**Fonte: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.728.2463&rep=rep1&type=pdf**

**A perspectiva de colaboração centrada no código: Evidências do GitHub**

**ABSTRATO**

Sistemas de Controle de Versão Descentralizada (dvcs) permitem equipes para estruturar o seu desenvolvimento de forma independente, permitindo um trabalho paralelo. Serviços de hospedagem on-line para dvcs tornaram-se uma opção viável para as equipes estruturarem seu fluxo de trabalho, e o GitHub é atualmente o serviço mais amplamente utilizado, hospedando mais de 10 milhões de repositórios para código aberto e projetos de softwares proprietários. Quais são os mecanismos através do qual o uso de fluxos de trabalho baseados em dvcs suporta colaboração da equipe? Em nosso estudo, nós decompor a colaboração em seus elementos de coordenação, comunicação, conscientização, divisão de tarefas e resolução de conflitos. Fornecemos evidências qualitativas de equipes usando o GitHub em termos de colaboração, e explicar o benefício vendo como cada elemento de colaboração é suportado. Nós descobrimos que usar o dvcs anda de mãos dadas com uma perspectiva de colaboração centrada no código, inspirada na auto-organização e na mediação das necessidades de comunicação e coordenação.

1. **Introdução**

A escolha de práticas de desenvolvimento tem implicações para o processo de colaboração de uma equipe, e mesmo que certas decisões sejam de natureza técnica, elas têm impacto. Por exemplo, o destino Version Control Systems (vcs) a gestão da base de código, mas andam de mãos dadas com o processo de colaboração da equipe, destacando como as contribuições dos membros da equipe são tratadas. vcs fornecem uma base para as regras e etapas que os membros da equipe devem seguir quando gerenciamento, revisão e integração de alterações nos artefatos do projeto. O fluxo de trabalho resultante descreve como o trabalho em equipe é organizado [23] e aponta quando e quais interações são necessário entre os membros da equipe, estruturando a colaboração.

Os Sistemas de Controle de Versão Descentralizados (dvcs) permitem o desenvolvimento de maneira mais independente. Membros do time ter seus próprios clones de um projeto e usar filiais para trabalhar em tarefas paralelas a outros membros, minimizando o toque pontos entre eles. Ao mudar para uma configuração peer-to-peer, os dvcs fornecem um gerenciamento de código alternativo modelo, resultando em seus próprios fluxos de trabalho [7], mas os mecanismos pelos quais a colaboração é influenciada não são estudados. Nosso objetivo é entender a escolha de uma equipe de software usar dvcs se traduz em sua colaboração.

Atualmente, o Git é o dvcs mais reconhecível e tem ganhou impulso ao fornecer fluxos de trabalho que facilitam para os desenvolvedores trabalharem isoladamente, bem como colaborar integrando seu trabalho com os colegas quando estiverem prontos. Desenvolvedores sem acesso de gravação ao sinal do repositório original eles têm mudanças em seu clone através de um pedido de pull e, após a inspeção, esses podem ser integrados por um mantenedor do projeto. Além disso, a ramificação e fusão leves permitem que os desenvolvedores com acesso de gravação para ainda separar suas atividades por finalidade, em um espaço de trabalho dedicado e mesclar com frequência.

O GitHub é um serviço de hospedagem de código baseado na web para projetos usando o Git, desenvolvendo sua funcionalidade. Fornece codehosting em repositórios públicos ou privados, e técnicos e recursos sociais para apoiar o desenvolvimento colaborativo. Estudos sobre o GitHub se concentraram em como seus recursos são usados ​​para avaliar as propriedades do desenvolvedor e do projeto [13, 21, 33], ou eles influenciam as práticas de desenvolvimento [17, 24]. Embora os benefícios do desenvolvimento sejam claros, as implicações mais sutis colaboração não foi explicitamente estudada.

Nosso estudo qualitativo tem como objetivo descrever e compreender colaboração em equipes seguindo fluxos de trabalho orientados por dvcs, usando o GitHub como proxy. Nós decompor a colaboração em elementos, rastreando onde os problemas geralmente aparecem, conforme a literatura. Observamos uma visão de colaboração centrada em código nas equipes que estudamos, e a mediação da colaboração problemas. Abstraindo de nossas descobertas, explicamos como os fluxos de trabalho de dvcs minimizam os desafios de colaboração em geral ao segmentar especificamente cada elemento de colaboração. Nossas descobertas incluem dados provenientes de organizações de software, com atividade não limitada a repositórios públicos.

1. **ANTECEDENTES E TRABALHO RELACIONADO**

A colaboração de equipes de software é composta de membros ações interdependentes, coordenadas para um objetivo comum [34]. Nós seguimos a visão de coordenação da Malone & Crowston [20] como as ações diretas ou indiretas necessárias para gerenciar interdependências. Nós sintetizamos a colaboração de uma equipe processo observando como ele lida com a coordenação divisão de tarefas, conscientização, comunicação e conflito resolução. Nós nos referimos a estes como os elementos de colaboração.

Coordenação e comunicação envolvem desafios conhecidos que resultam em falhas e tempos de resolução mais longos [4, 6, 5, 19]. Manter a consciência das interdependências também é um desafio; ferramentas podem fornecer alertas de coordenação potencial necessidades e recomendam comunicação entre desenvolvedores interdependentes [27, 29], mas não é sem sobrecarga [3]. Conflitos ocorrem mesmo na presença de ferramentas de conscientização e as equipes gastam esforços para resolvê-las. Finalmente, embora a modularização de tarefas seja uma maneira aceita de minimizar as interdependências [35], é impossível removê-las completamente.

As equipes da OSS parecem ser mais resistentes às tensões do trabalho distribuído porque são estruturadas por padrão a premissa de que a distribuição é a única forma viável de interação [10]. Mecanismos de coordenação baseados na autogovernação [11], um conjunto de ferramentas de comunicação simples [18], e uma cultura meritocrática baseada em tarefas [36] são facilitadores para produção eficaz de software de qualidade, mesmo em grande escala. Pesquisas buscaram maneiras de transferir alguns dos benefícios do OSS para o ambiente de desenvolvimento comercial por propondo frameworks que constroem sobre a estrutura, processos e cultura das equipes de OSS [30, 26].

Tem havido um número crescente de estudos focados no uso do GitHub e suas implicações para os desenvolvedores. A adição de recursos sociais aos serviços de hospedagem de código do GitHub tem sido o foco de estudos qualitativos, como esses recursos são usados ​​para avaliar a atividade e o potencial nos projetos [13, 14, 21, 33, 22]. Estudos quantitativos, examinamos a estrutura do ambiente do usuário do GitHub [32, 31] e as implicações da adoção de modelos de dvcs para o processo de desenvolvimento.

O GitHub (baseado em dvcs) está permitindo um processo de desenvolvimento diferente, gerenciando contribuições de código por meio de um modelo baseado em pull [17], com benefícios identificados em termos de processo automação e separação da construção de artefatos de a integração à base de código. O fluxo de trabalho resultante é um processo onde os membros da equipe são isolados durante a codificação atividades e interagir em ocasiões específicas do contexto durante integração. Nós antecipamos que isso tem implicações para os elementos de colaboração, por ex. a quantidade de coordenação e comunicação necessária para ocorrer na equipe.

Nós, portanto, pretendemos responder à questão de pesquisa

*Como os fluxos de trabalho baseados em dvcs dão suporte à colaboração?*

observando como cada elemento de colaboração funciona nesses fluxos de trabalho e se a colaboração conhecida os desafios persistem. Fazendo o uso do GitHub nosso proxy também nos permite aprender mais sobre como as equipes usam seus recursos para moldar ou apoiar seu processo de colaboração.

1. **DESIGN DE ESTUDO**

Como a questão de como funciona o processo de colaboração em equipes usando dvcs e seus fluxos de trabalho resultantes tem sido em grande parte inexploradas, optamos por realizar um estudo qualitativo exploratório. A colaboração em equipe é complexa para capturar e maior que a soma de ferramentas e etapas, portanto a riqueza de informações é crítica. Nós projetamos nosso estudo para começar com uma pesquisa com perguntas abertas para moldar nossas observações iniciais, seguida de entrevistas com 35 participantes da pesquisa para uma investigação aprofundada de suas práticas de colaboração. Ao longo do nosso estudo, analisamos iterativamente os dados que coletados em um processo baseado na Grounded Theory [9].

Baseamos nossa curiosidade inicial em trabalhos anteriores, que mostrado que os modelos de controle de versão descentralizados podem mudar processos de desenvolvimento [1] e que interfaces em cima de dvcs, como o GitHub, podem influenciar o comportamento do desenvolvedor [24]. Como O Git é atualmente o dvcs mais popular e o GitHub é o maior serviço de hospedagem de projetos Git, optamos por usar o GitHub como nosso proxy. Por meio da API pública do GitHub, adquirimos uma lista de usuários ativos recentemente em maio de 2013.

**3.1** **PROCEDIMENTO DE PESQUISA**

Com base em nosso interesse em como a colaboração acontece em projetos que adotaram dvcs, criamos uma pesquisa com questões em aberto. O objetivo da pesquisa foi atuar como uma ferramenta de recrutamento para entrevistas, colocando-nos em contato e nos dando uma sensação inicial de nosso público. Por isso, manteve a pesquisa curta e as perguntas simples para manter nossos entrevistados interessados ​​em continuar com uma entrevista. Suas respostas também podem ser usadas como “assunto ganchos" no caso, tivemos dificuldades em fazer com que os entrevistados fornecessem informações detalhadas. Perguntamos aos participantes por que eles adotaram o GitHub, quer o use para colaborar, como ajuda eles colaboram, como usam o GitHub para rastrear atividades de outros, e como o GitHub afetou seu desenvolvimento processo. No final da pesquisa, perguntamos aos participantes se eles seriam voluntários para uma entrevista conosco.

Depois de pilotar a pesquisa com membros de nossa pesquisa grupo, convidamos 100 usuários do GitHub para responder à pesquisa um piloto externo. Isso levou a 19 respostas (19% de resposta taxa) e quatro entrevistas piloto. Nós ajustamos nossa pesquisa depois o piloto, adicionando uma pergunta sobre dependências entre membros da equipe para avaliar as práticas de coordenação da equipe. Nós enviamos nossa pesquisa revisada para 1000 usuários do GitHub. Para esta principal pesquisa, recebemos 240 respostas (taxa de resposta de 24%).

**3.2 PROCEDIMENTO DE ENTREVISTA**

Criamos um roteiro de entrevista para conduzir semiestruturado entrevistas, permitindo-nos buscar insights adicionais. Através das respostas ao piloto externo e as entrevistas-piloto, fomos informados do fato de que alguns participantes usaram GitHub para o seu trabalho, enquanto outros usaram as ferramentas apenas para projetos pessoais e oss. Nós usamos a oportunidade para obter insight daqueles que usam o GitHub principalmente para desenvolvimento de como o GitHub é usado em um ambiente comercial, através da atividade em repositórios privados. Baseado no que nós aprendido com o piloto, ajustamos nossa entrevista roteiro para explicar a distinção entre trabalho e projetos de interesse pessoal. Dos participantes que se ofereceram para entrevistas, decidimos conversar com 35 deles. Todas as entrevistas foram gravadas e transcritas para facilitar a análise iterativa. Em média, uma entrevista durou uma hora. As perguntas da entrevista foram formuladas em torno de nossa:

*Como o GitHub suporta colaboração?*

A primeira parte da entrevista começou com perguntas gerais sobre o que o entrevistado usou principalmente para o GitHub, em qual configuração (relacionada ao trabalho ou não) e que tipo de repositórios. Em seguida, perguntamos sobre informações descritivas, conforme aplicável: situação e responsabilidades do trabalho, tamanho da empresa, tamanho da equipe e distribuição geográfica. Finalmente, perguntamos entrevistados para descrever seu fluxo de trabalho típico, incluindo interações com sua equipe e fornece um exemplo, as ferramentas e tecnologias que eles usam diariamente como parte de sua colaboração, e quaisquer desafios que eles enfrentam.

Em seguida, fazer perguntas relacionadas à maneira como a equipe manipula cada um dos elementos de colaboração. Nosso objetivo era reunir o máximo possível de informações sobre como cada equipe colabora e, portanto, se concentra especificamente apenas no GitHub no final desta parte da entrevista. Nesse ponto nós perguntou aos entrevistados para refletir sobre o que é o papel do GitHub sua colaboração, como ele suporta isso para eles, e o que benefícios ou desafios que eles notaram. Nós incluímos as perguntas usadas para esta parte da entrevista na Tabela 1 abaixo. O conjunto completo de perguntas pode ser encontrado [aqui].

Seguimos uma abordagem sistemática para organizar os dados não estruturados das entrevistas, com base nos princípios da Grounded Theory para processamento de dados qualitativos [9]. Nós agrupamos nossos dados livremente em torno dos elementos de colaboração e começamos nossa análise com codificação aberta, na qual nós códigos hierárquicos atribuídos a partes interessantes das entrevistas. O sistema de código desenvolvido em um processo iterativo, em que codificamos, discutimos códigos, códigos combinados e divididos, e escreveu memorandos sobre fenômenos mais abstratos como eles surgiu a partir dos dados. Nós então usamos codificação axial para iterar sobre os dados que coletamos, o que nos permitiu construir um responder a nossa pergunta de pesquisa.

* 1. **PARTICIPANTES**

Nossa pesquisa teve 240 entrevistados com uma taxa de resposta de 24%. 148 entrevistados (62%) disseram que usam o GitHub principalmente para colaboração, e 90 (38%) disseram usá-lo principalmente para projetos solo. Nós codificamos as respostas para a pergunta da pesquisa sobre razões para adotar o GitHub, levando às categorias na Tabela 2. Os resultados da entrevista são apresentados na Seção 4.

Para contribuir com oss ou compartilhar código 67 (28%)

Porque eles usaram o git 52 (22%)

Para colaborar com os outros 35 (15%)

Apenas como espaço de armazenamento 26 (11%)

Por causa da popularidade do GitHub 24 (10%)

Por causa da interface / facilidade de uso do GitHub 24 (10%)

A adoção do GitHub foi decidida no trabalho 8 (3%)

Outras razões 4 (2%)

Tabela 2: Razões dos entrevistados para usar o GitHub

A maioria dos entrevistados usa o GitHub principalmente para projetos de equipe relacionados ao seu trabalho (cerca de 68%), alguns usam principalmente mantendo seus próprios ou contribuindo para os projetos dos outros (cerca de 18%), e apenas poucos usam exclusivamente para hospedar projetos pessoais. 28 entrevistados (82%) disseram explicitamente que use o GitHub para colaborar com outras pessoas. Destes entrevistados, 25 (71%) eram desenvolvedores profissionais, 4 (11%) desenvolvedores e / ou gerentes, 3 (9%) eram estudantes e 3 (9%) foram desenvolvedores em estágios. No geral, embora tenhamos segmentado usuários de um site centrado em oss, a maioria nossos entrevistados eram desenvolvedores profissionais e focados sobre como eles usam o GitHub para colaborar como parte de seu trabalho. Isso também nos permitiu investigar com profundidade os processos de trabalho usados ​​em projetos privados e comerciais usando o GitHub infraestrutura - um detalhe valioso normalmente escondido da visão pública.

Os entrevistados fazem parte de pequenas equipes de desenvolvimento, principalmente 4-8 desenvolvedores. Mesmo nos casos em que os desenvolvedores estão trabalhando para grandes organizações executando vários projetos com equipes de desenvolvimento com mais de 15 membros, eles relataram que uma prática padrão é formar subequipes com um máximo de 4-5 desenvolvedores trabalhando em qualquer projeto. Todos os entrevistados estão ativos em projetos OSS, seja mantendo seus próprios repositórios públicos ou contribuindo com código para os repositórios dos outros.

No restante do artigo, apresentamos nossas descobertas (Seção 4), abstraindo-as (Seção 5.1), fornecer uma resposta nossa pergunta de pesquisa (Seção 5.2), e discutir sua origem (Seção 5.3) e implicações (Seção 5.4).

**4.0 RESULTADOS**

Nesta seção, relatamos as evidências que surgiram através analisar as respostas dos entrevistados, desencadeadas por cotações. A contribuição que reunimos se concentra em quatro áreas: a perspectiva dos entrevistados sobre a colaboração, que fornece o quadro de interpretação para suas respostas, a adoção da funcionalidade de solicitação pull do GitHub para organizar seu trabalho, a comunicação relaxada e os requisitos de coordenação do trabalho em equipe como um benefício do uso do GitHub para colaborar e os desafios que os entrevistados encontraram ao usar o GitHub.

**4.1 Trabalhando juntos por meio de trabalho independente**

Descobrimos que os entrevistados têm uma visão centrada em código do trabalho de desenvolvimento e da colaboração que entra nele; sua visão é focada e limitada pelos artefatos de código.

Embora houvesse casos em que os entrevistados também tivessem funções gerenciais que incluíam conversar com clientes, tomar decisões sobre a direção do projeto ou escolher novos recursos para implementação, a grande maioria dos participantes foi somente envolvido nas atividades de programação. Sua perspectiva e experiência é, portanto, focada nas atividades de implementação que começam após o planejamento e a tarefa de alto nível. A divisão é concluída e termina com a implantação na produção. O ponto de partida mais comum de seu fluxo de trabalho era uma lista de recursos inseridos em uma forma de sistema de tickets ou outro, e sua subdivisão subsequente para os problemas inseridos no rastreador de problemas do GitHub.

Os entrevistados veem o desenvolvimento como uma atividade de solução de problemas que cumpre uma tarefa específica e não se concentra no negócio lógica por trás ou o valor que oferecerá ao cliente. As atividades de desenvolvimento que se enquadram nessa área foram atribuídas o gerenciamento, analistas ou até mesmo designers, e não foram reconhecidos como a preocupação dos programadores ou usando o GitHub. Em alinhamento com essa perspectiva centrada em código, a visão de colaboração dos entrevistados é a de uma configuração que permite compartilhar e integrar facilmente código proveniente fontes separadas quando necessário, mas permitindo a codificação independente com o requisito mínimo possível coordenação entre programadores ou entre programadores e gerenciamento.

|  |  |
| --- | --- |
| Coordenação | O que significa coordenação para você? 🡪 Como você definiria isso?  Exemplo. 🡪 Você pode dar um exemplo de coordenação em sua equipe?  Necessidades de coordenação. 🡪 Quais são as ocasiões em que você acha essencial coordenar com sua equipe?  Problemas / Soluções 🡪 Você consegue se lembrar de um problema de coordenação? Como você resolveu isto? Você tem alguma convenção sobre a coordenação que você segue na sua equipe? |
| Divisão de tarefas | Critério 🡪 Como você decide como dividir o trabalho entre os membros da equipe? |
| Consciência | Atividade 🡪 Acompanhar a atividade em seu projeto?  Pessoas/Artefatos 🡪 você prefere rastrear a atividade de pessoas ou artefatos? Por que?  Manter a consciência 🡪 Como você está ciente de ações ou alterações nos artefatos?  Desafios / problemas 🡪 Isso é demais? Alguma coisa está faltando? |
| Comunicação | Necessidade de Comunicação 🡪 Quais são as ocasiões em que você acha essencial comunicar-se com seu time?  Desafios / problemas 🡪 Isso é demais? Alguma coisa está faltando? |
| Resolução de conflitos | Conflitos 🡪 Você se depara com conflitos? Como você os resolve? |
| Suporte do GitHub | Função 🡪 Como o GitHub se encaixa com todas essas peças de colaboração?  Benefícios 🡪 Como o GitHub ajudou sua equipe a colaborar?  Evolução do usuário 🡪 Seu uso do GitHub mudou com o tempo? Como?  Problemas 🡪 Há algo faltando? O GitHub impede alguma coisa? |

**Tabela 1: perguntas da entrevista correspondentes aos elementos de colaboração e como o GitHub os suporta.**

**4.2 Puxar pedidos para confirmar, mesclar para coordenar**

Descobrimos que os pedidos pull não apenas fornecem um modelo de desenvolvimento, mas também destacam os momentos em que a coordenação é necessária. O Git, combinado com uma estratégia de ramificação e fluxo de trabalho para gerenciar contribuições, permite aos entrevistados um alto grau de codificação independente e paralela, e o GitHub ajuda eles organizam seu trabalho, no nível individual e de equipe.

As solicitações pull foram o recurso mencionado com mais frequência pelos entrevistados, utilizados para duas finalidades: como ferramenta e modelo de acesso quando há contribuições de desenvolvedores menos experientes ou desconhecidos, e como um modelo de commit alternativo para encorajar revisões de código mais cedo do que depurar mais tarde. Quase todos os entrevistados que usam o GitHub, como parte de seu trabalho, relataram que sua empresa usa uma combinação de repositórios privados e públicos. Repositórios privados são usados ​​para hospedar o código que faz parte a vantagem comercial do projeto, enquanto partes do código são publicado como bibliotecas ou pedras preciosas e abertamente disponíveis para qualquer desenvolvedor interessado em usá-los ou estendê-los. Quase todos os entrevistados usando o GitHub para contribuir com projetos de oss relataram que as contribuições são quase exclusivamente tratadas através de pedidos pull.

*"O repositório compartilhado é ótimo se você tiver um alto nível confiança nas pessoas que têm acesso a ele, ou pessoas com quem você agora confia. eu iria prefere ter uma espécie de período probatório com puxar pedidos."* [P5]

Entrevistados tratando de direitos de acesso para o desenvolvimento A equipe falou sobre o acesso ao repositório principal como uma questão de confiança nas habilidades de programação de um colaborador. Na maioria casos, o repositório principal é o lugar de onde o código é empurrado para a produção e alguma forma de gatekeeping é necessário, que é o que o repositório bifurca e puxa solicitações, apoio, suporte. Os membros da equipe mantêm seu próprio garfo do repositório principal, trabalhe com mudanças lá e, em seguida, puxar pedido quando estiver pronto. Isso permite que o código dos desenvolvedores menos experientes seja filtrado antes de ser adicionado à ramificação de produção, seja por meio de testes ou com base no GitHub. sinal para saber se o código vai ser uma fusão limpa ou não. O mesmo princípio se aplica às contribuições recebidas de voluntários que são desconhecidos para a equipe e querem contribuir para os seus repositórios públicos.

*“Quando eu comecei todos nós empurramos para um ramo, mas o problema é que você empurra e ninguém sabe que mudanças estão passando e não há chance de uma revisão. Nosso sistema atual é aquele você não faz alterações sem enviar um pedido de [...] puxar e, em seguida, um ou mais membros vai revisá-lo e você precisa de um polegar para cima de outro membro da equipe para mesclá-lo, caso contrário, não pode entrar. "* [P13]

Mesmo nos casos em que o código de todos os membros da equipe é confiável, os entrevistados disseram que são a favor da adoção de as solicitações de recebimento como seu modelo de confirmação. Emitir uma solicitação pull sinaliza a hora da revisão do código logo que o código está pronto e combinado com testes unitários e integração contínua, permitindo que o trabalho seja realizado de forma mais rápida e automática. Os entrevistados reconheceram incorporar o processo de fazendo revisões de código ao receber uma solicitação pull como melhor prática, eles estão em processo de discutir com seus equipe ou já adotado porque eles começaram a usar o GitHub.

*“Se as coisas continuarem em um ramo de recurso e divergirem em vez de se fundirem de volta a esse bem conhecido ponto GitHub, a consequência é que fica mais difícil para coordenar quando é hora de se fundir. "[*P25]

Fazer revisões de código ao receber uma solicitação pull foi relatado como uma prática recomendada pelos entrevistados. Mesclagens são os pontos de coordenação que exigem a atenção dos desenvolvedores e adotando uma prática de revisão de código e commits frequentes, é visto pelos entrevistados como minimizando a coordenação esforço, já que fazer revisões de código em pequenos pedaços de código é relatado como mais fácil do que depurar trechos de código maiores posteriormente. Além disso, na ocasião de conflitos de mesclagem, revertendo para um antigo, versão estável resulta no menor esforço desperdiçado / perdido, por causa da janela de tempo curto entre lançamentos.

**4.3 Minimização das necessidades de comunicação e coordenação**

Descobrimos que os entrevistados utilizam a visibilidade do GitHub ofertas, bem como solicitações de pull para minimizar e direcionar suas ações de comunicação e coordenação. Necessidades reconhecidas foram perguntas e problemas durante revisões de código e mesclagens.

Os entrevistados concordam que a conscientização do progresso e o status do projeto é importante, juntamente com possíveis bloqueadores provenientes do trabalho de outros membros da equipe ou subequipes. No entanto, eles são céticos sobre a sobrecarga de comunicação e coordenação para atender a essa necessidade de conscientização.

*“Todo mundo tem a oportunidade e é diretamente informado por meio de notificações de todas as alterações para o codebase. Nós nunca tivemos uma situação em que alguém foi “uau, onde essas coisas novas vieram de "ou “uau, quem quebrou meu código" porque todo mundo é coordenado por causa dessas ferramentas. "[*P16]

O status do projeto é visível por meio do rastreador de problemas do GitHub ou de qualquer ferramenta semelhante que se integre ao GitHub. Assim como a lista de commit, e os desenvolvedores podem ficar atualizados através de e-mails de notificação. Os entrevistados estão conscientes da sobrecarga de informação devido a um alto volume de notificações provenientes de projetos grandes e ativos, embora o pequeno tamanho da equipe ajuda a manter o volume gerenciável. No que diz respeito aos bloqueadores, ferramentas de comunicação como clientes de mensagens instantâneas ou salas de bate-papo atendem ao propósito de comunicação contínua, usada conforme necessário. Essas ferramentas são discretas e fornecer um meio adicional de manter-se atualizado fornecer notificações relacionadas a problemas ou se comprometer como mensagens recebidas sem interromper o trabalho de um programador. Entrevistados comentaram sobre o esforço consciente necessário para atualizar o status dos problemas para que seja atual e visível para todos, mas visto isso como um processo mais desejável em vez de ter uma comunicação adicional direcionada para a conscientização.

*“No que diz respeito à colaboração e comunicação, GitHub a maior parte da comunicação e colaboração tem a ver com uma das duas coisas: preocupações, ou os efeitos posteriores de uma mudança. "[*P6]

As principais necessidades reconhecidas de comunicação são questões que os desenvolvedores têm para outros membros da equipe quando encontrar problemas (abordados suficientemente através de chat, e-mail ou comunicação direta, quando aplicável) e discussões em torno de um item de codificação específico. A última necessidade de comunicação é tratada com sucesso através de comentários GitHub, em qualquer um dos problemas, confirma ou solicita solicitações. Esta comunicação centrada em código é vista como um facilitador para e melhor desenvolvimento pelos entrevistados, e eles o consideram como parte integrante de seu trabalho e processo de aprendizado, e não adicionado sobrecarga.

Outra prática que modera a quantidade de comunicação e coordenação necessária é seguir uma prática de autos significação em relação à divisão de tarefas entre os membros da equipe. Os entrevistados concordaram coletivamente que, quando as atividades de desenvolvimento são tratadas entre os programadores, há menos atrito quando se trabalha e se colabora. O desenvolvimento os membros da equipe conhecem seu nível de conhecimento e experiência, e se eles podem usar esse critério ao definir e selecionar tarefas para trabalhar, a equipe está trabalhando coletivamente com a melhor divisão de tarefas. Os casos em que as atividades da equipe de desenvolvimento se cruzam com o gerenciamento de projetos ou outras equipes ou departamentos criam comunicação e desafios de coordenação, como discutido mais adiante.

Deve-se notar que os entrevistados comentaram sobre auto-organização como uma prática bem-sucedida a ser adotada a partir de o mundo oss. Eles reconheceram duas organizações espinhosas questões para a autogestão funcionarem de forma eficaz, no entanto. Uma administração precisa confiar nos desenvolvedores para ter controle sobre o que eles trabalham, o que é algo que é atingido com relutância. Em segundo lugar, o recrutamento certo é essencial, onde o peso está na obtenção de indivíduos auto motivados que funcionam dentro de um grupo, compartilhando o mesmo objetivo.

Todos os entrevistados consideram essencial a coordenação para suavizar desenvolvimento colaborativo. Sua principal preocupação com a coordenação inadequada é que ela pode levar a um trabalho duplicado devido à falta de consciência da atividade dos outros, enquanto coordenação é vista como sobrecarga. As ações discutidas anteriormente para manter um status altamente visível do projeto mitigar o perigo de uma lacuna de conscientização no estágio em que desenvolvimento está acontecendo de forma independente. Ao mesmo tempo, uma estratégia de ramificação e modelo de compromisso incorporados no o fluxo de trabalho da equipe dita as etapas do processo de que precisam seguir e limitar as necessidades de coordenação a esses pontos. Puxar solicitações sinalizam pontos de fusão e necessidades de coordenação ações em fazer revisões de código, testar e mesclar o código de volta ao ramo mestre usando ferramentas e processos específicos, quando os desenvolvedores são alertados.

Os entrevistados em geral relataram ação conjunta muito suave. Conflitos de código são raros quando se trata de fusões, principalmente porque o processo de revisão de código e a funcionalidade do GitHub permitem que conflitos ou erros sejam detectados com facilidade e cedo. Conflitos entre membros da equipe raramente ocorrem e eles estão relacionados a desacordos de estilo de código ou sugestões de soluções alternativas para o problema de programação à mão. Nas ocasiões em que houve sérios conflitos que resultou em dificuldades para o projeto, ou atrasos, aqueles eram de natureza organizacional de nível superior (por exemplo, a direção de um projeto oss) ou discordâncias com gestão sem conhecimento técnico (por exemplo, gestão insistindo no uso de ferramentas específicas ou tendo irrealistas expectativas).

**4.4 Desafios com membros não técnicos**

Descobrimos que os entrevistados não relataram desafios ao usar ou adotar o GitHub na equipe de desenvolvimento. Os desafios relatados envolveram interação com membros que operavam fora do GitHub.

*“O CMO, os outros oficiais no negócio, praticamente qualquer um que não seja técnico, eles usem o Asana para registrar solicitações de recursos. Isso é uma coisa sobre o GitHub, pode ser difícil encontrar coisas, e eu nunca pediria a uma pessoa não técnica que usasse o GitHub. Nunca.*" [P5]

*“Nós temos 2-3 biólogos, eles estão acostumados com coisas como o Dropbox e compartilhar arquivos por e-mail. Isto seria ótimo se eles criassem um ticket no GitHub e eu não teria comunicação extrapor e-mail com eles. Por outro lado, posso apreciar que a interface do GitHub para biólogos, para pessoas externas, não desenvolvedores, não é ideal "*. [P9]

Um dos desafios que os entrevistados relataram sobre usar o GitHub era ocasiões em que eles não tinham membros em sua equipe que eles têm que interagir e trabalhar com os membros da equipe que não são programadores estão relutantes em usar o GitHub, pois acham complicado, um fato que é reconhecido pelos desenvolvedores. No entanto, o problema permanece como ele adiciona ao conjunto de ferramentas que os programadores precisam usar e as fontes de informação que eles precisam para acompanhar de modo que não há perda de consciência ou um pedido não é esquecido. Enquanto não há estratégia de enfrentamento relatada para lidar com esse problema, a medida tomada é o uso de um rastreador de problemas ou de uma sala de bate-papo separada dedicada a problemas não técnicos. Usuários, onde eles podem relatar bugs ou postar solicitações de recursos.

*“Associo coordenação com problemas e trabalho isso tem que ser feito, porque é disso que eu me lembro, a parte frustrante da coordenação. Como grandes projetos de código aberto gerenciam a si mesmos de alguma forma? Não é assim tão simples, eu sei porque existem desenvolvedores até lá que fazem as principais decisões, mas porque os desenvolvedores estão falando a mesma língua que é mais fácil "*. [P19]

Outra área de desafio para os programadores é a interseção entre suas atividades e gerenciamento de projetos, que na maioria das vezes cria requisitos adicionais de coordenação e comunicação para atender às necessidades da gerência para monitorar o progresso do projeto. Isso é menos grave problema quando pelo menos um membro da equipe de gerenciamento também é um programador, mas isso é relatado como a exceção em vez da regra. Na maioria dos casos, a falta de histórico de gestores coloca pressão sobre a comunicação com os desenvolvedores em uma tentativa de obter uma visão sobre a lógica por trás de decisões técnicas. A administração não técnica especialidade ocasionalmente também resulta em ter expectativas irrealistas dos desenvolvedores, um fato que se manifesta como tensão durante a comunicação.

**4.5 Resumo**

No geral, a opinião dos entrevistados sobre a colaboração da visão mais tradicional que exige ou assume que os membros da equipe estão trabalhando em conjunto em algo [15]. A visão de colaboração dos programadores consiste em tarefas que são realizados de forma independente até que haja necessidade de interceptar ou mesclar, restringindo a necessidade de coordenação e comunicação apenas àquelas ocasiões. Esta ação independente é suportada por dvcs e fluxos de trabalho que seguir um processo claro de confirmação, revisão, teste e implantação de código. Além disso, os requisitos de coordenação e comunicação dos pontos de fusão podem ser facilmente preenchidos pelo uso de ferramentas simples. O GitHub atende exatamente a essa perspectiva de programador, fornecendo a combinação de recursos necessários para que, por sua conta, os desenvolvedores possam cumprir todas as tarefas sem criar sobrecarga organizacional. A maioria entrevistados relataram que têm ou estão dispostos a mudar suas práticas de colaboração para melhor utilizar a funcionalidade e o conjunto de recursos do GitHub. A maioria relatou que sua equipe chegou ao seu processo de colaboração atual através de tentativa e erro (ou que eles ainda estão experimentando), mas aqueles que estão seguindo os fluxos de trabalho e as estratégias de ramificação ativadas pelo Git e pelo GitHub estão experimentando mais rápido e colaboração mais suave.

Na Tabela 3, resumimos as práticas que nossos entrevistados propuseram para ajudar em seu processo de colaboração suave, conforme foram habilitados pelo GitHub correspondente característica. Discutimos mais sobre como cada elemento de colaboração é apoiado por cada prática, respondendo a nossa pesquisa pergunta, na Seção 5.2.

**5. DISCUSSÃO**

Nesta seção, adicionamos contexto às nossas descobertas, caracterizando e analisando-as. Em primeiro lugar, extraímos três temas principais, comuns entre nossos entrevistados, representando os princípios por trás de sua perspectiva centrada no código: Foco, independência e exposição. Em segundo lugar, sintetizamos os resultados e temas para responder a nossa pergunta de pesquisa. Em terceiro lugar, interpretamos nossas descobertas à luz do gerenciamento do trabalho e da tomada de decisão que ocorre em diferentes níveis de organização. Finalmente, colocamos algumas questões abertas, isso forma implicações para futuras pesquisas baseadas em nossas descobertas atuais.

**5.1 Temas identificados**

*5.1.1 Foco*

Nossos entrevistados são sensíveis sobre distrações de suas principais atividades de codificação. O foco representa as necessidades específicas atendidas pelo conjunto mínimo de ferramentas e ações.

**Ferramentas simples**. Os entrevistados descreveram situações em que suas equipes experimentaram diferentes ferramentas como parte de seu processo de colaboração, mas estão considerando ou decidimos usar alguns e simples, alinhando seu processo de colaboração com a funcionalidade do GitHub. Sua visão tem mostrado que, enquanto o desenvolvimento e comprometendo processo é claro e respeitado, eles não reconhecem uma necessidade para usar várias ferramentas e eles têm uma preferência para configurações que integram a funcionalidade em um único local. O GitHub oferece-lhes uma série de variações de fluxo de trabalho, às regras de fusão e mesclagem do Git para gerenciar a base de código, contribuições recebidas e lançamentos. Relatórios de bugs e solicitações de recursos são tratadas por meio de um simples rastreador de problemas ferramentas enquanto a comunicação é suficientemente chat clientes e / ou chats.

A simplicidade das ferramentas não parece comprometer

a capacidade das equipes de manter a consciência do trabalho que é

realizado e usando isso como um mediador para a coordenação

necessidades. Gutwin et al. [18] observaram similarmente que o uso

de comunicação de texto simples e visível foi adequada para permitir que os desenvolvedores de projetos de oss mantenham uma boa consciência

atividades de outros desenvolvedores, uma descoberta que parece

cobrir a conscientização sobre o progresso, como é evidente no nosso caso. Apesar

em [18] artefatos de trabalho não foram representados no sistema de comunicação, nossos entrevistados conscientemente optaram por usar ferramentas de comunicação que se integram com o GitHub (como Campfire, que foi o exemplo mais frequente) e permitir para eventos como fusões, confirmações, etc., aparecerem como mensagens recebidas. Isso destaca uma tentativa de adotar princípios de comunicação do ambiente oss para as equipes desenvolvimento de software comercial também.

**Interação focada**. Todos os programadores em nosso estudo mostraram afinidade com a comunicação que é focada em torno de uma preocupação específica, atividade ou artefato. Eles são frugais quando se trata de atividades de coordenação que não atendem a um propósito específico, e a maioria relatou experiências negativas com reuniões de propósito geral, ainda mais com colegas ou equipes fora da equipe de desenvolvimento. O GitHub é considerado um bom local para comunicação centrada em código, assíncrona e discreta, especialmente por meio de comentários sobre qualquer tipo de artefato. Ao mesmo tempo, a opção de ter comunicação em tempo real ou discussões documentadas é aberta através do uso de chat ou listas de discussão. O GitHub atua como o local de registro de todas as comunicações relacionadas a códigos, bem como os resultados da discussão que ocorreu em outro lugar.

O GitHub destaca quando há necessidade de coordenar as atividades entre os membros. A funcionalidade de marcação alerta os desenvolvedores quando eles foram mencionados em conexão com um problema ou commit, chamando sua atenção quando necessário para agir sobre algo. O envio de solicitações de solicitação sinaliza a necessidade de um desenvolvedor revisar o código e mesclá-lo. O modelo fork & pull é, portanto, utilizado de forma estigmática [2] semelhante ao observado em projetos oss, gerando traços de atividades que informam o trabalho de outros desenvolvedores e minimizam, assim como concentram, as necessidades de coordenação.

|  |  |  |
| --- | --- | --- |
| **Prática de colaboração em código** | **Efeito de elemento de colaboração** | **GitHub Enabler** |
| Desenvolvimento independente | Necessidades de colaboração minimizadas | Repositórios de bifurcação e ramificação |
| Processo claro e conciso de comprimento | Necessidades de coordenação minimizadas | Fluxo do git (também mencionado como fluxo de trabalho do GitHub |
| Progresso e status visíveis | Consciência | Linha de tempo, notificações, integração com o cliente de bate-papo |
| Revisões de código | Necessidades de coordenação destacadas | Solicitações completas |
| Comunicação centrada em código | Comunicação direcionada | Comentário sobre confirmações, problemas e solicitações de recebimento |
| Auto-organização | Divisão de tarefas | Acompanhamento de problemas públicos |
| Teste e implantação automáticos | Resolução de conflitos | Ganhos de serviços nas ferramentas correspondentes |

**Tabela 3: práticas de colaboração centradas em código que dão suporte a elementos de colaboração por meio do GitHub**

**5.1.2 Independência**

Os entrevistados apreciam a Independência como a dissociação de seu trabalho dos outros, o que permite que eles se auto organizem.

**Trabalho descentralizado**. O uso do controle de versão descentralizado permite que os desenvolvedores trabalhem de forma independente, que foi a razão mais citada pelos nossos entrevistados e equipes herdeiras que decidiram usar o Git ou o GitHub, bem como o maior benefício que tiveram ao fazer isso. A capacidade de ter sua própria cópia local do repositório permite que eles trabalhem em alterações, correções ou recursos por conta própria, sem serem restringidos pelo trabalho de outra pessoa ou terem que estar necessariamente conectados a uma rede para fazê-lo. Desbloquear o trabalho de cada desenvolvedor da obrigação de esperar pelos commits de outras pessoas ou fazer o checkout da versão mais recente (como no caso de versões centralizadas) significa que a velocidade de desenvolvimento é maximizada. Essa liberdade é novamente uma reminiscência de projetos oss onde os desenvolvedores são fortemente independentes em seu comportamento de contribuição.

Um fator-chave para a independência do trabalho dos membros da equipe é a visibilidade da atividade. Dabbish et al. [13, 14] destacaram a transparência do espaço de trabalho como o principal direcionador para a observação da informação, em vez de obtê-la através da comunicação, de maneira muito semelhante ao princípio da estigmatização [2]. A transparência ajuda a apoiar a independência do desenvolvimento, complementando o trabalho descentralizado com a imagem clara do projeto que é necessário para que o esforço de desenvolvimento individual não seja redundante ou duplicado.

**Baixa necessidade de gerenciamento**. Todos os entrevistados evitam a rigidez da administração ou da estrutura da equipe e favorecem a organização quando se trata de selecionar o que trabalhar e como proceder para o desenvolvimento. Ao mesmo tempo, um processo de fluxo de trabalho e desenvolvimento claro, conciso e conhecido é reconhecido como essencial. Ter um processo, mas também a liberdade de se auto organizar, permite aos desenvolvedores um nível de independência e os mantém motivados, semelhantes aos projetos de oss, nos quais o papel do gerenciamento é geralmente baixo. Os entrevistados que lideram as equipes ou CTOs também confirmaram que o objetivo deles é que o gerenciamento seja o mais leve possível e que ele tome mais a forma de uma fonte de suporte para os desenvolvedores quando eles se deparam com problemas, em vez de decidir em que eles devem trabalhar. , quando ou como. Aqueles com responsabilidades gerenciais disseram que seu principal objetivo é fazer os desenvolvedores felizes e tentar recriar a sensação de trabalhar em algo que gostam com seus amigos ("um país das maravilhas de um desenvolvedor"), que é a melhor motivação para eles, mais forte que prazos ou métricas.

Os desenvolvedores reconhecem a necessidade de gerenciamento para monitorar o progresso das atividades no projeto e fazer estimativas sobre sua conclusão. Um estado desejável relatado pelos entrevistados foi aquele em que pelo menos um membro da equipe de administração ou de negócios é instruído a usar o GitHub, para que eles possam inserir solicitações e / ou rastrear o status do desenvolvimento sem criar sobrecarga de comunicação ou coordenação.

**5.1.3 Exposição**

Dado o forte interesse dos entrevistados na colaboração, o Exposure permite que eles compartilhem facilmente o trabalho com outras pessoas e os incentive a produzir um código melhor.

Processo de contribuição fácil. O Git e o GitHub fornecem aos desenvolvedores uma maneira fácil de contribuir para uma base de código. Os projetos de OSS são altamente detectáveis ​​devido ao amplo uso do GitHub e principalmente ao primeiro lugar nos resultados dos mecanismos de pesquisa. Uma vez descoberto, todo o conteúdo do projeto é altamente visível e seu progresso é refletido no rastreador de problemas, confirmações, etc. Os desenvolvedores podem prosseguir com o projeto, trabalhar nele localmente e enviar uma solicitação pull com suas alterações, um processo que tem foi reconhecido como diminuindo as barreiras à contribuição [22].

Para projetos comerciais, conforme relatado em nosso estudo, a mesma facilidade de uso beneficia o trabalho que ocorre em repositórios públicos, onde as contribuições externas podem variar de corrigir um erro de digitação em uma página wiki até contribuir com um patch ou comentário. Isso maximiza a exposição que a organização e o projeto comercial podem receber enquanto capitaliza os commits provenientes de desenvolvedores externos à equipe.

Pressão dos pares. Devido à exposição e visibilidade oferecidas através do GitHub, desenvolvedores, embora inclinados a contribuir mais, estão mais conscientes da qualidade do código que enviar. Entrevistados contribuindo para projetos de oss relatados que em muitos casos eles terão seu código em um privado repositório por algum tempo trabalhando em melhorias antes eles tornam isso público. Em projetos comerciais, os desenvolvedores são também mais cuidadosos porque há mais propriedade visível de código, muito parecido com projetos OSS, e é fácil de detectar quando alguém comete um código que introduz um erro ou é de baixa qualidade. Nessa linha, as equipes também adotam a prática de commits menores e mais frequentes, como mencionado anteriormente. Isto, portanto, é possível que essa visibilidade resulte indiretamente em software de alta qualidade sendo produzido como também em [13].

Na Seção 4, observamos como as equipes de desenvolvedores usam GitHub e dvcs na prática e os benefícios relacionados que eles ter experimentado. As práticas e fluxo de trabalho que eles seguem revelou uma perspectiva de colaboração deslocada, com os temas identificados acima representando os princípios sobre os quais opera. Abaixo, interpretamos como e por que esses princípios trabalho, olhando para as condições do ambiente colaborativo.

**5.2 Como os fluxos de trabalho baseados em DVCS suportam colaboração?**

A perspectiva centrada em código que nossos entrevistados expressaram tem como alvo o núcleo da colaboração. Para os desenvolvedores, **a colaboração é equivalente a gerenciar contribuições independentes para o todo comum**. Embora as contribuições individuais devam ser trabalhadas de maneira descentralizada e independente, um fluxo de trabalho que consiste em etapas claras e concisas para aceitá-las, analisá-las e integrá-las é crucial para o gerenciamento de contribuições. Esse senso de colaboração dissociado parece resultar em um comportamento colaborativo que beneficia os elementos de colaboração.

Em vez de gastar esforços para manter a conscientização sobre as ações de todos os desenvolvedores ou todas as alterações feitas, para garantir que o subconjunto de que eles precisam estar cientes esteja coberto, nossos entrevistados optam pela **conscientização do progresso e do status do projeto por meio da visibilidade**. Em um ambiente tão transparente [14], os desenvolvedores podem seguir as informações que julgarem relevantes, sem sobrecarga de comunicação ou gerenciamento, e de maneira discreta. Para manter esse estado visível, no entanto, todos os membros da equipe são individualmente responsáveis ​​por sinalizar o status de seu trabalho.

Em um fluxo de trabalho que suporte o trabalho descentralizado, as necessidades de **coordenação são minimizadas e focadas em fusões**, como pontos que requerem ação conjunta. Menor necessidade de coordenação também requer menos comunicação como o veículo. Isso não significa que a equipe perca o fluxo de informações ou o relacionamento entre seus membros. **A comunicação é centrada em código e focada em artefatos específicos** e é arquivada para fornecer registro de decisões e soluções. O benefício é a comunicação e coordenação que é suave e eficiente, porque é direcionado.

Os princípios do fluxo de trabalho descentralizado permitem que conflitos sejam detectados na revisão de código antes de serem mesclados ou manipulados automaticamente por meio de ambientes de teste e integração contínua. Isso transforma a **resolução de conflitos de uma preocupação de grupo em uma tarefa individual** que é capturada quando o processo seguido destaca a necessidade de fazê-lo. Devido à auto atribuição e propriedade de tarefas, as alterações de código necessárias para resolver conflitos são, por padrão, de responsabilidade do desenvolvedor original. A auto atribuição baseada na experiência otimiza o desempenho coletivo do grupo, já que os membros trabalham em tarefas ou corrigem erros de uma maneira que aproveita suas habilidades. O suporte para cada elemento de colaboração está resumido na Tabela 3, respondendo à nossa pergunta de pesquisa.

A combinação de um fluxo de trabalho como um processo estruturado para a equipe e a manutenção da consciência através da visibilidade, concorda com o conceito de coordenação contínua [25], um paradigma sob o qual as práticas de trabalho são estruturadas e flexíveis. A estrutura é traduzida em um protocolo especificado para gerenciar alterações de código e fornece pontos de verificação de sincronização de trabalho independente. A flexibilidade se baseia na transparência do trabalho, permitindo a auto-organização. Embora a coordenação contínua seja assumida como sendo servida por ferramentas sofisticadas [28], nossa evidência sugere que os princípios de coordenação contínua podem ser alcançados pelo uso de ferramentas muito mais simples, nos incentivando a repensar sua contribuição em relação ao poder do processo.

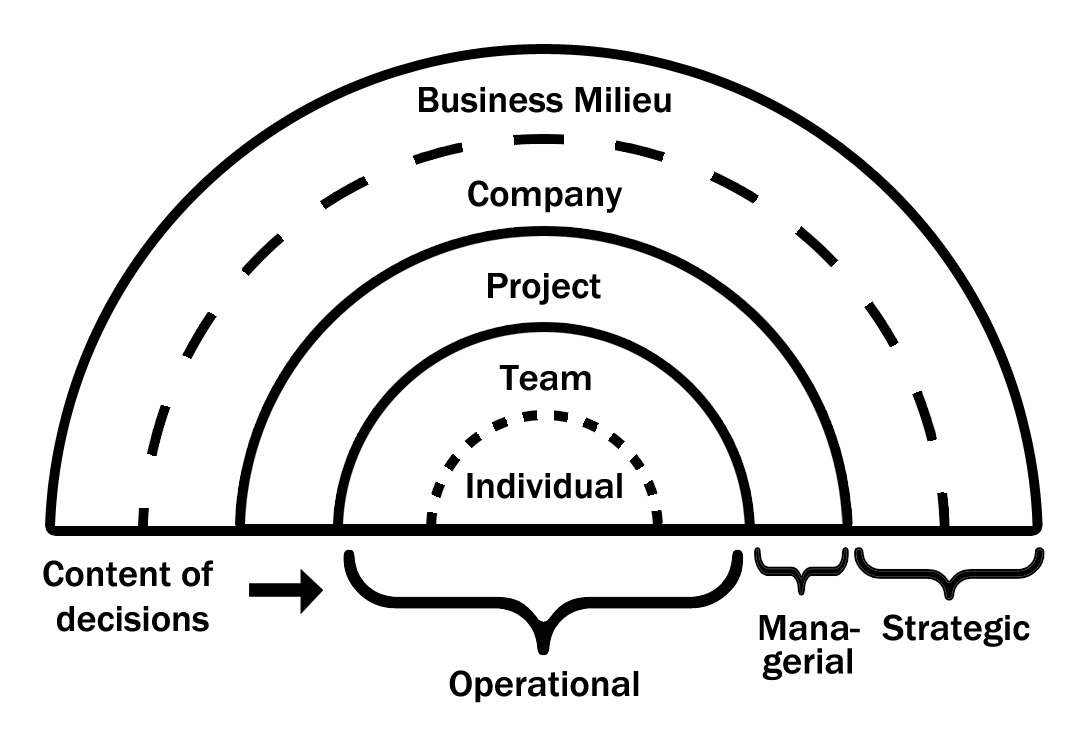
A aplicação dos princípios que identificamos para ver e praticar a colaboração parece simplificar o processo de colaboração. Esse é um efeito importante quando consideramos o quão caro é (em termos de tempo e dinheiro) criar equipes e experimentar a combinação de ferramentas e práticas que lhes permitam colaborar de forma eficiente. As evidências relatadas apontam para os benefícios que podem ser esperados se a perspectiva de colaboração for mudada.

**5.3 Níveis de decisão e mentalidade semelhante à OSS**

Para o resto da nossa discussão, olhamos para o projeto de desenvolvimento ou organização como uma estrutura estratificada de organização do trabalho e tomada de decisões, semelhante ao Curtis et al. modelo comportamental em camadas [12], adaptado como mostrado na Figura 1. Na base desta estrutura é a camada operacional, preocupado com as decisões e tarefas que são altamente de natureza técnica e foco na implementação de soluções. Na organização de equipes de programadores, as decisões técnicas podem ser individuais em termos do melhor caminho para a solução de problemas, mas também envolvem o restante dos membros da equipe quando se trata do uso de ferramentas e processos específicos para sua colaboração. Um nível acima é a camada gerencial, focada na descrição de problemas e no design de soluções, olhando o projeto como um todo. A camada gerencial também é responsável por decisões que permitem que a camada operacional funcione de maneira eficiente, tendo os recursos e processos necessários. Dependendo do tamanho da equipe, empresa ou organização, uma camada adicional pode estar em vigor acima da gerencial, preocupada com decisões estratégicas relativas à visão e direção da equipe dentro da organização, ou ter uma visão mais holística da organização e da organização. sua função e posição dentro de um ambiente corporativo mais amplo.

As evidências que coletamos provêm de desenvolvedores que fazem parte do nível operacional e estão preocupados com as porcas e parafusos da implementação das decisões do nível acima deles. No nível operacional, portanto, observamos que existe uma perspectiva de colaboração em que o trabalho é organizado de forma que possa ocorrer de forma independente, gerando benefícios em termos de aliviar parte da tensão que as equipes de desenvolvimento colaborativo frequentemente experimentam. Esses princípios de colaboração seguidos no nível operacional estão alinhados com a opção de usar dvcs e GitHub e são influenciados pela cultura OSS e organização do trabalho.

Ao perguntar por que esses princípios funcionam, devemos considerar os seguintes fatores:



**Figura 1: Camadas de decisão em nossa análise, de [12]**

• No nível operacional, a natureza do trabalho permite independência e auto-organização. Algumas decisões que pertencem às camadas estratégicas e gerenciais se beneficiam de vários pontos de vista e brainstorming, como a visão e a identidade da organização, a lógica de negócios por trás do software desenvolvido ou o valor que ele fornecerá ao cliente. Essas decisões também envolvem conhecimentos que não fazem parte do conjunto de habilidades dos desenvolvedores. As tarefas no nível operacional são altamente modulares e os desenvolvedores as dividem em questões que são claramente definidas, são inseridas no rastreador de problemas do GitHub e pode ser selecionado por membros da equipe.

• No nível operacional, os membros estão familiarizados com a ideia de trabalhar dessa maneira e compartilhar a mentalidade por trás disso. Os princípios da perspectiva codecêntrica da colaboração são fortemente influenciados pela forma como o trabalho é organizado e executado em projetos de OSS, com os quais nossos entrevistados têm muita experiência. Como os próprios desenvolvedores da oss, eles são autônomos e compartilham a mentalidade de auto-organização, minimizando as necessidades de comunicação e coordenação, possuindo a propriedade do código e operando em um modelo meritocrático e especializado [16].

Os casos de entrevistados com função gerencial confirmaram os benefícios e a perspectiva deslocada. Esses eram, no entanto, gerentes que também eram programadores, usando o GitHub para contribuir com código, bem como para manter uma visão holística do status e progresso do projeto para gerenciar recursos e dar suporte à equipe de desenvolvimento. Os desafios que ocorrem em interação com o gerenciamento, mencionados na Seção 4.4, são casos de gerentes que não são usuários do GitHub. No entanto, os desafios podem não ser devido à falta de uso do GitHub, mas à diferença de mentalidade e perspectiva de colaboração que os dois grupos / níveis possuem. No nível operacional, as fontes de inspiração dos programadores são dvcs, apoiando o trabalho descentralizado e oss, seguindo práticas de auto-organização, ambientes que geralmente não são relacionáveis ​​com membros do nível gerencial ou estratégico.

Estudos já examinaram os processos que se mostraram eficazes para projetos de oss e como eles podem ser implementados e beneficiar organizações tradicionais. Sharma et al. [30] examinaram os projetos oss baseados em uma estrutura de estrutura, processo e cultura. Sua proposta era a criação de um ambiente híbrido na organização que funcionasse com profissionalismo técnico, autogovernação, construção de comunidade e confiança. Mais recentemente, Riehle et al. [26] contrastaram os princípios da meritocracia e da auto-organização com os princípios de software de código fechado de status de recompensa em vez de mérito e imposição de processo. A sugestão deles era que a indústria de software deveria capitalizar no paradigma oss para trazer benefícios da colaboração aberta interna. Eles propuseram um esforço de canalização para ter colaboradores motivados, permitindo-lhes espaço para ação voluntária e promovendo melhor qualidade através do escrutínio quase público.

Nosso estudo traz evidências de como o paradigma de trabalho do oss é atualmente utilizado em um cenário comercial e quais são os benefícios potenciais. Dado que chegamos a este observando o uso de ferramentas específicas e a percepção de que essas observações se aplicam em um nível de trabalho de desenvolvimento, abaixo, colocamos algumas questões sobre a possibilidade de transferência e adoção de nossos resultados além do ambiente em que os observamos.

5.4 Implicações para o trabalho futuro

Com base em nossas descobertas, vimos que, no nível operacional de desenvolvimento, aplicar uma perspectiva de colaboração centrada em código, com base nos princípios da organização e gerenciamento do trabalho, ajuda a lidar com algumas dificuldades colaborativas conhecidas. No entanto, diferentes partes do ciclo de desenvolvimento ou níveis de tomada de decisão podem ter diferentes requisitos de comunicação e informação. Portanto, usamos o entendimento obtido ao descobrir como as equipes no nível operacional usam o GitHub, o paradigma de trabalho do oss, e pensamos em colaboração, para colocar outras questões.

uma. Quais são os fatores que fazem com que esse paradigma seja usado com sucesso? Nós identificamos alguns que estão intimamente ligados a ferramentas e ambientes de desenvolvimento, mas

mais pesquisas são necessárias para ver se existem outras, adotando uma visão mais holística das atividades de desenvolvimento.

b. Os princípios por trás da colaboração centrada em código são transferíveis para outros níveis?

Eu poderia um modelo focado em especialização que incentiva o trabalho independente e a auto-organização, fornecer o mesmo benefício lá?

As decisões tomadas nos níveis de decisão gerencial e estratégica estão mais longe da implementação e envolvem mais criatividade e abstração. Tais atividades geralmente se beneficiam do brainstorming e da discussão de ideias, o que parece correr contra os princípios dos membros trabalhando isoladamente. As vantagens e desvantagens de usar este modelo para organizar o trabalho precisam ser investigadas.

ii. A aplicação desses princípios depende do uso de ferramentas específicas? Poderia (deveria) o GitHub integrar-se com outros sistemas de gerenciamento de artefatos usados ​​em outros níveis?

Muitos dos entrevistados relataram desafios em conseguir que membros não técnicos adotassem o GitHub. Isso significa que, se a ferramenta não pode ser adotada, a perspectiva de colaboração em que ela opera não pode? Há espaço para ferramentas usadas em outros níveis para se integrar ao GitHub como meio de colher algumas das recompensas experimentadas no nível operacional?

iii. como as atividades gerenciais seriam afetadas se esse modelo de trabalho fosse adotado em todos os níveis? Quais seriam os trade-offs potenciais?

Central para a perspectiva de colaboração centrada em código é a noção de membros da equipe concentrados em tarefas que utilizam seus conhecimentos, enquanto confiam que os outros façam o mesmo de forma independente. O mesmo pode funcionar para níveis mais altos se a gerência estiver disposta a adotá-lo como um ponto de vista, aplicando os princípios não apenas horizontalmente, dentro do mesmo nível, mas também verticalmente, nas interações entre os níveis. A aplicação horizontal significaria que os membros dos níveis gerencial e estratégico se concentram nas tarefas que utilizam seus conhecimentos, enquanto separam suas preocupações de outros níveis. A aplicação vertical significaria que há menos necessidade de atividades de monitoramento e controle. Espera-se que o papel da gerência passe a apoiar o nível operacional, seguindo uma prática de confiança, visibilidade e recrutamento correto.

iv. A perspectiva está dirigindo a ferramenta ou vice-versa?

Certas capacidades técnicas permitem diferentes formas de organizar o trabalho individual e coletivo e, por extensão, a colaboração dos membros da equipe. No nosso caso, o uso de dvcs anda de mãos dadas com um alto grau de trabalho independente e uma visão diferente da colaboração. Pelo mesmo princípio, no entanto, pode-se argumentar que as necessidades colaborativas e as visões das equipes mudam, abrindo a arena para a criação e aperfeiçoamento de ferramentas, como a absorção da mentalidade de trabalho oss que presenciamos. Embora haja evidência de um relacionamento, é difícil identificar qual é a força motriz, já que os vemos evoluindo normalmente ao mesmo tempo. Um exemplo do mesmo princípio é visto quando interpretamos a Lei de Conway [8], onde a presença de uma relação holomórfica entre o sistema e a organização de projeto não responde à questão de qual estrutura procedeu a outra.

**6. CONCLUSÃO**

Neste artigo, apresentamos um estudo explorando o uso do GitHub por equipes de desenvolvimento colaborativo, no ambiente comercial e oss. Nós relatamos os mecanismos através dos quais o GitHub (construindo capacidades de dvcs) suporta cada um dos elementos de colaboração, respondendo assim às nossas questões de pesquisa de como ele suporta a colaboração em geral. Nossas evidências (Tabela 3) mostram que os principais efeitos parecem estar na minimização e na concentração da necessidade de coordenação, apoiando o objetivo do GitHub de promover "colaboração sem coordenação direta" 2.

Nosso trabalho oferece três contribuições. Primeiro, vimos como o GitHub é usado em projetos e empresas de software comercial, investigando atividades que estão além da análise de eventos públicos do GitHub. Descobrir como as equipes usam o GitHub para realizar suas atividades colaborativas revelou seu fluxo de trabalho e colaboração com menos atrito. As equipes que entrevistamos têm formulado práticas de trabalho em torno das ferramentas e recursos disponíveis, utilizando a visibilidade de seu ambiente colaborativo para progredir em seu trabalho individual e minimizando o esforço explícito de coordenação pela adoção de solicitações pull como sinais dos próximos pontos de sincronização.

Segundo, através de uma análise qualitativa sistemática das entrevistas, identificamos os princípios por trás dos benefícios: *Foco, Independência e Exposição*. A visão codecêntrica de colaboração dos programadores vai em uma direção diferente da visão mais tradicional de colaboração como trabalho conjunto, mas, no entanto, mostrou sinais claros de benefícios associados a ela. A inspiração para estes princípios de independência, trabalho auto organizado está fortemente relacionado com a mentalidade do ambiente de desenvolvimento oss e o uso de dvcs para gerenciar uma base de código colaborativo.

Abstraindo da perspectiva centrada em código, adotamos uma visão estratificada da tomada de decisão no ciclo de vida do desenvolvimento. A visão centrada em código é equivalente a um modelo de organização de trabalho que é baseado em conhecimento especializado e permite uma separação de interesses entre camadas de decisão que exigem conhecimentos diferentes. Como a adoção de um modelo de organização do trabalho descentralizado e do tipo oss está se mostrando bem-sucedida e benéfica no nível operacional de projetos de desenvolvimento comercial, contribuímos para futuras pesquisas, levantando questões para avaliar sua aplicabilidade em outros níveis de decisão.

**7. AGRADECIMENTOS**

Agradecemos gentilmente a todos os participantes do estudo por dedicarem tempo para nós fornece informações e insights valiosos. Sem eles, este estudo não teria sido possível.