Agentes de Codificação de IA como Prática Colaborativa: Implicações para Equipes de Desenvolvimento de Software

Lucas Feksa Hickmann
Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, Rio Grande do Sul, Brazil
lucas.fhickmann@inf.ufrgs.br

Barbara Hoch
Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, Rio Grande do Sul, Brazil
barbaradjhoch@gmail.com

Rafael Parizi
Instituto Federal Farroupilha (IFFar)
São Borja, Rio Grande do Sul, Brazil
rafael.parizi@iffar.edu.br

Leticia dos Santos Machado
Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, Rio Grande do Sul, Brazil
leticia.machado@inf.ufrgs.br

Resumo

Agentes de codificação baseados em Modelos de Linguagem de Grande Escala (LLMs) vêm transformando a engenharia de software ao atuarem de forma autônoma em fluxos colaborativos antes exclusivos de equipes humanas. Embora a literatura os avalie predominantemente por métricas técnicas, como qualidade do código e acurácia, seus impactos sobre a colaboração permanecem pouco compreendidos. Este artigo investiga como esses agentes reconfiguram práticas colaborativas em equipes de desenvolvimento de software, com foco em comunicação, coordenação, cooperação e workspace awareness. Para isso, realizamos um estudo qualitativo exploratório baseado na análise temática de dez threads públicas de pull requests no GitHub analisadas à luz do Modelo de Colaboração 3C e da teoria de Workspace Awareness. Os resultados mostram que agentes de IA assumem papéis colaborativos, reconfigurando mecanismos de interação e awareness, e apontam oportunidades para ampliar modelos colaborativos para a Engenharia de Software de sistemas agênticos.

Abstract. Coding agents based on Large Language Models (LLMs) have been transforming software engineering by acting autonomously within collaborative workflows once exclusive to human teams. Although the literature evaluates them predominantly through technical metrics, such as code quality and accuracy, their impact on collaboration remains poorly understood. This paper investigates how these agents reconfigure collaborative practices in software development teams, focusing on communication, coordination, cooperation, and workspace awareness. To this end, we conducted an exploratory qualitative study based on the thematic analysis of ten public pull request threads on GitHub, selected through systematic screening and analyzed in light of the 3C Collaboration Model and Workspace Awareness theory. The results show that AI agents come to act as participants in collaboration, reconfiguring roles, practices, and awareness mechanisms, and point to opportunities for extending collaborative models in Software Engineering toward agentic systems.

Palavras-chave

Agentes de codificação de IA, Colaboração humano-IA, Modelo de Colaboração 3C, Engenharia de software

1 Introdução

A adoção de sistemas baseados em Modelos de Linguagem de Grande Escala (LLMs) remodelou as práticas de engenharia de software nos últimos anos, com o surgimento notável de agentes de codificação de IA, que combinam percepção, planejamento, memória e integração de ferramentas para executar de forma autônoma tarefas antes reservadas a desenvolvedores humanos [11, 22]. Esses agentes têm sido aplicados em atividades que vão da geração de código e automação de testes até a elicitação de requisitos e a criação de artefatos centrados no usuário [2, 3]. Sua adoção crescente levanta questões não apenas sobre desempenho técnico, mas também sobre como alteram o tecido social e organizacional das equipes de desenvolvimento de software.

As avaliações recentes sobre agentes de codificação baseados em LLM concentram-se quase exclusivamente em métricas técnicas, como taxa de *merge*, qualidade e acurácia da tarefa [11, 23]. A dimensão colaborativa permanece pouco examinada no contexto de agentes autônomos de codificação e com a literatura carecendo de estudos empíricos que investiguem seu uso colaborativo no mundo real. Como a introdução de agentes reconfigura a comunicação entre membros da equipe, quem os supervisiona, como profissionais mantêm consciência da atividade paralela dos agentes, e como a confiança em artefatos gerados por agentes é negociada?

Este artigo investiga como agentes de codificação de IA reconfiguram práticas colaborativas em equipes de desenvolvimento de software e outras atividades de software nas quais tais agentes têm sido propostos [1]. Para alcançar este objetivo, realizou-se um estudo qualitativo exploratório por meio de análise documental e observação de interações públicas entre desenvolvedores e agentes no GitHub, reunidas por um pipeline documentado e replicável em um corpus de dez threads de pull request (PR, a unidade de contribuição do GitHub em que um autor propõe alterações de código para incorporação a um repositório) selecionadas por triagem sistemática, e analisadas pelo Modelo de Colaboração 3C [6, 7] e pela teoria de Workspace Awareness [10]. O artigo oferece caracterização preliminar de como a introdução de agentes reconfigura comunicação, coordenação, cooperação e *awareness*, evidencia tensões colaborativas não capturadas por avaliações técnicas, e delineia agenda de pesquisa na interseção entre CSCW e IA generativa.

2 Fundamentação Teórica

As dimensões fundamentais da colaboração identificadas em [6] — comunicação, coordenação e cooperação —, elaboradas por Fuks et al. [7] como o Modelo de Colaboração 3C, framework central para projetar groupware torna-se um contexto relevante para a Engenharia de Software pois, é uma atividade inerentemente social que exige cooperação estreita entre equipes[21], e agentes de IA acrescentam atores cujos comportamentos comunicativos diferem dos humanos, exigindo reexaminar como comunicação, coordenação e cooperação se manifestam.

Comunicação refere-se à troca de mensagens e à negociação que precede a ação coletiva; coordenação gerencia pessoas, atividades e recursos para que comunicação e cooperação não se percam; e cooperação é a produção conjunta em um espaço compartilhado, na qual os membros executam tarefas e geram artefatos [7]. As três dimensões formam um ciclo iterativo, não uma sequência linear: durante a cooperação, os participantes obtêm feedback de suas ações e *feedthrough* das ações dos companheiros por meio de informações de awareness [7], o tecido conectivo que permite reorientar continuamente comunicação e coordenação à medida que o trabalho evolui.

Nesse cenário, um conceito relevante é o de Workspace awareness, definida por Gutwin e Greenberg [10] como a compreensão atualizada que uma pessoa mantém sobre as interações de outras pessoas em um espaço de trabalho compartilhado, é o conhecimento contínuo sobre onde os outros estão trabalhando, o que estão fazendo e o que planejam fazer a seguir, essencial para coordenar ações, gerenciar o acoplamento entre tarefas, antecipar comportamentos e encontrar oportunidades de auxílio. O *framework* descritivo dos autores identifica os elementos que constituem awareness, como identidade, localização, ações, intenções e artefatos dos colegas, junto com os mecanismos perceptuais pelos quais as pessoas reúnem essas informações. Um achado central é que awareness é mais difícil de manter em groupware do que presencialmente, pois espaços de trabalho computacionais transmitem apenas uma fração da informação perceptual disponível [10]. Esses desafios se intensificam quando agentes de IA se tornam participantes do espaço compartilhado, operando de forma autônoma e paralela, com transparência limitada e sem as pistas comportamentais naturais nas quais as pessoas se apoiam para manter awareness.

2.1 Agentes de IA como Participantes Colaborativos

Um agente de codificação baseado em LLM opera por meio de um ciclo de percepção, memória, planejamento, ação e reflexão, interagindo com ferramentas externas e com o conhecimento compartilhado do projeto [22]. Diferentemente de uma chamada isolada a um modelo, o agente mantém estado, executa ações de múltiplos passos e produz artefatos concretos, como commits e pull requests, no mesmo espaço em que trabalham os desenvolvedores humanos.

Diversas propriedades desses agentes têm implicações diretas para a colaboração. Sua autonomia introduz um ator cuja decisão é parcial ou totalmente opaca para humanos, e a memória de que dispõe cria terreno comum que pode não ser visível para as pessoas [11]. Seeber et al. [18] antecipam esse cenário com a noção de *machine teammates*, agentes autônomos que fazem inferências e participam da tomada de decisão, argumentando que devem ser tratados como participantes, não como ferramentas passivas, o que levanta questões sobre papéis, confiança e responsabilização. Nem o modelo 3C nem workspace awareness foram sistematicamente aplicados a agentes de codificação de IA em uso colaborativo no desenvolvimento de software, lacuna que este artigo aborda ao tratar o agente como um artefato sociotécnico aberto à investigação qualitativa.

3 Trabalhos Relacionados

Banh, Holldack e Strobel [2] investigaram como a IA generativa transforma a engenharia de software por meio de entrevistas com desenvolvedores de dezessete empresas europeias, relatando ganhos individuais de produtividade e a percepção crescente dos LLMs como parceiros colaborativos, não meras ferramentas, o que levanta questões sobre responsabilidade e autoria. Ulfsnes et al. [21] corroboram esses achados com um estudo empírico de treze profissionais, caracterizando a adoção de GenAI como mudança de paradigma; criticamente, desenvolvedores recorrem cada vez mais a ferramentas de IA em vez de perguntar a colegas, rompendo o ciclo informal de aprendizagem das equipes ágeis, o que mostra que a adoção reconfigura o tecido social do desenvolvimento.

He, Treude e Lo [11] fornecem o survey técnico mais abrangente sobre agentes baseados em LLM em engenharia de software, revisando 71 estudos primários; a cooperação baseada em papéis é o padrão dominante, mas os autores deixam de lado as dimensões humana e colaborativa. Watanabe et al. [23] conduzem um estudo empírico de pull requests de agentes de codificação no GitHub focado no desempenho técnico taxa de merge e qualidade, sem examinar a dinâmica colaborativa investigada neste estudo, e Treude e Gerosa [20] propõem uma taxonomia de onze tipos de interação humano-IA focada em confiança e controle, restrita porém a ferramentas de agente único.

Este artigo aborda por meio de um estudo qualitativo exploratório que aplica o modelo 3C e workspace awareness a equipes de software que utilizam agentes de codificação de IA.

4 Metodologia

Este estudo segue um design qualitativo exploratório que combina duas estratégias complementares e aceitas para investigação de fenômenos colaborativos emergentes: observação de interações online e análise documental de logs de comunicação e históricos de contribuição [4]. A investigação apoia-se em traços digitais públicos de interação, que oferecem um substituto verificável para relatos autorreportados e permitem que terceiros auditem cada afirmação empírica na fonte original. A coleta de dados foi realizada entre 06 e 10 de julho de 2026 por um pipeline documentado em quatro etapas, descrito a seguir, cujos scripts, consultas e dados brutos integram um pacote de replicação público.

4.1 Seleção de Fontes e Caso de Estudo

O GitHub foi selecionado como fonte primária: threads de pull request e issues são simultaneamente interações observáveis e logs com timestamp, o conteúdo é público e acessível por API aberta sem credenciais especiais, e a plataforma preserva a identidade de

tipo de cada conta (humano ou aplicação), viabilizando a separação sistemática entre ações humanas e de agentes. Discord e Slack foram excluídos por não serem indexáveis nem verificáveis sem autenticação; fóruns e blogs serviram como fontes documentais secundárias. Como caso de estudo, adotou-se o agente de codificação nativo do GitHub [8], que opera sob a conta de aplicação `copilot-swe-agent[bot]`. A escolha se justifica pela escala de adoção, superior a 1,4 milhão de PRs mescladas até 06/07/2026, e pela identidade de bot estável e filtrável via API, que permite delimitar o universo de interações humano-agente de forma reprodutível. A unidade de análise é a thread de PR contendo troca direta entre desenvolvedores humanos e o agente.

4.2 Procedimento de Coleta

A coleta seguiu quatro etapas, cada uma implementada em um *script* Python disponível no pacote de replicação.

Etapa 1, caracterização do universo. A API pública de busca do GitHub foi consultada em 06/07/2026 com duas expressões documentadas nos scripts: `type:pr author:app/copilot-swe-agent is:merged`, que retornou 1.428.332 PRs mescladas de autoria do agente, e a mesma expressão com o recorte `comments:>8`, que retornou 134.678 threads com discussão acima desse limiar. O filtro `is:merged` garante ciclos completos de colaboração, da abertura à integração do código.

Etapa 2, amostragem por triagem sistemática. Ordenar as threads pelo número de comentários não capta bem a riqueza da interação, pois as mais volumosas tendem a ser dominadas por notificações automatizadas; e a seleção *purposive* inicial (três threads, uma excluída) era frágil demais para sustentar a caracterização. A seleção manual foi então substituída por uma triagem sistemática sobre um *pool* de candidatos extraído do mesmo recorte da Etapa 1. O *pool* une duas varreduras da lista de resultados ordenada por número de comentários, uma ascendente e uma descendente: a ascendente é decisiva por alcançar as threads na faixa de 9 a 25 comentários, onde se concentram os diálogos humano-agente moderados, enquanto a descendente privilegia as threads mais volumosas, dominadas por bots de integração contínua.

Sobre cada candidato aplicou-se, no canal de discussão da PR (os comentários gerais, onde ocorre a negociação humano-agente, distintos dos comentários de revisão *inline* e das revisões formais), uma regra de inclusão objetiva: (R1) a PR foi mesclada, (R2) o canal contém ao menos dois turnos de humano e (R3) ao menos dois turnos do agente. A regra reproduz as decisões da seleção inicial: T1 e T2 são incluídas, e T3 é excluída porque sua única troca humano-agente ocorre no canal de revisão *inline*, e não no de discussão. Por ser condição necessária mas não suficiente, os candidatos aprovados passam ainda por dois filtros qualitativos na leitura fechada: ser um projeto de software genuíno (e não um repositório cujo único propósito é orquestrar o próprio agente em ciclos automáticos) e conter troca direta em língua acessível aos pesquisadores.

Dos 54 candidatos triados, 25 satisfazem a regra objetiva; após os filtros qualitativos, oito novas threads foram incluídas, elevando o corpus para dez threads (Tabela 1), além de T3, mantida como exclusão documentada. O corpus cobre projetos e portes diversos, de correções pontuais de cerca de nove turnos a colaborações longas e sustentadas de cerca de quatrocentos turnos. Os scripts de busca, triagem e extração, o *pool* de candidatos e a planilha de triagem integram o pacote de replicação, permitindo reexecutar a seleção e auditar cada decisão.

Etapa 3, extração integral. Cada thread foi extraída por dois caminhos documentados no pacote de replicação: o canônico consulta quatro endpoints da API REST pública (metadados, comentários de discussão, comentários de revisão de código e revisões formais); um alternativo, salvaguarda diante dos limites de taxa do acesso não autenticado, captura a página pública e a converte na mesma estrutura.

Etapa 4, pré-processamento e separação de identidades. Os três fluxos de comentários de cada thread foram consolidados em uma linha do tempo única ordenada por timestamp, e cada turno foi classificado em três categorias de ator a partir dos campos de tipo de conta e login da API: *agente* (a conta em estudo), *bot* (demais contas automatizadas, como integração contínua e automação de release) e *humano*. Turnos de bots que não o agente em estudo foram marcados como ruído, mantidos nos arquivos como contexto mas excluídos da codificação, pois consistem em notificações automáticas que não contribuem para a decisão colaborativa. Essa triagem revelou que o canal de discussão de T3 continha apenas turnos de um bot de build, e que a única troca humano-agente ocorreu no canal de revisão inline, em um único par de turnos; por ficar abaixo do critério de múltiplos turnos, T3 foi excluída da leitura fechada, mas as condições observadas foram registradas como achados na Seção 5.

Como evidência documental complementar, foram coletadas duas discussões do fórum da comunidade GitHub (#159068 e #164267), uma issue do rastreador da ferramenta de linha de comando oficial (cli/cli#11362) e um relato de engenharia sobre a integração de um agente de codificação a um fluxo GitHub em produção [15], todas com URL pública arquivada na data da coleta. Nenhum dado privado foi coletado; os nomes de usuário citados são públicos e correspondem a mantenedores agindo em capacidade pública.

4.3 Procedimento de Análise

As linhas do tempo consolidadas foram analisadas por análise temática [4], tomando o turno de conversação como unidade de codificação. A codificação foi primariamente dedutiva, com categorias derivadas dos quatro construtos da Seção 2: comunicação, coordenação e papéis, cooperação e confiança, e awareness. Uma passagem indutiva complementar registrou padrões não antecipados pelo framework, seguindo o movimento de familiarização, codificação, agrupamento em temas e revisão contra os dados proposto pelos autores. A codificação foi conduzida por um único pesquisador, condição compatível com o escopo exploratório mas registrada como limitação na Seção 4.5; como mitigação, cada afirmação interpretativa da Seção 5 está ancorada em turnos identificáveis das threads públicas, permitindo verificação independente.

4.4 Considerações Éticas

O estudo utiliza apenas informações de acesso público, sem interação com participantes, enquadrando-se no art. 1º da Resolução CNS 510/2016, dispensada a apreciação pelo CEP/CONEP. As citações limitam-se ao necessário, os nomes reproduzidos são identidades

públicas de mantenedores de código aberto e não se emitem juízos sobre indivíduos, apenas sobre padrões de interação.

4.5 Ameaças à Validade

Quanto à validade de constructo e à confiabilidade, a codificação por um único pesquisador sem checagem entre avaliadores pode introduzir viés, e o GitHub é uma plataforma mutável sujeita a limites de taxa e aos riscos já documentados de minerar o GitHub [12]; ambas mitigadas pela ancoragem dos códigos no framework pré-estabelecido, pela verificabilidade pública de cada fonte e pelo arquivamento dos dados brutos com timestamp e paginação completa (Seção 4.2). Quanto à validade interna, embora ampliado para dez threads, o corpus não sustenta inferência causal e os achados devem ser lidos como padrões ilustrativos; soma-se que muitos repositórios são conduzidos por um único mantenedor, o que estreita a diversidade de configurações de equipe observadas. Quanto à validade externa, evidências de um único agente em repositórios de código aberto podem não generalizar para contextos proprietários nem para pipelines multiagente genuinamente orquestrados.

5 Resultados e Discussão

A Tabela 1 resume o corpus de dez threads incluídas, que cobre projetos e portes diversos: de correções pontuais de cerca de nove turnos, como T1 (um workflow de release) e T2 (publicação de pacote npm com autenticação OIDC), a colaborações longas e sustentadas de cerca de quatrocentos turnos (T6 e T7). T3 permanece como exclusão documentada, por limitar-se sua interação humano-agente a um único par de turnos no canal de revisão inline enquanto seu canal de discussão é povoado por um bot de build, condição também discutida na Seção 5.4. As subseções seguintes primeiro relatam o que foi diretamente observado e em seguida discutem o que a observação sugere à luz da Seção 2.

5.1 Reconfigurações na Comunicação

Nas dez threads incluídas, a comunicação com o agente ocorreu pelo mesmo canal já usado na discussão entre humanos: comandos por menção com `@copilot` em comentários comuns de pull request, sem interface dedicada. O padrão de o agente citar a instrução recebida antes de responder mostrou-se sistemático em todo o corpus, espelhando os movimentos de confirmação com que humanos demonstram escuta ativa e estabelecem terreno comum. O tom, porém, varia de perguntas de esclarecimento pontuais (T1) a longas sequências de instrução imperativa e correção (T6, T7). Em vários casos o comando dispensa qualquer preâmbulo — "Resolve conflicts" (T11), "fix all gate" (T8), "mark as experimental" (T5) — e a correção assume tom francamente diretivo, como em "no that wasn't it, undo that change" (T9) ou "I told you to remove the `node-version: lts/*` specification" (T2). O custo social de emitir ordens secas a um agente é nulo, o que sugere a suspensão do trabalho de preservação de face [9] que regula a comunicação entre colegas humanos, alterando as normas pragmáticas descritas pelo modelo 3C [7].

Duas leituras emergem daí. Primeiro, comunicar-se com um agente não elimina a clarificação iterativa característica do trabalho em par entre humanos; ela apenas se desloca da negociação de entendimento mútuo para a instrução explícita, repetida e progressivamente mais restritiva. Segundo, embora a pesquisa sobre respostas sociais a computadores mostre que pessoas tendem a aplicar normas sociais a máquinas [16], aqui essa aplicação parece suspensa, e a camada de negociação que precede a ação coletiva [6] é parcialmente substituída por comando direto.

5.2 Coordenação e Papéis Emergentes

Coordenar a atividade do agente exigiu infraestrutura sem contrapartida em equipes exclusivamente humanas. O relato de engenharia analisado [15] descreve listas de permissão de comandos, fluxos separados de leitura e escrita, aprovação humana obrigatória antes de escrita e lógica para impedir que o agente responda a si mesmo ou a outros bots, criando laços de retroalimentação. As evidências do fórum da comunidade mostram atrito recorrente ao endereçar o agente: a discussão #164267 trata de como atribuir uma issue ao agente via API REST, e a issue `cli/cli#11362` registra falha da ferramenta oficial de linha de comando ao atribuir-lhe trabalho. Esse conjunto indica que parte da sobrecarga de coordenação foi deslocada da negociação interpessoal, prevista pelo modelo 3C [7], para a configuração prévia de infraestrutura técnica.

No nível de papéis, o mesmo relato descreve tarefas rotineiras delegadas ao agente enquanto desenvolvedores assumem a revisão e supervisão do trabalho gerado, consistente com achados de que profissionais delegam trabalho repetitivo mantendo supervisão [2, 21]. As threads observadas confirmam o padrão no artefato: nas dez threads incluídas, o humano jamais editou o código diretamente, atuando exclusivamente por instruções e revisão, enquanto o agente executava as modificações. Observa-se ainda coordenação entre os próprios humanos em torno do trabalho do agente, como quando um mantenedor transfere a outro a condução da tarefa ("@lpcox I'll let you finish this one", T4).

5.3 Cooperação, Confiança e Custo da Delegação

A cooperação ocorreu no mesmo artefato compartilhado já usado para contribuições humanas, a pull request, um espaço reconhecidamente colaborativo no desenvolvimento social em plataformas como o GitHub [5], sem espaço de trabalho separado para o agente. A confiança observada e relatada é procedural, não interpessoal, aproximando-se da noção de confiança calibrada por processos e verificação da literatura de confiança em automação [14]: o trabalho do agente é aceito porque passa por mecanismos verificáveis, como assinatura de coautoria nos commits mesclados, escalonamento gradual de permissões a partir de modo somente leitura e aprovação humana obrigatória antes do merge [15]. Em T1, o commit final mesclado registra explicitamente a coautoria do agente, tornando a atribuição auditável no histórico do repositório.

Os traços observados, contudo, complicam a narrativa de delegação sem atrito presente em parte da literatura de produtividade. Em T2, um ajuste nominalmente pequeno de configuração de publicação contínua exigiu treze turnos de discussão, sete deles do mantenedor, com seis mensagens sucessivas de correção ao longo de cerca de dezenove horas até corresponder à sua intenção, incluindo a remoção de mudanças que o agente introduziu por iniciativa própria e que o humano considerou desnecessárias. Isso sugere que a cooperação com um agente pode demandar supervisão mais

Tabela 1: Corpus analítico: dez threads de pull request incluídas e T3 (exclusão documentada), coletadas em 2026-07-10, com turnos de conversação por tipo de ator (excluídas as descrições de abertura das PRs). *disc. h/a*: turnos de humano/agente no canal de discussão, base da regra de inclusão.

ID	Thread	Turnos	Humano	Agente	Bots	disc. h/a	Situação
T1	`mcp-typescript-template#69`	9	2	3	4	2/2	incluída
T2	`argent#6`	13	7	6	0	7/6	incluída
T3	`docs.microsoft.com-nuget#3582`	11	1	4	6	0/0	excluída
T4	`gh-aw#25135`	48	12	12	24	12/10	incluída
T5	`gh-aw#22183`	119	48	58	13	46/44	incluída
T6	`tlang#315`	408	199	207	2	137/142	incluída
T7	`ojp#195`	399	207	192	0	162/162	incluída
T8	`ublk-azblob#4`	321	45	242	34	8/9	incluída
T9	`jsondbg#2`	9	5	4	0	5/4	incluída
T10	`tinyusb#3235`	9	3	5	1	3/3	incluída
T11	`common#29`	9	5	4	0	3/2	incluída

próxima e mais persistente do que o termo delegação sugere, e que o custo economizado na execução retorna parcialmente como custo de verificação e correção.

A escala do corpus expandido reforça que a delegação a um agente pode retornar como custo de verificação e correção. Em T7 (unificação de conectividade XA em um proxy JDBC), um único mantenedor emitiu 162 turnos de instrução ao agente ao longo de vários dias, incluindo depuração conjunta em que o agente reconhece não dispor dos logs necessários e devolve ao humano um protocolo de coleta ("Since I don't have access to logs showing where exactly it's stuck, I need you to: 1. Run with … 2. take a thread dump …"). O mesmo balanço aparece em escalas menores: em T6 o mantenedor sustentou 137 turnos até um pedido explícito de saneamento antes do merge, e em T8 o humano reinstrui o agente ao detectar um falso-positivo de CI ("fix SKIP as pass, should not skip"). Em nenhum dos casos o humano editou o código diretamente, atuando por instrução e revisão — um padrão de supervisão granular que avaliações puramente técnicas [11, 23] não capturam.

5.4 Awareness: Lacunas e Autorrelato

Dois desafios distintos de awareness emergiram. O primeiro é a atribuição agregada: quando múltiplos membros da equipe acionam o mesmo agente, todas as ações aparecem sob uma única identidade de bot, e o relato de engenharia registra que isso pode obscurecer qual colega humano originalmente solicitou uma mudança [15]. Trata-se de uma instância concreta da redução de pistas de identidade que a pesquisa em groupware associa há décadas à colaboração mediada [10], agora reintroduzida por uma camada de intermediação algorítmica.

O segundo é um fenômeno que os frameworks clássicos de awareness não antecipam: o autorrelato de limitações pelo próprio agente. Observado inicialmente em um único turno de T1 — em que o agente, ao receber a instrução de reescrever a mensagem de um commit, divulga espontaneamente que seu mecanismo de push rejeita atualizações que reescrevem histórico —, mostrou-se um mecanismo recorrente no corpus expandido, com duas naturezas distintas. A primeira é operacional-ambiental: em T4 e T5 o agente anexa sistematicamente um aviso de que regras de firewall bloquearam seu acesso a determinados endereços ("Firewall rules blocked me from connecting to one or more addresses"), e em T10 o humano já formula a instrução em torno dessa limitação conhecida ("retry to fetch address that is blocked by firewall in previous attempt"). A segunda é epistêmica: em T7 o agente declara os limites do que pode inferir e condiciona o avanço a informação que só o humano pode fornecer, e em T8 reconhece não poder executar uma ação na plataforma ("I can't edit the PR title myself, but a fitting title would be …"). No vocabulário de Gutwin e Greenberg [10], elementos de awareness como intenções, capacidades e limitações — que entre humanos são inferidos de pistas perceptuais — são aqui explicitamente declarados pelo ator artificial, invertendo o mecanismo de aquisição da informação. A contraface é que a awareness assim obtida depende do que o agente escolhe ou consegue relatar, sem os canais de verificação consequencial disponíveis na observação de colegas humanos.

Por fim, a triagem de T3 produziu dois achados adicionais: a fragmentação por canal, já que a única troca humano-agente ocorria no canal de comentários inline (recolhido por padrão) enquanto o canal de discussão visível era povoado por um bot de build; e o emaranhamento de identidades, pois os comentários inline do revisor automático exibem o mesmo login do agente de codificação, embora a revisão formal seja assinada por outra conta de bot. Ambos reforçam a necessidade de separação de identidades em estudos de traços digitais.

5.5 Copresença de Múltiplos Atores Automatizados sem Orquestração

A passagem indutiva revelou um padrão não antecipado pelos construtos dedutivos. Em T1, quatro atores automatizados coocuparam a mesma pull request: o agente de codificação que a abriu, um revisor automático de código do mesmo fornecedor, um bot sumarizador de revisão de outro fornecedor e um bot de automação de release. O padrão se repete em T3, com o agente de codificação, o revisor automático e um bot de build povoando a mesma thread. Nenhum mecanismo central coordenava esses atores; a copresença resultou da adoção incremental e independente de ferramentas pelos mantenedores ao longo do tempo, um acúmulo de bots consistente com observações prévias de que a automação se infiltra de forma incremental no fluxo de desenvolvimento [19]. Essa copresença sem

orquestração central, observada em T1 e T3, repete-se no corpus expandido: em T4, T5 e T8 o canal é compartilhado por bots de integração contínua, sumarização e revisão que coabitam com o agente sem coordenação explícita.

Esse achado qualifica o conceito de orquestração da literatura técnica, que pressupõe uma plataforma central gerenciando comunicação e alocação de tarefas entre agentes [11, 17]. Na prática observada, ambientes com múltiplos atores automatizados já emergem de baixo para cima, sem orquestrador, e a coordenação entre eles é implícita, mediada apenas pelo estado compartilhado do repositório. Isso reforça a pertinência de estudar o agente individual em seu fluxo colaborativo, e não uma plataforma multiagente idealizada.

6 Considerações Finais

Este estudo examinou, de forma preliminar, como agentes de codificação de IA reconfiguram práticas colaborativas em equipes de desenvolvimento de software por meio de análise documental e observação de interações públicas no GitHub, coletadas por um pipeline documentado e replicável. Nos construtos analisados, os achados sugerem que: os agentes são integrados aos canais de comunicação existentes, mas alteram as normas pragmáticas da troca, deslocando a negociação para instrução explícita e repetida; a coordenação passa a depender de configuração técnica prévia, como listas de permissão e salvaguardas anti-retroalimentação, além da negociação interpessoal; a cooperação permanece sob supervisão humana granular, com o custo de execução economizado retornando parcialmente como custo de verificação; e a awareness ganha um mecanismo novo, o autorrelato de limitações pelo próprio agente, ao mesmo tempo em que perde granularidade de atribuição quando múltiplos humanos acionam a mesma identidade de bot. O modelo 3C [7] e workspace awareness [10] permanecem produtivos, mas requerem extensões para acomodar atores que declaram explicitamente intenções e limitações em vez de exibi-las por pistas perceptuais. Trabalhos futuros devem buscar corpora maiores com coleta autenticada, triangulação com entrevistas, observação de pipelines multiagente genuinamente orquestrados e a extensão da lente colaborativa a atividades como a Engenharia de Requisitos, incluindo a geração assistida de artefatos como proto-personas [1, 13].

Disponibilidade de Artefatos

Scripts, consultas, dados (brutos e processados) e instruções de replicação estão disponíveis em repositório anonimizado: https://anonymous.4open.science/r/Agentes-de-IA-como-Pratica-Colaborativa-Pacote-de-Replicao-9365/

Agradecimento

Este trabalho foi apoiado pela FAPERGS, Processos nº 25/2551-0000883-2 e nº 25/2551-0000925-1.

Referências

[1] Firas Ayach et al. 2025. Generating Proto-Personas Through Prompt Engineering: A Case Study on Efficiency, Effectiveness and Empathy. In *Anais do XXXIX Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, Porto Alegre, RS, Brasil, 126–137.

[2] Leon Banh, Florian Holldack, and Gero Strobel. 2025. Copiloting the Future: How Generative AI Transforms Software Engineering. *Information and Software Technology* 183 (2025), 107751. doi:10.1016/j.infsof.2025.107751

[3] Tuomas Bazzan, Benjamin Olojo, Przemysław Majda, Thomas Kelly, Murat Yilmaz, Gerard Marks, and Paul M. Clarke. 2024. Analysing the Role of Generative AI in Software Engineering — Results from an MLR. In *Systems, Software and Services Process Improvement*. Springer Nature Switzerland, Cham, 163–180.

[4] Virginia Braun and Victoria Clarke. 2006. Using Thematic Analysis in Psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. doi:10.1191/1478088706qp063oa

[5] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. Association for Computing Machinery, New York, NY, USA, 1277–1286. doi:10.1145/2145204.2145396

[6] Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. 1991. Groupware: Some Issues and Experiences. *Commun. ACM* 34, 1 (1991), 39–58. doi:10.1145/99977.99987

[7] Hugo Fuks, Alberto B. Raposo, Marco A. Gerosa, and Carlos J. P. Lucena. 2005. Applying the 3C Model to Groupware Development. *International Journal of Cooperative Information Systems* 14, 2–3 (2005), 299–328.

[8] GitHub, Inc. 2026. About GitHub Copilot Coding Agent. GitHub Docs. https://docs.github.com/en/copilot/concepts/agents/coding-agent/about-coding-agent

[9] Erving Goffman. 1955. On Face-Work: An Analysis of Ritual Elements in Social Interaction. *Psychiatry* 18, 3 (1955), 213–231. doi:10.1080/00332747.1955.11023008

[10] Carl Gutwin and Saul Greenberg. 2002. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Computer Supported Cooperative Work (CSCW)* 11, 3–4 (2002), 411–446. doi:10.1023/A:1021271517844

[11] Junda He, Christoph Treude, and David Lo. 2025. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Trans. Softw. Eng. Methodol.* 34, 5, Article 124 (2025), 30 pages. doi:10.1145/3712003

[12] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An In-Depth Study of the Promises and Perils of Mining GitHub. *Empirical Software Engineering* 21, 5 (2016), 2035–2071. doi:10.1007/s10664-015-9393-5

[13] Dilan Karolita, John Grundy, Tanjila Kanij, Hourieh Khalajzadeh, and Jennifer McIntosh. 2023. Use of Personas in Requirements Engineering: A Systematic Mapping Study. *Information and Software Technology* 162 (2023), 107264. doi:10.1016/j.infsof.2023.107264

[14] John D. Lee and Katrina A. See. 2004. Trust in Automation: Designing for Appropriate Reliance. *Human Factors* 46, 1 (2004), 50–80. doi:10.1518/hfes.46.1.50_30392

[15] Chamith Madusanka. 2026. How We Integrated Claude Code Into Our GitHub Workflow. Medium. https://chamith.medium.com/how-we-integrated-claude-code-into-our-github-workflow-97a5db8bcb8e

[16] Clifford Nass and Youngme Moon. 2000. Machines and Mindlessness: Social Responses to Computers. *Journal of Social Issues* 56, 1 (2000), 81–103. doi:10.1111/0022-4537.00153

[17] Angshuman Rudra and Manan Agrawal. 2025. Composable AI Stack for Intelligent Agents: Modular Orchestration Using Context Routing, Memory, and Tools. In *2025 International Conference on Computer and Applications (ICCA)*. IEEE, 1–6. doi:10.1109/ICCA66035.2025.11430973

[18] Isabella Seeber, Eva Bittner, Robert O. Briggs, Triparna de Vreede, Gert-Jan de Vreede, Aaron Elkins, Ronald Maier, Alexander B. Merz, Sarah Oeste-Reiß, Nils Randrup, Gerhard Schwabe, and Matthias Söllner. 2020. Machines as Teammates: A Research Agenda on AI in Team Collaboration. *Information & Management* 57, 2 (2020), 103174. doi:10.1016/j.im.2019.103174

[19] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting Developer Productivity One Bot at a Time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. Association for Computing Machinery, New York, NY, USA, 928–931. doi:10.1145/2950290.2983989

[20] Christoph Treude and Marco Aurelio Gerosa. 2025. How Developers Interact with AI: A Taxonomy of Human-AI Collaboration in Software Engineering. In *2025 IEEE/ACM Second International Conference on AI Foundation Models and Software Engineering (FORGE)*. IEEE, 236–240. doi:10.1109/Forge66646.2025.00033

[21] Rasmus Ulfsnes, Nils Brede Moe, Viktoria Stray, and Marianne Skarpen. 2024. *Transforming Software Development with Generative AI: Empirical Insights on Collaboration and Workflow*. Springer Nature Switzerland, Cham, 219–234. doi:10.1007/978-3-031-55642-5_10

[22] Yanlin Wang, Wanjun Zhong, Yanxian Huang, Ensheng Shi, Min Yang, Jiachi Chen, Hui Li, Yuchi Ma, Qianxiang Wang, and Zibin Zheng. 2025. Agents in Software Engineering: Survey, Landscape, and Vision. *Automated Software Engineering* 32, 2, Article 70 (2025), 35 pages. doi:10.1007/s10515-025-00544-2

[23] Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E. Hassan. 2026. On the Use of Agentic Coding: An Empirical Study of Pull Requests on GitHub. *ACM Trans. Softw. Eng. Methodol.* (2026). doi:10.1145/3798166