessa tarefa sem dar um commit, né? Ah, eu tô há dois dias. ai..tá.né....Aí você começa a buscar as causas, né?"

Vamos procurar nele alguma parte que tenha uma informação relevante sobre nosso tema. Essa é a parte de Seleção de Trecho Relevante. O exemplo pode ser visto destacado de vermelho na Figura 4.

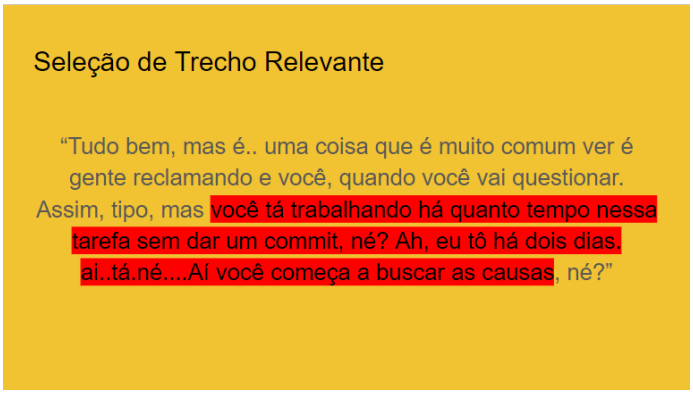

Fig. 4. Seleção de Trecho Relevante - Codificação

Dado que um trecho era selecionado, acabava-se a parte de Codificação e iniciava-se a parte de Análise de Trecho. No exemplo da Figura 5, é possível ver a primeira coisa a ser feita, criar um ID para identificar o trecho. No caso, nosso trecho vai ficar com o ID 4. Após isso, verifica-se a que Categoria aquele trecho pertence. Decidida a Categoria, que no exemplo foi: Prática causar conflito, era o momento de definir a Descrição, que é uma mensagem que serve para resumir o que tava sendo dito no trecho selecionado. Nesse exemplo, a Descrição foi: trabalhar muito tempo em uma tarefa. Por fim, caso a categoria do trecho fosse relevante para o nosso estudo, era o momento de definir o Fator. Nesse caso, a categoria: Prática causar conflito, era relevante então definimos o fator que se encaixava à descrição, que no caso foi Tempo. Mais adiante vemos quais categorias possuem relevância para o estudo.

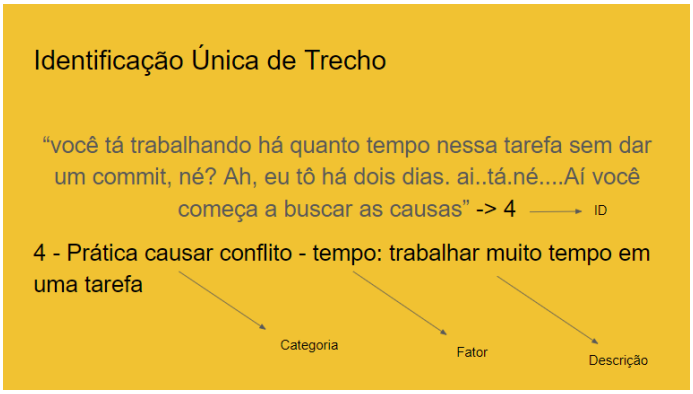

Fig. 5. Análise de Trecho - Análise

Para entender um pouco melhor a diferença entre Categoria e Fator, sugerimos olhar a Figura 6. Nela é possível ver que categoria serve para identificar o porquê um trecho é relevante. Lá também é possível observar que o fator inidica o tipo de contexto que envolve o trecho. Vale lembrar que tanto as categorias quanto os fatores não surgiram já todos prontos. Eles foram identificados a medida que as análises eram feitas. Contudo, ao final de todas as análises, obtivemos as listas mostradas. Outra ressalva importante é que apesar de termos classificados todos os trechos das entrevistas com as categorias citadas, somente as que estão destacadas em negrito na Figura são relevantes para nosso estudo. O motivo disso é que elas servem para responder nossas perguntas. Um outro alerta surge para a diferença entre as categorias: Prática causar conflito e Principal prática causar conflito. A segunda era resultado direto de um trecho referente a uma pergunta direta feita ao entrevistado e que serviria para responder nossa terceira pergunta de pesquisa enquanto que a primeira era resultado de observações sobre qualquer trecho que pudesse significar uma prática que, segundo o entrevistado, estava relacionada a causar conflito. O que era importante para as nossas primeiras e segundas perguntas de pesquisa.

Após a Análise de Trecho, foi feito o Agrupamento de Descrições. Nessa etapa, após a Análise de Trecho de toda a entrevista, procurava-se pelos agrupamentos que tinham categorias relevantes e procurava-se os que tinham descrições e fatores parecidos, como pode ser visto na Figura 7, onde os as descrições identificadas por 4 e 137 são levadas a um mesmo agrupamento, que a gente chama de Agrupamento de Descrições, e que no exemplo decidimos chamar de Tempo de Contribuição, visto que resume ainda melhor as descrições de 4 e 137. É possível notar também que o código utilizado para classificação das descrições ainda é mantido, como aparece na imagem. É observável também que várias outras descrições

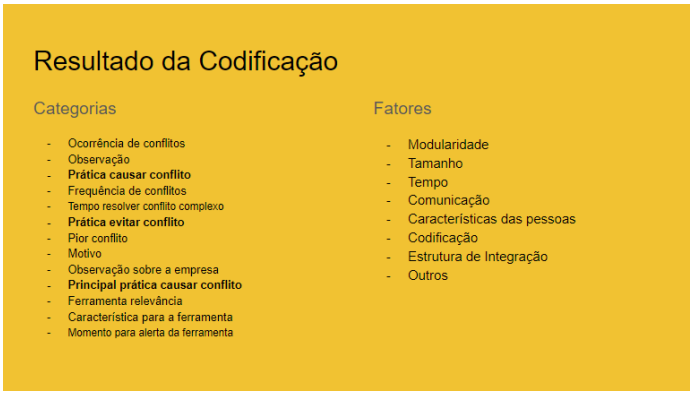

Fig. 6. Lista de Categorias e Fatores - Análise

pertencem a esse agrupamento, como as descrições 8, 36, 136. O símbolo ”+” indica que ali também estão descrições de outra entrevista, no caso, a entrevista seguinte. Uma dupla ”+ +” indica que uma entrevista seguinte não apresentou nenhuma descrição para esse agrupamento, mas que é necessário mostrar isso porque uma outra entrevista seguinte vai ter descrições pertencentes a esse agrupamento. Por exemplo, O Agrupamento aparece na entrevista 1, não aparece na dois, mas aparece na três. Então após os IDs da entrevista 1 vai ter um ”+ +” e depois vai ter os IDs da entrevista 3. Cada Agrupamento de Descrição está associado a um fator. Por exemplo, o agrupamento mostrado, Tempo de Contribuição, está associado ao fator tempo. Os números indicados no começo servem para diferenciar unicamente esse agrupamento dos outros agrupamentos do fator tempo. Então o número 2 do código 1.2, indica que ele é o segundo agrupamento de tempo do nosso estudo e o 1 indica que ele apareceu pela primeira vez na entrevista 1.

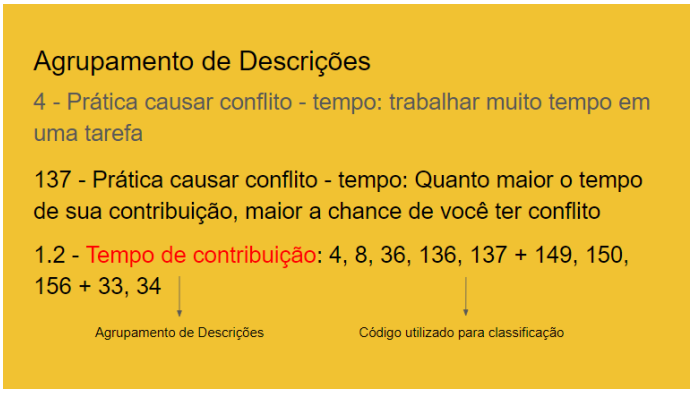

Fig. 7. Agrupamento de Descrições - Análise

Para os fatores dos estudos prévios já havia em mente que existiriam os Agrupamentos de Descrição para cada um deles. Então já sabíamos que existiriam os agrupamentos de modularidade, de tamanho e de tempo. Era necessário apenas ir nomeando, criando o identificador e mostrando as descrições que pertenciam a eles a medida que íamos analisando as entrevistas. Entretanto, para os fatores novos, assim como não se sabia das suas existências, também não havia em mente a criação dos seus agrupamentos. A medida que íamos realizando as análises e notávamos que elas podiam se relacionar com outras descrições criando uma relação maior de sentido, criávamos um novo fator e precisávamos iniciar a revisão de todas as outras enrevistas já feitas em busca de descrições que se encaixavam no novo fator e ao mesmo tempo íamos populando os Agrupamentos de Descrições desse novo fator. Por exemplo, quando observamos que descrições como: equipe por mesa se vendo o tempo todo, mesmo em países diferentes, através de televisão com câmera; comunicação o tempo todo; programar sozinho leva a não querer parar de codar pra nada; programar em par, o par avisa hora de push e commit; nós notamos que um novo fator que seria capaz de englobar todo um conjunto de descrições e agrupamentos tinha surgido, o fator de Comunicação.

Após o entendimento do nosso estudo, na próxima Seção apresentamos os resultados do mesmo.

IV. Resultados

Nessa Seção, mostraremos os resultados do estudo qualitativo. Para melhor entendimentos, ele será feito por partes, sendo elas referentes as perguntas de pesquisa.

P1: Qual a percepção dos desenvolvedores sobre os fatores já estudados na prática?

Nós verificamos que apenas os fatores de tamanho e tempo são percebidos. Para isso, observamos os Agrupamentos de Descrições dos fatores presentes nos estudos prévios: modularidade, tamanho e tempo, um por um, como mostraremos a seguir.

MODULARIDADE: Para o fator modularidade, tivemos 12 Agrupamentos de Descrições, como mostra a Figura 8.

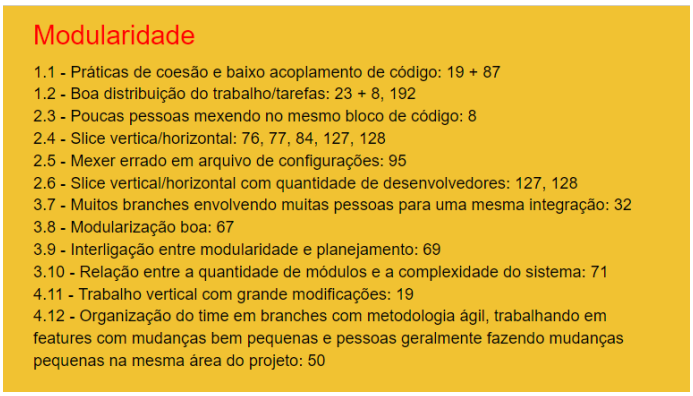

Fig. 8. Agrupamentos de Descrições - Modularidade

Entretanto, desses doze, apenas três se destacam em relação aos estudos prévios que é o objetivo dessa Seção.

- 2.4 - Slice vertical/horizontal: 76, 77, 84, 127, 128
- 2.6 - Slice vertical/horizontal com quantidade de desenvolvedores: 127, 128
- 4.11 - Trabalho vertical com grande modificações: 19

O item 2.6 e o item 2.4 são da mesma entrevista, no caso entrevista 2. Entretanto, é possível notar que são opiniões um pouco diferente do entrevistado 2. Em 2.4 ele estaria percebendo totalmente nosso fator estudado, mas em 2.6 ele está explicando melhor sua opinião e dando uma ressalva sobre sua percepeção. Nesse estudo, chamamos isso de aprofundamento da opinião do entrevistado ou simplesmente aprofundamento e essa relação nos faz poder descartar a opinião mais geral, por ela está incompleta, visto que uma mais específica está explicando melhor a opinião do entrevistado. Assim, o item 2.6 é um aprofundamento do item 2.4, então, ele é mais preciso. Por isso, para nosso próposito nessa Seção, podemos descartar o item 2.4. Dessa forma, os resultados mostram que das 4 entrevistas realizadas, apenas dois tiveram percepção sobre os nossos estudos prévios sobre modularidade e mesmo eles apresentam ressalvas sobre o fator. O segundo entrevistado fala que existe uma relação com a quantidade de desenvolvedores e o quarto entrevistado relaciona com a quantidade de modificações.

TAMANHO: Para o fator tamanho, tivemos 20 Agrupamentos de Descrições como mostra a Figura 9.

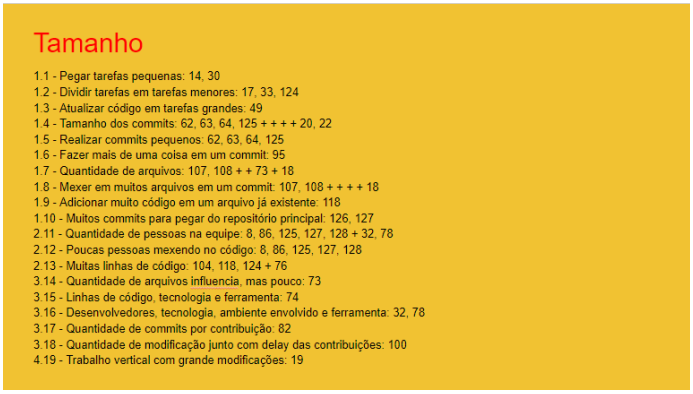

Fig. 9. Agrupamentos de Descrições - Tamanho

Para um melhor entendimento do fator tamanho, vamos chamar nesse momento todos os fatores relacionados ao tamanho como sub-fatores do mesmo. Para o sub-fator quantidade de arquivos, que faz parte dos estudos prévios de tamanho, destacam-se 3 agrupamentos:

- 1.7 - Quantidade de arquivos: 107, 108 + + 73 + 18
- 1.8 - Mexer em muitos arquivos em um commit: 107, 108 + + + + 18
- 3.14 - Quantidade de arquivos influencia, mas pouco: 73

Os item 1.8 e 3.14 são aprofundamentos do item 1.7. Dessa forma, os resultados mostram que das 4 entrevistas realizadas, três tiveram percepção sobre os nossos estudos prévios sobre quantidade de arquivos e um deles aponta uma ressalva sobre o fator. O terceiro entrevistado fala que existe a influência, mas ela é pouca. Assim, 50% do total, percebe o sub-fator.

Para o sub-fator número de linhas de código, que faz parte dos estudos prévios de tamanho, destacam-se 3 agrupamentos:

- 1.9 - Adicionar muito código em um arquivo já existente: 118
- 1.10 - Muitas linhas de código: 118 + 104, 124 + 76
- 3.15 - Linhas de código, tecnologia e ferramenta: 74

As descrições 76 e 74 do entrevistado 3 podem indicar uma dúvida sobre a opinião dele em um primeiro momento. Mas logo, nota-se que o item 74 é um aprofundamento do item 76, mostrando que essa é a opinião dele. Assim, pelo agrupamento 1.10, observa-se que apenas o entrevistado 2 concorda totalmente com a importância do sub-fator. Os entrevistados 1 e 3, concordam, mas com ressalvas. O entrevistado 1 sugere que as linhas de código são imporantes quando se está modificando arquivos já existentes, mas não arquivos do zero, enquanto que o entrevistado 3 diz que o fator está ligado a tecnologia e ferramentas de integração do projeto. Isso implica dizer que apenas 25% dos entrevistados concorda com o sub-fator.

Apesaar do pensamento mais voltado para a computação indicar que seria melhor diminuir os agrupamentos e colocar agrupamentos menores dentro dos maiores que os engloba, nossas análises vem mostrando o contrário. O motivo é que alcançando uma maior granularidade sobre uma descrição é possível entender melhor as opiniões e com isso entender mais fatos sobre os temas que abordamos, algo que não é possível, ou bastante difícil, de fazer nas análises de repositórios.

Para o sub-fator número de desenvolvedores, que faz parte dos estudos prévios de tamanho, destacam-se 3 agrupamentos:

- 2.12 - Quantidade de pessoas na equipe: 8, 86, 125, 127, 128 + 32, 78
- 2.13 - Poucas pessoas mexendo no código: 8, 86, 125, 127, 128
- 3.16 - Desenvolvedores, tecnologia, ambiente envolvido e ferramenta: 32, 78

Os item 2.13 e 3.16 são aprofundamentos do item 2.12. Dessa forma, os resultados mostram que das 4 entrevistas realizadas, apenas duas tiveram percepção sobre os nossos estudos prévios sobre número de desenvolvedores e mesmo assim, uma delas aponta uma ressalva sobre o fator. O terceiro entrevistado fala que existe a influência, mas ela está ligada a tecnologia, ambiente desenvolvido e ferramenta de integração. Assim, apenas o entrevistado 2, 25% do total, concorda com o sub-fator.

O último sub-fator que faz parte dos estudos prévios de tamanho é número de commits. Destaca-se apenas 1 agrupamento:

- 3.17 - Quantidade de commits por contribuição: 82

Dessa forma, os resultados mostram que das quatro entrevistas realizadas, apenas 25% do total, tem percepção sobre os nossos estudos prévios sobre número de commits.

TEMPO: O último fator dos estudos prévios é tempo. Para esse fator tivemos 10 Agrupamentos de Descrições como mostra a Figura 10. Para tempo tinhamos dois sub-fatores: tempo de contribuição e delay de contribuição. O primeiro tem 2 agrupamentos destacados. São eles:

- 1.2 - Tempo de contribuição: 4, 8, 36, 136, 137 + 149, 150, 156 + + 33, 34
- 3.8 - Tempo da contribuição alinhado com contexto: 95, 97

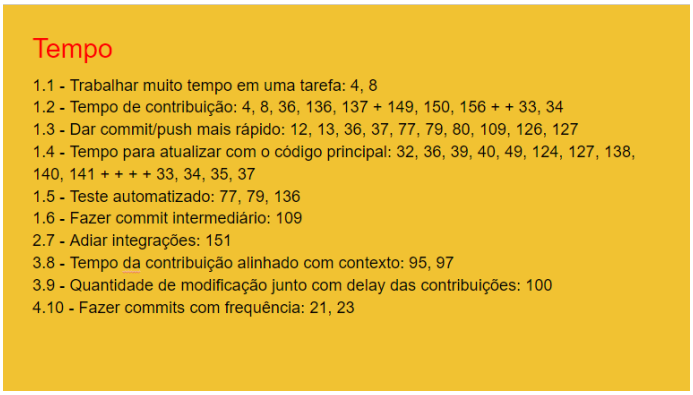

Fig. 10. Agrupamentos de Descrições - Tempo

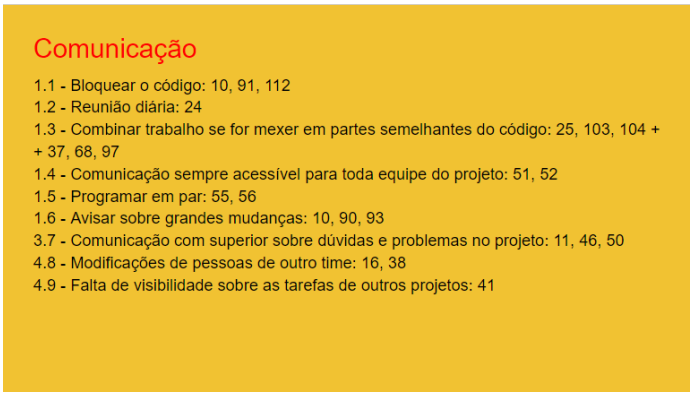

Fig. 11. Agrupamentos de Descrições - Comunicação

Isso mostra que o entrevistado 3 vê esse sub-fator como um problema, mas com a ressalva de que ele está alinhado ao contexto de desenvolvimento. Todos os outros entrevistados, 75% do total, tem uma percepção total sobre o sub-fator como influente na ocorrência de conflitos.

Para delay de contribuição, temos apenas 1 agrupamento destacado:

- 3.9 - Quantidade de modificação junto com delay das contribuições: 100

Em um primeiro momento é possível achar que o agrupamento 1.4 faz parte do delay de contribuição, mas após analisar melhor vemos que não porque esse agrupamento engloba descrições como ficar realizando pull/rebase enquanto ainda não acabou sua modificações e consquentemente não devolveu seu trabalho para o repositório compartilhado. Então, isso mostra que apenas o entrevistado 3 vê o sub-fator como um problema, mas mesmo assim com a ressalva de que ele está relacionado com a quantidade de modificação que acontece durante o tempo do delay.

P2: Existem outros fatores que interferem nos conflitos de merge?

Quando foram feitos os Agrupamentos de Descrições de práticas que causam - ou evitam - conflitos de merges, alguns dos agrupamentos do fator Outros, rapidamente, mostraram que tinham algo em comum. Isso originou o surgimento de outros fatores específicos. O primeiro a se identificar foi o fator Comunicação (Figura 11). O segundo a se identifcar foi o fator Pessoas, mas esse era muito génerico. Assim, decidimos dividi-lo em dois outros fatores: Características das Pessoas (Figura 12) e Codificação (Figura 13). Esse último, para esclarecimento, aborda descrições que envolvem escrita de código. Por fim, o último fator que notamos como agregador de várias descrições foi o fator Estrutura de Integração (Figura 14). Ele engloba descrições que envolvem formas de criar, usar branches, quando realizar merges, ferramentas de integração, trabalho na master, etc.

Olhando os resultados, é possível verificar ainda que Estrutura de Integração é um fator que está presente em 100% das entrevistas, enquanto que Comunicação e Características das

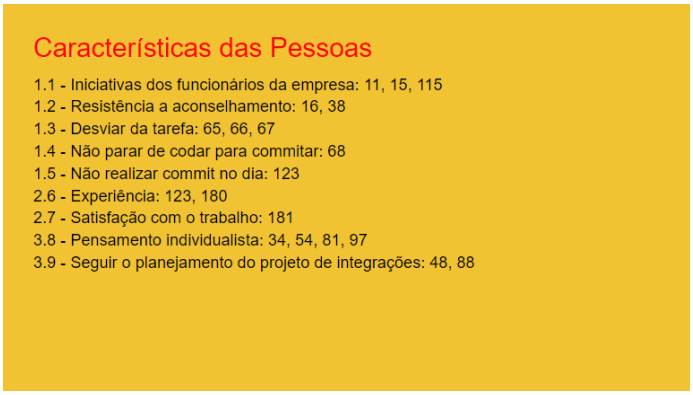

Fig. 12. Agrupamentos de Descrições - Características das Pessoas

Pessoas estão presente em 75% e Codificação em 50% das entrevistas.

Uma outra análise, permite observar que apenas 4 Agrupamentos de Descrições foram citados em mais de uma entrevista. O primeiro é do fator Comunicação, o segundo do fator Codificação e os dois últimos são do fator Estrutura de Integração.

- 1.3 - Combinar trabalho se for mexer em partes semelhantes do código: 25, 103, 104 + + 37, 68, 97

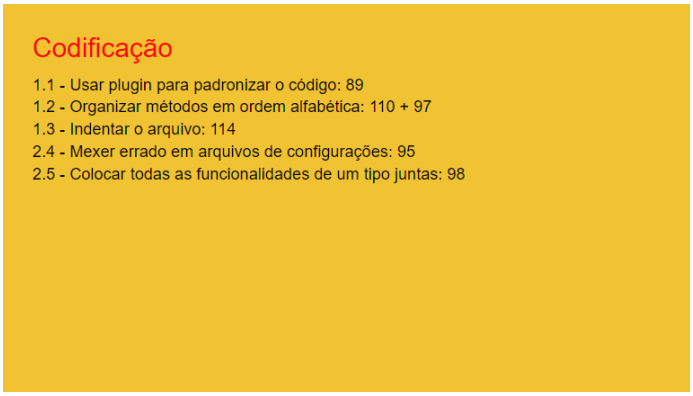

Fig. 13. Agrupamentos de Descrições - Codificação

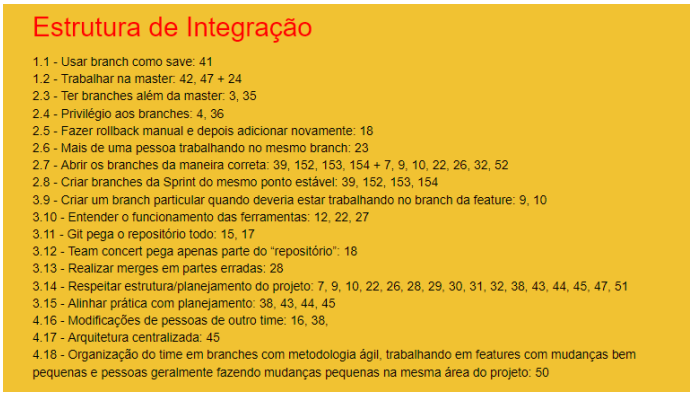

Fig. 14. Agrupamentos de Descrições - Estrutura de Integração

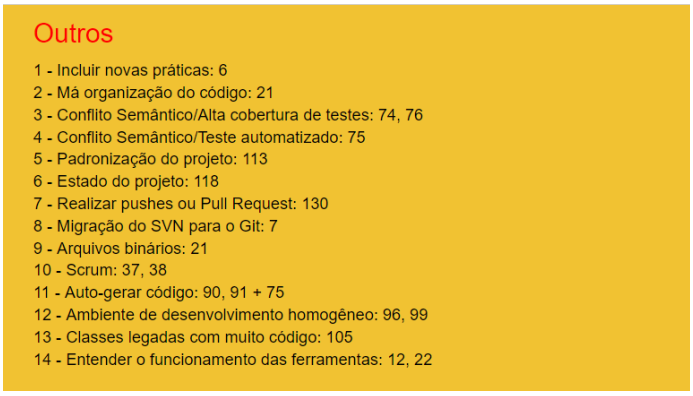

Fig. 15. Agrupamentos de Descrições - Outros

- 1.2 - Organizar métodos em ordem alfabética: 110 + 97
- 1.2 - Trabalhar na master: 42, 47 + 24
- 2.7 - Abrir os branches da maneira correta: 39, 152, 153, 154 + 7, 9, 10, 22, 26, 32, 52

P3: Quais os fatores mais percebidos pelos desenvolvedores?

Os resultados dessa Seção foram oriundos de falas diretas dos entrevistados. A Categoria que ficou responsável por englobar esse tipo de dado foi: "Principal prática causar conflito". Para o entrevistado 1, a principal causa para conflitos de merge é o tempo sem atualizar o código, que faz parte do fator tempo, mas que não estava recebendo um enfoque específico nos nossos estudos anteriores. Para o entrevistado 2, a principal causa são as pessoas. Seja a experiência em desenvolvimento, seja o quesito pressão de prazo x tempo para aprender a melhorar a qualidade do código. O entrevistado 3 alega uma causa que tem relação com Estrutura de Integração. O entrevistado 3 disse que a principal causa é abrir branches de maneira errada. Isto significa práticas como abrir um branch pessoal quando deveria estar trabalhando em uma feature branch compartilhada, por exemplo. Para o entrevistado 4, a principal causa são as modficações feitas por pessoas de outros times. Segundo ele, o fato de não ter contato com a outra equipe, mas ela mexer no mesmo repositório que ele, é a principal causa. Trata-se de uma causa ligada ao fator de Comunicação.

Apesar da constante tentativa de encontrar novos fatores que agrupassem as descrições e agrupamentos, infelizmente 14 agrupamentos não conseguiram ser encaixados em nenhum dos fatores já mencionados, nem mostraram força para originar um novo fator. Houve até uma iniciativa de criar um fator para Conflitos Semânticos, mas acreditamos que esse fugiria ao escopo do nosso projeto. Entretanto, apesar de não termos realizado um estudo a fundo desses outros agrupamentos, pois ele não se agruparam e também fugiria ao escopo analisá-los individualmente, decidimos por mostrá-los na Figura 15.

V. Trabalhos Relacionados

Nosso trabalho está relacionado com outros trabalhos sobre o tema. O trabalho de Dias [3] serviu como estudo prévio para o nosso estudo. A análise de repositórios guiou nossa entrevista, parte do nosso estudo e serviu para fazer a ponte entre os estudos sobre os repositórios e o lado prático. Como mencionado, esse estudo abordou os fatores de modularidade, tamanho e tempo. No fator modularidade, Dias [3] observou o aumento de conflitos de merge (5,98 vezes para a amostra Ruby e 4,55 vezes para a amostra Python) quando as contribuições, sobre seus respectivos frameworks MVC, não eram modular, no sentido de envolver arquivos do mesmo slice MVC. Para esse fator, os nossos estudos não apresentaram sintônia visto que o fator não foi percebido, isoladamente, pelos entrevistados. No fator tamanho, foi observado que contribuições que envolviam muitos desenvolvedores, commits, arquivos modificados ou linhas de código modificadas levavam a conflitos de merge nos projetos Ruby. O mesmo acontecia para os projetos Python, exceto pelo fator linhas de código modificada. O que mostra sintônia com o que o entrevistado 3 observou, que linhas de código influenciam, mas está relacionado com outros fatores como tecnologia e linguagem de programação. A percepção sobre arquivos modificados também foi alta. No fator tempo, tinha sido observado que contribuição de desenvolvedores com longos períodos de duração levavam a conflitos, como nossa análise também identificou o resultado e no quesito delay das contribuições, tanto o trabalho de Dias [3] como o nosso, mostraram que o fator não remetia a conflitos. Nosso estudo complementa o de Dias [3].

O trabalho de Leßenich et al [1] está relacionado com o nosso. Apesar do trabalho dele abordar mais aspectos como número de conflitos, ocorrência de conflitos e número de arquivos com conflitos a relação se deve mais aos fatores gerais que ele aborda, tendo em comum com nosso trabalho o fator tamanho. Mesmo assim, vale lembrar que nem todos os sub-fatores desse fator são vistos por ele. Além disso, outros fatores como modularidade e tempo não foram vistos por ele e nenhum dos novos fatores que surgiram: estrutura de integração, comunicação, características das pessoas e codificação.

Existe também uma relação com Cataldo e Herbsleb [4]. O que nós chamamos como fator modularidade, para esse outro

estudo, pode ser entendido de forma semelhante, mas com algumas diferenças, como fatores organizacionais. Lá, pode ser visto como a relação de criar times de feature pensando na estrutura do projeto pode ser importante para a diminuição de conflitos. Aqui, possíbilitamos que os entrevistados pudessem dizer se eles percebem isso na prática. Infelizmente, apesar dos resultados promissores de Cataldo e Herbsleb [4], aqui nenhum enrevistado mostrou uma percepção pura do fator como relevante. Além disso, os fatores técnicos mencionados por Cataldo e Herbsleb [4] são um aprofundamento de estudo também de campo modularidade sendo que para um viés mais do código em si. O nosso trabalho completa o correlato por estudar indicar outros quatro fatores nessa questão de fatores organizacionais.

Um outro trabalho relacionado é Accioly et al [5]. A relação ocorre principalmente no nosso novo fator apontado: Codificação e o objeto de estudo de Accioly et al [5] referente a: que tipos de mudança de código geralmente lideram a ocorrência de conflitos? Accioly et al [5] aponta como o principal causador de conflito, com 84.57%, as modificações dos desenvolvedores que são referentes a mesma ou várias linhas de código consecutivas no mesmo método. Isso também envolve o nosso sub-fator de tamanho que é quantidade de linhas modificadas. Entretanto, enquanto o estudo relacionado apresenta um alto percentual de impacto, só 25% dos nosssos entrevistados perceberam o sub-fator de número de linhas modificadas e 50% o fator de codificação. Entretanto, vale lembrar que o estudo de Accioly et al [5] é totalmente voltado para conflitos semânticos enquanto que o nosso, apesar de abrigar os conflitos semânticos, tem como foco os conflitos de merge. Assim complementamos o estudo por mostrar outros fatores que norteiam os conflitos de merge e que podem abrir um melhor entendimento sobre as diferenças com os conflitos semânticos.

	Com.	Estrut. Integração	Caract. Pessoas	Codificação	Modular.	Tam.	Tempo
Dias (2018)	X	X	X	X	V	V	V
LeBenich et al (2018)	X	X	X	X	X	V	X
Cataldo and Herbsleb (2011)	X	X	X	X	V	X	X
Accioly et al (2017)	X	X	X	V	X	V	X

Fig. 16. Resumo da Relação de Abordagem de Fatores pelos Trabalhos Relacionados

Existem outros trabalhos relacionados no âmbito de ferramentas utilizadas para diminuição da ocorrência de conflitos. Nossos novos fatores podem servir para aprimoramentos futuros em ferramentas como Cassandra, proposto por Kasi e Sarma [6], Palantir [7] e Crystal, proposto por Brun et al [8].

VI. Conclusões

Após analisar as 4 entrevistas, nós concluímos que existem novos fatores que devem ser levados em conta na análise de estudos sobre conflitos de merge. Isso porque todos os novos apresentaram um percentual alto de percepção por parte dos entrevistados, o que garante relevância sobre eles, visto que eles surgiram espontaneamente. O fator Estrutura de Integração chegou a ser mencionado em todas as entrevistas (100%), os fatores de Comunicação e Características das Pessoas tiveram 75% e o de Codificação teve 50%. Apesar desse e outros dados estatísticos mostrados nesse estudo, lembramos que o nosso foco é exploratório e qualitativo. Tais observações percentuais são apenas para dar uma visão como todo do processo de análise.

Outra conclusão que chegamos faz referência a percepção dos desenvolvedores sobre os fatores já estudados. Infelizmente, a maioria obteve percentuais muito baixo de percepção. Os dois único fator/sub-fator que obtiveram percentuais altos foram o de tempo de contribuição, que obeteve 75% e o de quantidade de arquivos modificados, 50%. Uma possível razão para os valores baixo é o fato desses fatores estarem mais discriminados e não serem ainda tão genéricos quantos os novos fatores que indicamos, pois quando se observa os fatores como um todo é possível notar que os fatores prévios tem mais agrupamentos e muitos dos agrupamentos aparecem em mais de uma entrevista. Uma possível segunda razão se dá a quantidade de entrevistas analisadas. Apesar de termos realizado 16 entrevistas, contando com o auxilio de Klissiomara Lopes Dias, não foi possível fazer uma análise para todas elas. O motivo foi o tempo de escopo desse trabalho diante de eventuais questões de logísticas que a realização de entrevistas traz consigo. Um segundo motivo relacionado ao tempo diz respeito a transcrição das entrevistas. Apsar do auxílio de Dias e Marcos da Silva Barreto nesse segundo momento, muitas entrevistas levaram mais de 12 horas de trabalho para serem transcritas.

Outros estudos futuros podem seguir várias aspectos do nosso estudo. Existe a possibilidade de analisar a relevância prática, além da percepção - que foi já foi feito aqui - dos fatores dos nossos estudos prévios. Um outro caminho diz respeito a fazer a análise das outras entrevistas que já foram até realizadas e transcritas. Nesse aspecto é, possível uma expansão ainda maior do processo como todo se for motivo de interesse expandir até a quantidade de entrevistas para um número bem maior. Não menos importante, uma terceira possibilidade é aprofundar o estudo e análise sobre novos Agrupamentos de Categorias que identificamos, mas que não apresentaram relação com nenhum dos fatores indicados nesse trabalho, nem com nenhum outros agrupamentos isolados. Visto que agora existe um passo a passo de estudo, testado, para a análise sobre o tema, uma última possibilidade de trabalhos futuros seria a criação de uma ferramenta para mecanizar o processo de codificação e análise.

Finalmente, acreditamos que os novos fatores encontrados: estrutura de integração, comunicação, características das pessoas e codificação, bem como o tempo de contribuição das integrações, são o ponto chave desse trabalho. Isso porque eles podem servir de novos parâmetros para os estudos futuros da área diante da sua percepção e relevância prática como fatores que contribuem para conflito de merge. Dos fatores mais percebidos pelos desenvolvedores como causadores de

conflitos, sendo eles: tempo sem atualizar código, pessoas, abrir branches de maneira correta e modificações feitas por pessoas de outros times, é importante falar que 75% desses fatores são relacionados aos fatores novos que nosso estudos apontaram. Além disso foi possível observar que todos os fatores de tamanho sugeridos foram percebidos pelos entrevistados. Dos fatores de tempo, apenas tempo de contribuição foi percebido. E do fator de modularidade, onde se observava a interferência de trabalhar em slice vertical ou em módulos na ocorrência de conflitos, os desenvolvedores não perceberam o fator em sua experiência prática, pelo menos não de forma isolada.

REFERENCES

[1] Olaf Leßenich, Janet Siegmund, Sven Apel, Christian Kästner, Claus Hunsen1. 2017. Indicators for merge conflicts in the wild: survey and empirical study. Automated Software Engineering (09 Set 2017). https://doi.org/10.1007/s10515-017-0227-0

[2] Guilherme Cavalcanti, Paulo Borba, Paola Accioly. 2017. Evaluating and Improving Semistructured Merge. Proc. ACM Program. Lang. (Out de 2017). https://doi.org/10.1145/3133883

[3] Klissiomara Lopes Dias. 2018. Understanding Predictive Factors for Merge Conflicts.

[4] Marcelo Cataldo e James D Herbsleb. 2011. Factors leading to integration fail-ures in global feature-oriented development: an empirical analysis. In Proceedings of the 33rd International Conference on Software Engineering. ACM, 161–170.

[5] Paola Accioly, Paulo Borba, Guilherme Cavalcanti. 2017. Understanding semi-structured merge conflict characteristics in open-source Java projects. Empirical Software Engineering (21 Dez 2017). https://doi.org/10.1007/s10664-017-9586-1

[6] Bakhtiar Khan Kasi e Anita Sarma. 2013. Cassandra: Proactive conflict minimization through optimized task scheduling. In 2013 35th International Conference on Software Engineering (ICSE). IEEE, 732–741.

[7] Anita Sarma, David F Redmiles, Andre Van Der Hoek. 2012. Palantir: Early detection of development conflicts arising from parallel code changes. IEEE Transactions on Software Engineering 38, 4 (2012), 889–908.

[8] Yuriy Brun, Reid Holmes, Michael D Ernst, David Notkin. 2013. Early detection of collaboration conflicts and risks. IEEE Transactions on Software Engineering 39, 10 (2013), 1358–1375.