Um estudo sobre os impactos da implementação de uma camada de revisão lógica no Scrum

Bárbara Este Fernandez[1], **Fábio Silva Lopes**[1]

[1]Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie (UPM) – São Paulo, SP – Brasil

31937039@mackenzista.com.br, fabio.lopes@mackenzie.br

Abstract. *The Scrum framework can be adapted to help teams achieve the desired success. In the condition of software development, this study proposes to investigate the logical review phase and how it should be applied to provide quality to the generated artifacts. Through a questionnaire and interviews, we aimed to understand how this stage is used in Brazilian Information Technology companies. The analysis of the responses indicated that although the individuals of the agile teams consider it important, little is discussed about the necessary steps for its due efficiency. Therefore, it is recommended to minimally standardize the tasks within the framework with some existing technique, such as Behavior Driven Development, to improve communication between members of agile teams who find it difficult to adopt the logic review.*

Resumo. *O framework Scrum pode ser adaptado para ajudar as equipes a alcançar o sucesso desejado. Na condição de desenvolvimento de software, este estudo propõe-se a investigar a fase de revisão lógica e como ela deve ser aplicada para fornecer qualidade aos artefatos gerados. Por meio de um questionário e entrevistas, buscou-se entender como esta etapa é utilizada nas empresas brasileiras de Tecnologia da Informação. A análise das respostas indicou que embora os indivíduos das equipes ágeis considerem-na importante, pouco se discute sobre as etapas necessárias para sua devida eficiência. Recomenda-se então padronizar minimamente as tarefas dentro framework com alguma técnica já existente, como por exemplo o Behavior Driven Development, para aprimorar a comunicação entre os membros de equipes ágeis que encontram dificuldades na adoção da revisão lógica.*

1. Introdução

O processo ágil, em termos de desenvolvimento de software, é a prática utilizada no gerenciamento de projetos que possui como base o Manifesto Ágil [Project Management Journal 2013]. O Manifesto Ágil, por sua vez, é um documento composto por doze princípios de desenvolvimento que baseiam-se em quatro valores: indivíduos e interações a processos e ferramentas, software funcional à documentação completa, colaboração do cliente à negociação do contrato e resposta a mudanças a seguir um plano [Beck 2001].

Este estudo tem como base a metodologia Scrum, um framework criado por Ken Schwaber e Jeff Sutherland que propõe aplicações empíricas e adaptativas dos conceitos ágeis com o intuito de ajudar pessoas e times a gerar valor. Como cita o próprio *Scrum Guide*:

> O framework Scrum é propositalmente incompleto, apenas definindo as partes necessárias para implementar a teoria Scrum. O Scrum é baseado na inteligência coletiva das pessoas que o utilizam. Em vez de fornecer às pessoas instruções detalhadas, as regras do Scrum orientam seus relacionamentos e interações [Schwaber and Sutherland 2020].

A partir do momento em que uma tarefa é finalizada, entende-se que sua execução agregou uma parcela de valor ao produto final. Quando um time depara-se com uma manifestação negativa por parte do usuário sobre o que foi entregue, esta hipótese é refutada, gerando um impacto negativo no planejamento da *Sprint*, uma vez que pode implicar, por exemplo, na necessidade de repensar o *backlog* da mesma. Nesta situação, uma das maneiras de validar durante o processo possíveis disparidades entre o que foi solicitado e o que está sendo entregue é através da revisão lógica, onde examina-se o código em busca de implementações incorretas ou que não atendem suficientemente às demandas do *Product Owner*.

Apesar dos benefícios, é notória a dificuldade de implementação da revisão lógica nos times ágeis, principalmente pelo tempo que é necessário dedicar a este tipo de tarefa [Bernhart et al. 2010]. Sendo assim, o problema de pesquisa deste estudo consistiu em responder a seguinte pergunta: como é possível aplicar a revisão lógica considerando as características e ritos do Scrum?

[Dingsoeyr et al. 2019] citam que há discrepâncias com relação à interpretação e aplicação das práticas ágeis nos times de desenvolvimento, e que existem poucos estudos empíricos que analisam como os frameworks ágeis funcionam na prática. [Lindvall et al. 2005] orientam que é necessária atenção em aspectos como quão difícil será a manutenção da prática adotada, quais serão seus efeitos colaterais e qual será a satisfação com a utilização da mesma.

O objetivo desta pesquisa é compreender a aplicabilidade da revisão lógica dentro do Scrum, levando em consideração suas cerimônias, convenções e a adaptabilidade do time ágil com tal processo. Este artigo foi estruturado da seguinte maneira: a seção 2 apresenta o referencial teórico, a seção 3 aborda a metodologia utilizada, a seção 4 apresenta os resultados e discussões e por fim a seção 5 apresenta as conclusões do estudo.

2. Referencial Teórico

2.1. Métodos Ágeis

Muitas das pesquisas publicadas propõem maneiras de adaptar os processos atuais de acordo com as necessidades de cada time [Barlow et al. 2011, Cao et al. 2009, Lindvall et al. 2005].

[Pawlak 2021] e [Lindvall et al. 2005] citam que pouco se discute sobre quais aspectos da organização são necessários atenção para que seja possível que a transformação ágil seja feita com sucesso. [Barlow et al. 2011] mencionam a dependência organizacional do processo como um problema a ser considerado, explicando que é um fator decisivo para classificar sua adoção como sucesso ou falha. O custo de coordenar tarefas sequencialmente dependentes, por exemplo, certamente possui influência na adoção de um processo que as contém. Também citam como exemplo do que classificam como “in-

terdependência sequencial" a revisão lógica, onde um desenvolvedor precisa que outro analise e aceite seu código para então definir como "entregue" seu trabalho.

Assimila-se à "interdependência sequencial", também, a coleta de requisitos para o planejamento de funcionalidades. Sem o entendimento das necessidades do usuário, é difícil que se consiga entregar funcionalidades de valor. Apesar disso, métodos ágeis como o Scrum propõem que novos requisitos sejam estudados ao passo que funcionalidades já definidas sejam desenvolvidas [Maschietto et al. 2020].

2.2. A Etapa de Revisão de Código

Nesta pesquisa, a revisão lógica do código está de acordo com a definição de [Dooley 2017], onde a etapa de codificação de uma funcionalidade é procedida da revisão pelos pares. Apesar disso, Dooley descreve-a como uma reunião síncrona presencialmente executada. Aqui, refere-se especificamente ao exercício assíncrono e virtual, através de ferramentas de sistemas de controle de versão, como o *Git* e sua contrapartida de interface gráfica *GitHub*.

A revisão lógica possui as seguintes características: um repositório central que armazena todas as versões do código de um software na nuvem é disponibilizado para que o time trabalhe em funcionalidades de maneira paralela e sem conflitos. Quando um desenvolvedor deseja iniciar o desenvolvimento de uma nova funcionalidade, uma versão do código mais recente da nuvem é disponibilizada em seu computador, comumente chamada de versão "local", pois existe somente no seu dispositivo. Ao finalizar sua tarefa, o desenvolvedor precisa publicar essa versão no repositório central, para que todos os outros membros da equipe tenham acesso ao novo código. A revisão consiste na submissão do código ao que chama-se de solicitação de mesclagem (em inglês *Merge Request*) do novo código à versão mais atualizada disponibilizada no repositório. Do ponto de vista do autor, este processo consiste descrever quais foram as alterações realizadas, o objetivo da solicitação (usualmente relaciona-se com um requisito de negócio) e em escolher um ou mais revisores, que geralmente são outros desenvolvedores que trabalham no mesmo código, para a devida análise. Já aos revisores cabe a leitura, entendimento, testagem e aprovação/desaprovação da solicitação.

A falta de tempo e de recursos são os principais motivos para que os times ágeis não adotem revisões durante o processo de desenvolvimento [Bernhart et al. 2010, Ciolkowski et al. 2003, Harjumaa et al. 2005]. Além disso, uma pesquisa feita por [Ciolkowski et al. 2003] demonstrou que para 50% dos 865 respondentes a falta de conhecimento também influencia na adoção da tarefa.

Uma das qualidades desse esforço no desenvolvimento é que ele melhora a disseminação do conhecimento entre o time, resultando em maior protagonismo por parte dos desenvolvedores na hora de manter e melhorar o software. Em processos ágeis, é comum a utilização da revisão em pares (*peer review*), pois acredita-se que o código é uma responsabilidade de todos [Bernhart et al. 2010, Dooley 2017]. Por outro lado, [Harjumaa et al. 2005] indicam que a revisão do código deve ser uma atividade anônima, pois os desenvolvedores podem sentir-se constrangidos ao receberem críticas negativas.

Para que se aplique o exercício da revisão como parte da cultura ágil, as literaturas sugerem que a única maneira de os times adotarem-na é estabelecê-la como uma tarefa diária, obrigatória e com técnicas predefinidas [Ciolkowski et al. 2003, Dooley 2017].

2.3. Requisitos nos Métodos Ágeis

Segundo [Haigh 2010], observa-se na última década um interesse em compreender melhor o alinhamento entre a área de TI e de negócios, incentivado pela percepção de que os problemas do desenvolvimento de software estão gradativamente sendo atribuídos à comunicação entre quem o desenvolve e aqueles para quem está sendo desenvolvido. Para Haigh, um melhor entendimento dos requisitos leva a uma melhor comunicação entre as partes envolvidas, o que por sua vez permite a priorização correta das funcionalidades. [Bermejo et al. 2014] citam a comunicação com o cliente como um dos atributos diretamente ligados ao sucesso do desenvolvimento de software, uma vez que interações frequentes garantem uma maior confiança. Discutem também a importância da qualidade da documentação nesse processo, e que estes artefatos produzidos durante o desenvolvimento devem ser úteis e garantir o suporte adequado às partes envolvidas.

O sucesso de um projeto Scrum também está diretamente atrelado em como se aplica o exercício da engenharia de requisitos. A coleta dos requisitos pode ser aplicada de inúmeras maneiras, entre elas com a abordagem *Behavior-Driven Development* (BDD) e a construção de histórias do usuário [Maschietto et al. 2020]. [Cohn and Beck 2004] descrevem uma história de usuário como sendo um documento que representa uma funcionalidade que agregará valor tanto ao usuário quanto ao cliente que a solicita. Ela possui três aspectos: a descrição da história propriamente dita, as discussões em cima da descrição, que servem para explicá-la com detalhes e os critérios de aceitação, que são utilizados para validar quando uma história está completa. O emprego do BDD como técnica de desenvolvimento de software estende o uso das histórias de usuário a todo o ciclo de desenvolvimento de software. O BDD representa uma maneira de certificar que os requerimentos funcionais são tratados corretamente pelo código através da conexão das suas descrições com os testes, além de oferecer a possibilidade de automatização através de ferramentas [Solís and Wang 2011].

3. Método de Pesquisa

Esta pesquisa pautou-se em elaborar um estudo exploratório sobre revisão lógica de código em projetos ágeis, buscando definir o estado da arte nesta temática. De modo complementar, um questionário foi aplicado para alunos de cursos de graduação em TI, objetivando identificar se as práticas citadas na literatura estão em uso em empresas que atuam com desenvolvimento de software e entrevistas foram realizadas com participantes de times que utilizam o Scrum.

3.1. Questionário

O objetivo do questionário foi aumentar o entendimento sobre os processos ágeis hoje utilizados nas empresas brasileiras da área de Tecnologia da Informação e como estes realizam a revisão lógica frente às demandas. Por tratar-se de uma área fortemente presente na Internet, o mesmo foi disponibilizado nas redes sociais *LinkedIn*, *Facebook*, *Twitter* e também nos fóruns educacionais da Faculdade de Computação e Informática da UPM.

A fim de adquirir o maior número de respostas, o questionário é anônimo e composto por cinco perguntas. Primeiro, identifica-se o respondente como profissional de Tecnologia da Informação e em seguida são apresentadas três categorias para seleção: gestor de projetos, desenvolvedor ou analista de qualidade. A escolha de uma categoria condiciona o conteúdo da segunda etapa do questionário. Para cada uma delas, são

Tabela 1. Perguntas realizadas no questionário de acordo com a categoria aplicada, seu tipo e quais as alternativas (quando aplicável)

Categoria	Pergunta	Tipo de Pergunta	Alternativas (quando aplicável)
Geral	Você trabalha com Tecnologia da Informação?	Múltipla Escolha	Sim/Não
	Seu time utiliza alguma metodologia ágil como processo? Se sim, qual?	Texto de resposta curta	-
	Qual área melhor identifica o seu cargo?	Múltipla Escolha	Projetos/Desenvolvimento/Qualidade
Qualidade	Durante os testes de aceitação, encontro muitas funcionalidades que não estão totalmente de acordo com o que foi solicitado pelo Product Owner.	Escala Likert	1 (Discordo Totalmente) a 5 (Concordo Totalmente)
	Acredito que se os desenvolvedores validassem o código de acorod com o que foi solicitado, a quantidade de falhas encontradas durante os testes de aceitação seria reduzida.	Escala Likert	1 (Discordo Totalmente) a 5 (Concordo Totalmente)
Projetos	Tenho interesse em entender os processos que os times de desenvolvimento e de qualidade utilizam para entregar uma funcionalidade.	Escala Likert	1 (Discordo Totalmente) a 5 (Concordo Totalmente)
	Se vamos estourar a data de entrega, prefiro...	Múltipla Escolha	Deixar de lado a qualidade do código/Entregar um escopo menor
Desenvolvimento	Seu time realiza a revisão do código antes da entrega? Se não, qual seria o motivo para não adotarem essa prática?	Texto de resposta curta	-
	Como desenvolvedor, acredito que durante a revisão do código, além da busca por falhas e sugestões de melhorias, deve-se validar se o código está de acordo com o que foi solicitado. Por isso, é importante entender a demanda antes de revisar o código.	Escala Likert	1 (Discordo Totalmente) a 5 (Concordo Totalmente)

propostas afirmações baseadas na opção escolhida e o respondente deve classificá-las de acordo com sua opinião. Desta maneira, as perguntas tornam-se únicas e é possível sumarizar suas respostas de maneira detalhada. O uso da escala *Likert* de cinco pontos (1 - Discordo Totalmente a 5 - Concordo Totalmente) traz flexibilidade ao respondente, porém mantém a característica fechada da pergunta. Já as perguntas de natureza aberta resumem-se a ”Sim”ou ”Não”e uma breve justificativa da escolha da resposta.

Foi realizado um pré-teste do questionário com um pequeno grupo, cujos participantes são alunos dos cursos da Faculdade de Computação e Informática. A versão inicial do questionário apresentava duas perguntas à respeito da principal atividade exercida pela empresa onde o respondente trabalha. Deste pré-teste ficou evidente que a informação não trazia nenhum valor que pudesse ser associado às outras perguntas. A Tabela 1 apresenta as perguntas do questionário final, juntamente com o seu tipo, quais as alternativas (quando admissível) e em qual categoria estas foram aplicadas.

O processo de análise e interpretação das respostas iniciou-se com a tabulação dos dados extraídos. As duas principais classificações utilizadas foram o uso de metodologias ágeis e tradicionais (com atenção especial para o uso do Scrum) e a categoria escolhida pelo respondente. Em seguida, foi realizada a construção de gráficos de superfície em colunas para visualizar a porcentagem de respostas para cada pergunta. Este modelo permite associar com facilidade as duas classificações escolhidas. Excepcionalmente, como mostra a Figura 1, onde a pergunta realizada não permitiu a correlação das duas variáveis, utilizou-se o gráfico de setores. Por fim, efetuou-se a análise dos padrões de respostas, associando-os com o referencial teórico disponibilizado.

3.2. Entrevistas

As entrevistas, por sua vez, possuíram caráter aberto e foram realizadas de maneira síncrona, através de videoconferência, onde propôs-se discutir o papel da revisão lógica no contexto do entrevistado, discutindo pontos como quais são as dificuldades encontradas em sua aplicação e como esta pode aprimorar o produto final desenvolvido. O objetivo da aplicação desta segunda técnica de pesquisa foi o de aproximar a generalidade das res-

postas do questionário com as experiências detalhadas pelos entrevistados, assim trazendo uma camada mais aprofundada de argumentação baseada na exemplificação. Por este motivo, as entrevistas foram realizadas após a compilação das respostas do questionário.

O planejamento da entrevista constituiu-se na elaboração de alguns tópicos que foram utilizados como guia para direcioná-la. Foram estabelecidos os seguintes tópicos:

1. Traçar o perfil do entrevistado:
   (a) Qual sua função dentro da equipe;
   (b) Se utiliza o Scrum como processo;
2. O entendimento do entrevistado sobre o conceito de revisão lógica:
   (a) Como e se esta etapa é aplicada no contexto de seu trabalho;
   (b) Quais as principais características do código são analisadas durante a revisão;
   (c) Se enxergam alguma dificuldade ou impedimento na realização deste tipo de tarefa;
3. Como é realizada a engenharia de requisitos:
   (a) Se a documentação escrita dos requisitos é levada em consideração;
   (b) Como é feita a disseminação do conhecimento entre os membros da equipe e com novos integrantes;
4. Como é a relação entre os gestores de projeto com os analistas que trabalham diretamente com o código, principalmente no que diz respeito ao planejamento das tarefas.

As conversas foram gravadas com a permissão dos entrevistados e com a garantia de anonimato. Em seguida, os vídeos foram transcritos afim de facilitar a análise das respostas e a associação com os dados extraídos do questionário.

4. Resultados & Discussões

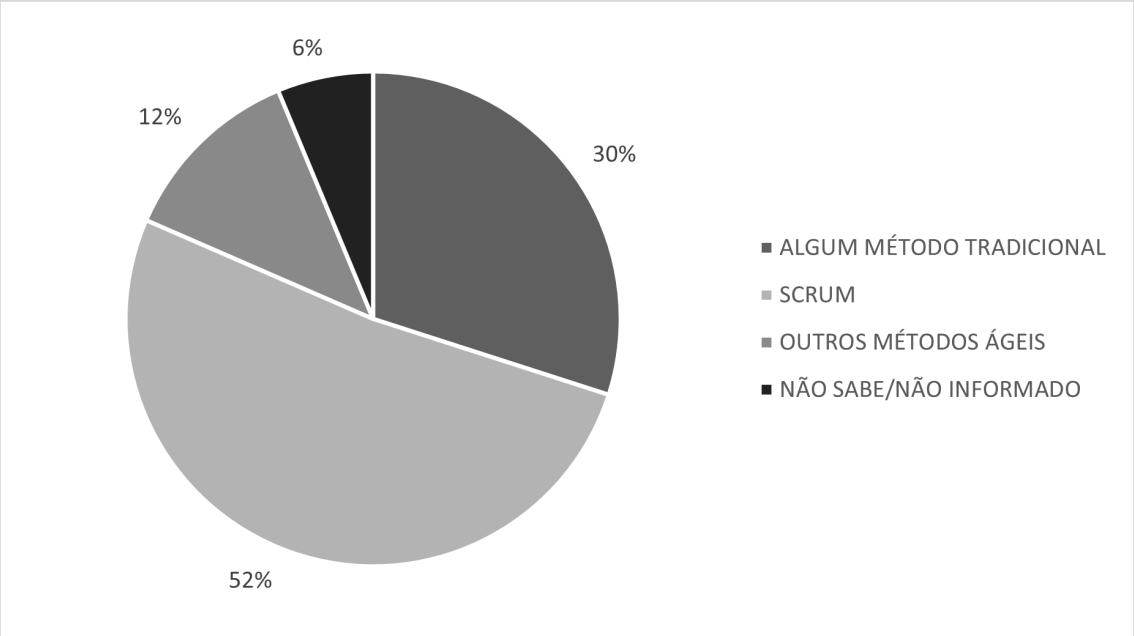
4.1. Questionário

O questionário disponibilizado obteve 384 respostas, das quais 75% (287) foram respostas válidas (respondentes que sinalizaram que trabalham na área de Tecnologia da Informação). Quanto à metodologia utilizada, 52% explicitaram o Scrum como método principal, 30% utilizam métodos tradicionais, 12% utilizam algum método ágil que não o Scrum, e 6% não sabem ou não informaram (Figura 1). A maior distribuição dos métodos ágeis vai ao encontro da hipótese de que estas são mais difundidas entre as empresas, como mencionam [Pawlak 2021, Dingsoeyr et al. 2019]. Como indica a Figura 1, nota-se considerável a presença dos métodos tradicionais nas empresas brasileiras de TI.

Para o prosseguimento da análise, foram excluídos os respondentes que não informaram qual metodologia ágil utilizam. Dentre os selecionados, 70% classificaram-se como desenvolvedores, 18% como gestores de projeto e 12% como analistas de qualidade.

As perguntas elaboradas para os gerentes de projetos procuraram entender o quanto estes estão integrados com as tarefas que o time exerce na construção de um software. Questionou-se aos gerentes se havia interesse em entender quais processos os times de desenvolvimento e qualidade utilizam em suas tarefas. 69% responderam que havia muito interesse em compreender como o software é construído. Apenas 4% indicaram

Figura 1. Distribuição dos métodos tradicionais e ágeis entre os respondentes

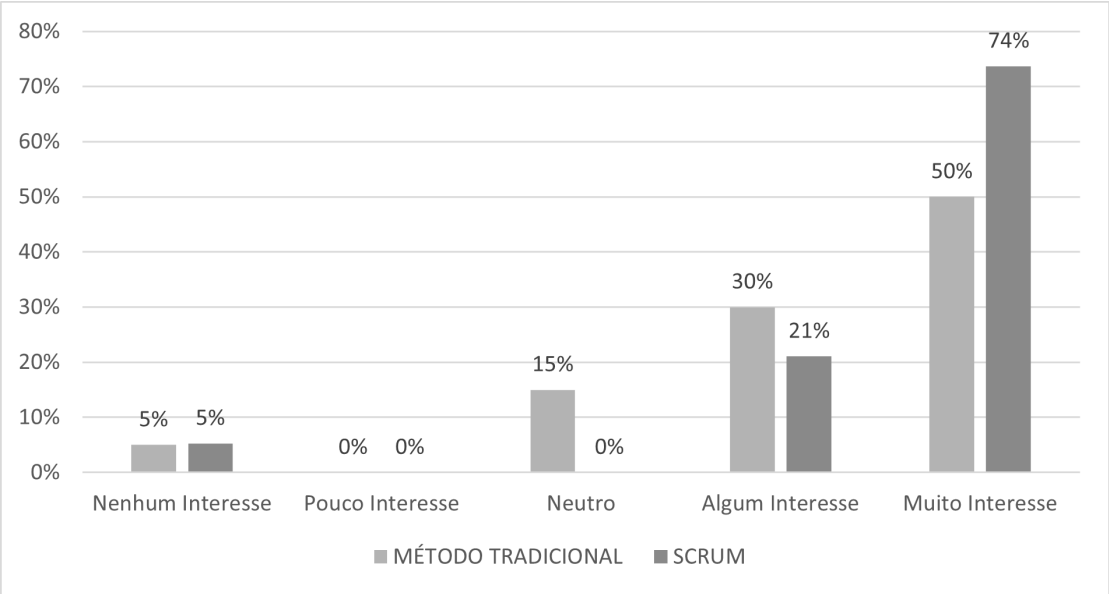


não haver nenhum interesse em conhecer o método. Pela Figura 2, nota-se que os gerentes que utilizam o Scrum têm uma tendência maior a se interessar pelos processos, quando comparados com aqueles que trabalham com métodos tradicionais, o que indica um ponto favorável à adoção da revisão lógica. Isto é justificado pela segunda pergunta realizada, onde todos os gerentes que utilizam métodos ágeis assinalaram que preferem entregar um escopo menor à reduzir a qualidade do código quando a data de entrega está próxima, o que indica que estes preocupam-se com a qualidade do software de modo geral, e entendem que o código é peça fundamental para atingir este objetivo.

Entre os respondentes, 32 identificaram-se como analistas de qualidade. Para estes, foram apresentadas duas afirmações a respeito do software entregue para análise e como isto afeta seus trabalhos.

Indagou-se se estes, durante os testes de aceitação, encontram muitas funcionalidades que não estão de acordo com o que foi solicitado pelo *Product Owner*. Apenas um respondente concordou totalmente com esta afirmação. 28% concordaram parcialmente e 50% mantiveram-se neutros. Porém, quando analisamos as respostas entre os que utilizam algum método tradicional contra os que trabalham com o Scrum, como mostra a Figura 3, nota-se que os analistas de qualidade que trabalham com o Scrum são os que mais identificam-se com a afirmação. Isto porque os métodos tradicionais trabalham com um extenso estudo no design do software antes de iniciar a sua implementação (*Big Design Up Front*, como citam [Maschietto et al. 2020]), e no caso de depararem-se com funcionalidades incorretas, é explicada pela alteração desta funcionalidade durante o percurso de desenvolvimento mais do que por terem sido entendidas erroneamente pela equipe. Já o Scrum, assim como outros métodos ágeis, propõem a construção incremental do software como maneira de mantê-lo atualizado frente às demandas, o que significa que o estudo do design e implementação andam paralelamente e assim aumentam as chances de erros de interpretação.

Figura 2. Interesse dos gerentes de projeto em entender quais processos o time utiliza para a entrega de um software



Além disso, independente da metodologia utilizada, os respondentes concordaram que a validação do código frente às demandas solicitadas, assim como propõe a revisão lógica, reduziria a quantidade de falhas encontradas durante os testes de aceitação. 88% dos respondentes concordam (parcialmente ou totalmente) com esta afirmação e 13% permaneceram neutros. Nenhum respondente discordou.

Por fim, a última categoria a ser analisada são os desenvolvedores de software, que consiste na categoria com a maior taxa de resposta, com um total de 189 respondentes.

A taxa de revisão do código é similar tanto para os que utilizam métodos tradicionais quanto para os que utilizam métodos ágeis, como mostra a Figura 4. Dos que não utilizam a técnica, a falta de interesse e/ou conhecimento foi o motivo mais citado, de acordo com a categoria ”Scrum”(Figura 5). Isso vem de acordo com a pesquisa feita por [Ciolkowski et al. 2003] em seu artigo, que apresentou a falta de conhecimento como o terceiro fator mais citado para a não adoção da revisão do código, logo atrás do prazo de entrega.

A falta de recursos disputa com o prazo de entrega como o segundo principal motivo para os times dessa categoria não adotarem a prática. Estas foram evidenciadas por [Ciolkowski et al. 2003] e [Harjumaa et al. 2005] em suas respectivas pesquisas como sendo as principais razões para os times ágeis.

Em um segundo momento, indagou-se se o desenvolvedor concordava com a importância de entender a demanda antes de revisar um código, uma vez que é preciso validar se este está de acordo com o que foi solicitado. 83% concordaram com a afirmação e apenas duas pessoas discordaram. Relacionando a pesquisa realizada por [Ciolkowski et al. 2003], onde 40% das respostas indicaram que realizam a leitura dos documentos de especificação e design, nota-se a preocupação dos desenvolvedores com a implementação dos requisitos. Pouco se fala, porém, de como esta validação é implementada.

Figura 3. Distribuição da concordância entre os analistas de qualidade à afirmação de que muitas funcionalidades não estão de acordo com os requisitos

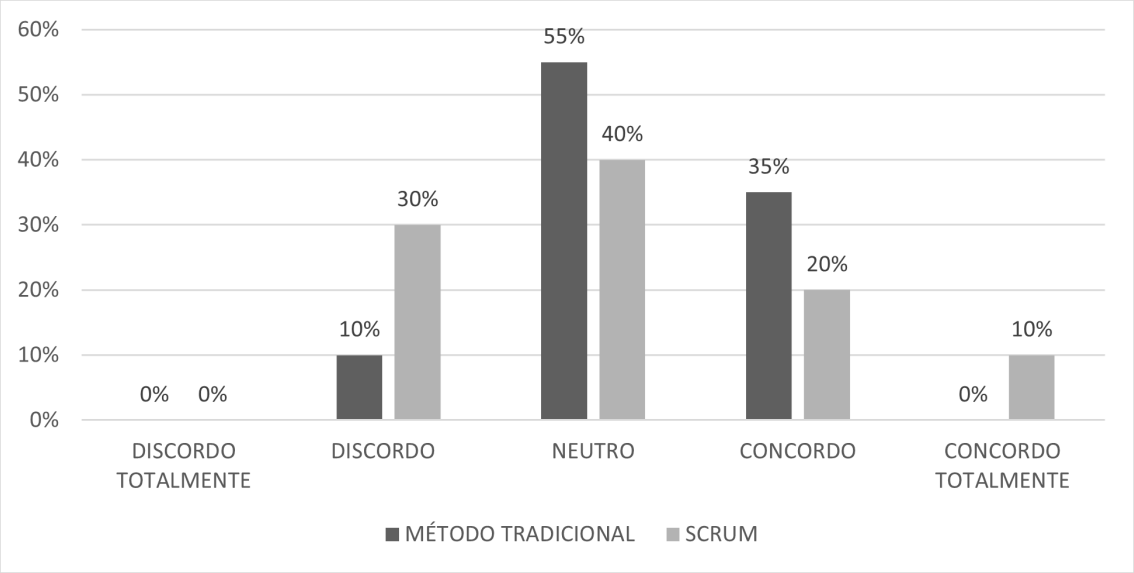


Figura 4. Distribuição das respostas dos desenvolvedores sobre a utilização da revisão lógica como etapa obrigatória entre as metodologias estudadas

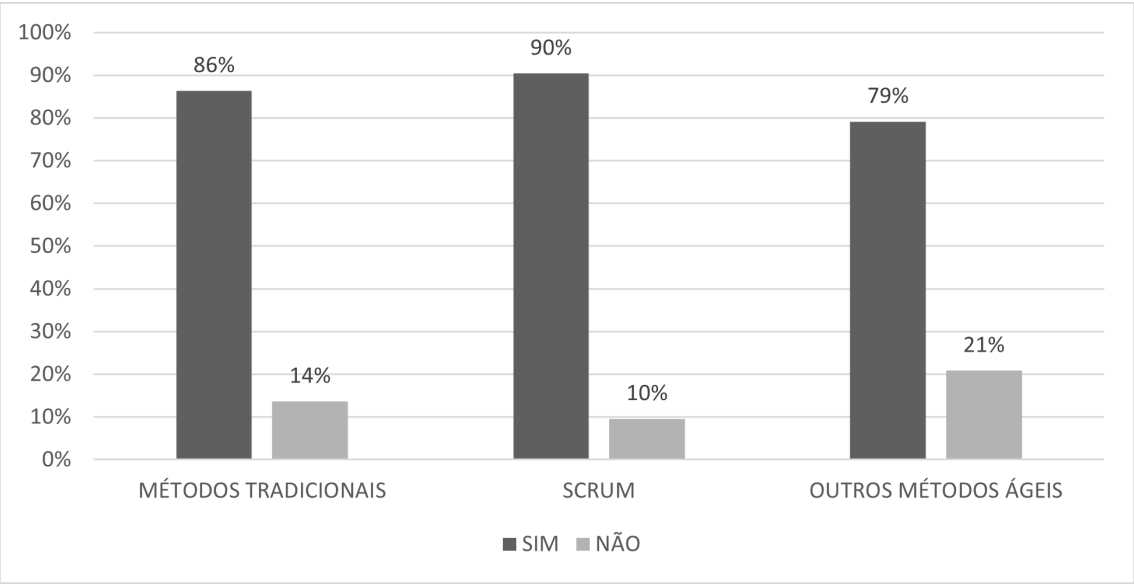
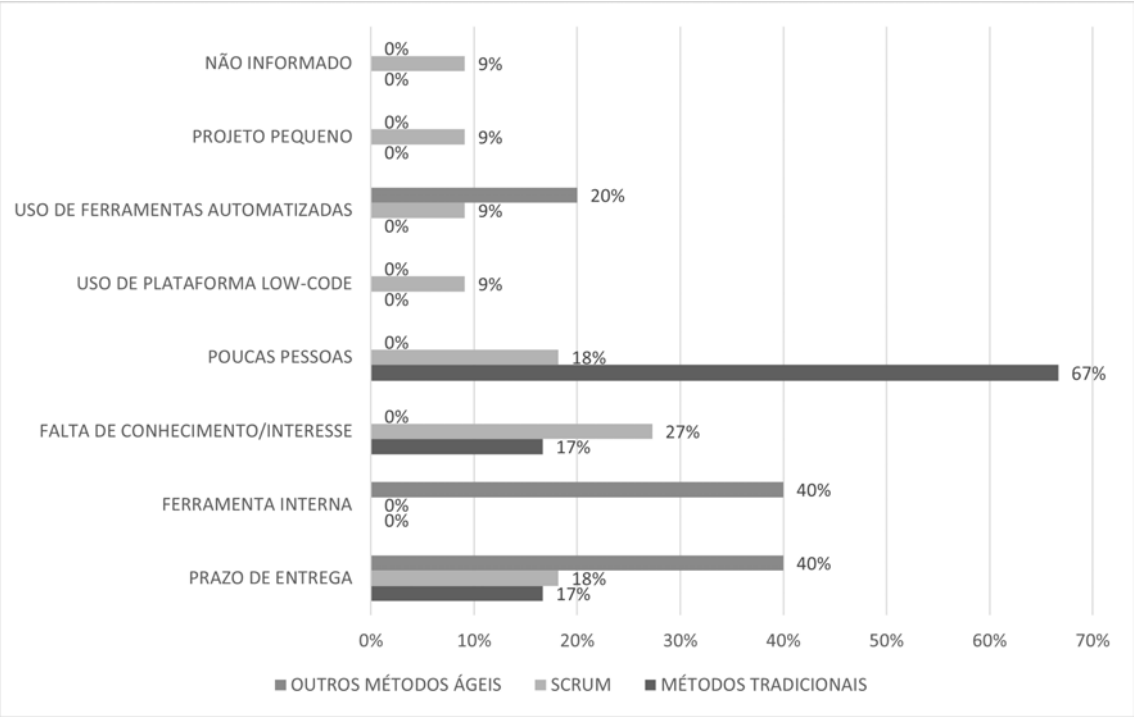


Figura 5. Motivos dos desenvolvedores para não utilizarem a revisão lógica como etapa obrigatória



4.2. Entrevistas

Foram conduzidas duas entrevistas síncronas, através de videochamada, com um desenvolvedor de uma grande empresa financeira, aqui denominado E1, e com uma gerente de projetos de uma *startup* que fornece software como um serviço, aqui denominada E2. Ambos os entrevistados trabalham com Scrum como metodologia ágil.

Com relação à etapa de revisão lógica do código, ambos a consideram importante. Para E1, é o momento de verificar com os desenvolvedores se os requisitos estão corretamente implementados, citando também ser indispensável a participação de todos neste processo, pois cada indivíduo tem um entendimento único do que foi solicitado. E2 acredita que é uma etapa que fornece um refinamento extra com relação às regras de negócio, mas que para ser eficiente é necessário o seu planejamento prévio, com um tempo delimitado para ser executada dentro da *Sprint*. A afirmação fornecida por E1 exemplifica o princípio de responsabilidade geral exemplificado por [Bernhart et al. 2010] e [Dooley 2017]. O apontamento de E2 vem de acordo com as explicações de [Ciolkowski et al. 2003] e [Dooley 2017], no que diz respeito à necessidade de padronização como forma de adaptação do time à sua utilização.

Ambos também concordam que falta atenção ao processo de coleta e descrição dos requisitos. E1 explicou que em seu trabalho não costumam descrever as tarefas com detalhamento suficiente, apesar de existir um modelo: ”No meu time isso não ocorre, não porque não gostamos, mas sim por fatores como falta de tempo e falta de priorização desse tipo de atividade, já que os integrantes do time conseguem executar uma demanda mesmo com a tarefa não estando muito bem detalhada. No entanto, sei da importância de a demanda estar muito bem explícita nas tarefas, porque se saio da empresa amanhã e

uma pessoa nova fica responsável pela tarefa, ela não terá o mesmo conhecimento que eu tinha". A entrevistada E2 compartilhou que frustra-se muito por as pessoas acharem que no Scrum não é necessário documentar: "Parece meio clichê, porque o Scrum tem muito de 'entrega de valor para o cliente', mas eu percebo que priorizar e entender essa parte do negócio é realmente complicado. As pessoas têm um problema, mas na hora de descrevê-lo às vezes acabam passando uma solução ao invés de apenas relatá-lo". Esta afirmação apresentada por E2 vem de acordo com o estudo de [Haigh 2010] sobre a comunicação com os *stakeholders*. E como explicam [Maschietto et al. 2020]: "*Stakeholders* podem não saber exatamente o que querem, expressando de maneira equivocada suas reais necessidades".

Por fim, discutiu-se sobre a presença do gerente de projeto durante a revisão lógica. E1 exprimiu: "Gostaria que ele (gerente) possuísse uma visão mais técnica sobre o produto, para que conseguisse revisar o código também. Acho que os desenvolvedores precisam ter o entendimento do negócio, mas o especialista sempre será aquele que está em contato direto com os *stakeholders*". Já E2 explicou que acredita ser desnecessária: "A comunicação entre os integrantes do time ágil é uma característica fundamental. Se um time se esforça para compartilhar suas inseguranças e dificuldades, não é necessária a validação do gerente no trabalho do desenvolvedor, pois a equipe está a par dos empecilhos". A visão exposta por E2 corrobora com o princípio ágil de que equipes autogerenciáveis são propensas a entregar artefatos de melhor qualidade [Beck 2001].

5. Conclusão

Apesar de o número de respostas ao questionário exceder às expectativas, a amostra é pequena com relação à população que procura caracterizar, portanto é notória a margem de erro que possui. Além disso, cabe para trabalhos futuros uma análise estatística mais aprofundada. Com relação às entrevistas, a ausência da entrevista com um analista de qualidade aponta para a falta de associação com as informações obtidas do questionário. Apesar disso, foi possível observar certo padrão nas respostas, que condizem com o referencial teórico apresentado.

[Dooley 2017] afirma que apenas a testagem do sistema consegue sozinha encontrar até 50% das falhas, mas que esse número pode chegar à 93% se combinada com a revisão. Para atingir o sucesso na implementação de uma etapa de revisão lógica, são muitos os fatores além do período de codificação do software que devem ser levados em consideração. A revisão lógica, na realidade, inicia-se logo no início do entendimento da demanda, assim que um cliente demonstra seu interesse em criar um produto.

A revisão lógica depende diretamente da qualidade com que os requisitos são descritos e de como essas informações passam de etapa por etapa no Scrum. É necessário, portanto, que a comunicação entre as partes do processo seja padronizada. Sugere-se então que essa padronização seja feita com abordagens como o BDD, que permitem que as informações sejam compreendidas por uma fórmula universal aplicada a todo o processo, dessa forma facilitando a revisão.

Este estudo é parte integrante do projeto "Objetos de aprendizagem para apoio em práticas de residência de software". Sua contribuição está atrelada a construção da biblioteca de objetos de aprendizagem em Engenharia de Software. Os artefatos gerados neste projeto são adotados no âmbito do Programa de Residência de Software do Grupo de Pes-

quisa de Ambientes de Produção de Software da Universidade Presbiteriana Mackenzie – MackLeaps.

Referências

Barlow, J., Giboney, J., Keith, M., Wilson, D., Schuetzler, R., Lowry, P., and Vance, A. (2011). Overview and guidance on agile development in large organizations. *Communications of the Association for Information Systems*, 29:25–44.

Beck, K. (2001). Manifesto for agile software development. `http://agilemanifesto.org/`. Acesso em: 18.10.2021.

Bermejo, P., Zambalde, A., Tonelli, A., Souza, S., Zuppo, L., and Rosa, P. (2014). Agile principles and achievement of success in software development: A quantitative study in brazilian organizations. *Procedia Technology*, 16:718–727.

Bernhart, M., Mauczka, A., and Grechenig, T. (2010). Adopting code reviews for agile software development. In *2010 Agile Conference*, pages 44–47.

Cao, L., Mohan, K., Peng, X., and Balasubramaniam, R. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems*, 18.

Ciolkowski, M., Laitenberger, O., and Biffl, S. (2003). Software reviews, the state of the practice. *IEEE Software*, 20:46–51.

Cohn, M. and Beck, K. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Dingsoeyr, T., Falessi, D., and Power, K. (2019). Agile development at scale: The next frontier. *IEEE Software*, 36:30–38.

Dooley, J. (2017). *Software Development, Design and Coding*, chapter Walkthroughs, Code Reviews, and Inspections, pages 271—-282. Apress, Berkeley, CA.

Haigh, M. (2010). Software quality, non-functional software requirements and it-business alignment. *Software Quality Journal*, pages 361–385.

Harjumaa, L., Tervonen, I., and Huttunen, A. (2005). Peer reviews in real life - motivators and demotivators. In *Proceedings - International Conference on Quality Software*, pages 29–36.

Lindvall, M., Muthing, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., and May, J. (2005). Agile software development in large organizations. *Computer*, 37:26–34.

Maschietto, L. G., Rodrigues, T. N., Bianco, C. M. D., de Moraes, D. M. P., de Souza Vettorazzo, A., de Andrade, A. L. C., Viviani, C. A. B., de Morais, I. S., de la Rocha Ladeira, R., and Bezerra, W. R. (2020). *Processos de Desenvolvimento de Software*. Grupo A.

Pawlak, R. (2021). Implementation aspects of agile methods in large organizations. *e-mentor*, 90:64–72.

Project Management Journal (2013). *Agile Project Management: Essentials from the Project Management Journal*. John Wiley & Sons, Inc.

Schwaber, K. and Sutherland, J. (2020). The 2020 scrum guide. `https://scrumguides.org/scrum-guide.html`. Acesso em: 15.10.2021.

Solís, C. and Wang, X. (2011). A study of the characteristics of behaviour driven development. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 383–387.